VoIP Networks Monitoring and Intrusion Detection

Mohamed Nassar

Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
UFR STMIA

VoIP Networks Monitoring and Intrusion Detection

THESIS

presented and defended publicly the 31st March 2009

in order to obtain

PhD from The Henri Poincaré University – Nancy 1

(speciality computer sciences)

by

Mohamed NASSAR

Composition of jury

Reporters : Georg CARLE
Ludovic MÉ

Examiners : François CHARPILLET
Olivier FESTOR
Radu STATE

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thloria.

Acknowledgments

I would first like to thank my advisor, Dr. Olivier Festor for the chance he gave me to work on a subject of remarkable importance and scientific openness. I also thank my co-advisor Mr. Radu State with whom I had the opportunity and pleasure to work during these three years and a half. Their advices at scientific and personal levels have always been clear and have resulted in the production of this work. I also thank them for their availability and regular discussions we have had. Thanks for having shown my weaknesses and also congratulated me every time I achieve a milestone.

I thank Mr. George Karl and Mr. Ludovic Mé for agreeing to report my thesis and for the efforts they made to accomplish this work. Their criticisms were constructive and their indications were very useful to me.

I present my sincere thanks to Mr. François Charpillet who did me the honor of presiding the jury and for the time he spent to read this document.

I express my sincere thanks to the members of the MADYNES team for the friendly atmosphere they did prevail in the team. I thank my colleagues at the offices of PhD students (some have already finished their PhDs): Humberto Abdelnur, Mohamed Salah Bouassida, Rémi Badonnel, Thibault Cholez, Vincent Cridlig, Guillaume Doyen, Jérome François, Abdelkader Lahmadi, Tom Leclerc, Cristian Popi, Julien Siebert et Gérard Wagner for their true friendship, for their listening and for their support in difficult times. Without them, these past years have not been as pleasant. I would like to thank Humberto Abdelnur, Balamurugan Karpagavinayagam, Nataraj Mocherla and Cristi Stefan who worked in parallel on other aspects of VoIP security. Their results and the discussions we have had were very useful and constructive to me. I also thank Frédéric Beck and Abdelkader Lahmadi for their technical assistance and their wise recommendations. I thank Mohamed Bouali and Anca Ghitescu that marked their internships at MADYNES with remarkable humor and friendliness.

I am grateful to Charbel Rahhal for his careful reading of the manuscript and for all the remarks he gave.

I am also thinking of my friends with whom I have had moments of complicity during this period in Nancy (in alphabetical order): Mohamed Ali Ahmad, Ali Attar, Toufik El Khatib, George Habib, Jamil Houhou, Rimond Hamia, Nazih Ouwayed, Samer Merhi, Ali Nassour, Jamal Saboune and Rami Saad. Surely, I forget someones. Would they excuse me.

I thank Mr. and Mrs. Kassem, Mr. and Mrs. Masri for considering me as a member of their families and for all their generosity and courtesy.

My thanks also go to Ms. Hania Rida for his optimism and encouragement that were an essential comfort during the drafting of this manuscript.

Finally, I want to express my gratitude to my parents and my family who have always supported and encouraged me. Without them I would not be able to accomplish this work.

I dedicate this thesis
to my mother "Hayat"
and my father "Khodor"

Contents

List of Figures

List of Tables

General Introduction

VoIP is a new and very attractive technology. Its definition bypasses the transmission of voice over packet-switched IP networks to meet other multimedia applications and to ensure dynamism, mobility and innovative applications. Due to its great flexibility and economical advantage over traditional PSTN, its deployment widely increases by both enterprises and individuals. Based on a set of standard and proprietary protocols, VoIP products are not limited to end-user equipments, but include as well call processors and managers, signaling and media gateways, proxies and firewalls.

VoIP inherits the adjacent security problems associated to the IP augmented with new specific ones. Vulnerabilities of signaling and media protocols could be exploited for eavesdropping, fraudulent usage and denial of service. VoIP is attractive for advertisers that can easily deploy Internet call centers and make automated calls at low costs.

Security considerations for VoIP have been and still are the subject of a large audience in the industrial, academic and government communities. In fact, VoIP security is strongly constrained by its special characteristics like quality of service issues, address translation and dynamic call establishment across firewalls. Conventional data networks policies like encryption, authentication and data integrity are highly recommended. However, they don't totally fit into practical deployments of VoIP architectures which are of large scale, open and dynamic nature. Such policies are incompatible with network address translation, increase latency, and need a key distribution infrastructure. More stress should be put on second line of defense policies like proactive defense, intrusion detection and monitoring mechanisms for the mitigation or the prevention of attacks. Important work in both host and network intrusion detection has already been done by the industrial and academic research communities, focused in scope towards network intrusion detection for transport, routing and application level protocols. However, specific approaches for VoIP are still in a preliminary stage.

Our thesis is motivated to leverage existing conceptual solutions for the VoIP specific application domain. We focus on the design, validation and implementation of new models and architectures for performing proactive defense, monitoring and intrusion detection in VoIP networks. The manuscript is composed of three parts and organized as follows: The first part contains the relevant state of the art in the domain of VoIP security. In the first chapter, we give an overall, yet brief introduction about VoIP signaling, media transfer and network address translation traversal protocols. In the second chapter, we go across a panorama of the VoIP threats. Finally, in the third chapter, we discuss the related works focusing on defense architectures against those threats, and we position our contributions towards them.

In the second part, we detail our contributions which are composed by three approaches: In the first chapter, we propose **a machine learning approach for VoIP signaling traffic monitoring**. Our monitor aims to detect anomalies in the incoming/outgoing traffic based on a machine learning algorithm. We propose two techniques to play the role of our system's brain: Support vector machines and Bayesian networks. In the second chapter, we present our proactive solution: a **VoIP-specific honeypot** and we describe its design and implementation. The honeypot is equipped by an investigation procedure about the received messages leading to a message classifier that we describe in details. In the third chapter, we propose a **distributed multi-layer intrusion detection solution** combining host-based and network-based intrusion detection systems by a central event correlation engine. Our works show how complementary approaches can jointly provide an in depth defense for VoIP architectures.

The third part is dedicated to the assessment of our contributions. In the first chapter, we present our VoIP agents to emulate good or malicious VoIP users. We highlight that if spread by a worm, our agents form a botnet that can be commanded to attack VoIP services and we discuss the appearing of such a malware in the future. In the second chapter, we validate our monitoring approach by exhaustive tests over real-world and test-bed collected network traces. We show how to configure and adjust monitoring parameters and we compare between the two proposed techniques (Bayesian networks and support vector machines) in term of classification accuracy and anomaly detection. The last chapter of the manuscript is to present the design of a network-based intrusion detection system acting as a first layer component in our event correlation framework. We show the ability of our prototype to detect a set of signaling attack signatures using event correlation rules. The results of our evaluation are promising in two directions: towards a real-time deployment of our monitoring scheme and towards efficient VoIP domain protection using event and alert correlation.

Part I

State of The Art

Chapter 1

Introduction to VoIP Protocols

Sommaire

1.1 Introduction

This chapter presents an overview of the standard protocols involved in delivering voice and multimedia services over packet switched data networks. The link, routing and transport Internet layers ensure different services to support three categories of application protocols:

- Signaling protocols: e.g. SIP (session Initiation Protocol) [82] and MGCP (Media Gateway Control Protocol) [6]. The signaling has a special value in any telephony architecture, it is the key mechanism of establishing, routing, switching and accounting the calls. Also in Internet telephony, it is the tool to create innovative services

and novel applications. Securing any VoIP architecture must give special interest to signaling. While MGCP is limited to manage media gateways by a call-agent at the Internet edge, we were especially interested by SIP which is the de-facto standard of end-to-end signaling over the Internet.

- Media protocols: e.g. RTP/RTCP (Real-time Transport and Transport Control protocols [85]).

- Utility protocols: e.g. DNS (Domain Name Service) [85], DHCP (Dynamic Host Configuration Protocol)[25] TFTP (Trivial File Transfer Protocol)[92], and ARP Address Resolution Protocol) [72].

SIP and H.323 (protocols umbrella for packet-based multimedia communication) [39] share by now the standardized VoIP market. In our work, we chose to focus on SIP as the de-facto standard protocol of Next Generation Networks (NGN) and Future Internet. The rest of this chapter is organized as follows: we compare between SIP and H.323 in Section 1.2. Section 1.3 overviews the SIP protocol. Media description and media transfer protocols used by SIP are introduced in Section 1.4. Utility protocols can not be all expanded in the scope of this manuscript. Network Address Translation (NAT) protocols are given in Section 1.5. Section 1.6 concludes the chapter.

1.2 A comparison between SIP and H.323

H.323 was designed by the International Telecommunication Union (ITU). As a member of the H.32x family of recommendations, it was destined to the communication over LANs with nonguaranteed QoS. H.323 is an umbrella of protocols including H.225 for signaling, H.245 for media negotiation and capability exchange, H.235 for privacy and encryption, and H.450 for supplementary services (e.g. call transfer). An H.323 domain architecture is composed from: endpoints (terminals or phones), gateways (to interface other telephony domains), Multipoint Control Units (MCU) (responsible of conference calls) and Gatekeepers. The terminals register to a gatekeeper in order to be reachable and to make calls. The gatekeeper offers several services to terminals of its **zone** like gateway localization and address translation. H.323 suite protocols inherit many aspects of their design and implementation from the circuit-switched PSTN (Public Switched Telephony Networks) and ISDN (Integrated Services Digital Network). Their use of binary encoding like ASN.1 (Abstract Syntax Notation One) results in smaller message size but more difficult implementation as Internet application developers found. Compared to H.323, SIP has the following strength points:

- SIP is professionally developed by the IETF (Internet Engineering Task Force) to be scalable over Internet and to cooperate with Internet standards and capabilities (e.g. DNS, URL);

- SIP is text-based with heritage from HTTP and SMTP so that it can be easily scripted, logged and inspected;

- SIP has presence and instance messaging capabilities and presents a high potential to invent new applications in the future;
- SIP is well supported by the industry: it has been adopted by mobile operators in their third generation networks and services (3GPP or third Generation Partnership Project);
- SIP takes advantage of Internet security mechanisms like encryption, authentication, integrity and certificates, even if it inherits a large set of Internet vulnerabilities from another side.

While H.323 dominates right now the Internet video conferencing market based on its ISDN heritage, it is widely deployed in small PSTN replacements where no much complexity to be handled. SIP -which was a little bit late to start its development cycle- has learnt many lessons from H.323: it added support for TCP, wireless, conferencing capabilities and new message types helping complex situations. The approaches proposed in this work have been studied in a SIP-based deployment but they can be applied to H.323 in some cases.

1.3 Introduction to SIP

The SIP protocol is exhaustively defined by the RFC 3261 [82]. Basically, SIP allows two communicating parties to set up, modify and terminate a phone call. Text-based with heritage from HTTP and SMTP, SIP is a request-response transaction-based protocol. SIP's latest version (2.0) defines thirteen type of requests and six families of responses as depicted in Table 1.1 and Table 1.2. A SIP Dialog is composed from one or more transactions. A transaction is formed by a request and a final response. Transactions are classified as client transactions (from the request sender side) and server transactions (from the request receiver side). They are INVITE transactions (where optional informational responses can precede the final response) or non-INVITE transactions. A final response is acknowledged by an ACK message only in the case of an INVITE transaction. An ACK message in response to a success final response (200 OK) is considered as a separate transaction. The SIP addressing scheme is based on the URI (Uniform Resource Identifier) e.g. `sip:user@host:port;parameters`. Contrary to a URL (User Resource Locator), a URI refers to a logical entity with a strong notion of mobility. In this section, we present briefly SIP network entities and we give examples of call setup and registration.

1.3.1 SIP network entities

Different SIP clients and servers are introduced next.

- User agents: A user agent (UA) sets up or tears down media sessions with other user agents and must maintain the state of calls. A UA is composed form two applications: The User Agent Client (UAC) to initiate requests to other UAs and receive their responses, and the User Agent Server (UAS) to process received requests from

other UAs and generate responses. A UA should advertise its media capabilities and features in the body of its SIP requests allowing other UAs to learn of them.

- Presence agents: A presence agent notifies its subscribers about some presence information. Sources of this information can be SIP UAs which register to the presence agent and publish their states or also non-SIP entities.

- Back-to-back user agents: A back-to-back UA (B2BUA) hides a number of UAs behind behaving as if it is the origin of their requests. When it receives a request from one of its UAs, a B2BUA replaces signaling and media information within the request and sends out a reformulated one. The inverse operation is done in the opposite direction when responses for this request are received. Also, the B2BUA relays the media sessions for its UAs. B2BUAs are used to implement anonymous services, private branch exchanging and PES (PSTN Emulation Service).

- Gateways: A gateway interfaces SIP to other networks (e.g. ISUP (ISDN User Part), CAS (Circuit Associated Signaling) and H.225). It could be defined as a user agent acting on behalf of another protocol. A gateway terminates the signaling path but not necessarily the media path (for example, in the case of a SIP/H.323 gateway, the SIP end-point and the H.323 end-point exchange the media session directly or through a media gateway). In a MGCP context, one MGC (Media Gateway Controller) controls one or several MG (Media Gateway). The MGC manage opening and closing media endpoints in the MG through the MGCP protocol. Non-SIP protocols like TRIP (Telephony Routing over IP) and TGREP (Telephony Gateway Registration Protocol) provide routing information to SIP proxies and UAs in order to find their way towards gateways when they need to interconnect to other telephony networks.

- Proxy servers: A proxy server helps routing SIP messages on behalf of their senders. It is allowed to make small modifications in the requests while preserving the end to end transparency between the sender and the receiver. A proxy server has access to one or several databases in order to query presence or location information. Also, it can use DNS to map domain names in the SIP requests. Special DNS cases are ENUM [28] which maps between global (E.164) telephone numbers and SIP URIs (within a NAPTR (Naming Authority PoinTeR) DNS record) in order to interconnect the PSTN, and DNS SRV (SeRVice locater) [37] which allows to locate SIP proxies within a domain (e.g. in regard to the used transport protocol). We have two types of proxy servers: statefuls and stateless. A transaction stateful proxy must keep the state of ongoing transactions while for example, a stateless proxy should never retransmit a lost message. A stateful proxy must start a timer whenever a request is forwarded and wait for its response. If no response is received within a certain time period, it steps forward and retransmits the request.

- Redirect servers: A redirect server doesn't route any message. Instead, it responds with a redirection message containing location information.

- Registrar: A registrar (or a registration server) is responsible of processing SIP REGISTER messages. It creates a temporary binding between the IP phone number (also known as the Address of Record AoR) and the IP address (or aliases) of the registering device. A registrar authenticates a registering device using a shared secret and a HTTP-like challenge scheme. A UA can use special REGISTER messages to retrieve a list of its current registrations, clear all registrations or add one alias to its registrations.

Note that these entities are logical, for example a registrar and a proxy server can be installed on the same physical support. These different entities are shown in action in the examples below.

1.3.2 SIP call example

An example of a session establishment between two SIP UAs is shown in Figure 1.1. The session starts by an INVITE request sent from the caller to the callee. The callee responds automatically by informational responses (TRYING then RINGING). One time the call is answered, the callee sends a success response (200 OK). The caller acknowledges this response by sending an ACK message then the media session starts immediately. The Session Description Protocol (SDP) [41] is responsible for media description and negotiation between the two parties. In our example, the caller inserts an SDP body in the INVITE message while the callee inserts another in the 200 OK response. As an example of negotiation, the caller chooses U-law as encoding scheme while the callee chooses A-law. Having A-law capability as well, the caller accepts the choice of the callee and starts sending an A-law encoded flow. The caller chose 10502 as the RTP port where it whishes receive the media session and the callee chooses the port 34154. More details about SDP are given in Section 1.4. At the end, one of the two parties has to send a BYE request. The other party acknowledges this BYE by a 200 OK response.

The INVITE message of the example is expanded next:

```
INVITE sip:1000@192.168.1.10 SIP/2.0
Max-Forwards: 10
Via: SIP/2.0/UDP 192.168.1.2:5060;branch=z9hG4bK1248BB87
CSeq: 7599 INVITE
To: <sip:1000@192.168.1.10>
Content-Type: application/sdp
From: <sip:bob@192.168.1.2>;tag=2AC98585
Call-ID: 580112446@192.168.1.2
Subject: Direct Call
Content-Length: 256
User-Agent: thomstream-7240
Contact: <sip:bob@192.168.1.2:5060;transport=udp>
P-hint: outbound

<body not shown>
```

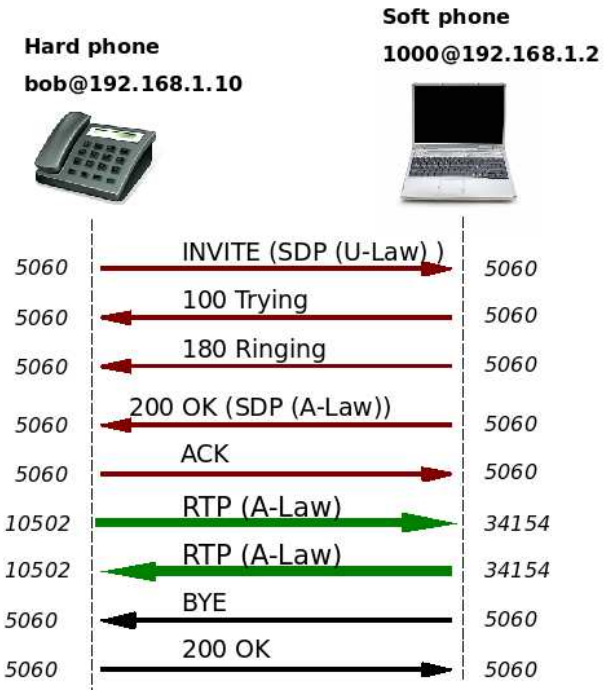


Figure 1.1: SIP call example

The Call-ID and the tag in the From header identify the dialog of the message. The callee adds a tag in the To header to fully identified this dialog. The remote (To) tag is particularly important when a request is forked towards different destinations. The branch parameter in the Via header identifies the transaction of the message. The first line of the request contains the destination URI and the method (INVITE). The subsequent lines are SIP headers of the form *header:value;parameter=value*. The To and From headers identify respectively the callee and the caller. The Contact header gives the address of the caller so that it can be reached directly (useful in case there are proxies). The Via header indicates transport information like the transport protocol, the IP and port used to send the message. The Command Sequence (Cseq) is formed by a sequence number and a method. The sequence number is incremented by one if a new request with the same method is sent. The User-Agent header carries the vendor string of the used SIP device.

In the previous scenario, the caller knows the IP address of the callee. This is not true in general since the request URI could not be directly reachable. The general form of a SIP call is presented in Figure 1.2 (known as the SIP trapezoid).

1.3.3 SIP registration example

Bob needs to register its Address of Record (AoR) to the registrar sever of his VoIP domain. He sends a REGISTER message where the *To* header contains the AoR and the *Contact* header contains the IP address where he is reachable. The registrar binds between the logical and physical addresses in a location database so that any proxy server having

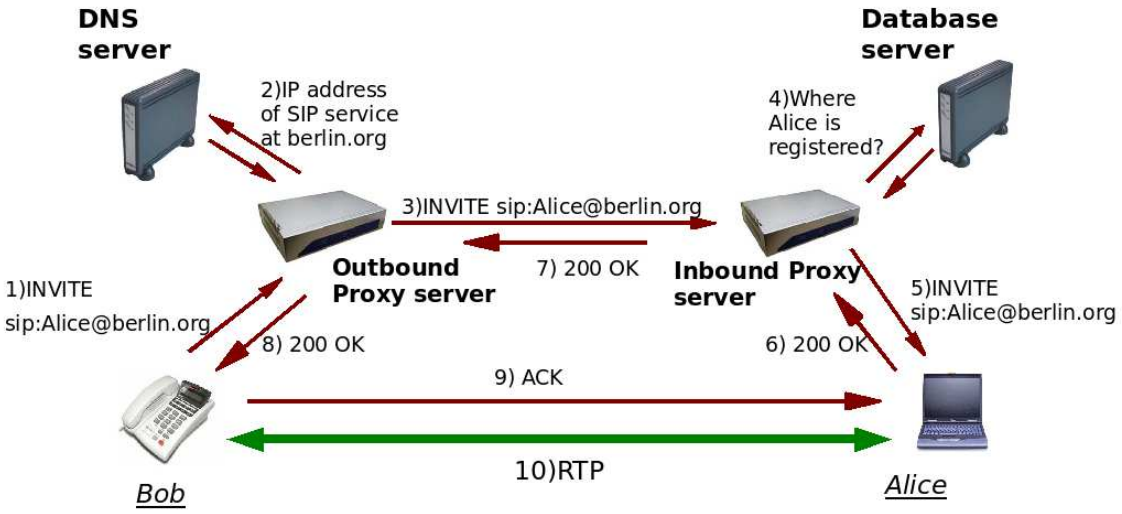


Figure 1.2: SIP trapezoid (from [45])

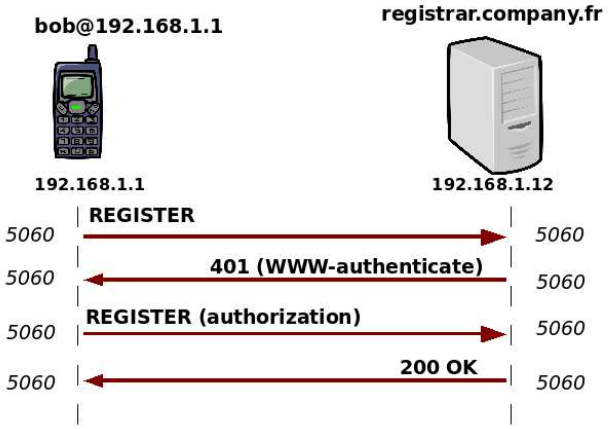


Figure 1.3: Registration example

access to that database is able to route incoming calls to Bob. An `expires` parameter in the Contact header indicates how long the registration has to be kept unless it is refreshed by a new REGISTER. In most deployments, registrars ask for authentication. In our example (Figure 1.3), the registrar responds with a 401 Unauthorized where a *WWW-authenticate* header contains a challenge for a shared secret. The challenge is based on the HTTP digest (composed by a realm, a nonce and an encryption algorithm e.g. MD5). Bob sends a new REGISTER where an *Authorization* header contains the requested authentication credentials (composed by its user-name and password, the realm and the nonce). SIP supports mutual authentication but it is rarely used which is vulnerable to multiple risks.

Table 1.1: SIP methods

Method	Description
INVITE	To establish media sessions between user agents
REGISTER	To notify a SIP network of its current contact URI
BYE	To terminate an established media session
ACK	To acknowledge final responses to INVITE requests
CANCEL	To terminate pending searches or call attempts
OPTIONS	To query a user agent or sever about its capabilities
REFER	To request a user agent to access a URI or URL resource
SUBSCRIBE	To establish a subscription for the purpose of receiving notifications
NOTIFY	To convey information about the occurrence of a particular event
MESSAGE	To transport instant messages
INFO	To send call signaling information within a media session
PRACK	To acknowledge receipt of reliably transported provisional responses
UPDATE	To modify the state of a session without creating a new dialog

Table 1.2: SIP response families

Family	Description
Informational	Call progress
Success	The request is received and processed with success
Redirection	Used by a redirect server or in call forwarding
Client Error	The request can not be processed as it was submitted
Server Error	The request can not be processed because of an error with the server
Global Error	The request will fail wherever it is tried

1.3.4 SIP security mechanisms

Authentication, encryption and integrity of SIP messages can be ensured using hop-by-hop or end-to-end mechanisms. IPsec suite [47] provides confidentiality, integrity and origin authentication for IP packets using a Security Association (SA) between the IP source and destination. A successful deployment of IPsec requires a scalable and automated SA and key management. Transport Layer Protocol (TLS [23]) which is based on the Secure Socket Layer (SSL) and runs over TCP enables confidentiality and integrity. Mutual authentication is also assured by exchanging certificates during the TLS handshake procedure. While self-signed certificates can not be always trusted, maintaining a Public Key Infrastructure (PKI) requires much management effort. IPsec and TLS works in hop-by-hop to secure a SIP session but their cryptographic computations are not trivial. TLS adds multiple round trips to open a connection increasing message latency.

The secure SIP (SIPS) URI scheme requires the use of end-to-end TLS transport (with the exception of the last hop, which may use some other encryption mechanism). This allows a UA to know that the SIP message path does not traverse any unencrypted links through the presence of intermediary proxies [45].

S/MIME [76] allows end-to-end encryption and data integrity. Some SIP headers can not be encrypted if they are needed by intermediary proxies for SIP routing. However, the

encryption of message bodies containing SDP prevents eavesdropping and media injection attacks. For an introductory material about network security in general, please refer to Chapter 7 in [53]. For more discussions about SIP security mechanisms, the reader is invited to [35, 52], or Chapter 8 of [91].

Other aspects of SIP are debated in the Internet community today. Several IETF design groups study NAT traversal, Application Layer Gateways (ALG) and firewalls, Internet and application mobility, incorporation in the 3GPP and wireless issues but we do not discussed them in the scope of this chapter.

1.4 Media description and media transfer protocols

1.4.1 SDP: Session Description Protocol

SDP [41] is more a description syntax than a communication protocol. It was initially intended to describe multicast sessions with the Session Announcement Protocol (SAP [42]). An example of an SDP body of a SIP message is shown next:

```
v=0
o=bob 26372 26372 IN IP4 192.168.1.1
s=session
c=IN IP4 192.168.1.1
t=0 0
m=audio 18400 RTP/AVP 3 0 8 101
a=rtpmap:3 GSM/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
```

Table 1.3: Mandatory SDP fields

Field	Name	Description
v	Version	The current version of SDP is 0
o	Origin	Uniquely identifies the session and the session originator
s	Subject	Name of the session
c	Connection	The connection type and address
t	Time	The start time and the stop time of the session
m	Media	The media session type, port, protocol and the *format-list*
a	Attributes	Describes one item in the *format-list*

Each line in the SDP message is of the form *field=parameter1 parameter2 ... parameterN*. A field is represented by one lower-case letter. For example, the letter `o` stands for origin and its parameters identify the origin of the session: `o=username session-id version network-type address-type address`. The letter `c` stands for connection and its parameters describe the source that will be sending the media packets: *c=network-type address-type connection-address*. The SDP fields that are mandatory are shown in Table

1.3. More details can be found in [45]. SDP is typical for media negotiation with SIP: the caller embodies its media capabilities in the INVITE (or the ACK) while the callee embodies it's in the 200 OK. Some fields are not really relevant like `subject` and `time` but they are mandatory. This is because SDP was not initially intended to SIP. Nevertheless, other fields are very relevant like `connection`, `media` and `attributes` because they allow the listing and the selection of supported media types. `Connection` or `attributes` can also be used to enable a put-on-hold option.

1.4.2 RTP and RTCP: Real-time Transport and Transport Control Protocols

RTP [85] is based on UDP to transport data in real time. RTP adds a small binary header of 12 octets to the transported packet as shown in table 1.4. Actually, RTP doesn't add any quality of service over UDP but it allows properly handling packet loss, jitter (variations in packet transport delay) and unordered packet arrival due to asymmetric routing.

Table 1.4: RTP header fields

Name	Bits	Description
Version (V)	2	The current version of RTP is 2.0
Padding (P)	1	This bit is set in case of padding octets
eXtension (X)	1	This bit is set if the RTP header is extended
Content source identifier Count (CC)	4	The number of contributing sources
Marker (M)	1	Indicates the start of a new frame
Payload Type (PT)	7	The codec in use
Sequence Number (SN)	16	To be incremented for each sent RTP packet
Timestamp (T)	32	The payload sampling time
Synchronization SouRCe Identifier (SSRCI)	32	The sender of the RTP packet
Contributing SouRCe identifier (CSRCI)	32	Identifies a contributing source

RTCP [85] allows the two parties of an RTP session to exchange statistics (e.g. jitter, Number of packets sent, received, or lost) within quality reports, and basic identity information. RTCP uses different packet types like SR (Sender Report), RR (Destination Report) and SDES (source DEScription).

SRTP and SRTCP [12] provide security services to RTP and RTCP like encryption, authentication and integrity. RTP can also be secured using IPsec. Securing RTP is important to protect VoIP calls from media spamming, media replay and eavesdropping.

1.5 NAT Traversal Protocols

Firewalls and NATs pose several difficulties to the work of the SIP protocol. A Firewall has to dynamically open and close media-transport ports (pin holes) as required by SIP/SDP messages, a feature that is not supported by traditional static firewalls. A NAT is not aware about the IP addresses used in the SIP headers while it is mapping between an internal/private IP address and an external/public IP address. Because the Via headers still contain private addresses, routing of the SIP message becomes impossible at the outside of the NAT. Similarly, the media session can not be established because its announced IP

and port can not be reached. Moreover, a NAT may change the mapping scheme during one SIP session if it is not transported over TCP. Application Layer Gateways (ALGs) are designed specially to bypass such kind of problems by handling properly application layer packets. Other solutions are NAT traversal protocols such as the Simple Traversal of Udp through Nats (STUN) [83] and the Traversal Using Relay Nat (TURN) which is still an Internet draft. Next, we give a short description of these two protocols.

1.5.1 STUN

The STUN protocol defines the communication process between a STUN client which is behind a NAT and a STUN server which has a public IP address. A STUN client opens a TLS connection with a STUN server in order to exchange a shared secret. Then, the STUN client sends one or several `Binding Requests` over UDP using the shared secret. The STUN server issues a `Binding Response` containing the IP and port of the source of one UDP packet as it was received. Hence, the client is able to discover the type of an intermediate NAT, if any, and to bypass it.

This scheme is effective to bypass **full cone** (constant bounds) and **restricted cone** (constant bounds but an outgoing packet has to be sent in order to open an incoming path) NATs. However, it is not useful in the case of **symmetric** NATs (different bounds are used depending on the IP destination) or when two communicating parties are both behind different NATs. For the latter, the TURN protocol proposes a solution.

1.5.2 TURN

TURN is very similar to STUN. In addition, a TURN client sends an `Allocate Request` to reserve an IP address and port at a STUN server and use them as point of relay. Hence, the TURN server has to relay the signaling and the media from and to the client. TURN impacts the quality of service due to its triangular routing scheme.

From another side, some SIP and SDP extensions have been defined in order to mitigate some NAT problems. One SDP extension suggests using a symmetric RTP session (the callee sends RTP packets to the IP and port in the RTP packets received from the caller). Another extension allows to explicitly announce an RTCP port because it can not be necessarily calculated from the announced RTP port after mapping (normally the RTCP port should equal the RTP port plus one). Finally, one SIP extension suggests that it would be better for a UA to register to an inbound proxy/registrar by a permanent TCP session. Requests destined to this UA would be routed through this open session.

1.6 Conclusion

In this chapter, we tried to give an overview over the VoIP protocols suite. Such an overview is essential to address the topics covered along this manuscript. However, we are far away from giving an overall survey. Many books and references were dedicated in the recent years for VoIP and VoIP security issues. Particularly, this bibliography [45, 91, 77]

was very instructive for us. In the second chapter, we address the problem statement: the threats that we must be careful about to secure the new VoIP arena.

Chapter 2

Problem Statement: The VoIP Threats

Sommaire

2.1 Introduction

The appearing of new telecommunication technologies is changing dramatically our style of life. The emergence of the Internet, the traditional telephony and the mobile telephony networks (through Internet Multimedia Subsystems (IMS)) into one global network provides new and innovative facilities to data, voice and video applications. However and more than in any past time, major security concerns have to be considered. VoIP inherits the security lacks of the Internet layers (Network, OS/software and human vulnerabilities) and extend them as well. Unlike closed PSTN networks, the Internet is open to hackers at low costs and high expertise. VoIP calls issued from (or received by) a LAN are highly exposed to eavesdropping by someone on the same LAN. Updates to a VoIP software are exposed to malicious modifications that can not be detected without good code integrity checks.

In this chapter, we try to accomplish the first task of any security project which is the **risk identification and classification**. Among all possible threats, we focus on VoIP specific ones. Based on their target, specific attacks can be classified into three categories: attacks against supporting services, media and signaling. Our attack tree which is shown in Figure 2.1 is expanded block by block.

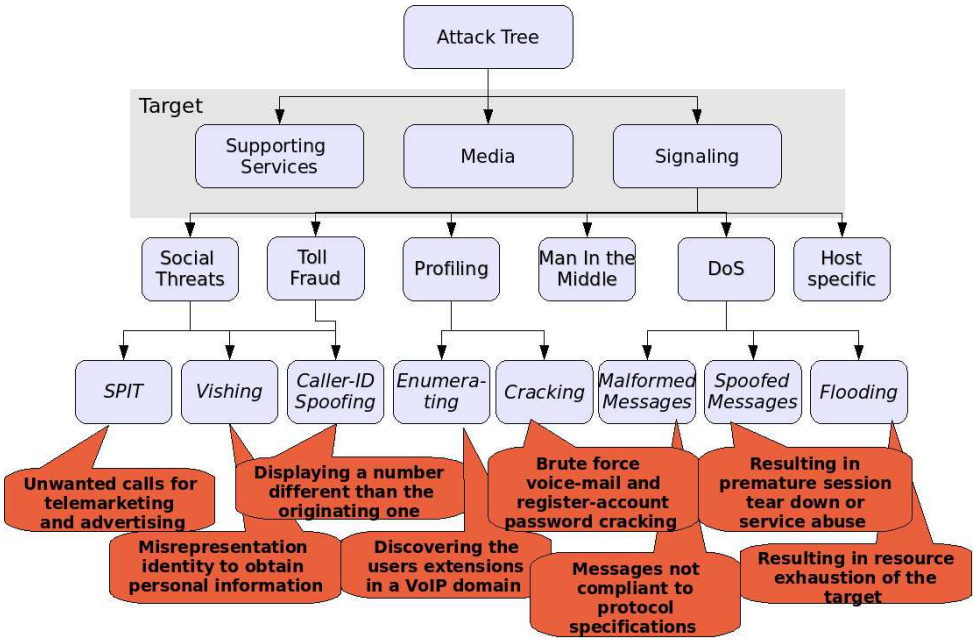


Figure 2.1: Tree of VoIP-specific threats

2.2 VoIP security risks

2.2.1 Supporting services

The deployment and the functioning of a SIP UA is usually supported by the ARP, DNS, DHCP and TFTP protocols (example in Figure 2.2). The ARP manages the data link layer in a LAN. The DNS offers translation services between textual representation of domain names and IP addresses. The TFTP is used to download new firmwares and configuration files. The DHCP allows the dynamic assignment of IP addresses within the domain. These protocols are the target of many attacks like poisoning and spoofing. A successful attack leads an attacker to take the control over the supported UAs and servers. In an ARP poisoning, an insider (intruder who has access to the LAN) tries to exploit a race condition in order to bind between its physical address and the IP address of a server in the cache of a UA. To become a man in the middle, he tries to bind between his physical address and the IP address of the UA in the cache of the server. For example, let's consider a UA that after a reboot connects to a TFTP server to download its configuration. An ARP-cache poisoning leads to substitute a rogue TFTP server to the correct one. As similar, this attack is possible over DHCP. A DNS-cache poisoning consists on injecting falsified records. The attacker tries to bind the IP of a rogue server to the hostname of the good one. MAC and IP spoofing are among the fundamental flaws in the Internet that cause many problems to the VoIP infrastructures.

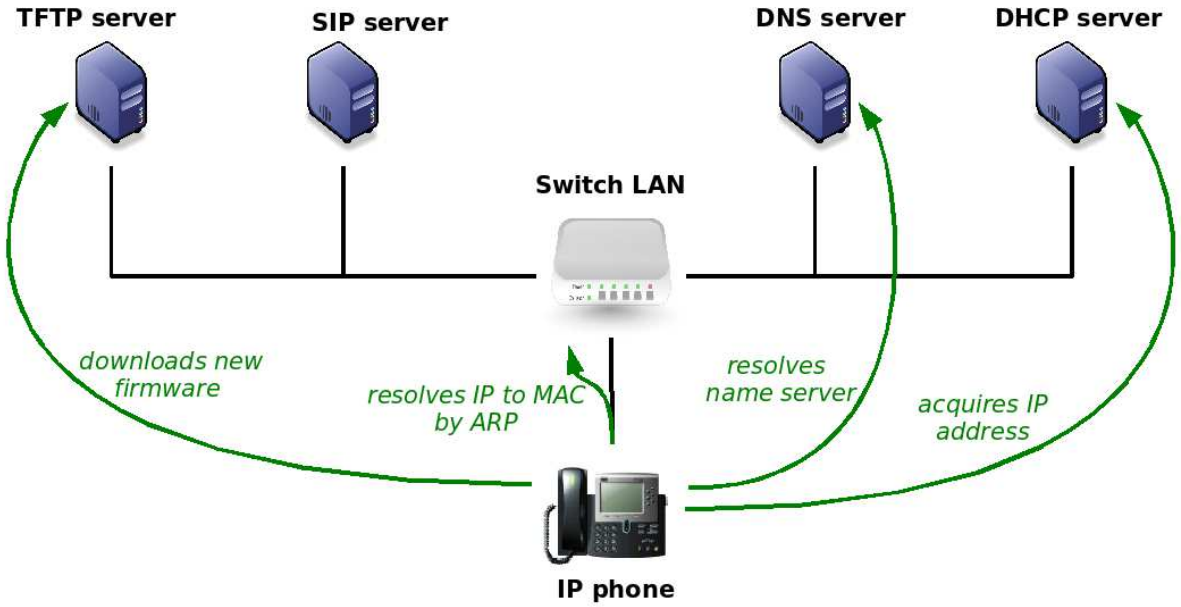


Figure 2.2: Supporting services for a SIP UA

2.2.2 Media protocols

VoIP applications are real-time so that altering their QoS would make their use inconvenient if not impossible. Under a 150 milliseconds delay or a 5% packet loss, a VoIP call becomes incomprehensible even when efficient codecs are used. As such, to cut the transmission between two communicating parties, an attacker only needs to congest the network at some edge in the transport path.

If an attacker is able to predict the sequence numbers of an RTP flow (e.g. by sniffing when the flow is not encrypted), he can inject extraneous media into the flow. A demonstrative tool of an RTP playout attack is found in [1].

SIP signaling is out-of-band. There is no mechanism to monitor and control the media session established by SIP. E.g. any modification in the used codecs and therefore the QoS conditions are not transparent to SIP. Therefore a QoS-based billing approach would be easily abused.

2.2.3 Signaling protocols

The signaling plane is the backbone of any telecommunication architecture including VoIP so that it is the subject of a wide range of security risks.

Profiling

Sniffing and interception of the SIP traffic would reveal many sensitive pieces of information. The interception of an unencrypted INVITE message provides the caller (in the `From` header), the callee (in the `To` header) and the route of the call (in the `Via` headers). The SDP body contains the media description of the call which allows filtering of

the corresponding packets, call reconstruction and eavesdropping. Sometimes, the SDP body contains the encryption key of the media session which makes eavesdropping to an encrypted session also possible.

Cracking consists on a brute force algorithm which is trying through recursive requests to guess the password of a SIP account in a registrar or a proxy server. Cracking attacks are usually preceded by a scanning for SIP devices (open 5060 ports) within a domain (IP range) and enumeration of existing users. The enumeration consists on issuing requests to bind between a VoIP number (SIP URI) and an existent user in the location database. Depending on the case (whether a user exists, exists but not available or doesn't exist) the response of the location server would be different. By analyzing the responses, the existent users would be filtered and used in further steps of attack.

While the above form of enumeration is active, there are passive forms like web browsing (e.g. VoIP numbers of an enterprise are found in its web site) and sniffing (i.e. extract the valid extensions from the SIP messages).

Man In The Middle

Due to the lack of good authentication practices, many situations allows a Man In The Middle (MITM) to "park" between a caller and his callee. SIP implements a strong authentication scheme similar to HTTP. A UAS, proxy, redirect or registrar server might authenticate a UAC by a cryptographic challenge based on a shared secret. Inversely, a UAC can authenticate the server. In most deployments, only authentication in one direction is applied. This attack is possible:

Bob is calling Alice. Bob sends an INVITE message to his outbound proxy. Trudy intercepts the message and sends a crafted redirection response to Bob leading to her IP address. Bob does not require the authentication of his proxy, accepts the response and redirects the call to pass by Trudy. The true proxy can be neutralized by a denial of service or by exploiting a race condition. In the same time, Trudy replaces Bob's location by her IP in the `Contact` header and routes the INVITE towards Alice. All further signaling between Bob and ALice will pass by Trudy. Hijacking the signaling path is a prestep to hijack the media one. Trudy can impersonate Alice regarding Bob and can impersonate Bob regarding ALice.

In another scenario, Trudy exploits the lack of end-to-end authentication. Trudy intercepts the INVITE from Bob to Alice, waits the session to be established, then steps in and sends (to Bob or Alice) a new INVITE with the same `Call-ID`, `To` and `From` headers and an incremented command sequence number (in `CSeq`). By the mean of this crafted message, Trudy can change the media conditions (e.g. replacing the reception port, adding or deleting streams), or the signaling path (by the mean of `Via` and `route` headers). Briefly, the session is hijacked by Trudy. Many such situations resulting in toll fraud and billing attacks are found in [115].

Malformed messages

VoIP software vulnerabilities are the target of remote exploits like buffer overflow and crafted messages. Possible damages are system crash, remote code execution and unau-

thorized access.

Spoofed messages

CANCEL and BYE attacks are a type of denial of service against the SIP call establishment procedure. We give here four examples:

- If whenever someone tries to call Bob, Trudy sends a spoofed CANCEL (as if it was originated from the caller) to Bob, then Bob will be prevented from receiving any call.
- If whenever Bob tries to make a call, Trudy sends a spoofed CANCEL (as if it was originated from Bob) to the destination, then Bob will be prevented from making any call. Moreover,
- Trudy sends a spoofed BYE to terminate a session after a few moments of its starting.
- Trudy impersonates an outbound proxy by sending spoofed error responses like 4xx (client error), 5xx (server error) or 6xx (global error) in order to persuade Bob that there is some error situation preventing the proceeding of his calls.

Flooding Attacks

Flooding attacks target the signaling plane elements (e.g. proxy, gateway, etc.) with the objective to take them down and produce havoc in the VoIP network. Based on the used technique, these attacks can be classified as flooding of legal messages, randomly malformed messages, or malformed messages exploiting target-specific vulnerabilities.

The authors of [27] categorize some of these attacks based on the request URI and perform a comparative study of these ones against popular open source VoIP equipment. We adopt the same categorization, i.e.:

- UDP flooding: Since the vast majority of SIP systems use UDP as transport protocol, a large amount of random UDP packets are sent in an attempt to congest the network bandwidth of the target. Such attacks produce a high packet loss. Legitimate call signaling has thus a reduced probability to reach the target and to be processed.
- INVITE flooding with a valid request URI: The attacker calls one SIP phone[1] registered at the target. The target relays the calls to the phone. If the proxy is stateful it will manage a state machine for every transaction. The phone is quickly overloaded by the high rate of calls and is no more able to terminate the calls. As a result, the server is allocating resources for a long time and it will run out of memory.

[1] We use the term SIP phone to represent a SIP UA with moderate processing and memory capabilities like in the case of a hardphone

- INVITE flooding with a non existent SIP phone: If the attacker doesn't know a valid SIP URI registered on the target, it can send calls to an invalid address. The target responds with an error response like "user not found". When the attack rate is higher than the target capabilities, the resources are exhausted. This type of flooding is less disturbing than the previous one but the target CPU is loaded with useless transactions and legitimate requests may be rejected.

- INVITE flooding with an invalid IP domain address: The attacker calls a user with a false IP address of the destination domain. The target is trapped to connect several times to an unreachable host/network while keeping the state of the current SIP transaction. This attack is efficient against some proxies like OpenSER [27].

- INVITE flooding with an invalid domain name: also known as SIP-DNS attack. The attacker calls a user with an irresolvable destination domain name. The target is trapped to ask the DNS server to resolve the domain name and to wait for the response. The DNS response could take a long time especially under some configuration errors. Some proxies are synchronous (i.e. they bloc the process until a response from the DNS is received) while other are asynchronous (i.e. they save the state of the call and take a new request, one time a response form the DNS is received, they are notified). Synchronous proxies are easily blocked by such an attack while asynchronous proxies will deplete their memory at some moment. For a profound comprehension of this attack, please refer to [114].

- INVITE flooding with an invalid SIP phone in another domain: The attacker calls a SIP phone located in another domain than the target's one. The target relays all requests to the server/proxy of the other domain which replies with an error response. In this way, Multiple targets are hit at the same time and cascading failures may occur.

- INVITE flooding with a valid SIP phone in another domain: The attacker calls a user/phone registered in another domain. The target relays all requests to the server/proxy of the other domain which sends them to the phone. The phone is quickly out of service and transactions in progress fill up the memory of the servers in the forwarding chain.

- INVITE/REGISTER flooding when authentication is enabled: The attacker sends INVITE or REGISTER messages and then stops the handshaking process. The proxy/registrar responds with a challenge and waits that the request is sent again with authentication credentials. This process is costly for the proxy/registrar in term of computing (Generating challenges and nonces) and memory (dialogs/transaction state machines). For these and other CPU-based DoS attacks against SIP servers, a survey can be found at [56].

In figure 2.3, we show the response of an OpenSER[2] (version 1.1.0) SIP proxy to an INVITE flooding with an invalid destination domain name in one of our experiments.

[2]Recently forked into two projects: OpenSIPS (`http://www.opensips.org/`) and Kamailio (`http://www.kamailio.org/`)

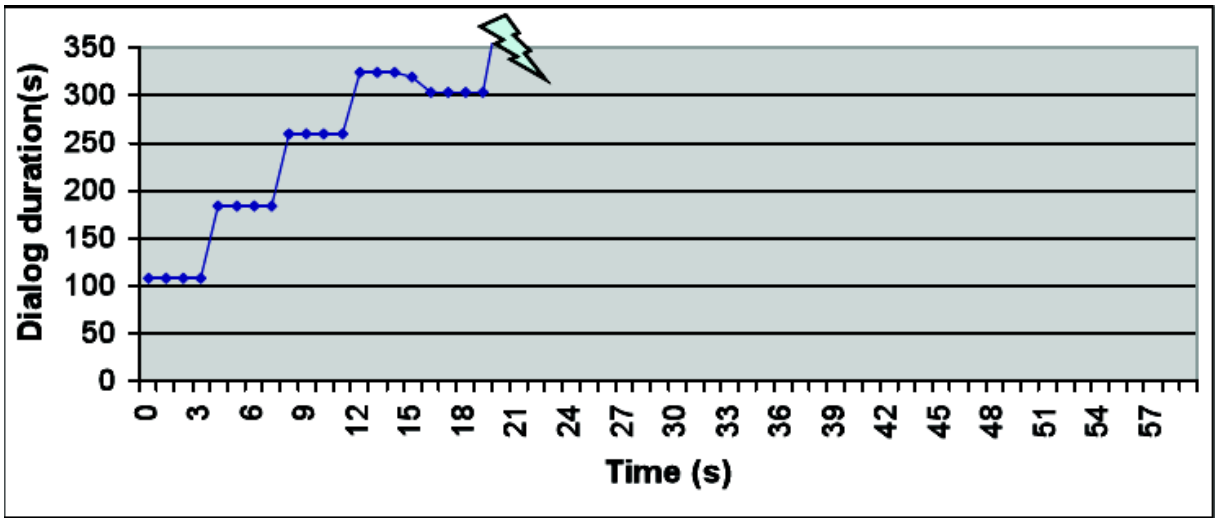


Figure 2.3: OpenSER response to an INVITE flooding with an invalid domain name

In order to resolve the domain name, the OpenSER issues DNS requests with different types (A (IPv4), AAAA (IPv6), SRV (Service locater) and NAPTR (Naming Authority Pointer)) recursively but without receiving a valid response. The behavior of the server can be divided into two successive phases. In the first phase, the first few requests are correctly handled (the call is rejected) but the mean session duration is increasing and the proxy is slowing down. The number of messages by session is increasing because of response retransmissions (no ACK is sent by the attacker). In the second phase, the proxy is no more able to handle the requests (sessions are in a CALLSET state) so the proxy is taken down. The take down time is about 20 seconds for an attack of just 1 INVITE/s rate.

Gateways, Private Branch Exchanges and firewalls

Media Gateway Controllers (MGC) and Private Branch Exchange (PBX) servers are critical points in the VoIP architectures. The MGC are responsible for bridging between different networks such as between a SIP-based network and the ISUP/SS7-based PSTN. The translation between a SIP message and an ISUP message consists of mapping the corresponding parameters. Malicious management of this process is another source of threat which is caused by the lack of authentication and integrity mechanisms in the SS7-based network. A comprehensive threat model of the integrated signaling between VoIP and PSTN is found in [87]. PBX is a term used in PSTN networks and adopted by VoIP to represent a provision of VoIP services (call switching, voice-mail, automated responders, ...). One example of a VoIP PBX is the open source Asterisk[3]. From the SIP point of view, a PBX is equivalent to a B2BUA. In PBX, voice mail and accounting servers, Call Detail Records (CDRs) and users' confidential information must be protected against host-based intrusions like unauthorized access or data destruction.

VoIP requires a new generation of firewalls with dynamic features to open and close

[3] `http://www.asterisk.org/`

media ports with respect to the session negotiated parameters. Attacking the dynamic firewalls or compromising them discloses the protected devices. One way is to exhaust the firewall resources by many requests for port opening and closing. In another direction, malwares could hide their traffic by transferring it through an open media port. If attacked, some VoIP applications can be used as backdoors.

Caller-Id spoofing

In PSTN, Calling Number Delivery (CND) is a telephony service which allows the calling party to provide the CPE (Customer Premises Equipment) of the called party with his directory number during the ringing cycle. Automatic Number Identification (ANI) is a system used by telephone companies and customers to identify the numbers of calling parties. Traditionally, Caller-id/ANI spoofing was a complicated process requiring access to the operator company's PBX or switches. With VoIP, it could be easy to spoof the caller-id (e.g. `From` header in SIP) and deceive the callee with the spoofed number. Caller-id spoofing promotes identity theft and bypassing some verification systems (e.g. gaining access to voice mails).

Social threats

Social threats are attacks ranging from the generation of unsolicited communications which are annoying and disturbing for the users to more dangerous data stealing attacks. The threat is classified as social since the term "unsolicited" is strictly bound to user-specific preferences. This makes this kind of attack difficult to identify.

An example of this is a threat commonly referred to as SPam over Internet Telephony (SPIT). SPIT is similar to SPAM in the email systems but delivered by means of voice calls. SPIT calls can be telemarketing calls used for guiding callees to a service deployed to sell products. They are of more disruption to the victim since they require his momentary reaction to stop the disturbance.

The daily quantity of received publicity in a mail box is far more inferior to the SPAM mass received each day in an electronic mail. This is intuitively because SPAM is almost cost free for the advertisers. Now what happens when phone calls will be of negligible, if any, cost? SPIT leverages on the cheap cost that VoIP has with respect of legacy phone systems. It is currently estimated that generating VoIP calls is three order of magnitude cheaper than generating PSTN calls.

A subtle variant of SPIT is the so-called Vishing (VoIP phishing) attack. In general, phishing is about masquerading as a trustworthy third party in an attempt to acquire confidential information from victims. For example, the phisher uses a link on an email as a lure, deceives the victim by a web page seeming to be its bank home page and asking him to fulfill some fields with sensitive information. With VoIP, Vishing aims either to make the callee dialing expensive numbers in order to get the promised prize or to collect personal data redirecting the users by a DID (Direct Inward Dialing) number towards an Interactive Voice Responder (IVR) pretended to be trusted.

Most of these attacks are going to be generated by machines (bot-nets) programmed to do such a job. Unsolicited communications (like SPIT or Vishing) are, from a signaling

point of view, technically correct transactions. It is not possible to distinguish from the INVITE message if such a transaction is SPIT or not. From a technical point of view the challenge is even more complicated since the content is not available to help in the detection until the phone rings (disturbing the user) and the callee answers the call. For this reason, techniques successfully used against e-mail SPAM like text filtering are hardly reusable in the VoIP sphere. Even if a transaction is identified as unsolicited its processing highly depends on the legal environment in the country of the caller.

2.3 Conclusion

VoIP is exposed to security vulnerabilities that are not fully understood yet. We emphasized the most important threats in this chapter. A complete VoIP security and privacy threat taxonomy has been published so far by VoIPSA (VoIP Security Alliance) in [104]. VoIPSA maintains a list of security tools supporting a wide range of VoIP-specific attacks: sniffing, scanning and enumeration, packet creation and flooding, fuzzing, signaling manipulation, media manipulation and other[4]. To go further, "VoIP hacking exposed" [27] shows, step by step, how online criminals ranging from script kiddies to professional crackers perform VoIP network penetration and vulnerability testing. Therefore, serious precautions must be adopted by users and enterprises aiming to deploy VoIP.

Any security process has a very trivial cycle: protection, detection and response. While our work is mainly focused on the detection, we have to stress on deploying effective protection mechanisms otherwise any detection procedure would be useless. In addition to general policies addicted to data networks security, best protection practices include developing appropriate network architecture by separating voice and data on logically different networks, using strong authentication and access control on the gateways and PBX systems; encrypting as possible signaling and media traffic; dealing with specific issues of NAT, Firewall and QoS; using secure tunneling (SSH, TLS, IPsec) for all remote management and auditing access; preferably using hard-phones; maintaining systems up-to-date by downloading new patches and version updates and checking their code integrity, finally giving special consideration to emergency communication services [52, 105, 77]. Monitoring and intrusion detection systems are needed as well. In the next chapter, we debate the recent approaches on detection and prevention in the context of our problem statement.

[4] `http://www.voipsa.org/Resources/tools.php`

Chapter 3

VoIP Intrusion Detection Systems in The State of The Art

Sommaire

3.1 Introduction

Even if VoIP attacks are not in the headline news in security reviews yet they soon will be of major harmfulness for the Internet telephony services. Enterprises, research and government communities started to investigate the best ways of assuring different security aspects. The first step was to define general recommendations for a safe VoIP deployment [105, 7, 52], then to analyze the security mechanisms applied to VoIP protocols and to highlight their weaknesses [35]. Sengar et. al. [87] present a threat model for integrated signaling between VoIP and PSTN and propose an integral protection solution. Up to overall books on VoIP security have been published (e.g. [77]). Many reasons make this field special and open to research studies. Indeed, Some security components can not be used "as is": application-aware firewalls are required to enable dynamic port allocation regarding the call setup; UAs behind NAT can not communicate without NAT traversal procedures; efficient encryption is challenging because of the derived packet size expansion, processing delay and complex key distribution schemes. Also, VoIP intrusion detection is challenging for many reasons:

- VoIP systems are distributed (UAs, servers, proxies, PBXs). Monitoring network entry points is insufficient because of possible insider malicious activities.
- VoIP systems employ a multitude of protocols for signaling, management, configuration and media delivery;
- VoIP systems are heterogeneous (different implementations, manufacturers, configurations ...) and typically placed under different autonomous administrations;
- VoIP systems just settle in the Internet where hackers have easy access and much experience. They are exposed to a wide set of classical Internet attacks.
- Some detection has to be coupled in real-time with prevention. Offline SPAM detection techniques can not be used in the case of SPIT because we need to block the calls before disturbing the users.
- Some checks must meet strict requirements. Sometimes, only one SIP message is needed to exploit a vulnerability. Low-rate floods using specially crafted messages can escape from loose detection and they are very efficient against some servers.

This chapter is mainly devoted to discuss the intrusion detection approaches proposed in the literature. We are specially interested by defense against SPIT and DoS as well as VoIP traffic monitoring. In the next section, we introduce the concept of intrusion detection. Then recent approaches are discussed in Section 3.3. We place works specific to DoS in Section 3.4 and those specific to SPIT in section 3.5. Section 3.6 concludes the chapter.

3.2 Intrusion detection concept

An Intrusion Detection System (IDS) is a second line of defense behind protection systems like authentication, access control and cryptography. It is defined by Anderson [5] as the process of monitoring computer networks and systems for violations of the enforced security policy. Automating the analysis of large audit trails and network traces either online or offline is the goal of intrusion detection. A first model was founded by Denning [22].

Regarding the data source, IDSs are categorized into host-based and network-based. Host-based IDSs (HIDS) are feed by audit trails from the operating system and log trails from the different applications. Network-based IDSs (NIDS) are feed by network traffic.

Regarding the detection method, IDSs are categorized into misuse detectors (knowledge-based) and anomaly detectors (behavior-based). Misuse detection is based on pattern matching to identify intrusions. These known patterns are referred to as signatures. Signatures are extracted from the known attacks. Anomaly detection builds a model of normal data and observes deviations form this model as intrusive. Misuse detection has the advantage of a negligible false positive rate while anomaly detection has the advantage of detecting previously unknown attacks. New and mixed methods have emerged as well. Specification-based detection pretends having the strengths of the two previous

approaches while maintaining a low false positive rate [98]. In specification detection, we define a set of security behaviors against which a user behavior is compared.

Whatever is the detection method, a strong knowledge in the VoIP domain is needed for the design of a VoIP IDS. Having a good knowledge in the application domain is not specific to VoIP. Krügel et al. [51] highlighted the need for a profound application-specific knowledge of the protected network services and gave practical examples on DNS and HTTP. With the proliferation of attacks against web services, such knowledge has been found necessary to design a multi-model detection approach by the same authors [50].

3.3 Recent approaches

Niccolini et. al. [66] suggest the using of network-based intrusion detection and prevention at the entry point of a SIP network. They proposed two stages of detection in order to improve the accuracy of the overall system: the first one is based on knowledge-based techniques while the second is based on behavior-based techniques. Using the Snort[5] system, the authors implement a SIP preprocessing logic composed from three steps: SIP syntax analysis check, SIP security analysis check (on mandatory fields) and SIP stateful analysis check. The results of performance testing of a first prototype are evaluated using the BRUTE traffic generator [17].

SCIDIVE [106] is a knowledge-based NIDS intended to VoIP. The system is based on a rule matching engine supplied by a rule set to perform two kinds of detection: stateful and cross-protocol. In stateful detection, the state from multiple incoming packets is assembled and the aggregated state is used in the rule matching engine. In cross-protocol detection, the matching rule spans multiple protocols, e.g. a pattern in a SIP packet is followed by another in a RTP packet. SCIDIVE can be installed at multiple points of the VoIP network (clients, servers, proxies). It is shown to detect different classes of intrusions including masquerading, denial of service, media-stream attacks and prematurely tear-down of sessions.

Sengar et. al. [89] propose a specification-based VoIP NIDS based on the interaction between extended finite state machines of different protocols. By tracking deviations from interacting protocol state machines, this solution is able to detect a series of SIP, RTP and cross-protocol attacks. Similarly, Ding et al. [24] propose a timed Hierarchical Colored Petri Net (timed HCPN) model to detect albeit the same set of attacks. A similar specification-based intrusion detection approach for H.323 networks is published in [97].

The previous approaches are tested over a limited set of attacks and don't show high scalability potential. A most promising architecture is "VoIP Defender" [30]: a highly-scalable intrusion detection and prevention system for SIP. This architecture builds a generic framework for detection algorithms without defining them. The authors impose strict requirements that a Point of Protection (PoP) must fulfill to assure the security of a Point of Service Provisioning (PoSP):

- Transparency: The PoP must be completely invisible from the networking point of view. Thus it shouldn't do neither IP routing, nor SIP proxying so that its existence

[5] `http://www.snort.org`

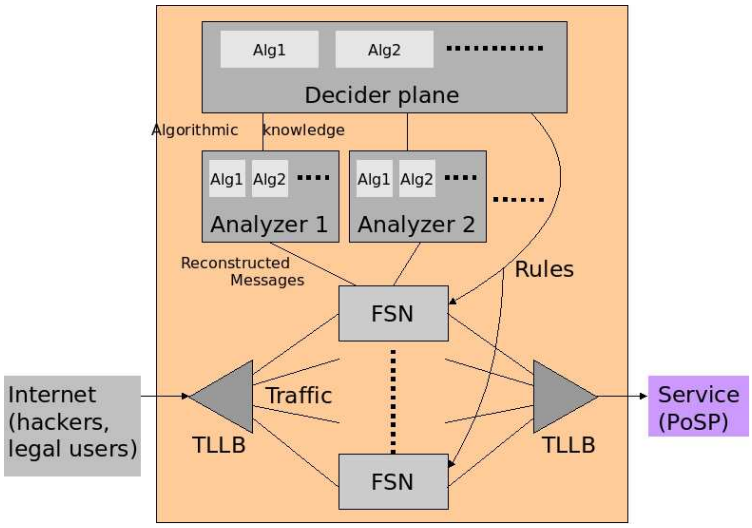


Figure 3.1: VoIP Defender architecture (from [30])

can not be guessed from the outside;

- Line Speed: All traffic must be dealt with in real time;
- Scalability: The PoP must be able to handle flooding attacks which can reach a gigabit per second bandwidth;
- Independence: The PoP must not rely on a specific SIP proxy implementation, as a consequence it must implement a subset of the SIP logic. It must monitor not only the incoming traffic but also that outgoing from the PoSP.
- Extensibility: The architecture must allow new detection algorithms to be integrated easily and safely.

VoIP defender is placed between the service provisioning platform and the customers as shown in figure 3.1 from [30]. It is based on a load balancing scheme composed of:

- a Transport Level Load Balancer (TTLB): performs load balancing between different FSNs in order to not expose them to high network and thus processing load;
- Filter/Scanner Nodes (FSN): have a two-fold mission, the first is to fork the packets, the second is to apply packet filtering rules set up by the decider;
- Analyzers: represent the low level functions of detection algorithms. they must deal directly with the flows (transactions, sessions, ...). Reports from analyzers working in parallel are aggregated at the decider node;
- a Decider: represents the upper layer of detection algorithms. It has to recognize the actual situation and apply suitable filtering rules to the FSN nodes.

Test measurements on the FSN nodes show that application of simple IP checking rules is feasible while rules involving matching of regular expressions influence to some degree the latency and the throughput. We find that this approach is still much abstract regarding VoIP even that we agree on its defined requirements and load balanced architecture.

Traffic profiling based on statistical properties helps a behavior-based detection of many network attacks. To our knowledge, [46] is the only work published so far on VoIP signaling traffic profiling and its applications. The authors propose a general methodology for profiling SIP traffic at several levels: server host, functional entity (registrar, proxy) and individual user. Authors develop an entropy-based anomaly detection algorithm and prove its efficiency in two direct applications: problem diagnosis and security protection. Network traces from a VoIP production environment are used in the experiments. Among the achieved results, the proposed approach detects the presence of SPIT calls even at low rate (10 concurrent calls generated from one source). Such a low rate attack is only detected when monitoring of individual user activities is enabled. We notice that such a monitoring is not scalable over real-world scenarios where multiple sources can participate to the attack. Monitoring of all the sources would end up by attacking the detection system itself. We suggest that more security-oriented SIP traffic profiling should be investigated based on machine learning techniques and we consider such an approach in our contributions. SPIT and DoS are the subject of independent security approaches so we extend their related works in the two following sections.

3.4 Works on DoS

We divide works related to denial of service into two categories: defense against flooding attacks and defense against intelligent malformed messages.

3.4.1 Defense against Flooding

One of the earlier studies in this domain is [78]. The authors present the design and the analysis of a multi-layered protection scheme in the context of VoIP enterprise networks. Both application and transport sensors are designed based on previous works on TCP flooding detection [108]. Different recovery algorithms are considered and evaluated for their performances. The approach seems interesting but unfortunately doesn't show a strong knowledge in the VoIP domain. Only SIP attacks which are TCP-like are studied like broken handshaking. Authors seem ignorant of the layered structure of the SIP protocol: they base their approach directly on SIP dialogs rather than on SIP transactions.

Chen [19] modifies the original state machine of SIP transactions by adding a new state called "Error". Then he applies thresholds on four variables :

- the number of transactions per second,
- the number of packets per transaction,
- the number of transactions per node,
- and the number of SIP application errors per second (SIP error responses).

A similar approach appeared in [26] and is implemented as a module of the aforementioned VoIP defender architecture. The authors define three different scopes of detection: at the transaction level, the sender level and the global level. They apply thresholds on a list of measurements calculated on a modified UAS transaction state machine. The list of measurements contains active transactions, active replies, new transactions, transaction termination, intra-state events and state transitions.

A detection simply based on thresholds is not very effective against stealthy attacks. More adaptive to such attacks is the work of Sengar et. al. [88]. Their VoIP Flood Detection System (vFDS) models a collection of packet streams (e.g. INVITE, BYE, 200 OK) tied together by a common factor (e.g. SIP protocol) as an evolving probability distribution and measures the difference between distributions over two time intervals by the mean of the Hellinger Distance (a measure of variability between two probability distributions). vFDS measures the Hellinger distance between a training period and a testing period and tags important deviations as anomalies. Experiments show efficient detection of TCP SYN, SIP INVITE and RTP floods. The approach is tested to not increase call-setup time regarding SIP and to not introduce jitter regarding RTP. Good performances were shown with 100% accuracy under 500 INVITE/s flooding. However, we notice that this approach would not be expressive in a real world environment. Not every deviation of the Hellinger distance is necessarily a flood. We suggest that more important results of the statistical machine learning theory should be investigated.

Zhang et.al. [114] addresses the SIP DNS attacks against SIP proxies (flooding by SIP requests containing irresolvable domain names) and evaluates possibilities to mitigate their effects. The authors show that over-provisioning is not sufficient to handle such attacks and propose a solution based on a non-blocking DNS caching (when a proxy starts to be overloaded, it only looks up entries in its cache and handles new domain names as irresolvable). They evaluate different caching strategies and show that the Least-Frequently-Used caching strategy gives best results.

3.4.2 Defense against Malformed messages

Two different approaches are proposed to struggle against SIP malformed messages: [36] proposes a misuse-based detection framework while [79] proposes an anomaly-based one. The authors of [36] argue that malformed messages can be effectively described through specific structures (signatures). Based on Perl regular expressions, a SIP message is firstly checked to comply with the SIP standard syntax as described by RFC 3261 [82], then it undergoes additional checks for malicious content. Any message which does not conform to the SIP standard is directly discarded. Additional checks perform specialized inspection over the message content looking for typical exploits like SQl injection. Human efforts are needed to write and update rules which are a laborious task.

Authors of [79] utilize a different approach since they propose a self-learning system and thus no human efforts are needed. Their system is capable of detecting unknown and novel attacks. They identify anomalous contents by embedding SIP messages to a feature space and determining deviation from a model of normality. Tokens or N-grams are used as features. The system supports two detection models: Global and local. Global detection is based on the smallest enclosing hypersphere. Local detection is based on

the K-nearest neighbors. The system adapts well to network changes by automatically retraining itself while beings hardened against targeted manipulations.

3.5 Works on SPIT

Identification and blocking of SPIT calls are challenging because of their "social" nature. Classifying callers into white and black lists embraces unsolved situations like in the case of a new caller/identity. Such a system can be infinitely supplied by "black" identities. Moreover, a "white" identity can be spoofed. Very recently, a framework for consent-based communications (permission request/ permission grant) in SIP is proposed [81]. Also, consent-based communications could be disturbing when users are flooded by permission requests.

Some researchers argue that strong cryptographic assertions are required in order to authenticate callers in an inter-domain context. The IETF SIPPING WorkGroup suggest defining of two new SIP header fields [70]: `Identity`, to convey a certificate validating the identity, and `Identity-Info`, to convey a reference to the authority which signed the certificate. Otherwise, circle of trusts can be drawn among VoIP domains using authenticated TLS connections. These approaches are hardly scalable over the Internet which is highly open and highly dynamic.

A progressive and multi-gray level algorithm (called PMG) is proposed in [90]. Each caller is attributed a gray level based on his call history. The gray level is composed from scores over a short and a long term periods. When the gray level of a caller bypasses a certain threshold, his calls would be blocked. The algorithm has the same weakness that we have mentioned: it can not treat calls carrying new identities. The authors suggest coupling PMG with strong authentication mechanisms. Also, we don't found this very practical. PMG is based on one statistical measure (call frequency) while lessons from SPAM detection suggest using multiple ones.

Quittek et. al. [75] apply hidden Turing tests to received calls and compare them to typical human communication patterns. The idea is that automated callers will not pass the tests. Otherwise they are forced to consume significant resources, contradicting the objective of placing as many calls as possible. The authors implement a prototype of their approach and integrate it as a module in their VoIP SEAL system [84]. VoIP SEAL filters SPIT calls by a two-stage decision process: The first stage analyzes the information available before actually answering the call (e.g. the INVITE message). The second stage interacts with the caller and/or the callee. The second stage represents some inconvenience (e.g. the callee is disturbed, Turing tests add some delay to the call). To mitigate this inconvenience, the first stage has a scoring system indicating if tests from the second stage are required or not. Each stage is composed form several modules. Besides Turing tests, there are white/black list, simultaneous calls, call rate and URI's IP/domain correlation. A modified SIP client allows the end-user to give his feedback to the system.

Authors of [48] argue that SPIT mitigation should be based on a socio-technical defense. The authors describe a multi-stage, adaptive SPIT filter based on the user presence (location, mood, and time), trust and reputation. They describe a closed-loop feedback control between different stages and a formalism for VoIP-specific trust and reputation

analysis. Their formal model is based on the human intuitive behavior for accepting or rejecting the calls based on his direct and indirect relationships with the caller. A similar approach so-called "CallRank" is found in [9]. The call rank is based on call duration, social networks and global reputation. Integration of the CallRank algorithm in SIP requires modifying the INVITE message to carry call credentials and proxy-appended reputation scores. Moreover, CallRank assumes that each user has a public/private key. We exclude that this solution is feasible for the moment.

3.6 Conclusion

In this chapter, many works related to VoIP intrusion detection, DoS and SPIT fighting were presented and criticized. In our opinion, these systems would not get mature before being in direct contact with the attacks. In the same time, it is very important to prepare our security platforms and to be one-step ahead of the attackers. It was clear for us that some important issues have not been sufficiently studied (e.g. only one paper on SIP traffic profiling) and other are neglected. We base our contributions on the missing pieces in the VoIP security puzzle. We work in three different axes:

- We assess SIP traffic monitoring for security purposes. Particularly, we incorporate a machine learning approach for traffic classification and anomaly detection. Two models are investigated: the first is based on Bayesian networks [63]. The second uses support vector machines [65].
- With the absence of early warning systems specific to VoIP, we propose the design of a honeypot and honeynet [64].
- We also propose an event correlation framework in order to assure a holistic security perspective within a VoIP domain [62].

Our contributions are expanded in the next part.

Part II

Contributions

Chapter 4

Monitoring SIP Traffic Using Machine Learning

Sommaire

4.1 Introduction

Traffic monitoring in general is imperative for service providers in terms of security, management and provisioning. Many attacks can be detected by monitoring the signaling traffic. These include for instance: SPIT, Vishing, VoIP numbers enumerating, remote

brute force password cracking and message flooding. A defense approach based on monitoring the incoming/outgoing signaling traffic is very effective since many attacks can be prevented before they reach the domain under protection.

In this chapter, we propose a SIP traffic monitoring system that analyses the ongoing traffic and reveals abnormal and attack situations. In addition, we impose the system to meet the following requirements:

1. an extremely low intervention from the human operator,
2. a real-time performance,
3. an extremely low false alarms rate,
4. a high adaptation to changes in the network environment such as the presence of new terminal nodes or services and to their effects on the traffic,
5. a high sensitivity to detect stealthy attacks that don't lead to a noteworthy deviation from what is normal,
6. a protection from attacks targeting the monitoring process itself.

Because rule-based and fixed/adjustable threshold systems fail to meet such requirements in several similar situations, we base our system on a machine learning approach. Machine learning has been studied over the recent years to be incorporated in many computer security areas ranging from malware behavioral learning and classification to malformed packet inspection. In particular, it has been shown that machine learning techniques help improving intrusion detection systems in many ways: detection of new and previously unknown attacks, reducing false alarm rates and enhancing performance while using a more compact representation [57, 10].

The monitoring scheme can be quite simple and flexible to support different machine learning techniques and algorithms. In a representative list (from [57]), there are instance-based learners, Bayesian networks, kernel density estimation, neural networks, support vector machines, learning decision rules, learning decision trees and mining association rules. Although we must evaluate as many algorithms as possible because it is impossible to determine a priori which algorithm will perform the best, we select Bayesian networks (BN) [69] and Support Vector Machines (SVM) [102, 103] to experiment with. Our selection is reinforced by several elements. Regarding BN:

1. BN are today one of the most complete and the most coherent formalism for knowledge acquisition, representation and exploitation in computer systems.
2. BN are employed more and more in diverse applications ranging from the control of autonomic vehicles to the operational risk modulation, data mining and gene localization [61].
3. BN are successfully applied in IDSs: Valdes et. al. [100] propose a naïve Bayesian classifier to dynamically analyze TCP sessions. Kruegel et. al. [49] criticize Valdez's model and propose a new model making full use of BN to analyze operating system

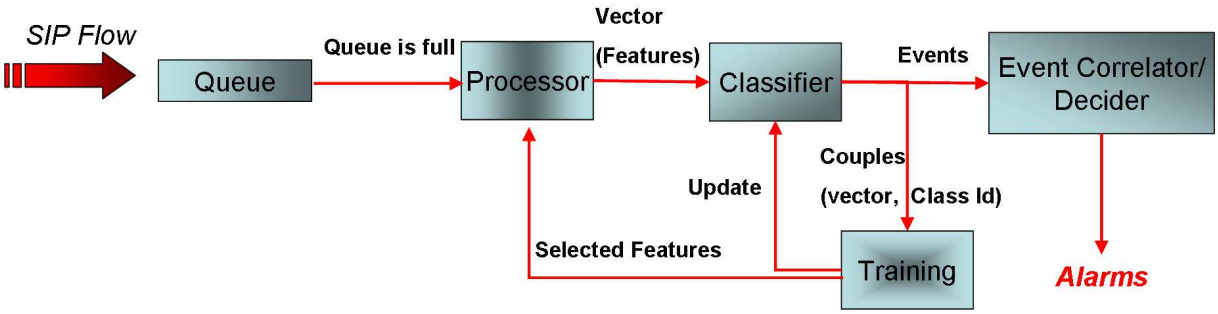


Figure 4.1: SIP traffic monitoring approach

calls and detect attacks against daemon applications. Barbará et. al. [11] integrate pseudo-Bayes estimators into their anomaly detection system.

On the other side:

1. SVM are known for their advanced ability to process high dimensional data. They support classification, regression and exploration of data.

2. SVM demonstrate great performance in many domains like bioinformatics and pattern recognition (e.g. [38, 80]).

3. SVM are successfully applied to network-based IDSs. They demonstrate better performance than neural networks in term of accuracy and processing proficiency [59]. Unsupervised SVM have been used for anomaly detection as well [55]. Very recently, Yu et. al. [113] propose using SVM to detect and classify traffic flooding attacks (TCP-SYN, UDP and ICMP) with SNMP MIB statistical data (rather than raw packets from network links). Their approach shares several points in common with ours.

The remainder of this chapter is composed as follows: In the next section, we describe our monitoring system. In section 4.3, we present our Bayesian model and in Section 4.4 we present our SVM model. Section 4.5 closes the chapter.

4.2 Monitoring Approach

Our approach is depicted in Figure 4.1. We track SIP messages in a queue of predefined size (as proposed by one of the earliest papers in IDS design [44], two types of window can be established: Chronological window e.g. we track messages for a period of 5 seconds, and count-related window e.g. we track a number of 10 messages). Once the queue is full, this slice of messages is used to compute a list of attributes (vector). The classifier decides if the vector belongs to the normal or one of the attack classes. In result, it issues an event towards the event correlator. Finally, the event correlator has to filter and correlate the events to issue alarms and trigger prevention response when necessary. E.g. The event correlation will generate an alarm for a group of events if their number bypasses a certain threshold during a sliding window of time $alarm\_window$. The motivation behind this

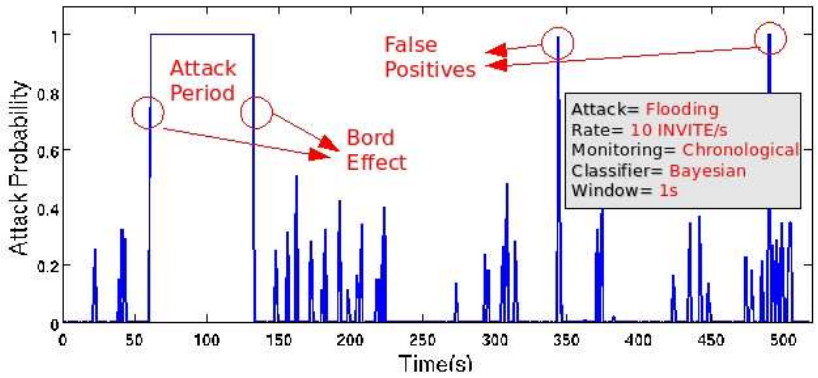


Figure 4.2: Real-time attack detection

approach is to reduce to minimum the number of false alarms. As we can reveal from Figure 4.2, an attack is characterized by a high number of events within a small period of time. In contrast, some events generated because of false positives of the classifier are time-sparse and should not raise an alarm. Stealthy attacks can be characterized by periodic events which can be correlated by applying appropriate rules.

To be suitable for an online deployment, a monitoring system implementing such an approach must guarantee that a slot (messages in the queue) will be processed in real-time. Let t_{pace} the time that the system takes to make a decision about one slot. If we neglect the time needed by the event correlation stage, t_{pace} is composed of two components:

- the analysis time ($t_{analysis}$): the time taken by the processor to evaluate the vector (attributes),
- and the machine time ($t_{machine}$): the time taken by the classification engine.

Given that:

$$t_{pace} = t_{analysis} + t_{machine}$$

our design achieves a real-time pace if: in case of a count-related window, t_{pace} is less than the size of the queue S divided by the mean message arrival rate λ

$$t_{pace} < \frac{S}{\lambda}$$

and in case of a chronological window, t_{pace} is less than the monitoring window D

$$t_{pace} < D$$

After experiments we notice that the machine time is much smaller than the analysis time (due to packet inspection). The number and the complexity of evaluated attributes are crucial from this point of view. For example, using attributes related to SIP dialogs and transactions require the maintenance of some data structures like lists and state machines which is costly in terms of memory and analysis time.

The size of the queue (or the monitoring window) should not be fixed for different kinds of attack detection. Some attacks like flooding require a small monitoring window

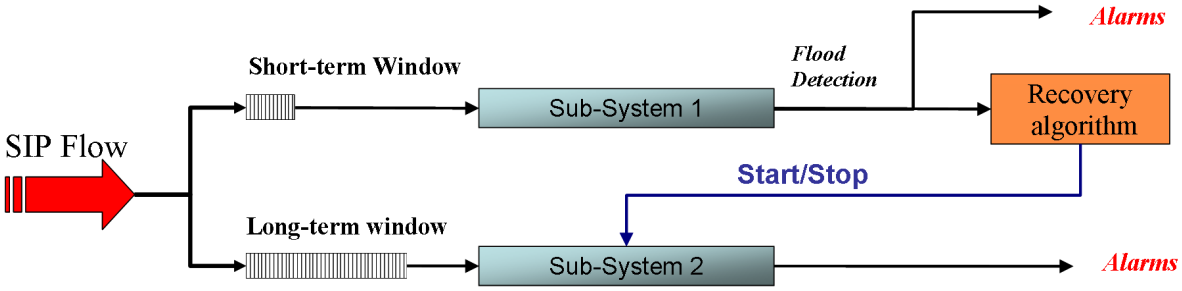


Figure 4.3: Short-term/long-term SIP traffic monitoring system

otherwise the learning machine will be submerged with a large number of messages, while other attacks extend for a certain time and require a large window to be detected. We suggest that our detection system must include two sub-systems (at least), one for short-term monitoring and other for long-term monitoring as depicted in Figure 4.3. The short-term monitor acts as a flooding armor. It must control a prevention response and a recovery algorithm. Possible recovery algorithms are out of our scope for the moment but those referenced in [78] can be adopted.

The classifier engine is based on an offline training phase during which labeled vectors have been used. If no labeled traces are available, unsupervised learning techniques should be applied. Both a classification algorithm and an anomaly detection algorithm can be used in conjunction where new anomalies can be added as new hypothesis to the classification scheme. The training process can be made on the fly during the operational phase by allowing updating the prediction model over time. This option can not be enabled if our system is not immune to adversary profiling and manipulation attacks (where the attackers try to "poison" the training process and therefore the prediction model). Many defense techniques are employed in the literature like randomization (i.e. training with samples randomly drawn from the traffic) and verification (i.e. ensuring that the new prediction model is not be far different from the old one). These issues are not completely investigated in our scope. We focus on studying our two proposed approaches (BN and SVM) for their detection accuracy, performance and attribute/feature selection.

4.3 A Bayesian networks model for SIP traffic monitoring

4.3.1 Introduction to Bayesian Networks

Bayesian networks also known as graphical models combine between the probability theory and the graph theory since their initial objective was to add uncertainty to expert systems. An important aspect of this combination is that probability properties can be seen as graph properties and vice-versa.

A Bayesian network is represented by a directed acyclic graph where nodes represent random variables and arcs reflect a relation of causality between the variables. Formally,

a Bayesian network is defined by:

- a directed acyclic graph G, $G = (V, E)$, where V is the set of nodes and E is the set of arcs,
- a probability space (Ω, Z, p), where Ω is the set of events, $(Z, \cap, \cup)$ is an algebra over Ω, and p is a probability over (Ω, Z).
- a set of random variables associated to the nodes of G and defined over (Ω, Z, p) such as:

$$p(V_1, V_2, \ldots, V_n) = \prod_{i=1}^{n} p(V_i \mid C(V_i))$$

  where $C(V_i)$ is the set of causes (parent nodes) of V_i in the graph G.

The conditional independence between random variables (which is a probabilistic property) is equivalent to a graphical property called `d-separation` (the d stands for directional). In other words, if two nodes X and Y are d-separated by Z then their associated variables X and Y are conditionally independent knowing Z (we use the same letter to represent the node and its associated variable):

$$< X \mid Z \mid Y > \Rightarrow (X \perp Y \mid Z \Leftrightarrow P(X \mid Y, Z) = P(X \mid Z))$$

Without entering in the d-separation definition, it is proven that a node is d-separated by its parents, its children and the parents of its children from the remaining nodes of the graph. This is one of the founding results of the Bayesian theory, because now probability calculations can be simplified by topological properties in the graph. More details are found in [61].

`Inference` is the process to calculate conditional probabilities of events that are tied together by cause-effect relationships. In this context, a `diagnostic` reasoning is a bottom-up inference where we predict causes based on our knowledge of their effects. A `causal` reasoning is a top-down inference where we predict effects from already known causes. The algorithmic complexity of exact inference is strongly dependent of the BN structure. In some situations, it is impossible. So, many approximate algorithms are proposed by the research community.

The essential difficulty using BN consists on learning: structure (graph topology) learning and parameters (conditional probabilities) learning. The goal of structure learning is to learn a directed acyclic graph that best explains the data. This is an NP-hard problem. The goal of parameter learning is to find the Conditional Probability Distribution (CPD) of each node knowing its parents. There are frequentest methods to estimate these parameters from the training data.

In front of these complexities, we choose a BN model known as `Naïve Bayesian Classifier` for its well defined structure and performance. We give more details about this model in the next section.

4.3.2 Naïve Bayesian Classifier

A Naïve Bayes classifier is a tow-layers tree (A tree is a particular graph where each node has only one parent and where there are no loops) where strong independence assumptions are assumed between leaf nodes knowing the root node. The state at the root node is unobserved (so-called belief or hypothesis) and has to be inferred from the states at the leaf ones (so-called observation or evidence). The conditional probability of a Hypothesis (H=h) knowing evidence (e) is formulated by `the Bayes theorem`:

$$P(H = h/e) = \frac{P(e/H = h) \cdot P(H = h)}{P(e)}$$

$P(H = h/e)$ is called the posterior belief. $P(H = h)$ is the prior belief. $P(e/H = h)$ is the likelihood. $P(e)$ is a normalization factor. It represents the summation of the numerator $P(e/H = h) \cdot P(H = h)$ over all possible hypothesizes:

$$P(e) = \sum_{h} P(H = h) \cdot P(e/H = h)$$

Let's apply the Bayes theorem to a "Naïve" structure having several leaf nodes $(X1, X2, \ldots, Xn)$:

$$P(H/X1, \ldots, Xn) = \frac{P(X1, \ldots, Xn/H) \cdot P(H)}{P(e)}$$

Since the leaf nodes are conditionally independent knowing the root node:

$$P(H/X1, \ldots, Xn) = \frac{P(X1/H) \ldots P(Xn/H) \cdot P(H)}{P(e)}.$$

This calculation is modeled by a message (probability vector) passing scheme (known as Pearl message passing which is a general solution for inference in Bayesian trees). Two kinds of messages are defined:

- likelihood (or diagnostic) messages λ travelling upward,
- and prior (or causal) messages π travelling downward.

Each node Xi is linked to the root by a conditional probability distribution CPD_i. For discrete variables (Xi has a finite number of states), the CPD is transformed into a Conditional Probability Table (CPT) where $CPT_i(hx) = P(X_i = x/H = h)$. The λ message propagation is given by the formula:

$$\lambda_i = CPT_i.\lambda_i(\text{child})$$

where λ_i(child) represents the observation.

In the opposite direction, the π message propagation is given by the formula:

$$\pi_i = \pi(\text{parent}).CPT_i$$

where π(parent) represents the hypothesis.

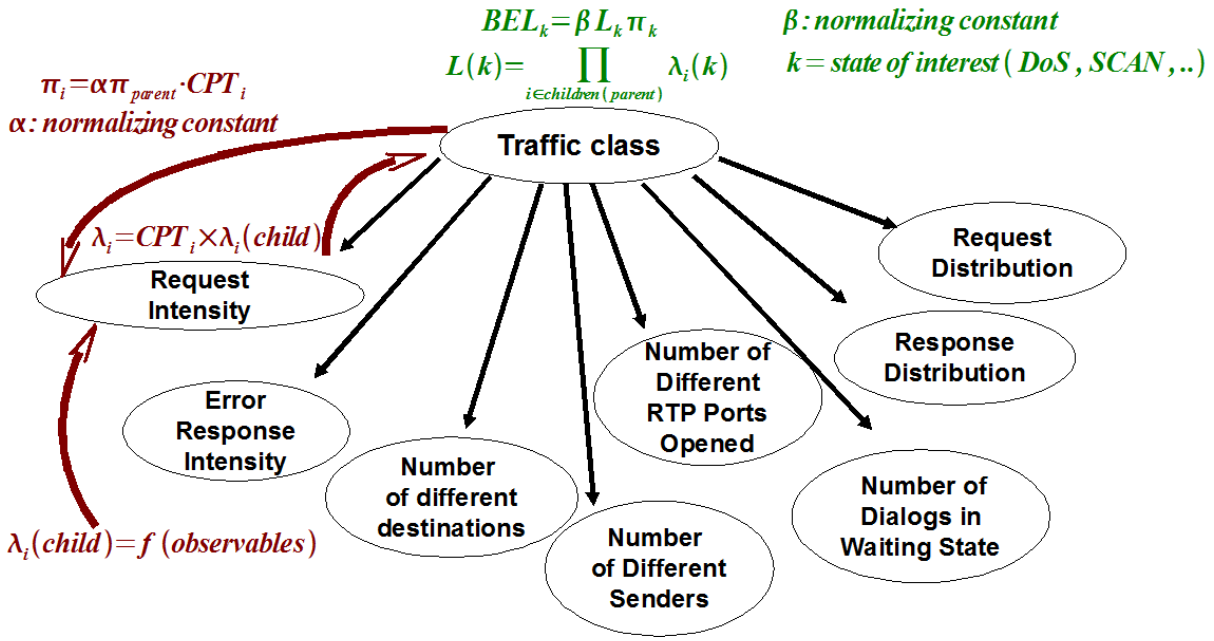


Figure 4.4: Naïve Bayes model for SIP traffic monitoring

To obtain the likelihood at the root node, the likelihood messages from all the leaf nodes are merged together by an elementwise multiplication and the resulting vector is normalized:

$$L(k) = \alpha \cdot \prod_i \lambda_i(k)$$

for $k = 1 \ldots K$ where K is the number of hypotheses, α is a normalizing constant. To obtain the posterior belief at the root node, L is multiplied in elementwise by the prior belief and the resulting vector is normalized:

$$BEL(k) = \beta \cdot L(k) \cdot \pi(k)$$

β is a normalizing constant. A state (k) is considered as `winner` if it has the highest belief value or if its belief value bypasses a certain threshold (e.g. $BEL(k) > 0.6$).

4.3.3 Model structure

Our model is a Naïve Bayesian classifier where the leaf nodes are defined by attributes calculated over a slot of messages and the root node represents the pertained class of traffic as illustrated in Figure 4.4.

The model is based on cause-effect relationships between the attacks and their direct effects on the traffic flow. We consider the following points:

- DoS (flooding): is characterized by high message arrival intensity,
- Scan (SIP URI enumerating): is characterized by a high number of different destinations and a high number of error responses,

- Password cracking (INVITE and REGISTER brute force with credentials): is characterized by a high number of error responses,
- Firewall traversal: is characterized by a high number of requests for opening RTP ports (which can be extracted from the SDP bodies in the SIP requests),
- SPIT: is characterized by a high number of different destinations.

Hence, we define the following hypothesizes (states of interest) at the root node: `Normal`, `DoS`, `Scan`, `Password cracking`, `Firewall traversal` and `SPIT`. Each leaf node represents an attribute or feature. As it is proposed in [100], we define three types of attributes: Intensity measures, High watermarks and distributions. Intensity measures are exponentially decayed counts. We define three intensity measures: `Request Intensity (RI)`, `Error Response Intensity (ERI)` and `Parse Error Intensity (PEI)`:

$$\begin{aligned} RI_{req} &= e^{k\Delta t} \cdot RI_{req-1} + 1.0 \\ ERI_{resp} &= e^{k\Delta t} \cdot ERI_{resp-1} + I(resp\_code) \\ PEI_{err} &= e^{k\Delta t} \cdot PEI_{err-1} + 1.0 \end{aligned}$$

Where Δt is the time between the current event (request, response or parsing error) and the immediately previous one. $k \leq 0$ is a decay constant. Exponentially decayed counts are practical because they don't increase above certain limit if the decay constant is properly chosen. $I(resp\_code) = 1$ if $300 \leq resp\_code$ and $I(resp\_code) = 0$ otherwise, because an error SIP response have a code number between 300 and 699.

A high watermark is a variable that we track its maximum value. We define four variables of this type: `Number of different senders`, `number of different receivers`, `number of open RTP ports`, and `number of dialogs in the waiting state`. For each variable of this type, we initialize a counter. As the slot is processed, the counter is incremented for each new event and the maximum it reaches is tracked. At the start of a new slot, all the counters related to high watermark variables must be reset.

A distribution is a group of categories where we calculate the frequency of appearance of each category during the slot. We define two distribution variables: `request type` and `response family type`.

For each variable that is continuous (intensity measures) or ordinal (high watermarks), we discretize its range into several intervals or bins. The dimension of its λ message is equal to the number of its intervals. Once a value is calculated for one of these variables, the λ message at the corresponding node is composed by setting 1 to the interval where the value falls below and zero to all others. For a distribution variable, the λ message is composed by the frequencies assigned to its different categories. For this reason, distribution variables are often called "soft evidence".

This model must be trained to learn the CPTs between the root and the leaf nodes. In testing mode, each slot of messages triggers a diagnostic inference that calculates the belief in each state of interest using the λ messages presented at the root nodes. The prior belief for each state of interest is proportional to the size of its data set used in the offline training mode.

This model is preliminary in terms of states of interest, chosen attributes and structure. We discuss adding new hypothesis later in this section. When training data are missing for different classes, the essential task of the system remains to distinguish what is normal from what is attack. Selection of the most appropriate attributes from a large set of them can be performed using an attribute/feature selection algorithm.

The structure can be improved by performing χ square tests over a set of labeled data in order to ensure the conditional independence property between the leaf nodes. When this property is not satisfied, arcs between leaf nodes must be drawn. A method found at [61] consists on building a tree over the leaf nodes such that the summation of mutual information carried by the arcs of the tree is maximized. The mutual information carried by an arc linking two nodes is the mutual information between their corresponding variables Xi and Xj knowing the root variable H as formulated below:

$$I(Xi, Xj \mid H) = \sum_{xi,xj,h} p(xi, xj).log\frac{p(xi, xj \mid h)}{p(xi \mid h).p(xj \mid h)}$$

However, such an approach is not supported by our model. In the next section, we present an example of the attack detection process.

4.3.4 Example of a diagnostic inference for attack detection

In this example, we show the attack detection process by going through a diagnostic inference starting from a slot of a network trace. But before going into the inference process, let's first set up the parameters of our model (the CPTs). These parameters are normally evaluated during a training phase, but in this example we set them manually according to protocol semantics. It is often desirable to not have parameters set exactly to 0, because this would neglect the influence of other parameters during the inference. This remark is not respected in this example, but in general small corrections are required (e.g. set a parameter to 0.001 instead of 0). For simplicity sake, we divide the range of each variable (except distributions) into only two intervals. The parsing error intensity (`PEI`) is excluded assuming no parsing (or decryption) errors have been noted.

Let's set the decay rate of the `Request Intensity` to -0.35 in order to ensure a half-life time of 2 seconds. To set up the CPT of the `Request Intensity`, we have to answer questions of the kind: If the traffic represents a DoS, what is the probability to have $RI > 10$?

Request intensity	0-10	> 10
NORMAL	1	0
SCAN	1	0
SPIT	1	0
DOS	0	1
PASSWORD CRACKING	1	0
FIREWALL TRAVERSAL	1	0

Let's set the decay rate of the `Error Response Intensity` to -0.15 in order to ensure a half-life time (count-related) of 5 events (5 non error responses). The CPT of this variable is shown next:

Error response intensity	0-4	> 4
NORMAL	1	0
SCAN	0.2	0.8
SPIT	0.2	0.8
DOS	0	1
PASSWORD CRACKING	0	1
FIREWALL TRAVERSAL	1	0

The CPT of the `Number of Destinations` variable is shown next:

Number of destinations	0-7	> 7
NORMAL	1	0
SCAN	0	1
SPIT	0	1
DOS	0.8	0.2
PASSWORD CRACKING	1	0
FIREWALL TRAVERSAL	0.8	0.2

The CPT matrix of the `Number of Opened RTP ports` is shown next:

Number of opened RTP ports	0-10	> 10
NORMAL	1	0
SCAN	1	0
SPIT	0.8	0.2
DOS	0.8	0.2
PASSWORD CRACKING	1	0
FIREWALL TRAVERSAL	0	1

The CPT matrix of the `max number of dialogs in the waiting state` is shown next:

Max Nb. of dialogs in the waiting state	0-10	> 10
NORMAL	1	0
SCAN	0.8	0.2
SPIT	1	0
DOS	0.1	0.9
PASSWORD CRACKING	0.8	0.2
FIREWALL TRAVERSAL	0.8	0.2

Also for simplicity, we restrict the `request type` distribution to the most used SIP methods: INVITE(I), REGISTER(R), ACK(A), CANCEL(C) and BYE(B). The corresponding CPTs are shown next:

Request	I	R	A	C	B
NORMAL	0.30	0.10	0.30	0.10	0.10
SCAN	0.40	0.05	0.40	0.10	0.05
SPIT	0.40	0.00	0.40	0.00	0.20
DOS	0.90	0.10	0.00	0.00	0.00
PA. CR.	0.10	0.40	0.40	0.00	0.00
FI. TR.	0.40	0.00	0.40	0.00	0.20

The CPT of the `response family type` distribution is shown next:

Response family	1xx	2xx	3xx	4xx	5xx	6xx
NORMAL	0.30	0.30	0.05	0.05	0.05	0.05
SCAN	0.10	0.05	0.05	0.70	0.10	0.00
SPIT	0.30	0.20	0.05	0.20	0.20	0.05
DOS	0.20	0.10	0.20	0.20	0.20	0.10
PA. CR.	0.20	0.00	0.10	0.60	0.05	0.05
FI. TR.	0.30	0.20	0.05	0.20	0.20	0.05

The following slot is an example of an INVITE `Scan` (SIP URI enumerating) composed by 9 calls/dialogs destined to 9 different extensions. The attacker targets a SIP server in order to discover the different SIP URIs registered at the server. The requests are launched periodically every 5 seconds in order to escape from the detection. The attacker requests and the corresponding server responses are collected for each dialog along with our interpretation.

Dialog 1: `INVITE` → `404 Not Found` → `ACK`. *the extension is not found.*
Dialog 2: `INVITE` → `484 Address Incomplete` → `ACK`. *The extension is the start of a valid URI.*
Dialog 3: `INVITE` → `100 Trying` → `503 Service Unavailable` → `ACK`. *This could be due to a server overload or to some load balancing configuration.*
Dialog 4: `INVITE` → `100 Trying` → `180 Ringing` → `CANCEL` → `200 OK(CANCEL)` → `487 Request Terminated` → `ACK`. *Good number, the attacker hangs up before the call is answered.*
Dialog 5: `INVITE` → `404 Not Found` → `ACK`. *the extension is not found.*
Dialog 6: `INVITE` → `484 Address Incomplete` → `ACK`. *The extension is the start of a valid URI.*
Dialog 7: `INVITE` → `100 Trying` → `503 Service Unavailable` → `ACK`. *The number could be right but his owner is not registered at the moment.*
Dialog 8: `INVITE` → `100 Trying` → `180 Ringing` → `200 OK` → `ACK` → `BYE` → `200 OK`. *Good number, the call is answered but the attacker hangs up.*
Dialog 9: `INVITE` → `404 Not Found` → `ACK`. *the extension is not found.*

Let's evaluate the λ messages at each child node. The calculus of the `request intensity` gives a value of $2.424 < 10$ so we set $\lambda = (1\ 0)$ at this node. The message passed to the root is $\lambda_{to\_parent} = (1\ 1\ 1\ 0\ 1\ 1)$. The calculus of the `error response intensity` gives a value of $2.70 < 4$ so we set $\lambda = (1\ 0)$ at this node. The message passed to the root is $\lambda_{to\_parent} = (1\ 0.2\ 0.2\ 0\ 0\ 1)$. The `number of destinations` is $9 > 7$ so $\lambda = (0\ 1)$ and $\lambda_{to\_parent} = (0\ 1\ 1\ 0.2\ 0\ 0.2)$. The `number of RTP ports opened` is $9 < 10$ so $\lambda = (1\ 0)$ and $\lambda_{to\_parent} = (1\ 1\ 0.8\ 0.8\ 1\ 0)$. The `number of dialogs in the waiting state` is 1 at any time so $\lambda = (1\ 0)$ and $\lambda_{to\_parent} = (1\ 0.8\ 1\ 0.1\ 0.8\ 0.8)$. The `Request type` distribution gives the following table:

Method	I	R	A	C	B
λ	9	0	9	1	1

So $\lambda_{to\_parent} = (5.6\ 7.35\ 7.4\ 8.1\ 4.5\ 7.4)$. The `response family type` distribution gives the following table:

Response family	1xx	2xx	3xx	4xx	5xx	6xx
λ	7	2	0	6	2	0

So $\lambda_{to\_parent} = (3.1\ 5.2\ 4.1\ 3.2\ 5.1\ 4.1)$.

Now we can fuse all the λ_{to_parent} from children by elementwise multiplication: $L_i = \prod_c \lambda\_to\_parent_i(c)$, $1 \leq i \leq 6$. We obtain $L = (0\ 6.11\ 4.85\ 0\ 0\ 0)$. If a priori the different states of interest are equiprobables, the posterior belief is obtained by normalizing L to

the unit sum: $BEL = (0\ 0.56\ 0.44\ 0\ 0\ 0)$. In conclusion, the trace in question is detected as either a `SCAN` or a `SPIT`. This result is totally justified because our scan uses INVITE messages similarly to what a SPIT does. The difference between the two attacks is that a scan may result in a higher `error response intensity` (the attacker doesn't now the valid extensions a priori) while a SPIT may result in a longer `mean dialog duration` (the attacker doesn't hang up immediately because he wants to deliver an advertising). This difference will be caught if we refine the scale of the `error response intensity` by discretizing it to a greater number of bins and if we introduce the `mean dialog duration` as an observed variable represented by a leaf node.

4.3.5 Using of Gaussian Nodes

As noticed in the previous example, categorical nodes are not suitable to represent continuous (the variable can take any real value within a certain range) or ordinal (the variable can take any integer value within a certain range) variables. In fact, two slightly different values may fall under completely two different categories misleading the inference process. In other words, categorical nodes make a strong approximation on the real probability distribution of a variable knowing a certain hypothesis. In addition no order relation exists between the different categories. Instead, we propose using of Gaussian nodes. The CPD linking a Gaussian node X_i to each possible hypothesis $H = h$ at the root node is modeled by a Normal distribution $P(X_i \mid H = h) \sim N(\mu_i^h, \sigma_i^h)$ where μ_i^h and σ_i^h are the mean value and the standard value of the variable X_i respectively under the hypothesis h. In general, the Normal distribution is given by:

$$N(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Using of Gaussian nodes increases the accuracy of our model in the spite of adding more complex computations.

4.3.6 Model adaptation

Any change in the network environment (e.g. when new services and users are added or suppressed) has an impact on the traffic and thus must be noticed by the monitoring system. We enable our Bayesian model to adapt its parameters (CPDs) progressively according to the ongoing traffic. A model for CPT adjustment is already proposed in [100]. Hence, we propose a procedure for CPD adjustment when Gaussian nodes are used. This procedure is launched periodically after each diagnostic inference. In fact, we initialize an effective count for each hypothesis: $\forall h, \text{counts}(h) = 1$. After an inference, the new counts of the different hypothesizes are set as follows:

- $\text{counts}_{new}(h) = \gamma\text{counts}(h) + 1$ if h is "winner" (or "dominant"),
- and $\text{counts}_{new}(h) = \gamma\text{counts}(h) + (1 - \gamma)$ for all other hypothesizes. $0 < \gamma < 1$ is a decay factor.

Then, σ and μ of each CPD linking between a leaf node and the "winner" hypothesis are adjusted as follows:

$$\begin{aligned} \mu_{new} &= \frac{\gamma \text{counts}(h) * \mu + x}{\gamma \text{counts}(h) + 1} \\ \sigma^2_{new} &= \frac{\gamma \text{counts}(h) * \sigma^2 + (x - \mu)^2}{\gamma \text{counts}(h) + 1} \end{aligned}$$

where x represents the last observed value for the node in question.

The proposed scheme is characterized by an effective count that never decreases below 1, and never increases beyond $1/(1-\gamma)$ when one hypothesis is always "winner". Only the "winner" hypothesis has the ability of adjusting its parameters. In addition, the model has certain immunity against some targeted manipulations. In fact, a slightly different version of an attack will result in the adaptation of the attack detection parameters, but it will be never considered as normal.

4.3.7 Adding New Hypothesizes

We give to our model the ability to add new hypothesizes to represent new attacks or anomalies. In fact, we initialize a state called `dummy` by making all the possible evidences conditionally equiprobables. In case of a categorical node, this is traduced by adding a new row of equal probabilities to each CPT. In case of a Gaussian node, this is traduced by adding a new Normal distribution to each leaf node such that:

- σ is infinite (by choosing a relatively large number)
- μ is the same as μ for the `normal` hypothesis.

Then, these new parameters are progressively adjusted as described in our adaptation model. The first few "winner" cycles of the new hypothesis dramatically change these parameters since the effective count is still small. When these parameters are stabilized, a new Dummy hypothesis is added again. This scheme is particularly important when only `normal` data are available for training. In this case, the model must be bootstrapped with only two states of interest: the `normal` state and a `dummy` state representing the attacks.

4.3.8 State Transition

As we have mentioned before, we initialize the prior belief of each hypothesis at the root node proportionally to the size of its data set used in the offline training phase. Moreover, we propose a discrete time and state Markov chain to model the state transition from the posterior belief into a slot to the prior belief into the next slot (similarly to what is proposed in [100]). This Markov chain is based on two facts:

- The "winner" state persists over some period of time. In other words, if a state i "wins" a slot j, it is probable that i will "win" the slot $j + 1$;

- The tendency of all states to decay to some initial status. E.g.: at most of the time, we assume the traffic will be in the `normal` state.

In the next section, we give a short introduction about SVM then we illustrate our SVM-based model.

4.4 An SVM model for SIP traffic monitoring

4.4.1 Introduction to SVM

Given a set of couples $S = (\overrightarrow{x_l}, y_l)_{1 \leq l \leq p}$, with $y_l \in \{-1, +1\}$, which denotes the correct classification of the training data, the SVM method tries to distinguish between the two classes by mean of a dividing hyperplane which has as equation $\overrightarrow{w}.\overrightarrow{x} + b = 0$. If the training data are linearly separable, the solution consists in maximizing the margin between the two hyperplanes, $\overrightarrow{w}.\overrightarrow{x} + b = +1$ and $\overrightarrow{w}.\overrightarrow{x} + b = -1$, such that for all points either $\overrightarrow{w}.\overrightarrow{x} + b \geq +1$ (1) or $\overrightarrow{w}.\overrightarrow{x} + b \leq -1$ (2). This is equivalent to minimizing the module $|\overrightarrow{w}|$ because the distance between the two mentioned hyperplanes is $2/|\overrightarrow{w}|$. The resulting Quadratic Problem (QP) where the conditions (1) and (2) are aggregated is formulated as:

$$\boxed{\begin{array}{c}\text{Find } \overrightarrow{w} \text{ and } b \text{ to minimize } \frac{1}{2}\overrightarrow{w}.\overrightarrow{w} \\ \text{so that } y_l(\overrightarrow{w}.\overrightarrow{x_l} + b) \geq 1 \forall (\overrightarrow{x_l}, y_l) \in S\end{array}}$$

The support vectors are the data points that lie on the margin. The problem of non linear separation has a similar formulation except that we replace each dot product by a kernel function. The kernel function maps the data set to a transformed space (can be high dimensional) where we search for the optimal linear classifier. The maximum-margin hyperplane in the new space is transformed back to the original space. The well known kernel functions are :

- linear $K_l(\overrightarrow{x}, \overrightarrow{z}) = \overrightarrow{x}.\overrightarrow{z}$
- polynomial $K_d(\overrightarrow{x}, \overrightarrow{z}) = (\gamma \overrightarrow{x}.\overrightarrow{z} + r)^d$, $\gamma > 0$
- radial basis function $k_{rbf}(\overrightarrow{x}, \overrightarrow{z}) = exp(-\gamma|\overrightarrow{x} - \overrightarrow{z}|^2)$ where $\gamma > 0$
- sigmoid $k_s(\overrightarrow{x}, \overrightarrow{z}) = tanh(\gamma \overrightarrow{x}.\overrightarrow{z} + r)$, $\gamma > 0$ and $r < 0$

The C-SVC (C parameter - Support Vector Classification) approach is particularly interesting when the training data is not linearly separable.

C-SVC

For the general case where the data S is not separable, the solution allows some points to be mislabeled by the separating hyperplane. This method, so called soft margin, is expressed by the introduction of slack variables $\xi_l (1 \leq l \leq p)$ where an ξ_l measures the degree of misclassification of the point $\overrightarrow{x_l}$. The target function is then increased by a function which penalizes non-zero ξ_l, and the optimization becomes a trade off between a large margin, and a small error penalty.

$$\text{Find } \overrightarrow{w}, \text{ b and } \xi \text{ to minimize } \frac{1}{2}\overrightarrow{w}.\overrightarrow{w} + C\sum_l \xi_l \text{ such that } \begin{cases} y_l(\overrightarrow{w}.\overrightarrow{x_l} + b) \geq 1 - \xi_l, \forall(\overrightarrow{x_l}, y_l) \in S \\ \xi_l \geq 0, \forall l \end{cases}$$

υ-SVC

While the C parameter in the C-SVC is independent of the number of data points, with υ-SVC we can control the number of support vectors (i.e. points that constraint the hyperplan) relatively to the number of examples. The parameter $\upsilon \in]0, 1]$ is an upper bound on the fraction of training errors and a lower bound of the fraction of support vectors. Here, instead of the margin band [-1, +1], we are looking for a band $[-\rho, \rho]$ to obtain the ratio of desirable support vectors.

$$\text{Find } \overrightarrow{w}, \text{ b, } \xi \text{ and } \rho \text{ to minimize } \frac{1}{2}\overrightarrow{w}.\overrightarrow{w} - \nu\rho + \frac{1}{l}\sum_l \xi_l \text{ such that } \begin{cases} y_l(\overrightarrow{w}.\overrightarrow{x_l} + b) \geq \rho - \xi_l, \forall(\overrightarrow{x_l}, y_l) \in S \\ \xi_l \geq 0, \forall l \end{cases}$$

Each of these problems can be transformed to a dual form if we multiply the constraints by Lagrange multipliers and add them to the target function. There are several implementations of SVM based on specialized algorithms for quickly solving such quadratic problems. One of the founding algorithms is Platt's Sequential Minimum Optimization (SMO) [71]. In our experiments (chapter 8), we make use of the well known tool LIBSVM [18] which implements a SMO-type algorithm proposed in [29] and supports a multi-classification scheme.

4.4.2 Model structure

SVM doesn't allow representing knowledge and causal relationships as BN does. Thus, we use another strategy to build our SVM model. In fact, we define a large set of features characterizing the SIP traffic, then we employ a feature selection procedure to extract the most representative (discriminative) ones. The selected features are highly dependent on the network environment, and the nature of attacks to be detected. Thus we define a general approach that can be instantiated depending on the case. We define in the following the important attributes/features that we extract from a slot of SIP messages and motivate why we collect them. We divide these features in five groups:

- **General statistics :** are number of requests, number of responses, number of requests carrying an SDP (Session Description Protocol) body, average inter requests arrival time, average inter response arrival time and average inter request arrival time for requests having SDP bodies; these statistics represent the general shape of the traffic and indicate the degree of congestion. The fraction of requests carrying SDP bodies (normally INVITE, ACK or UPDATE) represent the call signaling traffic which is a good indicator. An excessive use of re-INVITE or UPDATE for media negotiation (which can indicate a QoS theft) increases the number of exchanged SDP bodies and decrements the average inter-arrival of them. Flooding attacks are associated with peaks of one or all these statistics.

- **Call-Id based Statistics:** are number of Call-Ids, average of the duration between the first and the last message having the same Call-Id, the average number of messages having the same Call-Id, the number of different senders (the URI in the From header of a message carrying a new Call-Id) and the number of different receivers (the URI in the To header of a message carrying a new Call-Id). Similar to the Erlang model used in the telecommunication networks, where the arrival rate of calls and the average duration of a call characterize the underling traffic, the arrival rate of Call-Ids (can be starting a call or any kind of SIP dialogs) and the interval time of messages having the same Call-Ids, can be used to characterize the overlay SIP traffic. Nevertheless, we notice that non-INVITE dialogs have shorter durations and fewer number of messages than INVITE dialogs. We consider taking their Call-Id statistics as different features in future works.

- **Distribution of final state of dialogs/Call-Ids:** Since we are using a limited number of messages by unit of traffic to be analyzed, dialogs can be partitioned into two or several units. The final state of a dialog at the analysis moment is considered and this one is not necessarily the final state when all the messages of the dialog can be taken into account. The following states are defined: NOTACALL: for all non-INVITE dialogs, CALLSET: for all calls/INVITE dialogs those do not complete the initiation, CANCELED: when the call is cancelled before it is established, REJECTED: for all redirected or erroneous sessions, INCALL: when the call is established but not realized yet, COMPLETED: for a successful and ended call and RESIDUE: when the dialog does not start with a request. This latter is a residual of messages from a previous slice. In a normal situation where the size of the unit is large enough, NOTACALL, COMPLETED and REJECTED (for "busy" or "not found" situations) dominate this distribution. Major deviations may indicate an erroneous situation.

- **Distribution of SIP requests:** are INVITE, REGISTER, BYE, ACK, OPTIONS, CANCEL, UPDATE, REFER, SUBSCRIBE, NOTIFY, MESSAGE, INFO, PRACK. Although the first five types represent the main methods used in SIP, every other type may point out a specified application running above. The number of REGISTER sent by a user within a time interval is inversely proportional to the period of registration (`expires` parameter or `Expires` header). Obviously, the total number of REGISTER messages is proportional to the number of users of the domain and inversely proportional to the average period of registration among all users. We consider taking registration parameters as a group of features in future works. The existence of SUBSCRIBE and NOTIFY messages indicates SIP presence services. Instant messaging can also be revealed by MESSAGE requests. REFER requests may reveal a SIP peer to peer application or some call transfer running above. INFO requests are normally used to carry out of band DTMF tones within PSTN-VoIP calls. Finally, PRACK requests may reveal VoIP to PSTN activity.

- **Distribution of SIP responses:** are Informational, Success, Redirection, Client Error, Server Error, Global Error. An unexpected high rate of error responses is a good indication for error situations.

The set of features is enumerated in Table 4.1.

4.4.3 Feature selection

A larger number of features does not imply greater accuracy in all cases. From another side, it may affect the performance of the system as it increases both analysis time and machine (classification) time. Feature selection consists in choosing the most representative and the most discriminative set of features. It aims to optimize the performance of the system while improving its accuracy. Many strategies can be deployed using different filters and classifiers. These strategies are quantitatively compared based on several criterion measures. One of the most known measures is the Balanced Error Rate (BER) which is defined as:

$$BER = \frac{1}{2}(\frac{\#\text{false negatives}}{\#\text{positive instances}} + \frac{\#\text{false positives}}{\#\text{negative instances}})$$

We adopt a feature selection procedure based on obtaining the best support vector machines classification accuracy over a labeled data-set by filtering the most discriminative features using their F-score [111]:

In a context where a feature i cuts a data set $x_k, k = 1, \ldots K$ into two sets x_k^+ and x_k^- of respective cardinals n_+ and n_-, The F-score formula is given by:

$$F(i) = \frac{(\overline{x}_i^{(+)} - \overline{x}_i)^2 + (\overline{x}_i^{(-)} - \overline{x}_i)^2}{\frac{1}{n_+ - 1}\sum_{k=1}^{n_+}(x_{(k,i)}^{(+)} - \overline{x}_i^{(+)})^2 + \frac{1}{n_- - 1}\sum_{k=1}^{n_-}(x_{k,i}^{(-)} - \overline{x}_i^{(-)})^2}$$

Where : $\overline{x}_i, \overline{x}_i^{(+)}$ and $\overline{x}_i^{(-)}$ respectively represent the average value of the whole set, the one of the positive set and the one of the negative set. $x_{(k,i)}^{(+)}$ and $x_{(k,i)}^{(-)}$ respectively represent the kth value of the positive set and the kth value of the negative set. The formula is built such that the numerator increases when the positive set and the negative set are farther from one another, and the denominator $(\sigma_+^2 + \sigma_-^2)$ decreases when the vectors of each set are more clustered together. In this case, the F-score formula gives a high value. We use the following procedure[6]:

1. we calculate the F-score for each feature and we put them in a decreasing order (we eliminate those having zero values);

2. we calculate the SVM accuracy over the data set using the remaining list of features (by dividing randomly the set into two subsets, using the first subset for training and using the second subset for testing, repeating this process many times and averaging the results);

3. we eliminate half of the features by excluding those having the lowest F-scores;

4. we repeat step 2 and 3 many times;

5. we choose the feature list giving the best SVM accuracy.

[6] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/fselect/fselect.py`

Table 4.1: List of features

Number	Name	Description
Group 1 - General Statistics		
1	Duration	Total time of the slice
2	NbReq	# of requests / Total # of messages
3	NbResp	# of responses / Total # of messages
4	NbSdp	# of messages carrying SDP / Total # of messages
5	AvInterReq	Average inter arrival of requests
6	AvInterResp	Average inter arrival of responses
7	AvInterSdp	Average inter arrival of messages carrying SDP bodies
Group 2 - Call-ID Based Statistics		
8	NbSess	# of different Call-IDs
9	AvDuration	Average duration of a Call-ID
10	NbSenders	# of different senders / Total # of Call-IDs
11	NbReceivers	# of different receivers / Total # of Call-IDs
12	AvMsg	Average # of messages per Call-ID
Group 3 - Dialogs Final State Distribution		
13	NbNOTACALL	# of NOTACALL/ Total # of Call-ID
14	NbCALLSET	# of CALLSET/ Total # of Call-ID
15	NbCANCELED	# of CANCELED/ Total # of Call-ID
16	NbREJECTED	# of REJECTED/ Total # of Call-ID
17	NbINCALL	# of INCALL/ Total # of Call-ID
18	NbCOMPLETED	# of COMPLETED/ Total # of Call-ID
19	NbRESIDUE	# of RESIDUE/ Total # of Call-ID
Group 4 - Requests Distribution		
20	NbInv	# of INVITE / Total # of requests
21	NbReg	# of REGISTER/ Total # of requests
22	NbBye	# of BYE/ Total # of requests
23	NbAck	# of ACK/ Total # of requests
24	NbCan	# of CANCEL/ Total # of requests
25	NbOpt	# of OPTIONS / Total # of requests
26	Nb Ref	# of REFER/ Total # of requests
27	NbSub	# of SUBSCRIBE/ Total # of requests
28	NbNot	# of NOTIFY/ Total # of requests
29	NbMes	# of MESSAGE/ Total # of requests
30	NbInf	# of INFO/ Total # of requests
31	NbPra	# of PRACK/ Total # of requests
32	NbUpd	# of UPDATE/ Total # of requests
Group 5 - Responses Distribution		
33	Nb1xx	# of Informational responses / Total # of responses
34	Nb2xx	# of Success responses / Total # of responses
35	Nb3xx	# of Redirection responses / Total # of responses
36	Nb4xx	# of Client error responses / Total # of responses
37	Nb5xx	# of Server error responses / Total # of responses
38	Nb6xx	# of Global error responses / Total # of responses

Table 4.2: Features selected for short and long term monitoring

Short Term Monitoring		Long Term Monitoring	
Number	Name	Number	Name
11	NbReceivers	16	NbREJECTED
14	NbCALLSET	4	NbSdp
20	NbInv	20	NbInv
4	NbSdp	23	NbAck
2	NbReq	36	Nb4xx
3	NbResp	34	Nb2xx
13	NbNOTACALL	7	AvInterSdp
12	AvMsg	35	Nb3xx
		13	NbNOTACALL

One of the weaknesses of the F-score is that it doesn't reveal mutual information between features. However, since we are working in a real-time environment, we found that it accomplishes a good trade-off between performance and accuracy.

By referring to our experiments, this procedure gives out two set of features:

- The first set is suitable for short monitoring based on our experiments for the detection of different rates of flooding. This set is composed by 8 features as depicted in the left branch of Table 4.2. These features had the highest F-score in all the experiments but they were not always in the same order.

- The second set is suitable for long term monitoring, based on experiments on detection of different rates of spamming delivery. This set id composed from 9 features as depicted in the right branch of Table 4.2. These features had the highest F-score in all experiments but they were not always in the same order.

We have to emphasize that the best set of features is very dependant on the characteristics of the underlying traffic and those of attacks to be detected. In a first interpretation of these selected features, we notice that they are almost the same as the variables we had intuitively chosen for our BN model.

4.4.4 Anomaly detection using unsupervised SVM

SVM enable unsupervised learning where no labeled data set is available. One approach is to consider the training data as a cluster of points and to consider each point falling outside this cluster to be of different class. The two main methods of unsupervised SVM are the one-Class SVM and the smallest embodying sphere.

In the one-Class SVM model, we divide the data into two sets by a hyperplane: the `normal` data and the `outliers` (containing the origin) as formulated below:

$$\text{Find } \overrightarrow{w}\text{, b, } \xi \text{ and } \rho \text{ to minimize } \frac{1}{2}\overrightarrow{w}.\overrightarrow{w} - \rho + \frac{1}{\nu l}\sum_l \xi_l$$
$$\text{While respecting } \begin{cases} \overrightarrow{w}.\overrightarrow{x_l} \geq \rho - \xi_l, \forall \overrightarrow{x_l} \in S \\ \xi_l \geq 0, \forall l \end{cases}$$

Where ν is the tolerable fraction of data instances falling beyond the hyperplane (i.e. considered as `outliers`).

In the smallest embodying sphere SVM, the goal is to find the smallest hypersphere including a ν fraction of the data points. Because this problem is NP-complete, slack variables are used again to allow some degree of misclassification. This is formulated below ($\overrightarrow{w}$ designs the center of the sphere, r is the radius):

$$\boxed{\begin{array}{c}\text{Find } \overrightarrow{w}\text{, r and } \xi \text{ to minimize } \nu r^2 + \frac{1}{l}\sum_l \xi_l \\ \text{While respecting } \begin{cases} |\overrightarrow{x_l} - \overrightarrow{w}|^2 \leq r^2 + \xi_l, \forall \overrightarrow{x_l} \in S \\ \xi_l \geq 0, \forall l \end{cases}\end{array}}$$

We use the solutions of these formulations and their implementations to build an anomaly detection model in situations where only traces of `normal` traffic are available for training. In fact, we train the SVM machine with normal data instances and a small outlier ratio (e.g. 1%) to build a prediction model. The prediction model is then used in the testing mode. When a traffic slot is predicted as an outlier we consider it as an abnormal event.

The drawback of anomaly detection is that it misses a direct explanation about what kind of attack or abnormal behavior is performed; while its great benefit is the detection of new and previously unknown forms of attack. An anomaly detector can work together with a multi classifier to detect new attacks and supply them to the classification scheme. An approach using two-level hierarchical structure for anomaly detection and classification of network-layer and transport-layer flooding attacks is proposed in [113]. This approach is based on one-class SVM for both anomaly detection and multi-classification.

4.5 Conclusion

In this chapter, we presented an overall short-term and long-term monitoring system for VoIP signaling traffic in order to detect a range of VoIP-specific attacks. Two machine learning approaches were proposed to be in the heart of our system: Bayesian networks and SVM. For each approach, we gave a theoretical background, we presented a model for both multi-classification and anomaly detection, and we highlighted its peculiarities. Experimentations and comparison of the two techniques are presented later in the third part of this manuscript (Assessment) in Chapter 8. Our monitoring system is an essential component of a distributed defense solution for a VoIP enterprise domain which is presented in Chapter 6.

Chapter 5

VoIP Honeypot Architecture

Sommaire

5.1 Introduction

Even if VoIP attacks are not yet in the headline news in security reviews they soon will be of major harmfulness for the Internet telephony services. This has become certain with the growth and spread of these services at both enterprise and residential levels. However, the nature of these attacks and the possible scenarios are not fully understood yet. We must be well prepared for this battle in order to save our services and infrastructures from paying high costs. Information gathering and early warning systems constitute the most important factor in this preparation: We have to know who are the enemies, how powerful they are, and what their plans are. Early warning systems are of strategic importance to announce a state of emergency, alert defenses, estimate the impact of possible raids, and at the extreme, launch a proactive campaign to eliminate the attackers or to weaken

them. Honeypots and honeynets are widely deployed in computer networks as a main component of such a defense strategy. They have a specific value in attack detection and deflection as they are special systems that normally should not see any legitimate traffic or activity.

As a second contribution, we design a VoIP-specific honeypot. We present an individual "honeyphone" as a functional unit of a complete honeynet architecture emulating a VoIP enterprise domain. This chapter is organized as follow: Section 5.2 presents an intuitive introduction about the concept of honeypots and honeynets. In section 5.3 we explain the practical functionalities of our honeypot. We exhibit the internal design of the honeyphone in Section 5.4 and we show how it works by unrolling an example in Section 5.5. Section 5.6 closes the chapter.

5.2 What is a honeypot?

In literature, a Honeypot is a trap set to detect, deflect and monitor attacks against information systems. "A honeypot consists of an environment where vulnerabilities have been deliberately introduced in order to observe attacks and intrusions" [73]. Another definition states that "A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource" [43]. The concept of honeypot is that it doesn't present any production value and so nobody should interact or use it. Login attempts, data file access, data base connections to (or from) a honeypot are most likely malicious or unauthorized. This is very flexible definition since it doesn't restrict the application or the environment to be represented. In general, implementing a honeypot consists on providing a computer system, data or a network site that appear to be part of a production environment but which are actually isolated and monitored. Physically, implementing a honeypot doesn't require much resource thanks to emulation and virtualization. A honeynet is a high-interaction honeypot meaning that it provides real systems, applications and services to lure the attackers into interacting with them. The honeypots have the advantage to detect new and unseen attack signatures and to recognize attack plans. Nevertheless, they have a limited field of view because they only see attacks that are directly guided towards them. Thus, many honeypot projects have developed cooperation and collaboration relationships around the world. Maintaining a honeypot is sometimes dangerous because there is a risk that the honeypot would be compromised by a worm and therefore used to propagate the worm and attack real systems. The honeypot has to be well isolated, physically and logically from the production environment.

5.3 Functional aspects of our honeypot

We propose a VoIP-specific honeypot that can be installed within any VoIP network. A typical enterprise domain interconnecting the Internet with other telecommunications networks has a PBX system representing the VoIP services (Gateway, voice mail, call switching, answer machines ...) and a SIP proxy representing the routing infrastructure inwards and outwards the domain. We have implemented such a domain in our test-bed

composed of several SIP clients, Asterisk PBX [7] and SER proxy [8].

The Asterisk PBX is growing fast and replacing other SIP servers within Small Office/Home Office environments (SOHO). Asterisk is a gateway to different VoIP protocols (SIP, H323, IAX (Inter Asterisk eXchange)) and telephony networks (ISDN, PSTN). In addition, Asterisk supports call switching, voice mailboxes, IVR (Interaction Voice Response), file playback and directory listing. It offers codec translation, scheduling and management capabilities. According to the SIP point of view, Asterisk is a registrar, location server, and a back-to-back user agent.

SER is a high performance SIP proxy since it is able to manage a high number of registered users and thousands of calls per second if installed on a dual-CPU PC. SER can act as a SIP proxy, registrar or redirect server. SER features a variety of database backends, management features, NAT traversal and telephony features like least cost routing and speed dial. Based on its advanced routing logic, SER is flexible for complex routing configuration and based on its modular design, it can be easily extended with additional modules.

Many real-world VoIP solutions deploy SER and Asterisk simultaneously in order to serve specialized purposes such as load balancing. We installed both on our test-bed: Asterisk serves internal communications, user preferences and accounting, while SER is responsible for external communications. The honeypot registers itself with a number of SIP URIs at the registrar sever (at both SER and Asterisk) and is therefore able to receive calls as shown in Figure 5.1. Within SER, the registrar queries the location database to find the IP address corresponding to the request URI upon receiving an INVITE message. We properly configure SER and Asterisk so that SIP messages will be forwarded to the honeypot with minor modifications and that media sessions will be directly established with it instead of being relayed by Asterisk

Based on this low cost infrastructure, our honeypot has several functional aspects performing enumeration detection, SPIT mitigation and attack signature collection. We expand these functionalities next and we end by discussing the honeypot tuning options.

5.3.1 Enumerating Detection

Let's consider the following scenario: an attacker "Bob" is trying to gather information about a VoIP domain "abc.com". Bob succeeds to intercept the signaling of a limited number of calls and uses the intercepted data to identify a sequence of SIP extensions: "sip:300@abc.com", "sip:303@abc.com" and "sip:380@abc.com". "Bob" infers that all users from the "abc.com" domain have extensions of the form "sip:3xx@abc.com". To be able to identify all the users, he launches an enumeration attack. Such an attack would normally be formed by OPTIONS (SIP ping) messages with URIs of the form "sip:3xx@abc.com". The responses to these OPTIONS messages can be easily interpreted in order to map the target domain. However, Bob is not aware of our hidden honeypot which -in a previous time- has registered a sequence of unused "3xx" extensions to the domain registrar. The honeypot will receive one or multiple OPTIONS and alarm the

[7] Asterisk: The Open Source PBX, `http://www.asterisk.org`

[8] SIP Express Router, `http://www.iptel.org/ser`

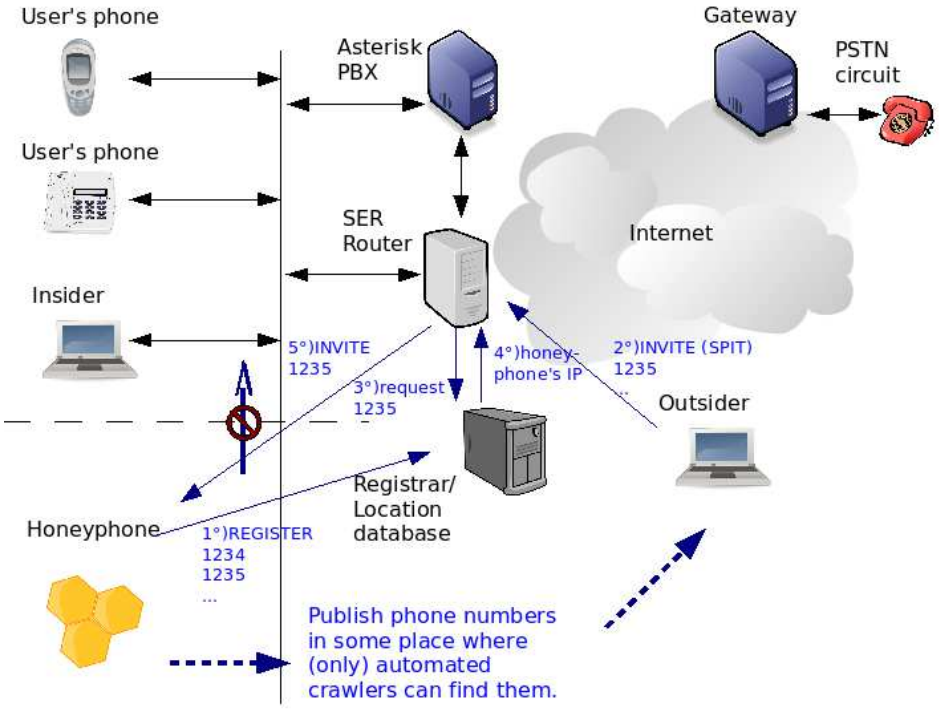


Figure 5.1: Reception of an INVITE

attack. Any enumeration of the domain can be detected within real time. We have multiple choices to configure the honeypot response to an OPTIONS message. The first choice is to not respond keeping secret the existence of the honeypot (an ICMP error message can be sent). In result, the proxy relaying the messages sends a "NOT FOUND" response. The second choice is to respond with a "200 OK" allowing the attacker to go further in his plans.

5.3.2 VoIP SPAM mitigation

Our honeypot feeds SPIT detection and prevention systems with many useful pieces of information. The SIP URIs declared by the honeypot play the role of attack deflectors. Since they should normally not experience any traffic activity, all calls destined to these deflectors are interpreted as malicious. The callers can be added to a black list either temporarily or permanently. The members of the black list are not allowed to initiate calls to the real users. Different honeypots can form a federation to share activity reports / black lists. A caller seen by different honeypots is more likely to be malicious than to have made a dialing error. Sharing activity reports within a federation of honeypots have several advantages that are not only limited to SPIT prevention.

In order to make our deflectors target of SPIT/Vishing attacks, we have to prepare scripts that systematically publish them in some place where the attackers and their automated crawlers can find them (web pages, emails, newsgroups, etc.) such that legal users can not see them (e.g. we write URIs in white over a white background in a web-page) or easily recognize that they should not call them (e.g. URIs with the word

do-not-call). Only bots automatically looking for semantics of URIs will harvest these deflectors.

The mitigation of the attack is an increasing function of the number of published fake URIs. In effect, the attacker will retrieve not only the real `x` users but also the fake `y` ones. The mitigation percentage is proportional to the ratio of the number of fake users to the number of all users `y/(x+y)` since the SPIT/Vishing calls will be distributed among all of them.

The honeypot registers every received call. Media content analysis using machine learning techniques (e.g. building a model of SPIT calls versus a model of legal calls) is useful for voice-mail filtering within voice mail-boxes similarly to what is done for e-mail SPAM.

5.3.3 Signatures collection

VoIP equipment softwares have different kind of vulnerabilities that can be remotely exploited using special malformed messages. Collecting signatures of these exploits is one of the missions of our honeypot. To trap the attackers, the honeypot announces some fake fingerprints (e.g. manufacturer string, product name, firmware version) that are interesting for the defenders (e.g. a domain administrator may announce the fingerprints of the deployed devices in his domain, a manufacturer company may announce the fingerprints of its VoIP products). The honeypot logs every received message along with the targeted fingerprint and makes them available for further study and extraction of possibly found attack signatures.

Each SIP URI announced by the honeypot is given a fingerprint. Upon receiving of a request destined to this SIP URI, the honeypot puts the fingerprint in the response (in the `User-Agent` or the `Server` headers). This is going to deceive an attacker using passive or simple OPTIONS fingerprinting (In passive fingerprinting, the attacker intercepts a message sent by the target. In OPTIONS fingerprint, he sends an OPTIONS message to the target and interprets the response). The challenge is more complicated if the attacker is using an active fingerprinting scheme that learns the target state machine (e.g. as proposed in [112] or [3]). Also in the future, we consider collecting malwares by providing a virtual SIP UA that is isolated and monitored.

5.3.4 Tuning the honeypot: from passive to active reactions

We allow configuring the reactivity of our honeypot following three modes: passive, active and aggressive.

- In the passive mode, the honeypot plays the role of the victim. It dumps all the received calls and signaling messages for further analysis.

- In the active mode, the honeypot interacts with the callers in order to gather more information about them: e.g. the SIP fingerprint, the OS fingerprint, used IP addresses, IP path towards the source, opened ports (several opened RTP ports indicate that many calls are initiated in parallel), etc. The information gathering

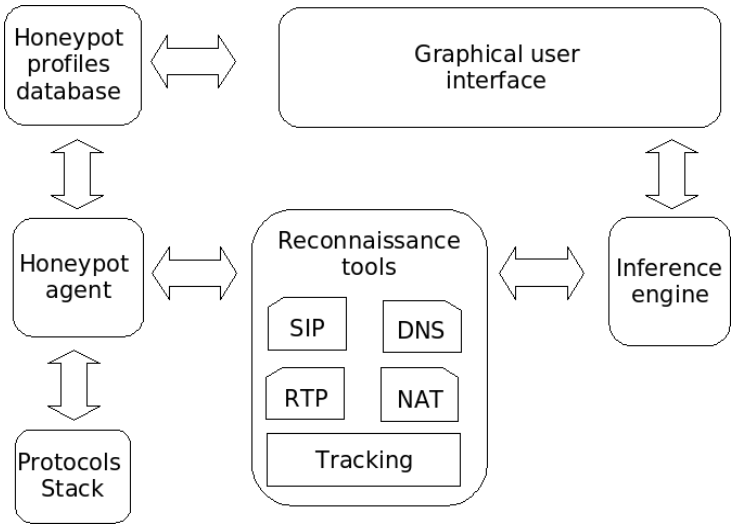


Figure 5.2: Honeyphone architecture

procedures are detailed later. In addition, the honeypot may check human communications patterns in the received call using Turing tests as proposed in [75].

- In the aggressive mode, information gathering is not sufficient: After estimating the risk of the attack and the identity of the attackers, the honeypot may found it necessary to proceed with a counter-attack to displace the danger from the real domain. E.g. It scans an attacker machine for vulnerability, finds a tool or script that can exploit that vulnerability, and uses this exploit to tear down or take control of the attacker machine. Legal aspects of such aggressiveness are out of the scope of our study. In the next section we present the design of our honeypot (we will use the term honeyphone) with provisioning for only two modes of reactivity: passive and active.

5.4 Internal Honeyphone Design

We strive for our honeyphone an efficient, modular and flexible design. The honeyphone architecture is composed from five main components as depicted in Figure 5.2:

- the agent is the core and the operational engine of the overall architecture. It handles incoming events based on a preconfigured state machine and takes corresponding actions e.g. : call reception, information gathering, call recording and alert generation;

- the protocol stacks (SIP, SDP, RTP, STUN ...) are based on protocol standard specifications, they are responsible for message manufacturing, message parsing and message transmission over the transport layer. STUN and TURN protocols are used to bypass classical NAT traversal problems. We however encourage the deployment of the honeyphone over a public IP infrastructure;

- the profile database allows the administrator to create, delete, save and restore profiles depending on his preferences and the network circumstances. Each profile is constituted of several configuration files in order to set up the honeyphone in the network environment and to control the behavior of the agent. Profile custom settings are very flexible so that they can assign different roles to the agent (e.g. network assessment);

- the reconnaissance tools are used in the information gathering procedures (e.g. nmap[9] for network and port scanning, SIPSAK[10] for SIP ping, traceroute and messages crafting, pOf[11] for passive OS fingerprinting);

- the inference engine represents the intelligence of the system. It is able to automatically interpret received messages and make autonomous decisions. In our design, the inference engine is constituted of a Bayesian model that interprets a received SIP message based on the message content and eventual results of investigation.

- the Graphical user interface (GUI) provides a user friendly interface to the administrator allowing him to configure profiles, reactivity and inference parameters, and to visualize eventually collected results and statistics.

5.4.1 Profile configuration

A profile supports two types of parameters: the environment parameters and the behavior parameters. The former are used to set up the honeyphone in the surrounding environment similarly to any SIP phone. We use an `attribute: value` grammar to define the following settings:

- identity settings like for instance the IP address, the used ports and the SIP accounts to be registered;

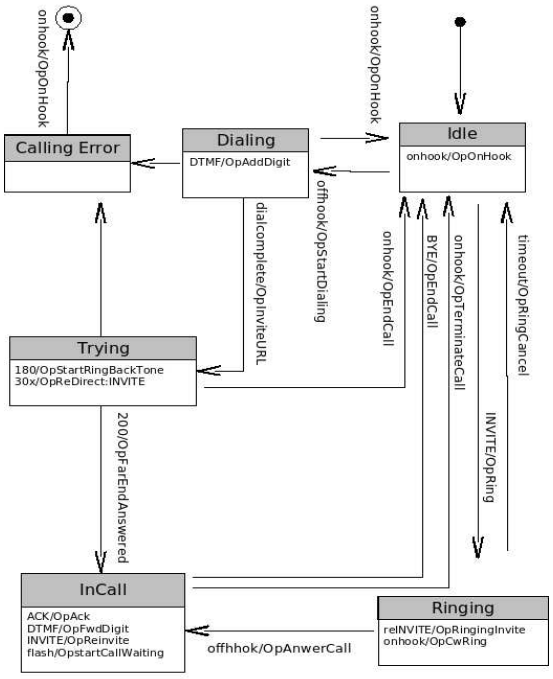
- network preferences like for instance the registrar server, the DNS server, the outbound SIP proxy and the voice mail server;

- audio preferences like the codecs and the RTP settings;

- SIP preferences like the life time of registration for each URI, the fingerprints, etc.

On the other hand, the behavior parameters define the honeyphone behavior upon receiving a message or a call e.g. to take or reject the call, to fix a time of ringing and a time to listen to the caller, etc. These parameters can be set as random (within a certain interval) or as determinist. Behavior parameters are used to preconfigure the state machine of the agent. This state machine is based on a SIP UA model. We adopt a model early published by the open source VOCAL project [21]. In that model, each state is within an object-oriented scheme and has a list of operators. The operators are

[9] Nmap:Network Mapper, `http://www.insecure.org/nmap/`
[10] Sipsak: SIP Swiss Army Knife, `http://sipsak.org/`
[11] pOf: passive OS fingerprinting tool, `http://lcamtuf.coredump.cx/p0f.shtml`




```
Idle {
        onhook/OpOnHook,
        INVITE/OpRing,
        offhook/OpStarDialing
        }
Ringing {
        reINVITE/OpRingingInvite,
        onhook/OpCwRing,
        timeout/OpRingCancel,
        offhook/OpAnswerCall
        }
Dialing {
        onhook/OpOnHook,
        DTMF/OpAddDigit,
        dialcomplete/OpInviteUrl,
        }
Trying{
        180/OpStartRingBackTone,
        30x/OpReDirect,
        onhook/OpEndCall,
        200/OpFarEndAnswered,
        }
Incall{
        onhook/OpTerminateCall,
        BYE/OpEndCall,
        DTMF/OpFwdDigit,
        INVITE/opReinvite,
        flash/OpstartCallWaiting,
        }
Calling Error{
        onhook/OpOnHook,
        }
```

Figure 5.3: Behavior configuration of a simplified SIP UA state machine

executed at the entrance of the state, at the exit of the state or upon occurrence of defined events. Some operators are responsible of guiding the machine by passing it from one state to another. There are two main types of events: a `SipEvent` occurs upon the reception of a SIP message and `TimerEvent` occurs when a time period triggered by an operator has elapsed.

Our approach is to take this model and to extend it by adding events and operators so that different behaviors can be simulated. To describe the state machine, we define a special syntax where each state is described by a clause of the form:

```
State1 {
        Event1/Operator1[,
        Event2/Operator2 ...]
        }
```

In figure 5.3, we show a simplified state machine (taken from [21]) to the left and the corresponding description syntax to the right. This syntax is very simple. For example `Idle{onhook/OpOnHook}` means that in the `Idle` state the operator `OpOnHook` is executed upon the occurrence of the event `onhook`. We reserve two keywords to represent the state entrance and the state exit events: `entry` and `exit`, respectively.

Using this syntax, we are able to preconfigure the agent state machine to fit a certain behavior. For example, the honeyphone unhooks automatically after 5 seconds of receiving a call if we change the `Ringing` clause as follows:

```
entry/OpStartRing(5),
ringtimeout/OpAnwerCall
```

In fact, the `OpStartRing` operator triggers a timer of 5 seconds duration. When the time is elapsed, a `ringtimeout` event is generated invoking the `OpAnswerCall` operator which takes the call and drives the machine to the `InCall` state. Let the honeyphone play back an audio (speech) file for 60 seconds after answering the call and let it record the conversation:

```
entry/OpRecord,
entry/OpPlayFile(speech0.mp3, 60)
```

Note that when two operators catch the same event, they are executed within parallel threads.

The advantage of such an approach is to automate events like `offhook` and `onhook` which are normally initiated by human beings. We provide tow ways for the administrator to define a certain behavior: The first one is a state machine description editor supplied with a documented help containing explanations of possible operators, states and events. The second one is by filling an application using widgets (listboxes, checkboxes, dialogs ...) of the GUI. Before launching the honeyphone, the state machine description is compiled into executable code and uploaded to the agent.

5.4.2 Information gathering

In the active mode, the honeyphone has to gather information about each received call. The INVITE message is of particular importance because it identifies the source of the call. This is translated in our description syntax by adding an `INVITE/OpInvestigate` statement in the `Idle` clause. The operator `OpInvestigate` releases the work of the reconnaissance tools while the honeyphone is processing the call.

The honeyphone forms a query list of important items in the INVITE: IP addresses, protocol ports, URIs, host names and domain names. These items are extracted form the SIP headers: `From`, `Contact`, `Via`, `Record-Route` and `Route` and from the SDP attributes: `Connection(c)`, `Owner(o)` and `Media(m)`. An example of investigation questions/answers about a received INVITE message is depicted in figure 5.4.

For each item in the query list, we inquiry the following data:

- `WHOIS` : We look up for the WHOIS record of the item in the Internet registries. The WHOIS contains the ownership and other real world information. Then we look for a trust score assigned to that record in a WHOIS trust table that we prepared earlier. The honeyphone can update the WHOIS trust table by decreasing the score of frequently seen WHOIS.

- Valid DNS : We check if a domain name item is resolvable or if an IP has a domain name by means of DNS, Reverse DNS, DNS SRV and ENUM requests.

- `AS` : We look up for the autonomous system number where the queried item resides. Then we look for a trust score assigned to that AS in an AS trust table that we prepared earlier.

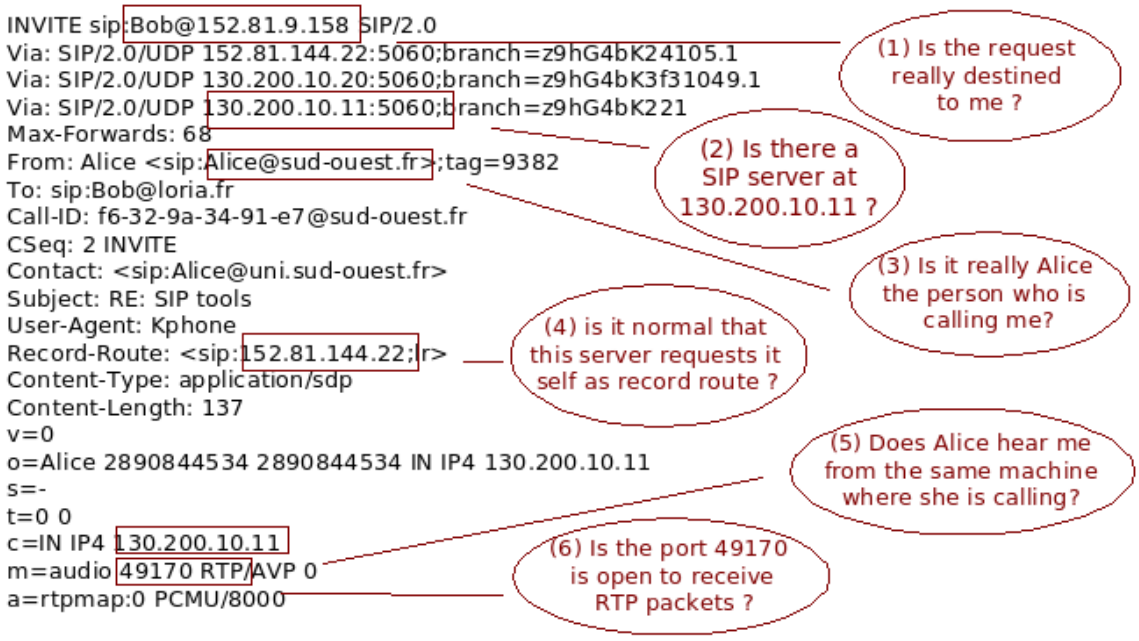


Figure 5.4: Questions about a typical INVITE message

To answer question 1:

- check if I am registered with the SIP URI: Bob

To answer question 2 :

- send OPTIONS request to 130.200.10.11:5060
- if the response is OK, some SIP stack is running there
- check for some DNS SRV record in the DNS server

To answer question 3 :

- hard to answer
- check if all the participating servers do not allow changing of the caller-ID

To answer question 4 :

- check if the server belongs to the Via chain
- evaluate a trust in the server

To answer question 5:

- check if 130.200.10.11 corresponds to uni.sud-ouest.fr in the Contact header using DNS or Reverse DNS requests

To a question 6 :

- send UDP packet to 130.200.10.11:49170
- if no ICMP Destination Unreachable response is received so the port is open

- `GL` : We look up the geographic location of the queried item. Then we look for a trust score assigned to that location in a location trust table that we prepared earlier.

- `Has Port` : We scan the queried item for the ports (SIP, RTP) it pretends to be open in order to check if they really are.

- `Fingerprint` : We fingerprint the queried item by an OPTIONS message. Then we look for a trust score assigned to that fingerprint in a fingerprint trust table we prepared earlier. We consider more advanced fingerprinting (by a sequence of OPTIONS as proposed by [112]). In particular, we fingerprint the source (the From header) and we check if our fingerprint matches that announced by the source (that can be found in the `User-Agent` header in the received INVITE). If no match, the fingerprint is totally distrusted.

- `Historical cache`: We look up how frequently the queried item has been seen before. The historical cache value of an item decreases by aging and increases with new occurrences of that item.

In addition, we trace SIP and IP routes towards the different items and we look for inconsistencies in the traced routes. SIP trace route is based on a sequence of OPTIONS with increasing `Max-forwards` (SIP TTL) value. Other characteristics of the SIP message that help identifying the source can be investigated as well (e.g. The structure of the message, the used codec). All the results of the inquiry procedures are input to the inference engine.

5.4.3 Inference engine

A priori, a message received at the honeypot must be considered as malicious. It can however result from an IP routing fault, a SIP routing fault, a bad configuration or a wrongly dialed number. Some attacks like SPIT or enumeration need to receive the responses of the target in order to proceed (e.g. a SPITer needs to know the RTP port where to send the media) while some other attacks like remote exploits doesn't. The results of the investigation procedures indicate if a queried item is spoofed and give us several trust scores. Based on these results, we reason as follow: if the message appears to be normal but distrusted, it is probably resulting from a SPIT or enumeration attack. If the message has several spoofed items, it is probably a crafted exploit. If the message appears to be normal and it is trusted, there is some unintentional fault. To release the human operator from interpreting a large amount of data, we propose an inference engine that automatically analyses all these results and give us a conclusion about each received message. A first prototype of this inference engine is based on a Bayesian model as depicted in figure 5.5. An introduction to Bayesian networks was presented in chapter 4.

Our model defines three hypothesizes at the root node:

- `Crafted`: the message is spoofed;

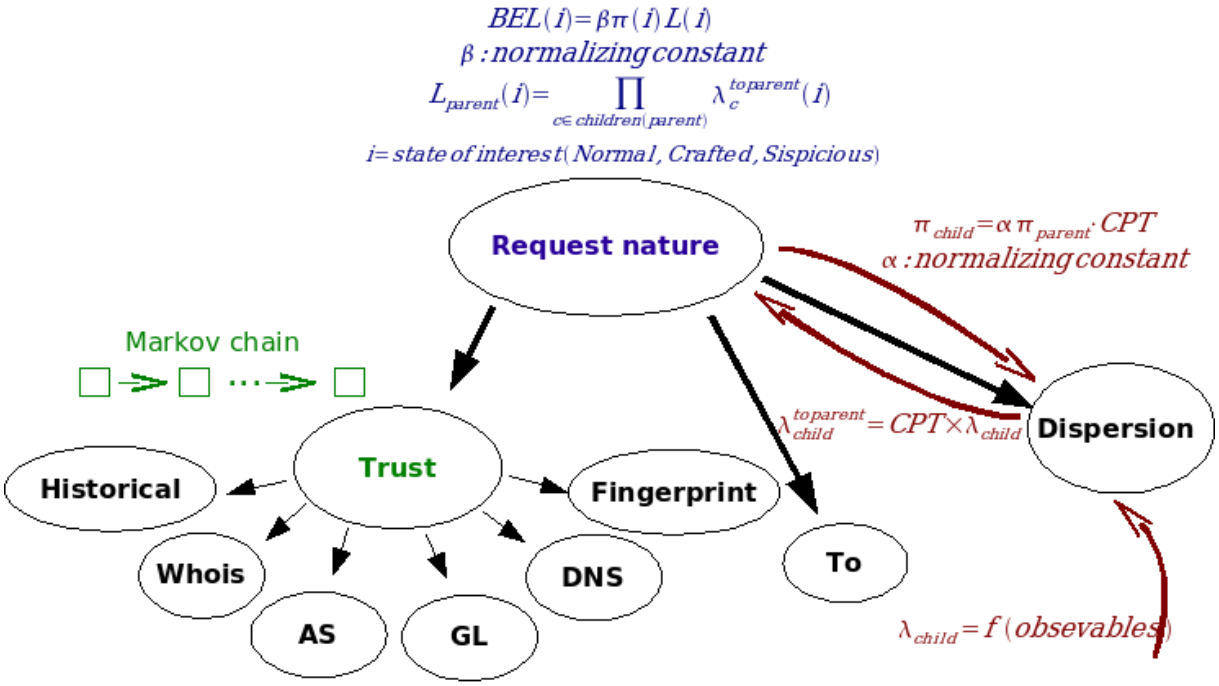


Figure 5.5: Artificial reasoning about the nature of a received SIP message

- `Suspicious`: the message is not spoofed but it is not trustable, it is classified as a SPIT or an enumeration;
- `Normal`: the message is not spoofed and it is trustable, it is classified as an unintentional fault.

We define the following variables at the leaf nodes:

- `To`: is a Boolean variable that indicates if the request URI in the `To` header of the message matches any of the initially announced URIs. If no match, the `normal` hypothesis is reinforced.
- `The dispersion of the source points`: is geometrically calculated in a one-dimensional space of origin O representing the honeyphone's IP and 5 other points: The point A represents the signaling source (the IP extracted from the first `Via`), the point B represents the IP extracted from the `Contact`, the point C represents the media source (the IP in the SDP's `Connection`), the point D represents the IP in the SDP's `Owner`, and finally the point E represents the IP of the `From` domain. The distance from each point to the origin is calculated as the number of IP hops by a special traceroute. Our traceroute consists of a sequence of OPTIONS with increasing TTL IP header (To not confound with a SIP traceroute formed by a sequence of OPTIONS with increasing `Max-Forwards`). The idea is that in a normal call, all these points represent the same host or they belong to the same domain and they are close to each other. But if some points are spoofed, they are sparse over

our axis. We define the dispersion metric as the standard deviation of the distances to these 5 points:

$$\sigma_{sources} = \frac{1}{5} \sum_{P \in \{A,B,C,D,E\}} \overline{GP}^2$$

where G is the barycenter of A, B, C, D, E.

- `Trust`: The overall trust on a queried item is inferred from the individual trust scores (of WHOIS, geographical location, autonomous system, DNS, "has port" variable, fingerprint and historical cache) based on a naïve Bayesian tree. The overall trust on a message is inferred from the trust scores of the queried items (the source and the proxies routing the call) based on a discrete state and time Markov chain. Each state in this Markov chain represents a certain level of trust. The initial state represents a medium level of trust. The state transition is based on the fact that only one distrusted item will raise significantly the distrust level of the message.

Next we show how our inference engine works by unrolling of a scenario.

5.5 Unrolling of a scenario

We consider the reception of an INVITE message (the INVITE of Figure 5.4) and we show step-by-step how the inference engine proceeds based on the inquiry results and infers a final conclusion. Let's assume that the INVITE message is totally correct (no items are spoofed) and that the honeypot has not registered with the SIP URI found in the To header ("Bob@152.81.9.158"). Thus, the reception of this message is due to a routing error (e.g. a legal user called Bob was assigned the IP "152.81.9.158", the user goes down and releases the IP. The honeyphone reboots and is reassigned the released IP by the DHCP server. The proxy receives the message and queries the location database which has kept the entry binding that IP to Bob, hence the routing fault. We however prefer that the honeyphone is assigned a static IP to avoid such problems). We show how our inference engine will arrive to the same hypothesis based on the collected data.

Let's start by defining the items of the inquiry list. The interesting items are shown in the following table:

Attribute	Value
From	Alice@sud-ouest.fr
Contact	Alice@uni.sud-ouest.fr
Via_0	130.200.10.11:5060
Via_1	130.200.10.20:5060
Via_2	152.81.144.22:5060
Record-Route	152.81.144.22
Connection	130.200.10.11:49170
Owner	130.200.10.11

This list is formed by 3 IP addresses, one domain name and one host name. Let's assume having the results depicted in table 5.1 after investigation.

Table 5.1: Results of investigation for the INVITE message

	sud-ouest.fr	uni.sud-ouest.fr	130.200.10.11	130.200.10.20	152.81.144.22
Frequency (Historical)	0 (not seen before)	0	0	0	0
WHOIS	Universite Pole nord	Universite Pole nord	Universite Pole nord	Universite Pole nord	LORIA-INRIA
AS	-	1200	1200	1200	1100
G.L.	Toulouse	Toulouse	Toulouse	Toulouse	Nancy
Has port	-	yes	yes	yes	yes
SIP F.P.	-	Kphone	Kphone	SER	SER
DNS	true	147.210.9.155	uni.sud-ouest.fr	proxy.sud-ouest.fr	proxy.loria.fr

The next step is to calculate the individual trust in each item based on its individual trust scores. We setup manually the Conditional Probability Tables (CPTs) linking each child node to the parent node. For example let's assume that the following CPT is linking the `Historical` node to the `Trust` node:

$$CPT(Historical/Trust) = \begin{array}{c|ccc} & 0 & [1-2] & [3-\propto] \\ \hline T & 0.9 & 0.1 & 0.0 \\ D & 0.1 & 0.2 & 0.7 \end{array}$$

Where `T` stands for `Trusted` and `D` stands for `Distrusted`. The effective number of times the item has been seen (we used the term "the effective number" in contrast to "the real number" in order to indicate the aging process) is divided into 3 intervals: 0 for previously unseen, $[1-2]$ if it has been seen for one or two times and $[3-\propto]$ if it has been seen for three or more times. The first line of the CPT is based on the following reasoning: Under the 'Trusted' hypothesis, there is a 0.9 probability that the item has not been seen before, 0.1 probability that the item has been seen for one or two times before, and 0.0[12] probability that the item has been seen for three or more times before.

Let's evaluate the trust in the first item "uni.sud-ouest.fr", shortly "uni". If we assume it is never seen before, the λ at the child node is:

$$\lambda_{Historical}\left|\begin{array}{l} 0 \\ [1-2] \\ [3-\infty] \end{array}\right. = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

The λ message passed to the parent node is calculated using the CPT(Historical/Trust). We obtain:

$$\lambda^{to\_parent}_{Historical}\left|\begin{array}{l} T \\ D \end{array}\right. = \begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix}$$

[12]It is often desirable to not have parameters set exactly to 0, because this would neglect the influence of other parameters during the inference. In general small corrections are required (e.g. set a parameter to 0.001 instead of 0)

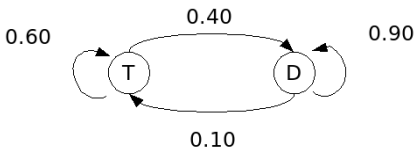
Likewise, we present the λ messages to the other children of the `Trust` node. The likelihood messages at the `Trust` node are then merged together and normalized. We obtain the following result (overall calculation is not shown):

$$\lambda_{Trust}\left|\begin{array}{l} T \\ D \end{array}\right. = \begin{pmatrix} 0.91 \\ 0.09 \end{pmatrix}$$

We assume equal prior probabilities for `T` and `D`. The belief (the trust on the item "uni") is obtained by multiplying (element by element) the prior and the likelihood messages:

$$Bel_{Trust}\left|\begin{array}{l} T \\ D \end{array}\right. = \begin{pmatrix} 0.91 \\ 0.09 \end{pmatrix}$$

A Markov chain is used to merge together the trust values on the different items. The simplest example is a Markov chain with only two trust levels hence two states 'T' (for trusted) and 'D' (for distrusted):



Let's assume the following transition matrix:

$$M = \begin{array}{c|cc} \curvearrowleft & T & D \\ \hline T & 0.60 & 0.10 \\ D & 0.40 & 0.90 \end{array}$$

The first column of this matrix is based on the following reasoning: If we are in the state `T` for the item of index i, there is a probability of 0.60 to be in state 'T' and a probability of 0.40 to be in state `D` for the item of index $i+1$. The prior trust of the item $i+1$ is evaluated based on the posterior trust of the item i and the transition matrix M:

$$Trust_{i+1}^{prior} = M.Trust_{i}^{posterior}$$

As such, the overall trust on the message is obtained after inferring all the items:

$$\lambda_{Trust}\left|\begin{array}{l} T \\ D \end{array}\right. = \begin{pmatrix} 0.95 \\ 0.05 \end{pmatrix}$$

This result is interpreted as that the INVITE is coming from a trusted source and has been routed through trusted proxies.

At the second level of our model, we insert global variables characterizing the whole message in the diagnosis. In this prototype we are only using the `To` and the `Dispersion` variables. Since the `To` header does not correspond to any of the URIs announced by the honeyphone, we assume the following λ:

$$\lambda_{To}\left|\begin{array}{l} Yes \\ No \end{array}\right. = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

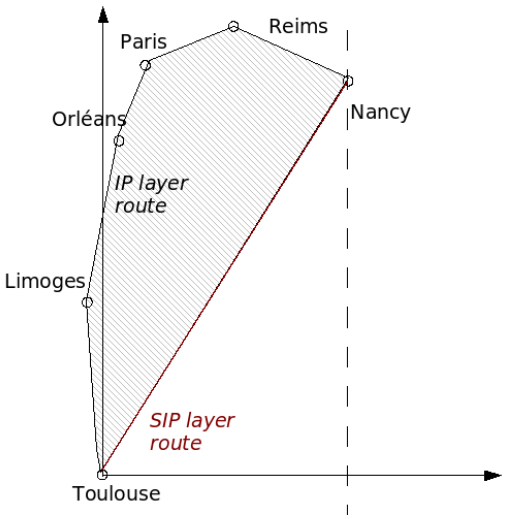


Figure 5.6: SIP Vs. IP routes towards the source of the INVITE

We set manually the CPT linking the `To` node to the `Nature` node as follows:

$$CPT(To/Nature) = \begin{array}{l|cc} & Yes & No \\ \hline Normal & 0.1 & 0.9 \\ Suspicious & 0.5 & 0.5 \\ Crafted & 0.5 & 0.5 \end{array}$$

The first line of this matrix is based on the following reasoning: under the 'Normal' hypothesis, there is a 0.1 probability that the `To` value is `Yes` (the message is really destined to us) and 0.9 probability that it is `No`. The λ message passed to the root node is:

$$\lambda_{To}^{to\_parent}\Big|\begin{array}{l} N \\ S \\ C \end{array} = \begin{pmatrix} 0.5 \\ 0.25 \\ 0.25 \end{pmatrix}$$

The source dispersion metric is 0 since all the items are on the same distance from the honeypot ($\sigma_{sources=0}$). A corresponding λ message is passed to the root node.

The likelihood messages form the 'Trust', 'To' and 'Dispersion' nodes are merged together and normalized. We obtain:

$$\lambda_{Nature}\Big|\begin{array}{l} N \\ S \\ C \end{array} = \begin{pmatrix} 0.9 \\ 0.05 \\ 0.05 \end{pmatrix}$$

If we assume that a priori, the `Suspicious` and `Crafted` hypothesizes have more chance to be true than the `Normal` hypothesis (since by default, any activity seen at the honeypot is considered as malicious), we assume the following π:

$$\pi_{Nature} = \begin{pmatrix} 0.3 & 0.4 & 0.4 \end{pmatrix}$$

The belief in the nature of the INVITE request is calculated by multiplying (element by element) the λ and the π messages:

$$Bel_{Nature}\begin{vmatrix} N \\ S \\ C \end{vmatrix} = \begin{pmatrix} 0.88 \\ 0.06 \\ 0.06 \end{pmatrix}$$

Obviously, the `Normal` hypothesis is dominant. This result helps to make the right decision about the message (e.g. e-mailing the administrator that there is some configuration error).

In addition, We can draw the SIP route and the IP route back to the source as depicted in figure 5.6. Such a geographical view makes it easy for the administrator to detect an abnormal routing situation (e.g.: A SIP message coming from Nancy to Toulouse is routed using a proxy in Brazil). To quantify this feature, we can use a geo-metric like the area bounded by the IP route and the SIP route.

5.6 Summary and future works

In this chapter, we presented a VoIP-specific honeypot as an important security component in our combat against the VoIP threats. The Honeypot has a rich set of functionalities rather than observing and recording the attacks: It can contribute to SPIT mitigation, user enumerating and signature collection. The reactivity of the honeypot can be tuned to investigate about the source of an attack and at the extreme, to pass a counter-attack. The decisions about received messages are autonomously taken based on investigation procedures and an artificial intelligence model. We detailed the internal design of our honeyphone and we unrolled a scenario of a diagnostic.

The honeyphone software is partially implemented using the Jain-SIP library [67]. The project is still considered to be incorporated in the Madynes high security laboratory [13]. In future works, we aim to scale our approach to design a high-interactivity honeypot (a honeynet). As the main goal of a honeypot is to attract attacks into a secured and observed environment, it has to offer attractive services that are worth to be attacked. A single honeyphone has a limited field of view so a limited set of attacks to be detected. To have an important defense value, we think a SIP honeypot has to simulate a whole SIP network offering several fetching values: SIP infrastructure (e.g. SER proxy), SIP services (e.g. Asterisk), and SIP users (e.g. honeyphones).

Chapter 6

Distributed VoIP Intrusion Detection

Sommaire

6.1 Introduction

VoIP intrusion detection is challenging for many reasons:

- VoIP systems are distributed (UAs, servers, proxies, PBXs). Monitoring network entry points is insufficient because of possible insider malicious activities.
- VoIP systems employ a multitude of protocols for signaling, management, configuration and media delivery;
- VoIP systems are heterogeneous (different implementations, manufacturers, configurations ...) and typically placed under different autonomous administrations;
- VoIP systems just settle in the Internet where hackers have easy access and much experience. They are exposed to a wide set of classical Internet attacks.
- Some detection has to be coupled in real-time with prevention. Offline SPAM detection techniques can not be used in the case of SPIT because we need to block the calls before disturbing the users.
- Some checks must meet strict requirements. Sometimes, only one SIP message is needed to exploit a vulnerability. Low-rate floods using specially crafted messages can escape from loose detection and they are very efficient against some servers.

Thus, VoIP intrusion detection is difficult to tackle by a centralized approach. In this chapter, we propose a distributed intrusion detection solution based on the correlation of events and alarms provided by host-based and network-based detection components (called sensors). The host-based components are deployed at several points of interest (gateway, user agent, proxy) to monitor the states of their systems and the application behavior of their users. The network-based components monitor the network traffic of different VoIP protocols.

Therefore this chapter is organized as follow: Section 6.2 presents a host-based sensor for a VoIP PBX system. Section 6.3 presents a network-based sensor for the SIP protocol. Distributed event correlation is addressed in Section 6.4. Conclusions and future works are addressed in Section 6.5.

6.2 VoIP-specific Host-based Intrusion Detection System (VoIP H_IDS)

A Host-based IDS aims to protect a critical host by analyzing the audit trails of its operating system and applications. Security violations are detected by abnormal patterns of system usage (abnormal behavior of subjects with respect to objects as formulated by one of the pioneering works in this domain [22]). A VoIP-specific H_IDS is required to defend the running VoIP applications (user agent, PBX, gateway) from being compromised, abused, wrongly managed or misconfigured. The security of the application is strongly derived form the security of the underlying operating system and the supporting services (e.g. Database, DNS, Mail server ...). A VoIP PBX has two main application-specific sources of information:

- Log files : gives an overall overview about the state of the PBX and helps investigating suspect situations like successive login failures or successive registration attempts.

- Call Detail Record (CDR) : helps building user profiles and investigating toll fraud and errant account behavior.

For example, let's show a message of interest from an Asterisk system's log file:

```
Oct 20 15:46:40 NOTICE[4689]: Registration from
'"radu" <sip:radu@152.81.114.151> -'  failed for '152.81.114.132'
```

The occurrence of several similar notices targeting the same user within a short time period may reveal a password cracking attempt. An open source H_IDS called OSSEC[13] already includes a special module for Asterisk log monitoring which can detect similar situations.

On the other hand, important fields are found in a CDR. For example:

[13] http://www.ossec.net/main/

Call time	Mon oct 23 2006 14:08:07
Caller	A@Inria.fr
Caller location	100.101.102.103
Callee	Bob@loria.fr
Duration	00:02:19
Bill-duration	00:01:88
Disposition	Answered

The CDRs of a user account can be transformed into histograms to build a user profile. We propose an anomaly-based detection of malicious accounts based on the abnormal behavior of an account profile and a group of account profiles. Such an approach is efficient to detect fraudulent usage, virus propagation and social threats like SPIT. In fact, we create a statistical profile for normal behavior of each user, and we generate anomaly records when deviations from this normal behavior occur. We build a profile as follow: a day is divided into bins of a predefined time (e.g. one hour). For each bin, a predefined metric is calculated (e.g. number of calls, number of different recipients, average duration of a call) over predefined events (e.g. calls). In the training phase (e.g. a month), we generate a long-term profile (e.g. the average number of calls per each bin). In the testing stage (e.g. a day), we generate a short-term profile and we compare it to the long-term profile by using an appropriate distance function (e.g. Euclidean distance, quadratic distance, Mahalanobis distance[14]). A large deviation from the long-term profile is reported.

We can also study some non-stationary features of an account such as the distribution of the dialed calls over all the callees or the variation of the callee list size over time. The difference between two distributions characterizing two different time periods is calculated by using an appropriate distance function (e.g. Hellinger distance [4]). A large deviation between two distributions belonging to two successive time periods as well as sudden bursts in the shape of a variable are treated as abnormalities.

On the other hand, long term profiles of the users can be classified into communication groups (social cliques). Social attacks are then detected by looking for activities that break the borders of these groups. For example, a member of group A starts to make calls to the members of group B. Also if a member of one group is known as malicious, it raises the level of suspicion on the other members of the same group. Several behavior-based approaches to secure e-mail systems (e.g. [94]) can be leveraged to the case of VoIP.

6.3 VoIP-specific Network-based Intrusion Detection System (VoIP N_IDS)

A network-based IDS monitors packets on the network in order to discover intrusions into the interconnected systems (e.g. network scan, DoS, R2L, break-in ...). A NIDS may run

[14]Mahalanobis distance is based on correlations between variables by which different patterns can be identified and analyzed. It is a useful way of determining similarity of an unknown sample set to a known one (Wikipedia).

either on the target system to watches its own traffic (It can be completely transparent if acting as an Ethernet bridge), or on an independent device promiscuously watching all the network traffic.

We propose a SIP-specific N_IDS which analyzes SIP flows and looks for suspicious patterns of headers, unique messages and sequence of messages. Our system aims in the first place to detect SIP exploits and crafted messages which present a real damage for a VoIP/SIP network. In fact, they can result in:

- remote DoS on the target: formed by one (a ping of death) or several messages (stateful attacks),
- cross-script attacks: against vulnerable web servers embedded in the targeted SIP products,
- database injection: against vulnerable databases using these malformed messages in accounting and statistics,
- remote eavesdropping on the targeted phone.

Our contribution is not a standalone system but a complementary checking module that can be incorporated in proposed misuse-detection systems [66, 36, 30] because these systems lack important notions like the detection of "Stateful attacks". Considering SIP exploits, we distinguish three types of signatures:

1. A Type-1 signature is formed by a single malformed header.
2. A Type-2 signature is formed by multiple malformed headers in one message.
3. A Type-3 signature is formed by multiple malformed SIP messages.

We use a rule-based approach where a rule is defined by:

- a predicate: an event matching condition (e.g. regular expression),
- a list of actions: to be performed one time the predicate is true,
- a Boolean context: to dynamically activate or deactivate the rule.

The system has to support a bottom layer intended to monitor the state of network and transport protocols because they constitute an attack vector against the upper services and applications. At the same time, it has to serve a top layer intended to check the application behavior of the users (e.g. to compare a SIP trace with the corresponding bill invoices). We depict the overall system in Figure 6.1 with focus on the SIP layer. SIP checking is formed by the parser and the correlator which are two rule matching engines performing event generation and correlation:

- The parser has the SIP message as an input unit. It is responsible of detecting Type-1 and Type-2 signatures. It has two types of rules: the first type matches single malformed headers based on regular expressions and generates an alert if a signature of Type-1 is detected, or an event to be treated by the second type of

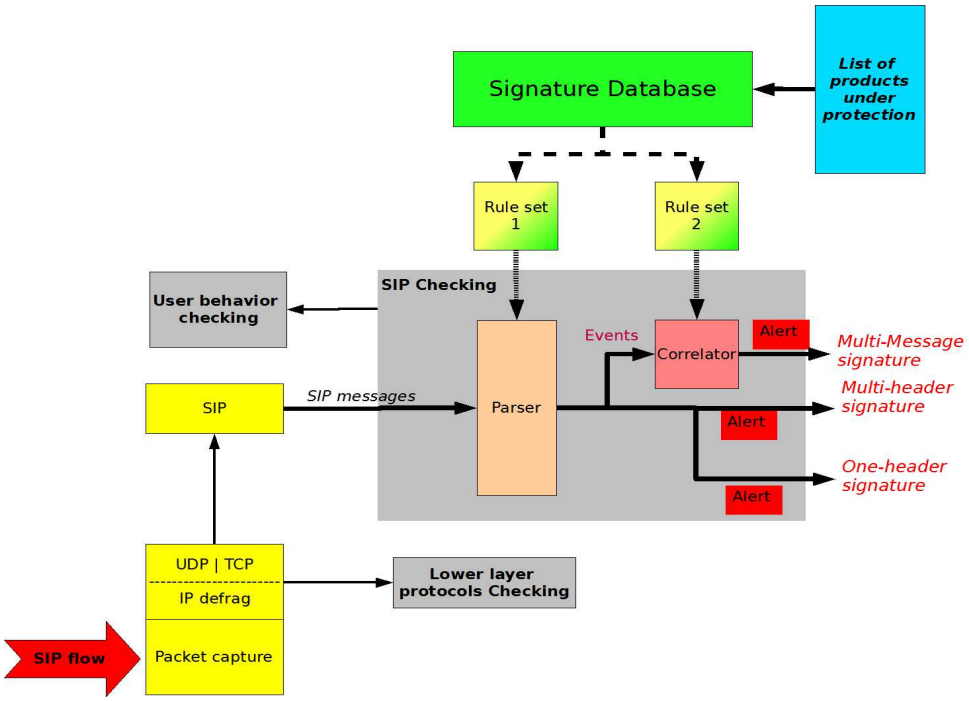


Figure 6.1: SIP-specific NIDS

rules. The second type of rules correlates events indicating malformed headers and generates an alert if a signature of Type-2 is completed. At the end, a special rule is responsible of generating an event about the overall message to the correlator. Events generated to the correlator have attributes. Attributes allow correlation of events belonging to the same call/session (Call-ID value) or events destined to the same target (To value).

- The correlator receives events from the parser (or the absence of events) and generates an alert if a Type-3 signature is detected.

As we need different types of correlation, we adopt the following rule types from the open source Simple Event Correlator (SEC) [99]:

- Single: matches the input event and execute a list of actions (e.g. generate a new input event that can be matched by other rules).
- Suppress: ignores the matching input event.
- Calendar: fires at a specific moment (CRON time format) to execute a list of actions. It is specially intended to schedule the management tasks of our N_IDS.
- SingleWithScript: matches the input event and executes an action list if an external script returns a certain exit value. This type is particularly useful for external checks (e.g. : to check if an IP address is well formed).
- SingleWithSuppress: matches the input event and executes a list of actions but ignores the following matching events for t seconds.

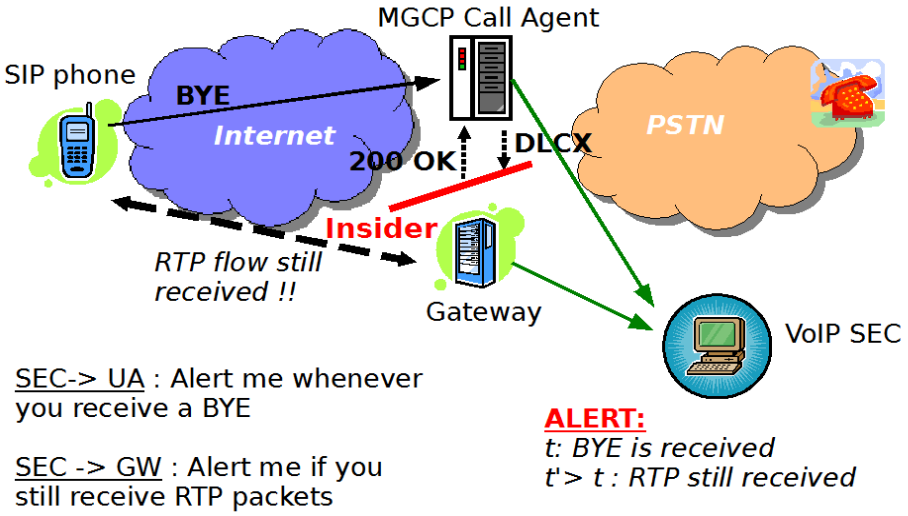


Figure 6.2: A cross-device attack

- Pair: works with a pair of events. On the first event arrival, it executes an action list, ignores the following similar events, waits for t seconds for the arriving of the second event and executes another list of actions.

- PairWithWindow: is slightly different than the previous type: It matches the first event and waits for the second in a time window of t seconds . If the second doesn't arrive within time, it executes an action list, otherwise (the second event arrives within time) it executes another list.

- SingleWithThreshold: counts matching input event in a time window of t seconds, executes a list of actions if a predefined threshold is exceeded, and ignores following matching events until the end of the window. This rule is useful to detect a flood of events.

- SingleWith2Thresholds: counts matching input events in a time window of t1 seconds and executes a list of actions if a threshold "max" is exceeded. In the following t2 seconds, it counts again the matching events. If they drop under other a threshold "min", it executes another list of actions. This rule is useful to recover from a stress condition.

As each product has its own vulnerabilities, we collect up-to-date attack signatures of each product along with their detection rule-sets in one database. We maintain a list of the protected devices in our domain. For example if we couple our N_IDS with a SIP proxy, whenever a user agent registers itself at the proxy, the N_IDS adds the user agent to the list of protected items. The N_IDS have to know the firmware version of the user agent. The is version is normally found in the vendor string of any SIP message sent by the UA, particularly the REGISTER message. This list helps us to select the relevant rule-sets and integrate them in the corresponding rule matching engines of our system in order to gain a better performance. The N_IDS monitors the incoming traffic and detect

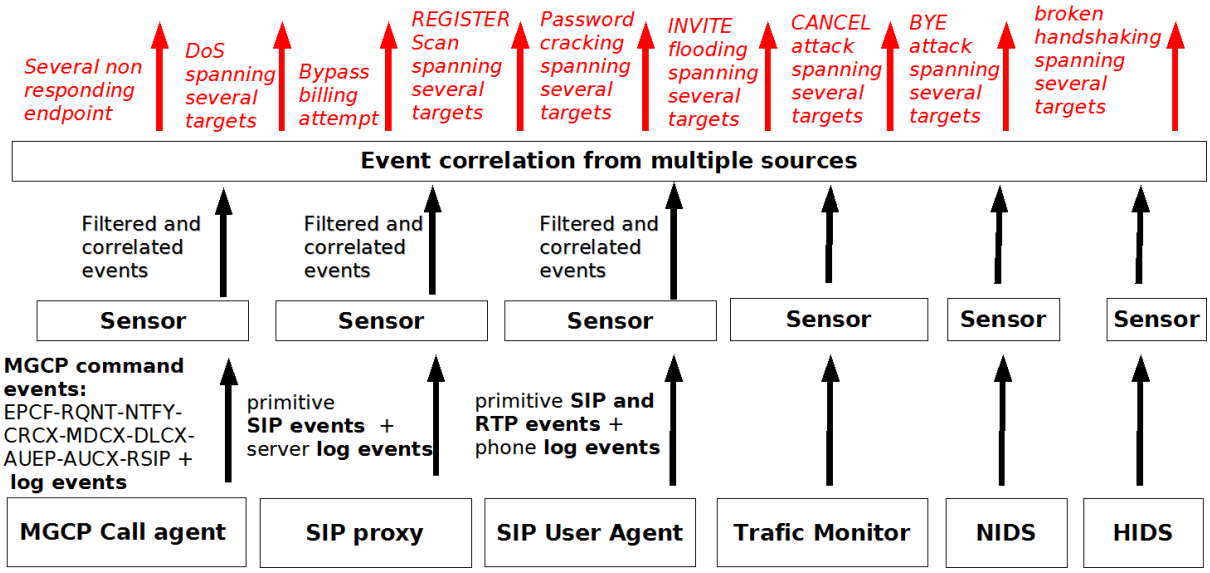


Figure 6.3: VoIP SEC: two-layers event filtering and correlation architecture

the prohibited attack signatures. Also this is of practical usefulness for prevention if the potential UA has an "Inbound-proxy" setting, which would prevent contact attempts except through the protected proxy.

It is clear that the design of our system is not restricted to the detection of malformed messages and exploits. Other attacks such as SIP request flooding and incomplete transactions can be detected as well. However, The system is not able to apply cross-protocol signatures like for instance prematurely session tear down, fake SIP instant messaging and call hijacking signatures similarly to what it was done with SCIDIVE [106]. In order to detect cross protocol attacks, our N_IDS must fulfill two additional conditions:

- the correlator must accept primitive network events from other protocol event recognizers (e.g. RTP);
- the primitive SIP events must include attributes from its lower layer protocols (e.g. IP sender).

6.4 Distributed event correlation

As mentioned before, VoIP systems are distributed, thus it is obvious that studying traces of only one VoIP user agent or server does not give an overall picture of what is happening to the whole domain. In particular, the signaling and the media don't follow the same path. The typical example of such a separation is the MGCP protocol [6] where a call agent is responsible of managing opening and closing connections in multiple gateways across the network. Let's take the following toll fraud/phreaking scenario:

A caller sets up a call towards the PSTN by contacting an MGCP call agent. The call agent chooses one gateway having free end-points, opens a media trunk in the gateway using a MGCP CRCX (CReate ConneXion) message allowing the user to send its RTP

flow across. Everything is normal until the caller sends a SIP BYE to the call agent. The call agent asks the gateway to release the connection by a MGCP DLCX (DeLete ConneXion) message. The caller intercepts the message and sends back a MGCP 200 OK instead of the gateway. The call is no more accounted by the call agent while the gateway is unaware of the DLCX command and still passes the media flow of the caller. This attack is possible because the MGCP protocol doesn't deploy encryption and authentication mechanisms. Such intrusions could not be detected if only the call agent network trace or the end-point network trace is separately monitored but a centralized event correlation approach detects them as shown in figure 6.2.

To generalize this approach and to be able to detect stateful, cross-protocol and cross-device attacks, we propose a VoIP Security Event Correlation (VoIP SEC) solution that is composed from sensors deployed at points of interest and a central unit component. The sensors (host-based and network-based) form a first (local) layer of event filtering and correlation. The central unit represents a second (central) layer of event filtering and correlation and applies security policies in the order of priority required by the administrator. Based on a specified protocol, the central unit remotely manages the sensors to filter the events of interest and send them to it. Our approach allows the detection of a wide range of distributed attacks as the central unit can see "the big picture" of the domain as depicted in Figure 6.3.

6.5 Conclusion and future works

In this chapter, we considered the intrusion detection challenge in VoIP domains and we proposed a holistic solution composed from both host-based components at points of interest (proxy, gateway, user agent) and network-based components at ingress points (inbound/outbound proxies) as well as a central unit for event correlation. We proposed a H_IDS to monitor the state of the underlying system and the user behavior profiles using log files and CDRs for sources of information. We proposed an event-driven N_IDS based on rule matching engines to detect different types of SIP malformed message attack signatures. Later in Chapter 9, we present the implementation of a prototype of our N_IDS coupled with an OpenSER SIP proxy using an open source event correlation tool called SEC (Simple Event Correlator).A solution coupling our VoIP SEC solution with a VoIP honeypot domain in parallel to the production domain is published in [62]. There, lessons learnt over the honeypot domain are applied in the SEC of the real domain.

As VoIP is still in its growing and deploying phase, VoIP intrusion detection is still in a preliminary stage. We have to be ready because we don't fully understand the VoIP threats yet: we don't know what will be the attackers plans and which specific weaknesses they will exploit once VoIP will be of real value to be attacked. We believe that alert analysis can play a major role in discovering attack strategies, prediction of forthcoming attack steps, and finding response/recovery plans. Because intrusions detection and other security systems generate a large quantity of low-level or incomplete security alerts, the domain administrators are overwhelmed by them and therefore unable to perform a comprehensive security diagnostic. Automated alert correlation is the process of filtering the less important alerts, reducing the redundant ones and integrating groups of them having

dependencies or common properties into high level meta-alerts in order to build attack scenarios. We consider applying different techniques of alert correlation (e.g. [101, 74]) in future works.

Part III

Assessment

Chapter 7

VoIP Bot and Botnet: Attack Tool and Attack Scenarios

Sommaire

7.1 Introduction

A bot, short for "robot" describes a program that executes commands when it receives specific input from an external master. A botnet is a large network of devices (usually end-hosts) where bots are installed. These devices called zombies are infected by malware and compromised for the benefit of hackers that can rent them without any knowledge of their real owners. The tenants orientate these bots to send massive SPAM or to launch denial of service attacks. With the growing of the VoIP technology, botnets are expected to extend their activities to attack VoIP services and infrastructures. Values of interest within a VoIP enterprise domain includes signaling and media infrastructure, accounting directories, PBX services (voice mailboxes, gateways) and individual user accounts. Compromising VoIP applications constitutes a bridge to bypass security mechanisms and attack internal networks. Attacks can be transmitted across gateways to integrated networks like mobile and traditional telephony ones. A typical overlay network of a "VoIP botnet" is depicted in Figure 7.1. In this chapter, we present such a framework of VoIP bot and botnet in the context of SIP. Our bots are not equipped with propagation mechanisms so they don't represent a real threat value rather than being used for research purposes namely the assessment of our proposed defense solutions. The remainder of this

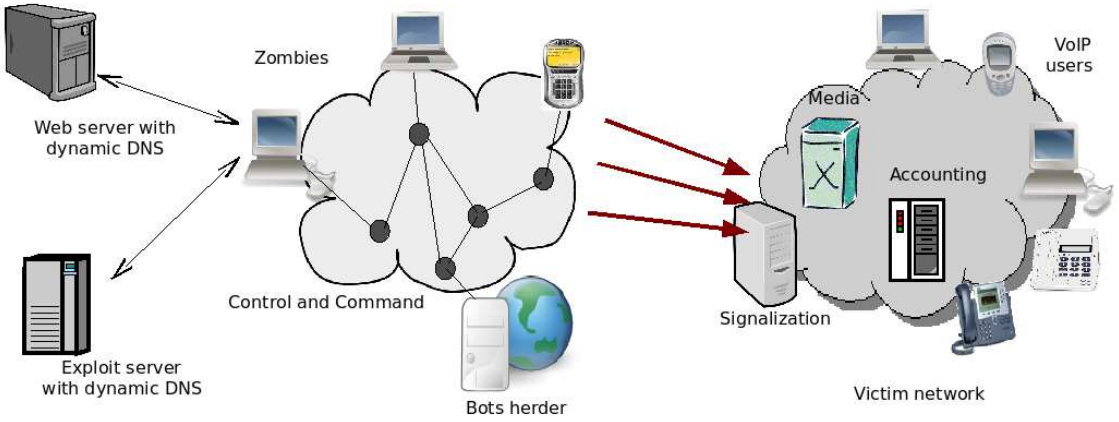


Figure 7.1: VoIP botnet overlay network

chapter is organized as follow: We illustrate our bot architecture in Section 7.2. We enumerate many attack scenarios and we show how they can be performed using our bots in Section 7.3 . We discuss implementation issues in Section 7.4. In section 7.5 we illustrate how works our tool. Section 7.6 concludes the chapter.

7.2 VoIP bot architecture

Our bot architecture is shown in Figure 7.2. Its components are described next:

- The stack of different protocols: provides the bot with an application interface to use these protocols. The SIP stack is responsible for sending and receiving, manufacturing and parsing SIP messages. The RTP stack is responsible for coding and decoding, compressing and expanding, packetizing and depacketizing media flows. Other stacks can be supported as well. For example, the STUN [83] protocol is useful to bypass NAT.

- The communication agent: allows the bot to exchange information and commands with the attacker. Most of the known botnets use IRC (Internet Relay Chat - RFC 1459) or peer-to-peer (P2P) networks for their control and command architecture. IRC is mainly designed for group communication and allows one-to-one communication (private discussion) as well. A channel (or a room) is supported by multiple servers building an application level spanning tree among them and relaying IRC messages between room visitors. P2P refers to a class of systems and applications that employ distributed resources to perform a function in a decentralized manner [58]. Bot masters moved towards P2P networks because of their high degree of anonymity and privacy. For example in Freenet [20], when a peer sends a message, the peer identity is rewritten as the message is relayed among a chain of peers. Another example is the LPWA system [34] which acts as a proxy server and allows

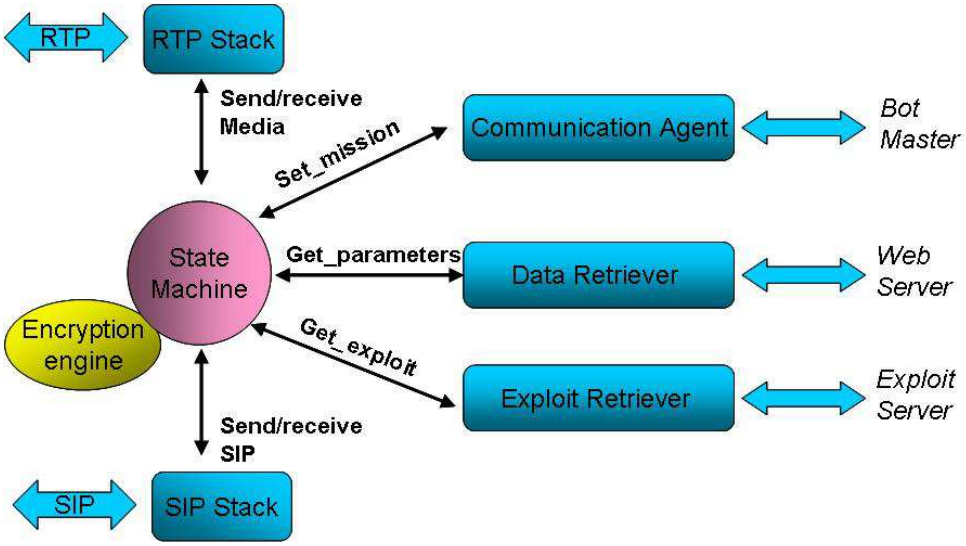


Figure 7.2: VoIP bot architecture

consistent untraceable aliases for clients from servers. A LPWA client opens an account and is recognized upon returning to this account while his true identity is hidden from the server. The Slapper worm [8] which builds a P2P overlay network has advanced features like reliable end to end message delivery, coping with network partitions and reshaping, and anonymous message delivery.

- The data retrieval component: allows the bot to retrieve different kinds of data (e.g. list of VoIP extensions (URIs), advertising audio files, list of default passwords to try, SIP messages to shoot, etc ...) using a data communication protocol (e.g. FTP or HTTP Client). Web servers using a dynamic DNS server (i.e. the DNS changes the IP corresponding to the web server over time) are preferred to avoid being tracked.

- The exploit retrieval component: allows the bot to retrieve specific exploits against vulnerabilities and software flaws in VoIP products. SIP is a strong candidate to become the UFBP (Universal Firewall Bypass Protocol) or the universal payload injector. This is assessed after the discovery of many cross-site scripting (XSS)[15], SQL injection[16], remote DoS on the target (similar to a ping of death)[17] and remote eavesdropping[18] exploits due to SIP vulnerabilities. These exploits are carried by malformed SIP messages to attack SIP servers, embedded web servers and databases in the targeted systems. Some exploits are stateless (consisting on shooting one SIP message) but others are stateful (based on the state machine of the target). Stateful attacks are formed by a series of messages such as the content and the sending time of each message depends on the previous sent message and the corresponding

[15] http://seclists.org/fulldisclosure/2007/Oct/0174.html
[16] http://voipsa.org/pipermail/voipsec_voipsa.org/2007-October/002466.html
[17] http://www.voipsa.org/pipermail/voipsec_voipsa.org/2007-August/002422.html
[18] http://www.voipsa.org/pipermail/voipsec_voipsa.org/2007-August/002424.html

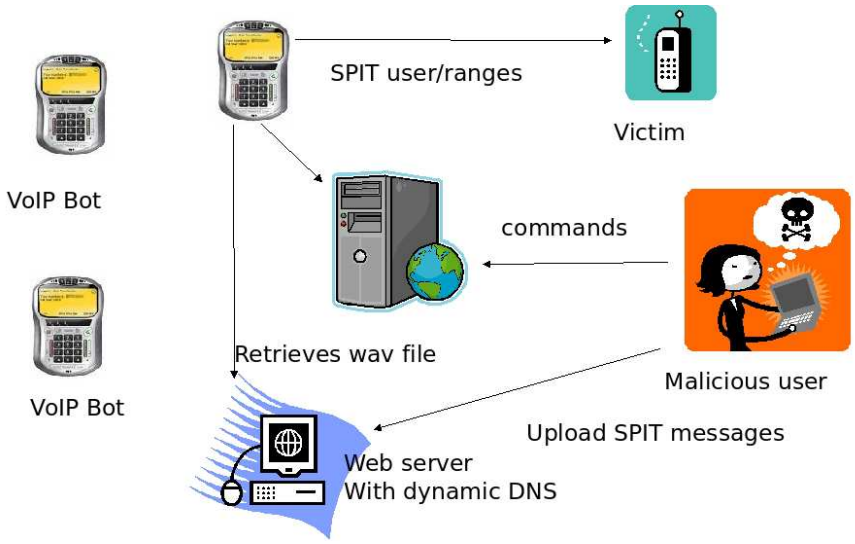


Figure 7.3: Botnet architecture for SPIT

reaction/response of the target to that message. The bot master defines a description syntax to describe stateful exploits and upload them to a server where they can be found by the bot. The bot parses the exploit description and builds a local state machine to perform the attack.

- The encryption engine: enables the bot to create digest authentication from credentials when authentication is required in the process of an attack or an attempt of registration. Typical use of this engine is password cracking and CPU-based flooding against the target authentication procedure.
- The SIP state machine: manages the operations of the bot with respect to the commands issued by the attacker. The `mission` of the bot as set by the attacker drives its behavior upon occurrence of SIP events (i.e. receiving a SIP request `RequestEvent` or a SIP response `ResponseEvent`) and `TimeOut` events. The transition from a state to another is constrained by predicates on a set of global and local variables. For example, when receiving a 200 OK message that belongs to some existing dialog, the bot's next step is based on the `Cseq_method` (which determines the method the 200 OK is in response for) and on the global attack parameters (mission, target IP, target SIP port ...).

Based on this architecture, different attack scenarios are possible as detailed in the next section.

7.3 Attack scenarios

SPIT

In a SPIT scenario, the attacker asks the bot to deliver an audio record to one or more destination URI. Similarly to e-mails being used in SPAM, URIs can be collected by web

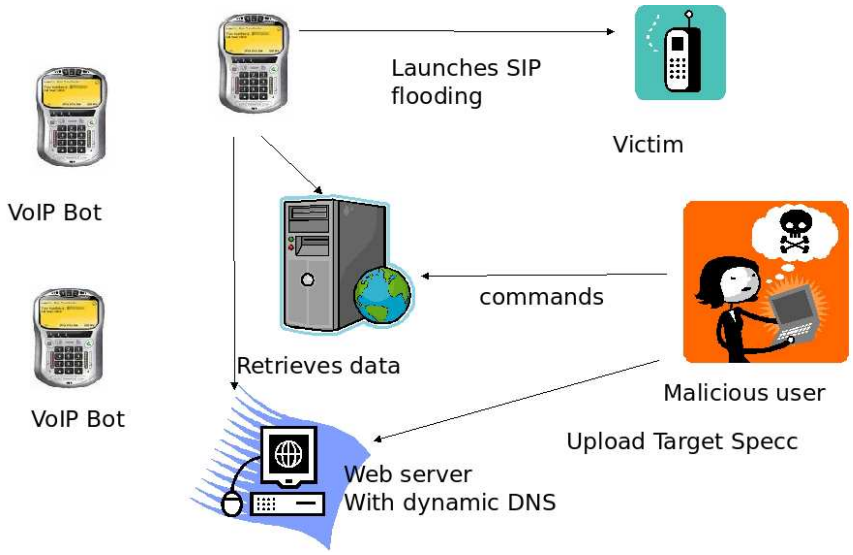


Figure 7.4: Botnet architecture for DoS

crawlers or in result to a VoIP domain enumeration. The bot builds an INVITE request carrying an SDP body given arguments like the destination URI, its IP address, its RTP port, the codec to be used and other media attributes. When the call is answered, the bot retrieves the audio record from the location as supplied by the attacker (e.g. from a URL or a local file in the compromised machine) and stream it to the callee as depicted in Figure 7.3.

Flooding

Flooding attacks can be categorized regarding their destination and their strategy. Whether the attack is destined to a valid URI in the target domain, a non existent URI in the target domain, a URI with an invalid domain or IP address, an invalid URI in another domain, or a valid URI in another domain, different damages are produced. The strategy is related to the nature of messages used during the attack: legitimate, malformed (carrying some exploits), invalid (non compliant to the SIP standard), and spoofed SIP messages (spoofed `From` and `Contact` header). Other attacks are targeted against the authentication process by using messages carrying valid `nonces` and requiring the target to compute digests which overwhelms its CPU capacity [56]. In a flooding scenario (Figure 7.4), the bot starts a thread which sends continuously request messages (INVITE or REGISTER) given the destination, the duration, the timing and the strategy as supplied by the attacker. Distributed flooding attacks can be easily organized by involving and synchronizing a number of bots.

Enumerating

Enumerating is the process of discovering valid SIP extensions (or URIs) in a SIP domain. Enumerating is usually preceded by a port scan to identify existent SIP proxies and user

agents. The standard port used by SIP is 5060. The first step of enumerating is to identify if the SIP service is really running on that port or not and what type and version of server is there. This is done by sending a simple OPTIONS message and interpreting its response. The searched information is usually found in the `Server` or the `User-Agent` header. In an enumerating scenario, the bot retrieves a list of extensions/URIs from a specified location, then it probes them using INVITE, REGISTER or OPTIONS messages. OPTIONS enumerating is preferable since it is stealthier (it does nor ring the phones, nor raise suspicions about the registration process). The bot has to match each request with the response it triggered based on the call-ID and/or transaction identifier. It analyzes the response to determine 1) if the dialed extension exists and if it is registered to the target, 2) exists and it is temporarily unavailable, or 3) doesn't exist at all. The interpretation of responses depends on the target's type and version. For example, if the target carries an `OpenSER sip proxy (version 1.1.1-notls)` fingerprint, the response to an OPTIONS message destined to an extension is interpreted as follow: a "200 OK" means that the extension exists and it is registered, a "404 NOT FOUND" means that the extension is invalid, but a received "100 TRYING" before a final error response means that the extension is valid but not available for the moment.

Cracking

Remote brute force password cracking consists in repeatedly trying guesses for an account's password using INVITE or REGISTER requests. Unchanged default passwords of deployed VoIP platforms make them strongly vulnerable to such attacks. In a cracking scenario (Figure 7.5) , the bot has to discover the registration or the voicemail password for a user name. Note that the user name used in the authentication process is not always the same as the one in the user's URI (which can be obtained by enumerating). The attacker has to know the user name or the voice mail extension before going to a brute force attack. The bot retrieves a list of default passwords corresponding to the target platform. For each guess, it sends a first REGISTER (or INVITE) asking for a challenge. An error response from the target gives him back a nonce. The bot calls its encryption engine to build a new request containing credentials based on the challenge and the temporal nonce. The target's final response decides if the guess was right or not.

Fingerprinting

Remote fingerprinting allows the attacker to identify the type and the version of the SIP target platform. The simplest method consists on sending an OPTIONS message and extraction of the manufacturer string from its response. This process can be fooled if the manufacturer string was intentionally falsified. Smarter fingerprinting schemes as described in [112] or in [3] can be supported as well. In a fingerprinting scenario, the bot is asked to fingerprint one or range of SIP extensions. The bot has to send back its results to the attacker or -in order to not disclose him- to put them in a specified location where he can access them later.

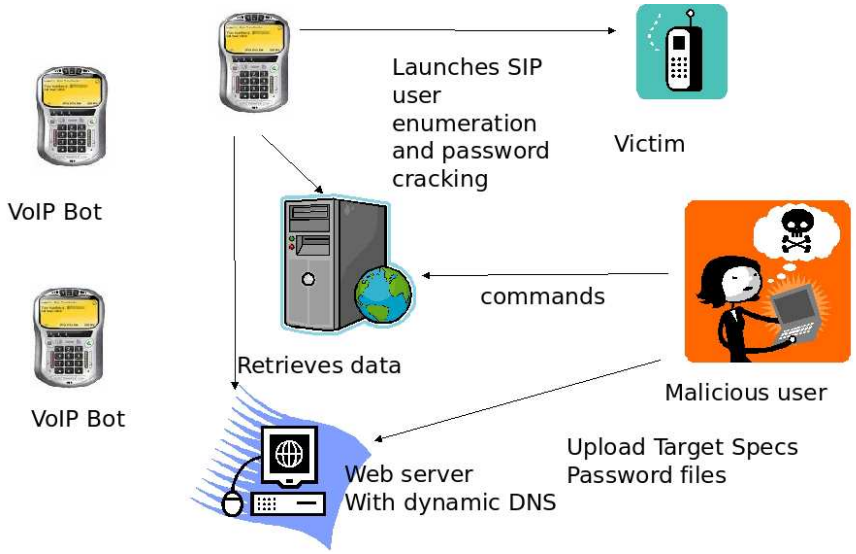


Figure 7.5: Botnet architecture for SIP cracking

Exploiting Specific Vulnerabilities

In an exploit attack scenario (Figure 7.6), the bot is either given the platform of the target or has to discover it by itself (by fingerprinting). The bot connects to an exploit server and retrieves a set of possible exploits corresponding to the target fingerprint. The bot and the exploit server must agree on the description syntax of an exploit (for instance, the messages to be sent and the fields that have to be replaced by the attack parameters). Each exploit should be tagged with some meta-data that the bot can transfer to the attacker giving him the choice. In case of stateful attacks, the bot builds a local attack state machine to execute the attack given local and remote parameters as reported by the attacker.

Interception and Modification

These attacks require an access to the internal network in the VoIP domain. If the attacker succeeds to compromise a machine inside the VoIP domain, many interception, eavesdropping, modification and man in the middle attacks are possible using ARP poisoning techniques. It is not just media and signaling traffic which is targeted, but also supporting protocols like DNS, DHCP, ICMP, and TFTP. In one scenario, if the bot knows the IP address of a phone and the IP address of the outbound SIP proxy, it can fool the phone into thinking that it is the proxy (by sending an RTP packet with the IP address of the proxy and the MAC address of the bot) and vice versa. Like that, the bot plays an MITM role by watching and forwarding messages between the two entities. Because typically no data integrity is deployed in current SIP implementations, the MITM can change an INVITE request before forwarding it. For example, it can redirect the call towards an IVR (Interaction Voice Response) to do Vishing (VoIP Phishing) scam.

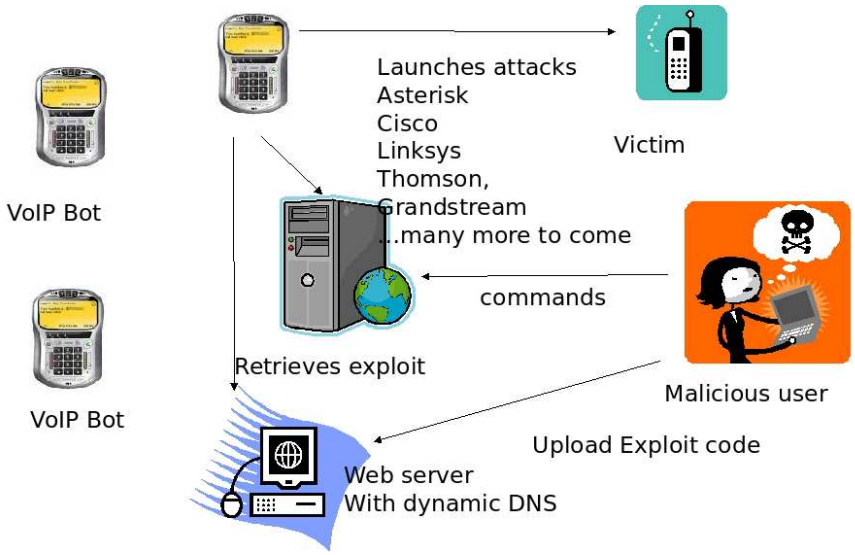


Figure 7.6: Botnet architecture for Takedown

Fraudulent Calls

The increasing financial and informational value of VoIP will attract more Internet hackers into attacking VoIP middlewares, take control over them and execute remote code. Future scenarios are to ask the compromised phone to dial an overtaxed number (similarly to the modem-based dialers in the near past) or to record all calls. Terms like "VoIP dialer" or "VoIP logger" are going to appear in the near future. A one million dollar idea arises immediately: let a large number of victims (one million) dial an overtaxed number resulting on only one additional dollar on the monthly bill of each caller. Most of the victims will not complain about.

Propagation

Propagation models of worms exploiting VoIP and mobile vulnerabilities and using directories in the compromised platforms to spread up are discussed in [31]. By its fingerprinting, exploit retrieving and executing capabilities, our bot constitutes a basis for such a worm using an algorithm like the following one:

```
Forall uri in PhoneBook
 fingerprint = Bot.Fingerprint(uri);
 exploit = Bot.
           retrieve_exploit(fingerprint);
 Bot.send_exploit (exploit, uri);
 Bot.upload_version(uri);
End_Forall
```

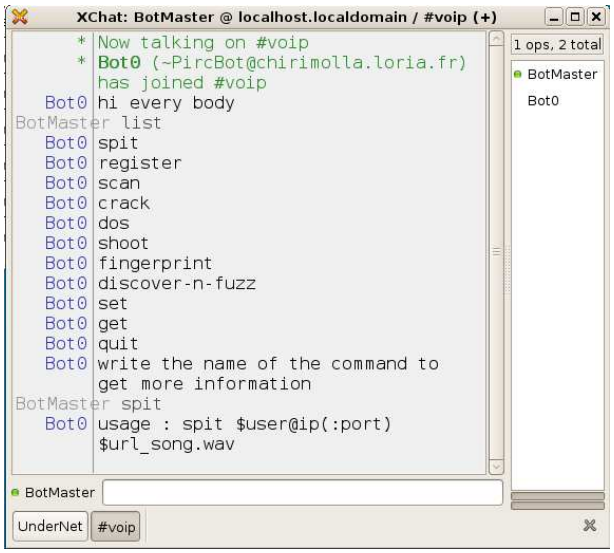


Figure 7.7: Screen view of the bot master

7.4 Implementation issues

We implemented a proof-of-concept worm-free IRC bot based on SIP and RTP using the JAVA language. For SIP we used the Jain SIP library [67]; for RTP we used the JMF library[96] and for IRC we used the PircBot library[19]. Our code is available under open source license at `http://gforge.inria.fr/projects/voipbot/`. The bot is currently able to perform DoS, SPIT, SCAN, CRAK, FINGEPRINT, SHOOT, EXPLOIT and REGISTER functionalities. Moreover, the bot master is able to perform a collective suicide of all the bots. The screenshot of Figure 7.7 shows the IRC client (Xchat[20]) of the bot master upon the connection of one bot.

Deployed on an Intel Pentium 4 CPU 3.40GHz and 2G RAM memory machine running a Linux kernel 2.6.18-1, the bot is able to send around 10,000 messages per second with different call-Ids. The call-ID seed is the number of the bot as set by the attacker so messages from different bots have different Call-Ids. When we used a similar machine with 3G RAM memory (hosting Asterisk and OpenSER) to be the target of a flooding attack (using legitimate messages and non existent URI destination), only one bot is able to raise the target CPU to 100% in case of both Asterisk and OpenSER, and only two bots are able to saturate the bandwidth of a LAN connection (about 12 MBytes/s). Asterisk consumes 25% of the host system memory (i.e. 750 MB) after 100 seconds of attack (i.e. 0.25% raise in memory/s), while OpenSER memory consumption depends on the number of its child processes as configured by its administrator (Each child reserves a 33 MB memory space).

[19] `http://www.jibble.org/javadocs/pircbot/index.html`
[20] `http://www.xchat.org/`

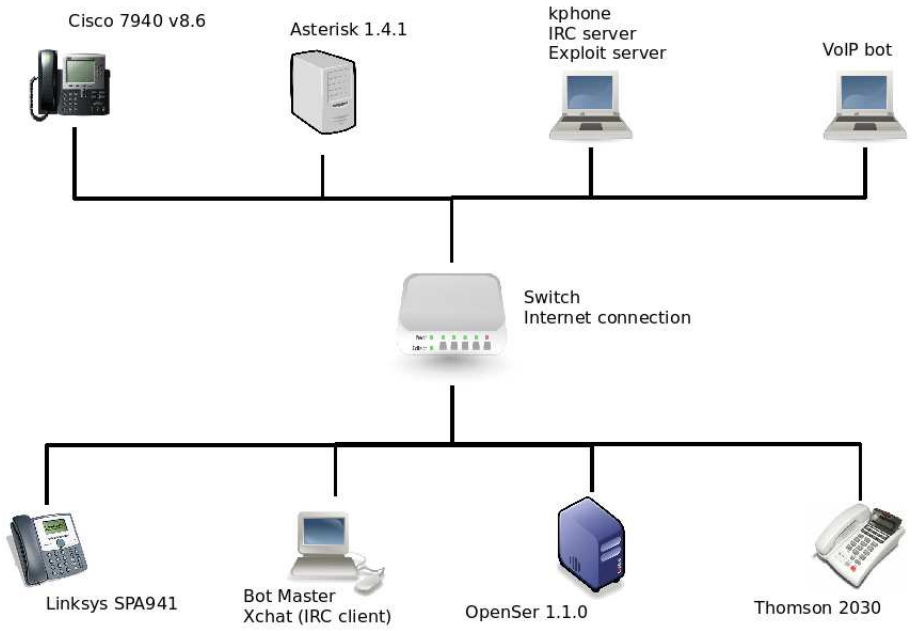


Figure 7.8: Local VoIP bot test-bed

Open issues

We still have some implementation problems. For instance, the code is compiled and executed under JDK version "1.5.0_07" but it is not working with java version 1.6 (perhaps because some of the used libraries are not compatible with java1.6). The media tasks are handled as stand alone processes which can cause some portability issues. Instead, we prefer to handle them by threads using JMF or an alternative library. Currently, the bot supports only PCMU audio format because PCMA (which is the PSTN telephony norm in Europe) is not supported by JMF (For our knowledge). Introducing new audio formats in the code nominates it to the assessment of additional VoIP products like media gateways. All contributions, recommendations, extensions to the project are welcome.

7.5 Tool illustration

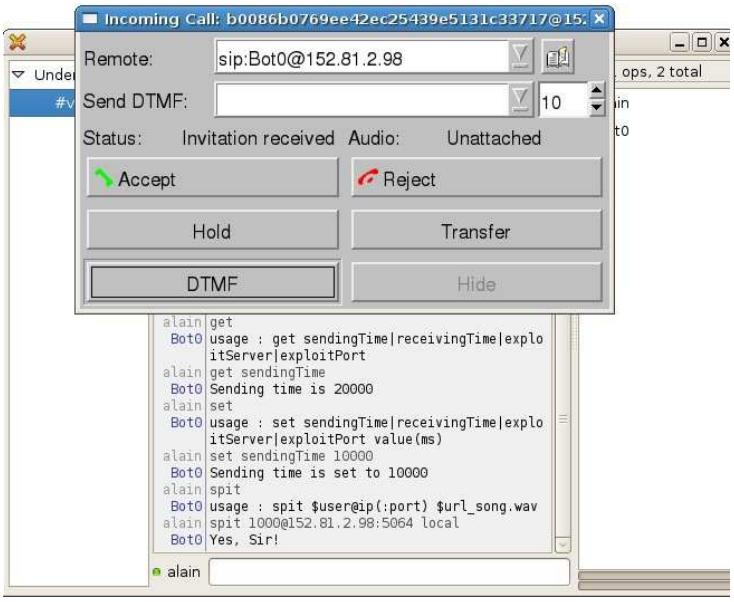
In this section, we illustrate our tool by executing some functional scenarios within a local test-bed. Our test-bed is shown in Figure 7.8.

SPIT

The bot calls a user and delivers an audio (.wav file) to him when it receives this command:

```
spit user@IP_address(:port) url_of_the_audio.wav
```

where the first parameter is the SIP URI of the user and the second parameter is the URL where the bot retrieves the audio file. The bot hangs-up after `sendingTime` ms. The following screenshot shows the SPIT functionality:



The word "local" prompts the bot to send a predefined local file (which is uploaded with the bot in the same archive).

DoS

The bot launches a flood of INVITE messages with different Call-IDs to the target when it receives one of these two commands:

```
dos user@IP_address:port duration_of_the_attack_in_ms
dos IP_address:port duration_of_the_attack_in_ms
```

where the first parameter defines the SIP target and the second parameter defines the duration of the attack. The following screenshot shows the DoS functionality targeting the OpenSER server:

```
alain dos
 Bot0 usage : dos $user@IP(:port) $time_ms
 Bot0 OR : dos $IP(:port) $time_ms
alain dos chirimolla.loria.fr 1000
 Bot0 Attack last 1132 ms
 Bot0 375 messages are sent.
```

SCAN

The bot starts a scan using OPTIONS messages when it receives this command:

```
scan local_file_list_of_usernames IP_address
```

where the first parameter is the path to a file containing the user-names (extensions) to use in the scan and the second parameter indicates the target. The following screenshot shows the SCAN functionality targeting the OpenSER server:

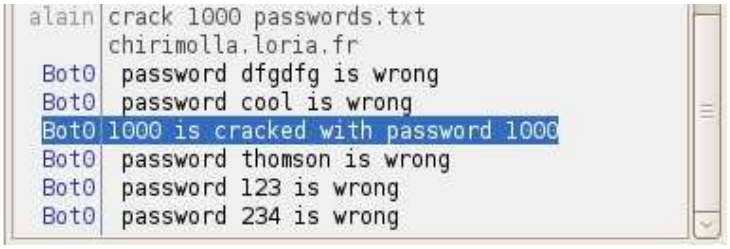
The users.txt file is uploaded with the bot in the same archive.

CRACK

The bots launches a password cracking attack using REGISTER messages when it receives this command:

```
crack username local_file_list_of_passwords IP_address
```

where the first parameter is the user-name of registration, the second parameter is the path to a file containing the passwords to use in the crack and the third parameter indicates the target. The following screenshot shows the CRACK functionality targeting the extension 1000 at the OpenSER server:



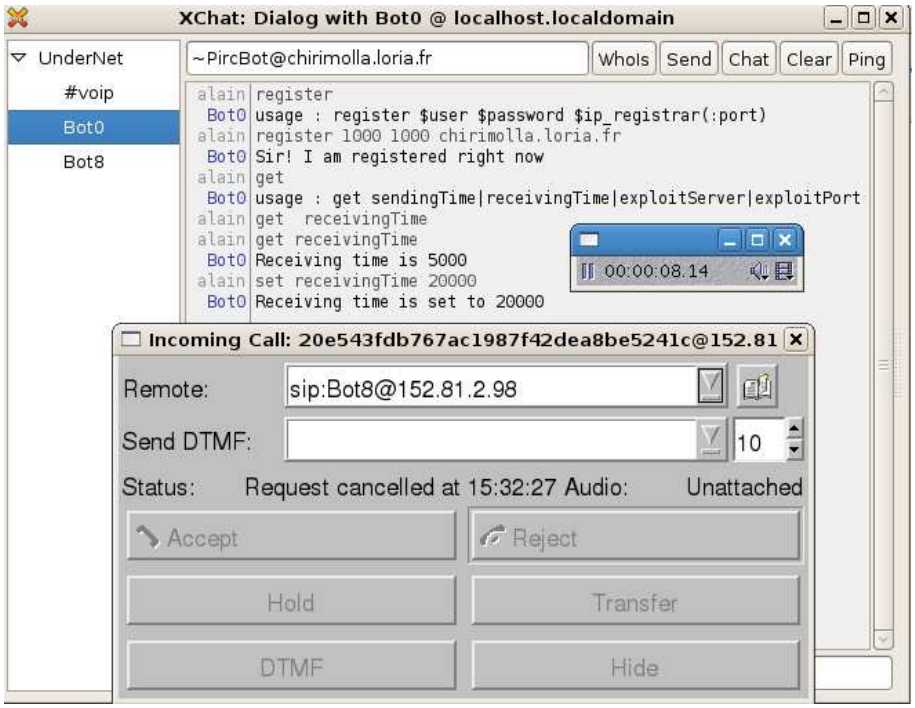
The passwords.txt file is uploaded with the bot in the same archive.

REGISTER

The bot can register to a SIP server using a valid username and password:

```
register username password IP_address_of_the_registrar
```

Once it is registered, the bot is able to receive and playback calls. When it receives a call, it hangs-up after `receivingTime` ms. When a bot registers it self using the user-name and the password of a legal user, it will receive calls instead of him as shown in the following screenshot:



After the Bot0 has registered itself claiming to be the legitimate user at the OpenSER server, the Bot8 makes a call towards Bob. This call has to be routed by the OpenSER. The OpenSer forks the received call towards the two registered locations (Bob's location and the Bot0 one). The first destination that picks up the phone will take the call and a CANCEL request will be sent to the other destination. In our case, Bot0 receives the call and a CANCEL is sent towards Bob.

SHOOT

The bot shoots a SIP message when it receives this command:

```
shoot user@IP_address url
```

where the first parameter is the target and the second parameter is the URL where the bot retrieves the message. This functionality can be used to send malformed SIP messages.

FINGERPRINT

The bot fingerprints the target using an OPTIONS message and returning the value of the "User-Agent" header or the "Server" header of the response:

```
fingerprint user@IP_address:port
```

We ask the bot to fingerprint a Kphone softphone at the extension 1000 (the message can be sent directly to the target or routed by the OpenSER) and the OpenSER server as shown in the following screenshot:

```
alain fingerprint
 Bot8 usage : fingerprint $uri(:port)
alain fingerprint 1000@chirimolla.loria.fr
 Bot8 message is sent
 Bot8 the finger print is kphone/4.2
alain fingerprint 1000@chirimolla.loria.fr:5064
 Bot8 message is sent
 Bot8 the finger print is kphone/4.2
alain fingerprint @chirimolla.loria.fr
 Bot8 message is sent
 Bot8 the finger print is OpenSer (1.1.1-notls (i386
alain fingerprint @chirimolla.loria.fr:5062
 Bot8 message is sent
 Bot8 the finger print is Twinke/1.0
```

When someone tries to fingerprint one of our bots, the bot returns a falsified "Twinke/1.0" fingerprint to deceive him.

DISCOVER_N_FUZZ

Using this command, we ask the bot to discover the implementation of a SIP target (by fingerprinting), connect to an exploit server and query an exploit against that special implementation (We use the word "Fuzz" because most of the exploits we tested are discovered using the KIF SIP stateful fuzzer [2]). We coded an exploit server and installed it in our test-bed. Our exploit server contains exploits against Asterisk 1.4.1, Cisco 7940 v8.6, Linksys SPA941 and Thomson 2030. We show this functionality in the following screenshots:

```
alain get
 Bot0 usage : get sendingTime|receivingTime|exploitServer|exploitPort
alain get exploitServer
 Bot0 ExploitServer is null
alain get exploitPort
 Bot0 ExploitPort is null
alain set exploitServer chirimolla.loria.fr
 Bot0 exploitServer is set to chirimolla.loria.fr
alain set exploitPort 5000
 Bot0 exploitPort is set to 5000
alain discover_n_fuzz
 Bot0 usage : discover_n_fuzz $user@ip(:port) ($choice)
      ($fingerprint)
alain discover_n_fuzz 1000@chirimolla.loria.fr 0
 Bot0 the finger print is kphone/4.2
 Bot0 Connecting to ExploitServer ...
 Bot0 Sending request : kphone/4.2 1000 chirimolla.loria.fr 5062
      Bot0 152.81.2.98 5062
 Bot0 Sorry, we dont have vulnerabilities for this phone
alain discover_n_fuzz 1000@chirimolla.loria.fr 0 thomson
 Bot0 Connecting to ExploitServer ...
 Bot0 Sending request : thomson 1000 chirimolla.loria.fr 5062 Bot0
      152.81.2.98 5062
 Bot0 maximal number_of_exploits = 3
 Bot0 plz re-ask with the chosen number as argument in the end
```

```
alain discover_n_fuzz 1000@chirimolla.loria.fr:5064 1 cisco
 Bot0 Connecting to ExploitServer ...
 Bot0 Sending request : cisco 1000 chirimolla.loria.fr 5064 Bot0
      152.81.2.98 5062
 Bot0 packet nb 3 is shooted
 Bot0 packet nb 2 is shooted
 Bot0 packet nb 1 is shooted
 Bot0 Attack is delivered to the destination
alain discover_n_fuzz 1000@chirimolla.loria.fr:5064 2 cisco
 Bot0 Connecting to ExploitServer ...
 Bot0 Sending request : cisco 1000 chirimolla.loria.fr 5064 Bot0
      152.81.2.98 5062
 Bot0 packet nb 6 is shooted
 Bot0 packet nb 5 is shooted
 Bot0 packet nb 4 is shooted
 Bot0 packet nb 3 is shooted
 Bot0 packet nb 2 is shooted
 Bot0 packet nb 1 is shooted
 Bot0 Attack is delivered to the destination
```

After setting the exploit server and port, we asked the bot to discover and fuzz the Kphone. The bot sent a negative response since it was unable to found a matching exploit in the exploit server. Then we asked the bot to attack the Thomson phone. The bot informed us that it has found three matching exploits and asked us to reformulates the request by choosing an attack number (We plan to add some meta-data in the future to describe the exploit and its damage giving more insight to the attacker in order to make a choice). Finally, we asked the bot to attack the Cisco phone twice using stateful exploits. The first attack (nb. 1) is composed of three SIP messages. The second one (nb. 2) is composed of six SIP messages. The phone rebooted after each attack.

7.6 Conclusion

In this chapter, we exposed a VoIP bot and botnet framework intended for the assessment of VoIP security solutions. Our VoIP bot is based on SIP/RTP/IRC protocols, connects to web servers, special exploit servers and local files in order to perform different attack scenarios in a remotely controlled and distributed manner. We illustrated the VoIP bot architecture, the attack scenarios and the implementation issues. We showed the currently deployed functionalities of the bot in live action by running typical examples. Our tool is of great helpfulness for the assessment of our contributions. It can be used as an attack tool and equally as a victim that receives the attack. We configured our bot to simulate a certain human behavior when it receives a call.

Chapter 8

SIP Traffic Monitoring Assessment

Sommaire

8.1 Introduction

In this chapter, we describe the assessment of our SIP traffic monitoring system (proposed in Chapter 4). We test the machine learning techniques in terms of detection accuracy and performance identifying attack and non-attack situations and distinguishing different types of attacks. We study the detection parameters and we compare between the two proposed models (The BN model and the SVM model) based on exhaustive offline tests.

Our data-set is composed of real-world traces given by a VoIP service provider. Due to the lack of attack traces, we build a local test-bed having the same architecture as the VoIP service provider domain (the traces are collected at an inbound/outbound OpenSER proxy). The real-world traces are assumed attack-free. The remainder of this chapter is organized as follow: In the next section, we present the tools we have used. Section 8.3 describes our data-set. In Section 8.4 we present our SVM-specific experiments. We compare the BN model to the SVM model in Section 8.5. Section 8.6 concludes the chapter.

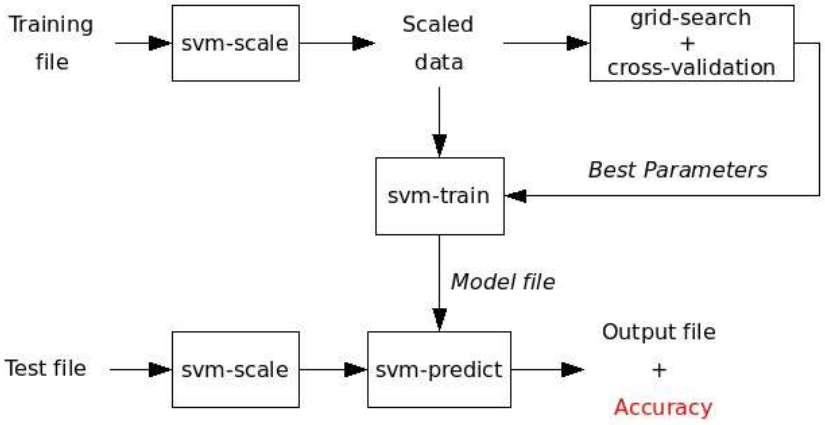


Figure 8.1: SVM flow chart

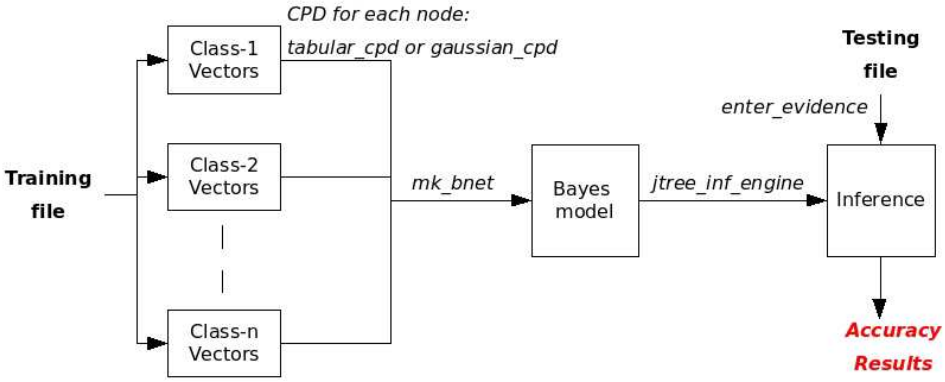


Figure 8.2: BN flow chart

8.2 Tooling

For our SVM model experiments, we use the popular LibSVM tool [18] which implements several algorithms for classification and regression. In addition, this tool supports multi-classification and probability estimates (so that a test vector x_i seems to be pertained to a class i with a probability p_i). The tool also provides support for one class SVM. LibSVM is bound to several other tools such as a grid search and cross validation algorithm. The algorithm searches for the SVM parameters giving the best accuracy. The cross validation for a set of parameters consists on dividing the data into n subsets, then for i going from 1 until n, training over all the subsets except the subset number i and testing over the subset number i. At the end, we have the accuracy ratio for each subset. The aggregation of these ratios is the accuracy resulting from the selected parameters. Using the LibSVM functions, we show our training/testing procedures for classification in Figure 8.1.

For our BN model experiments, we use the open source Bayes Net Toolbox for MATLAB [60]. BNT supports many types of conditional probability distributions including Gaussian, categorical and soft evidence. It supports exact and approximate inference (junction tree, belief propagation), parameter and structure learning, static and dynamic

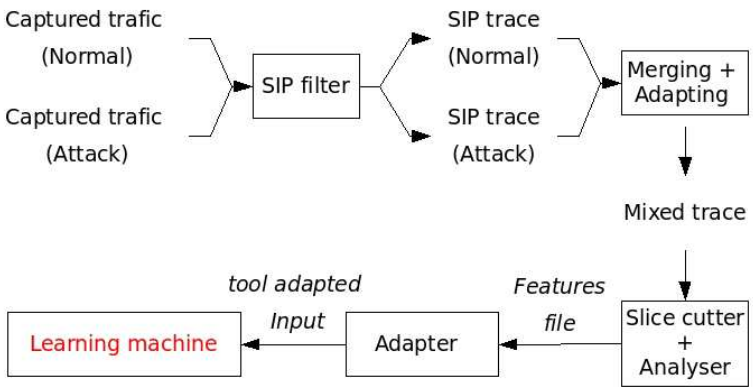


Figure 8.3: Data pre-processing flow chart

models. BNT is widely used in teaching and research (e.g. [33, 86, 116]). Using the BNT functions, we show our training/testing procedure for classification in Figure 8.2.

The training and testing files contains vectors representing traffic slots. Each vector contains the values for a set of predefined features. The analyzer is responsible of cutting up the input trace into slots, evaluating the features for each slot and putting them in an appropriate format to the tool (BNT or LibSVM). Our analyzer tool is developed using the JAVA Jain SIP API [67]. We work with mixed traces composed of normal traces where attacks are injected. This process is shown in Figure 8.3. In supervised training, the vectors are annotated as attack or normal based on the attack start and end times.

8.3 Data-set

8.3.1 "Normal" Data Set

The real-world trace represents a capture of two days traffic at an ingress OpenSER SIP proxy of a VoIP provider. We perform an analysis on the general characteristics of the trace by dividing it into bins of four hours duration and calculating statistics over these bins. In the four charts of Figure 8.4 are sketched the shapes of four statistics: the type of message distribution, the type of request Distribution, the type of response distribution and the inter-arrival timing distribution .

The type of messages distribution reveals that the call setup traffic (SDP requests) forms a relatively small fraction of the overall traffic. Some empty SIP messages are used as Ping or Keep alive so we class them under management messages. Some other messages are malformed (they throw parsing exceptions at the level of the used Jain-SIP parser). If we consider the distribution of the SIP requests, we can note the following:

- The two main components of the traffic are the OPTIONS messages in the first place and then the REGISTER messages.
- Some methods are absent from the capture such as MESSAGE, PRACK and UPDATE.

- Some methods like NOTIFY have constant statistics over all the periods of the day which can be explained by the existence of some SIP devices that remain always connected and periodically send notifications.

- The three main components of the call signaling (INVITE, BYE and ACK) have practically constant ratios over all the day, with an average ratio $\#INVITE/\#BYE = 2.15$ and $\#INVITE/\#ACK = 0.92$. While in ideal situations (successful calls setup) these two ratios are expected to be 1, the value of $\#INVITE/\#BYE$ is explained by failed calls, retransmission, re-INVITE and forking, the value of $\#INVITE/\#ACK$ is mainly caused by the retransmission of acknowledgments.

The type of response distribution is dominated by the 2nd family of responses (most of them belong to OPTIONS and REGISTER transactions). The 3xx, 5xx and 6xx response families are very rare while the informational responses (1xx) follow the INVITE messages because they are exclusively used in the INVITE transactions (the average ratio $\#INVITE/\#1xx = 0.59$ is explained by the fact that a call probably regroups one 100 Trying and one 180 Ringing therefore two 1xx responses).

The average inter-request arrival and the average inter-response arrival seems to be constant over all the periods (about 20 ms). The average inter-SDP (requests carrying SDP bodies) arrival moves inversely to the quadruple (INVITE-BYE-ACK-1xx) curve, it reaches 3s in quiet hours and decreases to 0.5s in rush hours. Note that these values don't represent the real traffic arrival because we involve messages arriving to and departing from the proxy.

8.3.2 The test-bed

The test-bed we use to generate the attack traffic consists of one OpenSER server machine and three other machines: the first machine plays the role of the attacker and uses a number of hacking tools (scanning, flooding, SPIT, VoIP bot). The two other machines play the role of victims where 100 VoIP bots are equally distributed and running. Our bots are profiled to simulate a human behavior as explained later in the experiments. All the VoIP bots and some hardphones are registered to the OpenSER server as belonging to the same domain. The traces of the attacks performed by the attacker machine are collected at the OpenSER server.

At this stage we are ready to conduct two sets of experiments: The first set aims to validate our monitoring approach and to study the different detection parameters. This set is performed using the SVM classifier. The second set compares the BN model to the SVM model and highlights the weakness and the stringiness of each model.

8.4 SVM-specific experiments

All experiments are done in a machine which has an Intel Pentium 4 CPU 3.40GHz and 2G byte of RAM memory running a Linux kernel 2.6.18-1. In term of performance, the SVM prediction time is very small: experiments show that a file containing 2449

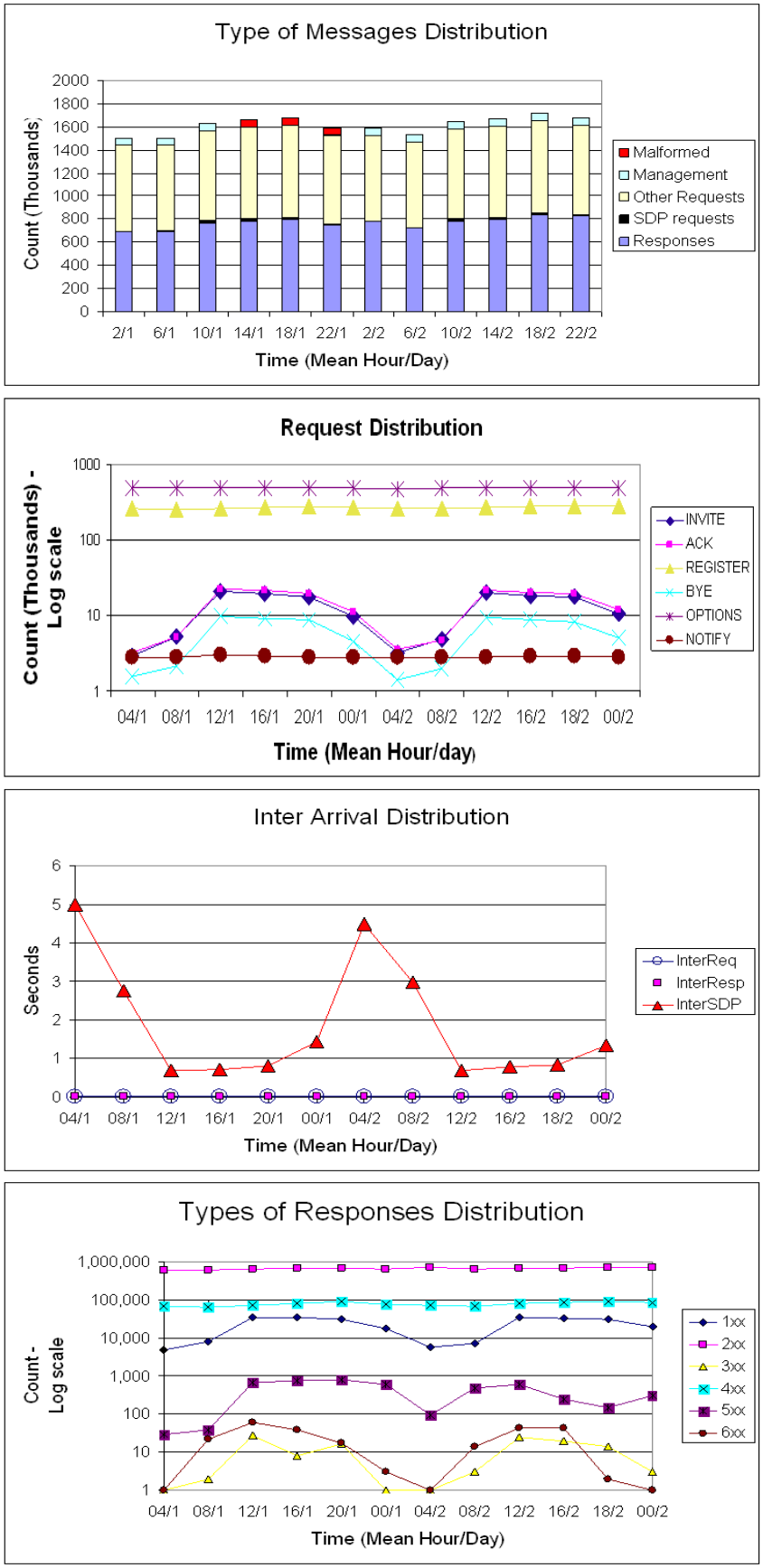


Figure 8.4: Overall statistics over real-world traces

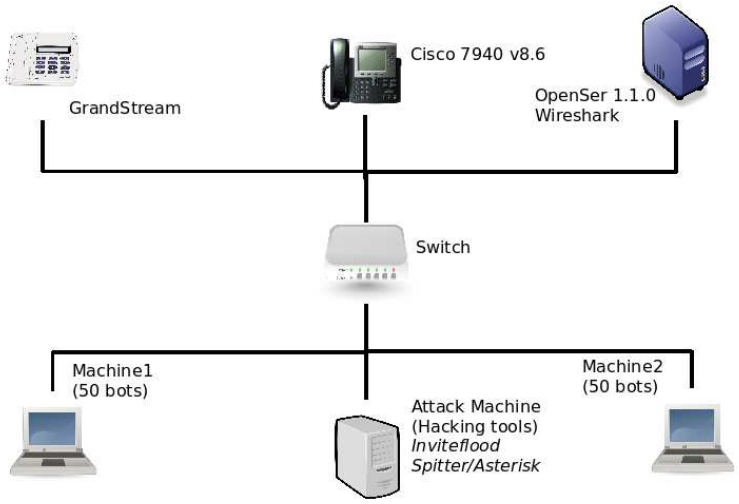


Figure 8.5: Test-bed of attack generation

slices/vectors of 38 features takes between 196 and 994 ms (depending on the used SVM kernel). Count-related type of window is used along these experiments.

Coherence test

The question we address here is how self-similar and consistent are the normal traces. For example, is the traffic from 22:00 to 02:00 in a day similar to the traffic of the same period in another day? To test the `coherence` between two given traces, we use the following procedure: the analyzer extracts the vectors from each trace. The vectors are then labeled with respect to the origin trace and scaled. We make a 2-fold test over all the vectors. In a 2-fold test, we randomly divide the set of the vectors into two parts, we train over the first part and we test over the second part. As long as the SVM can not distinguish between the two traces, they are tagged to the same class. This process is repeated 10 times and results are averaged. We define the coherence to be inversely proportional to the resulting accuracy of the 2-fold test. In Table 8.1, we summarize some results:

Table 8.1: Coherence test for two successive days

Day 1	06-10	10-14	14-18	18-22
Day 2	06-10	10-14	14-18	18-22
Accuracy(%)	55.91	53.72	52.83	56.90

We test the coherence of a period of a day with respect to other periods of the same day. In Table 8.2, we show the results of the same procedure for the period 02-06 of Day 1 compared to the other periods of the same day.

In conclusion, the SVM machine is not able to label 50% of the vectors in their correct classes while proceeding with the same period of two successive days and 40% of the

Table 8.2: Coherence test for different periods of the same day

Day 1	02-06	02-06	02-06	02-06	02-06
Day 1	06-10	10-14	14-18	18-22	22-02
Accuracy(%)	51.82	62.79	63.72	63.76	60.80

vectors proceeding with different periods of the same day. Also the second table reveals that the period 02-06 is more coherent with the neighboring periods (06-10 and 22-02) than with the other periods of the day. We can say that the coherence of the data is highly acceptable.

Multi-classification experiment

In this experiment, we test the ability of the SVM machine to distinguish between different classes of traffic: for instance traces coming form different VoIP platforms. We built a group of four traces, each representing a different traffic class : normal traffic, a burst of INVITE flooding attack, a trace generated by the KIF ([2]) test-bed, and a trace generated by an unknown test-bed. The traces are shown in Table 8.3.

We fix the size of the slot to 30 messages. After analysis, a 2-fold training/testing cross test is performed over all the respectively labeled vectors (2449 vectors). The test accuracy is defined as the percentage of the correctly classified vectors over all the vectors.

In results, when the RBF (Radial Basis Function) kernel is used with default parameters (C=1 and $\gamma = 1/38$), the accuracy is 98.24%.

Table 8.3: Multi-classification of a SIP traffic data-set

Trace	**Normal**	**DoS**	**KIF**	**Unknown**
SIP pkts	57960	6076	2305	7033
Duration	8.6(min)	3.1(min)	50.9 (min)	83.7(day)

Lee et. al. propose an innovative multi-step multi-class intrusion detection scheme based on one-class SVM [54]. Their system is able to detect novel attacks, and to provide incremental update and extension of new classes. We consider such an approach for multi-classification of VoIP attacks in the future.

Comparison between different kernel experiments

The RBF kernel is a reasonable first choice if one can assume that the classes are not linearly separable and because it has few numerical settings (small number of parameters, exponential function bounded between 0 and 1). On the other hand, linear and RBF kernels have comparable performance if the number of features is significantly higher than the number of instances or if both are too large [18]. Therefore, we test all the kernels with

their default parameters over our data-set (Table 8.3). The accuracy (already defined as the percentage of correctly classified vectors over all the vectors) of a 2-fold cross test and the machine prediction time are shown in Table 8.4.

Table 8.4: Testing results for different kernels

Kernel	**Parameters**	**Accuracy(%)**	**Time(ms)**
Linear	$C = 1$	99.79	196
Polynomial	$C = 1$; $\gamma = 1/38$; $r = 0; d = 3$	79.09	570
Sigmoid	$C = 1$; $\gamma = 1/38$; $r = 0$	93.83	994
RBF	$C = 1$; $\gamma = 1/38$	98.24	668
Linear	$C = 2$	99.83	157
RBF	$C = 2$; $\gamma = 0.5$	99.83	294

The last two lines of the table represent the RBF and linear kernels after a search for best parameters. The machine prediction time is averaged over 10 runs and shown for comparison purposes. The RBF and linear kernels have clearly the best accuracy and machine time.

Slot size experiment

The size of the slot to be analyzed is an important parameter for the attack detection and the real-time performance of the monitoring system. The size of the slot can be prefixed or varied with respect to other monitoring parameters. In this experiment, we report the accuracy of the classifier while changing the size of the slot to be analyzed. The results shown in Table 8.5 are obtained using a 5-fold cross test with the RBF kernel and its default parameters. The analysis time of the overall trace (Table 8.3) is only given for comparison purposes. As expected, the accuracy improves with larger window sizes, while incurring an increased analysis time.

Table 8.5: Testing results for different slot sizes

Window size	5	15	30	60	90	120	150
Accuracy (%)	95.4	99.32	99.30	99.67	99.63	100	100
Analysis Time (min)	1.12	2.40	2.56	4.31	6.39	7.42	8.51

Feature selection

The 38 features are chosen based on our domain specific knowledge and experience, but other features might also be relevant (e.g. transaction-based features like the transaction final state or registration-based features like the period of registration). The selection of highly relevant features is essential in our approach in terms of performance and accuracy. In these experiments, we rank the features with respect to their relevance. We can thus reduce the number of features by gradually excluding the less important features from the analysis.

In Table 8.6, we give the results of a preliminary experiment, where we exclude the groups of features one per each column in the following order: the distribution of the final state of dialogs (7 features) is excluded first, then the distribution of the SIP requests, the distribution of the SIP responses, and finally the Call-Id based statistics. The last column of the table represents the remaining group which is the general statistics. We perform a 5-fold cross test over the data- set (Table 8.3) using the RBF kernel and its default parameters. The test accuracy is the percentage of correctly classified vectors over all the vectors in the test data set.

Table 8.6: Results for different features sets

# of features	38	31	18	12	7
Accuracy (%)	99.30	99.39	98.90	98.65	89.22
Machine Time (s)	1.85	1.59	1.42	1.28	0.57

Although we notice a sudden jump between 12 and 7 features, the accuracy is not strictly decreasing as a function of the number of features being used. It is thus reasonable to inquire on the dependencies among the features.

Detection of flooding attacks

In these experiments, we use the Inviteflood tool [27] to launch SIP flooding attacks. We use an INVITE flooding with an invalid domain name in the request URI. We generate five attacks at five different rates, where each attack lasts for one minute. We assume that one machine of the real world platform is performing the attack and we inject each attack trace into a normal trace of two hours duration (background traffic $\sim$ 150 message/s, ratio attack_period/ normal_period = 1/120) , five minutes after its start. Each mixed trace is then cut into slots of 30 messages size, transformed into vectors and labeled properly by the analyzer (positively along the period of attack and negatively in all the remaining time).

In one experiment, we train the SVM machine with a mixed trace composed by a flooding at 100 INVITE/s and a normal trace. This means that we consider 100 INVITE/s as a critical rate that must launch an alarm. In the results, no false positives are held (all the normal traffic is detected as normal). We define the detection accuracy as the percentage of vectors correctly classified as attack over all the vectors of the attack period. As shown in the example of Figure 8.6 (for simplification and clarity sake a slot is sized to only three packets), the estimated probability of the attack trace is the average of the

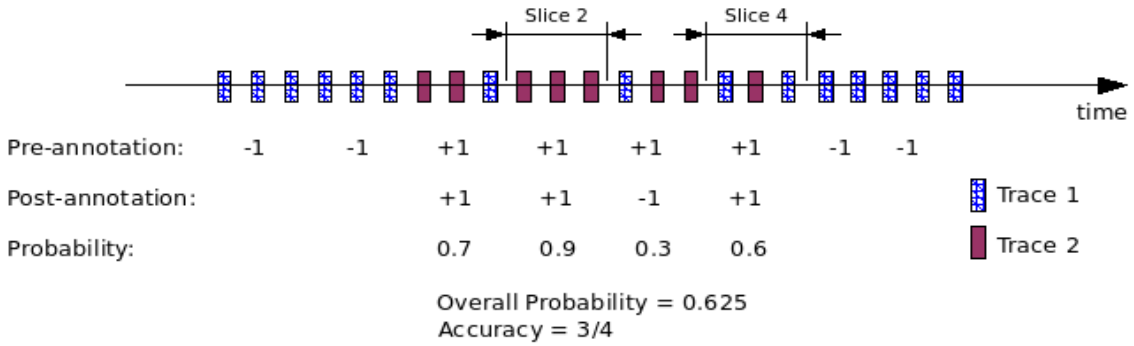


Figure 8.6: Attack detection in a mixed trace

Table 8.7: Attack probability estimation for different flooding rates

Flooding Rate (INVITE/s)	0.5	1	10	100	1000
Detection Accuracy-1 (%)	0	0	5.47	67.57	97.36
Detection Accuracy-2 (%)	0	1.48	30.13	88.82	98.24
Pr(Normal)	0.96	0.95	0.73	0.24	0.07
Pr(Attack)	0.04	0.05	0.27	0.76	0.93

estimated probabilities of its slots as given by the SVM machine. The results are in Table 8.7: the detection accuracy-1 is obtained without a search for the best parameters (Default parameters : $C = 1$, $\gamma = 1/38$, training accuracy: 90.95), the detection accuracy-2 and the estimated probabilities are obtained after a search for the best parameters ($C = 32$, $\gamma = 0.5$, training accuracy: 93.93).

Even though stealthy attacks cannot to be detected, the results show a promising opportunity to fine-tune the defensive solution. The threshold rate can be learnt by a dual trace : the ongoing normal/daily traffic and a stress condition where the server was troubleshooted or was noticed to be under-operating. In this way, SVM are promising for an adaptive online monitoring solution against flooding attacks.

Detection of SPIT attacks

To generate SPIT calls, we used a well known tool which is the Spitter/Asterisk tool [27]. The Spitter is able to generate call instances described in a ".call" file using the open source Asterisk PBX. The rate of simultaneous concurrent calls can also be specified as an option of the attack. We profiled our VoIP bots to receive the SPIT calls. Once an INVITE is received, the bot chooses randomly between three different responses :

- the first choice is to ring for a random time interval between one and six seconds and then to pick up the phone. This emulates two cases : a voice mail which is dumping a message or a human which is responding. The bot then listens during a random time between five and ten seconds and hangs up,
- the second choice is to respond with 'Busy',

Table 8.8: Detection of partial SPIT with different intensities

# of Concurrent Calls	True Positives (%)	True Negatives (%)
RBF; C= 1; $\gamma = 1/38$; Training accuracy = 99.0249		
1	0 (0/3697)	100
10	1.30 (10/766)	
50	10.01 (62/619)	
100	18.31 (102/557)	
Linear ; C=1 ; Training accuracy = 99.0197		
1	0 (0/3697)	100
10	2.09 (16/766)	
50	10.66 (66/619)	
100	19.39 (108/557)	

- the last choice is to ring for some time and then to send a redirection response informing the caller that the call has to be redirected to another destination (destination that we assume to not be served by our proxy). Other scenarios like forking (by the proxy) or transferring (by the bot) are also supported.

We generate two forms of attack: Partial SPIT and Full SPIT. In the former, the Spitter targets the proxy with 100 different destinations and among them only 10 are valid (representing registered bots). In this case the hit rate is just 10%. This emulates the real world scenario where attackers are blindly trying a list of extensions. Such an attack is similar to an INVITE scanning resulting in a high number of "user not found" responses. In the latter, the Spitter targets the proxy with 100 valid destinations (representing the 100 registered bots). We assume that attackers knew already the good extensions (either through a previous enumeration attack or from a web crawler), the hit rate is 100%.

In the partial SPIT experiment, we send 4 successive campaigns with different intensities (number of concurrent calls). For example, if the number of concurrent calls is set to 1, the Spitter does not start a dialog before the previous dialog has been finished. If the number of concurrent calls is set to 10, 10 dialogs are initiated at the same time and only when a dialog has been finished, a new dialog is started. The 4 resulting traces (duration about 2 minutes each) are injected in 4 normal traces (duration of two hours each). The traces are then cut into slots of 30 messages size, transformed into vectors and labeled by the analyzer (annotated positively in the attack period and negatively in all the remaining duration). We train the SVM machine with a mixed trace composed by a partial SPIT of 50 concurrent calls intensity and a normal trace. The SVM prediction results are shown in Table 8.8. The true positives rate is the percentage of vectors correctly classified as attack over all the vectors of the attack period. The true negatives rate is the percentage of vectors correctly classified as normal over all the vectors of the normal period.

These results should be considered under the larger umbrella of event correlation. For instance, in the example with 10 concurrent calls intensity:

- Most of the two hours traffic is normal and is correctly detected (47436 slices).

Table 8.9: Detection of full SPIT with different intensities

# of Concurrent calls	1	10	50	100
RBF; C= 1; $\gamma = 1/8$; Training accuracy = 98.9057				
True Positives	0.03 2/7015	3.05 15/492	12.18 85/698	23.41 184/786
True Negatives	100			

- 16 out of the 766 slices that compose the attack traffic are detected. This means that we have ten correct events in a period of two minutes, because the detection of one slice is highly relevant to all ongoing traffic around this slice.

In addition, the attacks are partial since they target a small fraction of the users of the VoIP server (more than 3000 users are identified in the normal trace). We agree that a stealthy SPIT of the magnitude of one concurrent call is never detected, but in the case of hundred concurrent calls, one over five positives is successfully detected when training is done using a half of this intensity attack.

With the help of a set of deterministic event correlation rules, our online monitoring system is able to detect the attacks efficiently:

Predicate	SPIT intensity
10 distributed positives in a 2 minutes period	Low
Multiple Series of 5 Successive Positives	Medium
Multiple Series of 10 Successive Positives	High

In the full SPIT experiment, the Spitter hits all the bots in 4 successive campaigns with increasing intensity. Results are slightly better than in the partial SPIT experiment (Table 8.9). We notice that the two types of SPIT generate an abnormal traffic at the same level.

8.5 Comparison between the SVM and BN models

We have to notice first that a comparison between the real-time performance of each model is not possible for many reasons: The tools we used are written in different languages (LibSVM is written in C while BNT is written in Matlab), and the tests are run over different platforms (LibSVM is installed on our local machine while BNT is installed on a common MATLAB server in our research center). Theoretically, both classifiers are computationally efficient: support vector machines are efficient because they only deal with inner products among feature vectors, and Naïve Bayesian classification basically involve simple counts.

In the following experiments, we compare the detection accuracy of the two models in three cases: classification Normal/DoS, classification Normal/SPIT and anomaly detection. Chronological windows are used along these experiments. The set of traces we are using is depicted in Table 8.10. We use mixed traces where an attack is injected into a

Table 8.10: Normal and attack traces

Name	Description
N	Normal trace (mean message arrival = 150 messages/s)
F1	2 minutes of flooding at 1 INVITE/s rate
F10	2 minutes of flooding at 10 INVITE/s rate
F100	2 minutes of flooding at 100 INVITE/s rate
F1000	2 minutes of flooding at 1000 INVITE/s rate
FS1	2 minutes of Full SPIT at 1 concurrent calls intensity
FS10	2 minutes of Full SPIT at 10 concurrent calls intensity
FS50	2 minutes of Full SPIT at 50 concurrent calls intensity
FS100	2 minutes of Full SPIT at 100 concurrent calls intensity
PS1	2 minutes of Partial SPIT at 1 concurrent call intensity
PS10	2 minutes of Partial SPIT at 10 concurrent calls intensity
PS50	2 minutes of Partial SPIT at 50 concurrent calls intensity
PS100	2 minutes of Partial SPIT at 100 concurrent calls intensity

four times larger normal traffic after one minute of its start. For example, **N+F1** means an 8 minutes normal trace where **F1** is injected.

Our event correlator is configured with the following parameters:

- *monitoring_window*: is the slot size.
- *alarm_window*: is the size of a sliding window which allows generating an alarm if a fraction α of its size is predicted as attack.

In other words, to generate an alarm at the end of an *alarm_window* period, the number of slots that must be predicted as attack is equal to

$$\alpha \cdot \frac{alarm\_window}{monitoring\_window}$$

8.5.1 Short-term monitoring experiments

We experiment with different mixed traces for training and testing, and different sizes of monitoring window. The results of our selection procedure (SVM + F-score) for this set of experiments are depicted in Table 8.11. To be comparable, the two models (SVM and BN) are both built using this set of features. We set α to 90%, *alarm_window* to 5s and we vary *monitoring_window* as a discrete value between 1 and 5s. A part of our results is shown in Table 8.12. We consider that an attack is successfully detected if at least one alarm is issued during the attack period and no false alarms are issued during the normal period.

8.5.2 Long-term monitoring experiments

We experiment with different mixed traces for training and testing, and different sizes of monitoring window. The results of our selection procedure for this set of experiments are

Table 8.11: Selected features for short-term monitoring

Number	Name	Description
11	NbReceivers	# of different receivers / Total # of Call-IDs
14	NbCALLSET	# of CALLSET/ Total # of Call-ID
20	NbInv	# of INVITE / Total # of requests
4	NbSdp	# of messages carrying SDP / Total # of messages
2	NbReq	# of requests / Total # of messages
3	NbResp	# of responses / Total # of messages
13	NbNOTACALL	# of NOTACALL/ Total # of Call-ID
12	AvMsg	Average # of messages per Call-ID

Table 8.12: Results of short-term monitoring experiments

Training	Testing	Detected by BN	Detected by SVM
N+F1	N+F1	No	No
N+F10	N+F10	Yes	Yes
N+F100	N+F100	Yes	Yes
N+F1000	N+F1000	Yes	Yes
N+F1	N+F100	No	No
N+F100	N+F1	No	No
N+F10	N+F100	Yes	Yes
N+F100	N+F10	Yes	No
N+F1000	N+F100	Yes	Yes
N+F100	N+F1000	Yes	Yes

Table 8.13: Selected features for long term monitoring

Number	Name	Description
16	NbREJECTED	# of REJECTED/ Total # of Call-ID
4	NbSdp	# of messages carrying SDP / Total # of messages
20	NbInv	# of INVITE / Total # of requests
23	NbAck	# of ACK/ Total # of requests
36	Nb4xx	# of Client error responses / Total # of responses
34	Nb2xx	# of Success responses / Total # of responses
7	AvInterSdp	Average inter arrival of messages carrying SDP bodies
35	Nb3xx	# of Redirection responses / Total # of responses
13	NbNOTACALL	# of NOTACALL/ Total # of Call-ID

Table 8.14: Results of long-term monitoring experiments

Training	Testing	Monitoring window	Detected by BN	Detected by SVM
N+PS50	N+PS50	1	No	No
N+PS50	N+PS50	2	No	No
N+PS50	N+PS50	3	No	Yes
N+PS50	N+PS50	4	No	Yes
N+PS50	N+FS50	5	Yes	Yes
N+PS50	N+PS50	6	Yes	Yes
N+PS50	N+PS50	7	Yes	Yes
N+PS50	N+PS50	8	Yes	Yes
N+PS50	N+PS50	9	Yes	Yes
N+PS50	N+PS50	10	Yes	Yes
N+PS50	N+PS50	11	Yes	Yes
N+PS50	N+PS50	12	Yes	Yes
N+FS1	N+FS50	10	No	No
N+FS50	N+FS1	10	No	No
N+FS10	N+FS50	10	Yes	Yes
N+FS50	N+ FS10	10	Yes	No
N+FS100	N+ FS50	10	Yes	No
N+FS50	N+FS100	10	Yes	Yes

depicted in Table 8.13. To be comparable, the two models (SVM and BN) are both built using this set of features.

We set α to 25%, $alarm\_window$ to 50s and we vary $monitoring\_window$ as a discrete value between 1 and 12s. A part of our results is shown in Table 8.14. We consider that an attack is successfully detected if at least one alarm is issued during the attack period and no false alarms are issued during the normal period.

8.5.3 Anomaly detection experiments

We experiment our approach for anomaly detection where only “normal” data are available for training. In our Bayesian approach, training data are used to build Gaussian distributions between the leaf nodes and the **normal** hypothesis at the root node. The **abnormal** hypothesis is represented by a "dummy" state at the root and linked to each leaf by a Gaussian distribution with the following parameters:

- σ is infinity (by choosing a relatively large number)
- μ is the same as μ of the Gaussian law of this leaf under the normal hypothesis.

In our SVM approach, we use one-class SVM with an outliers fraction of 1%. For each approach, we build tow models: one for short term monitoring based on features from Table 8.11 (to detect flooding attacks) and another for long term monitoring based on features from Table 8.13 (to detect partial and full SPIT attacks). A part of our results is depicted in Table 8.15.

Table 8.15: Results of anomaly detection experiments

Training	Testing	Monitoring window	Detected by BN	Detected by one-class SVM
N	N+F1	1	No	No
N	N+F10	1	Yes	No
N	N+F100	1	Yes	Yes
N	N+F1000	1	Yes	Yes
N	N+FS1	10	No	No
N	N+FS10	10	No	No
N	N+FS50	10	Yes	Yes
N	N+FS100	10	Yes	Yes
N	N+PS1	10	No	No
N	N+PS10	10	No	No
N	N+PS50	10	Yes	Yes
N	N+PS100	10	Yes	Yes

To summarize the results, Bayesian networks demonstrated better accuracy both in detecting variants of the same attack and in anomaly detection where only normal data are available for training. SVM demonstrated accuracy in classification where both labeled traces from normal and attack dataset are available for training. Both techniques are suitable for a real time deployment. The monitoring window has to be adjusted with the number and nature of extracted features to support such a deployment.

8.6 Conclusion

In this chapter, we studied the two proposed approaches (Bayesian networks and support vector machines) to monitor SIP traffic for intrusion detection. We presented two sets of experiments based on a data-set composed of real-world data and attacks locally crafted in our test-bed. Focusing on the SVM model, the first set aimed to validate our approach and to show how detection parameters (SVM kernel, slot size, feature selection) can be tuned to achieve a high accuracy. The second set compares the two models in term of normal/flooding classification, normal/SPIT classification and anomaly detection. Results show the high adaptability of our approach and its capacity to detect a series of stealthy attacks specially when coupled with efficient event correlation rules. Experimentations with additional types of attack and several real-world normal/attack data-sets are required to improve the system before going to a production environment.

In the future, we aim to enhance our system in several directions: to adapt to network changes such as the addition or suppression of terminal nodes or servers by system retraining on the fly, to enhance detection of stealthy attacks accuracy by refining our model and set of features, to deploy unsupervised learning techniques for anomaly detection, and finally, to be immune against adversary profiling and manipulation attacks where the attackers try to "poison" the learning machine by injecting gradually a traffic that is slightly different than normal.

Chapter 9

Signature Detection Using The Simple Event Correlator Tool

Sommaire

9.1 Introduction

The Simple Event Correlator (SEC[21] to not confound with our VoIP SEC (Security Event Correlation) approach) is an open source and platform independent event correlation tool. In this chapter, we test SEC for practical convenience in VoIP intrusion detection systems. While we know already that SEC is well adapted to host-based and log files monitoring, we take the initiative to use it in network monitoring and SIP packet inspection. We introduce SEC in section 9.2. We give an example of Asterisk log file monitoring in section 9.3. Furthermore in section 9.4, we present a method to monitor SIP traffic routed by an OpenSER proxy. We conclude the chapter in section 9.5.

9.2 Introduction to SEC: Simple Event Correlator

SEC is a lightweight online monitoring tool initially designed to fill the gap between home-grown and commercial event correlation solutions. SEC already proved its efficiency in

[21] `http://kodu.neti.ee/~risto/sec/`

several domains like network management, intrusion detection systems, system monitoring and fraud detection. The SEC engine uses static rules, accepts input from text streams named pipes and generates output events, log messages or executes shell commands. A SEC-rule contains an event matching condition based on a Perl regular expression, an actions list, and optionally a Boolean context that dynamically allows or not the application of the rule.

9.3 Using SEC to monitor Asterisk log

SEC is very efficient in monitoring log files and it can be used to monitor VoIP systems like Asterisk. Monitoring the Asterisk log file is useful to notice the state of the system, highlight configuration errors and detect a set of attacks. For example, the following rule detects a possible enumeration when successive notices of "No matching peer found" are logged in a short time.

```
type=SingleWithThreshold
ptype=RegExp
pattern=Registration from \'(\"\w*\")?\<(sip:\w+\@\d+.\d+.\d+.\d+)\>\' \
failed for \'(\d+.\d+.\d+.\d+)\' - No matching peer found
desc=$3
action= write - %t: Peer Scanning activity over $3
window=5
thresh=2
```

Still, changing log messages statements from one software version to another requires a parallel revision of the correlation rules. Emerging a standard for writing log messages would be very helpful for a wide deployment of such an approach.

9.4 Using SEC to monitor OpenSER SIP traffic

9.4.1 Setup

Our setup is based on a Unix machine. We wrote a new module to OpenSER called "tosec" whose mission is simply limited to fork each incoming SIP message towards the rule-matching SEC engine. The OpenSER configuration file must be changed to integrate the new module:

```
...
loadmodule "/usr/local/lib/openser/modules/tosec.so"
...
route{
tosec();
...
...
}
```

To correlate responses and not only requests, tosec must be declared in the reply routing clause as well. Likewise, SIP responses are also forked towards the SEC engine.

```
onreply_route{
...
tosec();
...
}
```

Also, we have to create the following FIFO queue where forked messages are routed:

```
mkfifo /tmp/ser2sec
```

Our SEC engine is formed by two components in cascade: the parser and the correlator. The SEC parser takes as input a SIP message, interprets each line as one alone event and matches it against a regular expression. A special rule matches the end of the message (an empty line) and writes an event with the important message attributes towards the correlator:

```
type=single
ptype=RegExp
pattern=^$
desc=$0
action=write /tmp/parser2correlator %a from %b to %c Call-ID %d Contact %e
```

Where parser2correlator is a FIFO created to link the parser output to the correlator input, and a, b, c, d, e are variables assigned during parsing. We start SEC by the following command:

```
perl sec.pl -input=/tmp/ser2sec -conf=correlator.conf
```

The parser is actually "spawn" by a special rule in the correlator configuration file:

```
type=Single
desc=$0
context=[SEC_INTERNAL_EVENT]
ptype=SubStr
pattern=SEC_STARTUP
action=spawn perl sec.pl -conf=parser.conf -input=/tmp/ser2sec
```

The correlator writes alarms of detected misuses to a GUI (WXPython interface). We used a Round Robin database library called RRDtool[22] to shape the SIP traffic. Our tool is shown in Figure 9.1 where the alarm corresponds to a DoS peak. The user can solicit an IP address or host within the alarm to be added to a black list. Items in the black list are temporarily or definitely blocked using the firewall capabilities of the command iptables.

Our solution is shown in figure 9.2

[22] http://oss.oetiker.ch/rrdtool

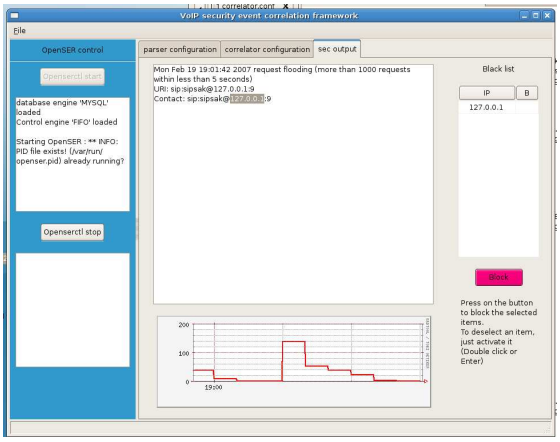


Figure 9.1: Detection of DoS attack

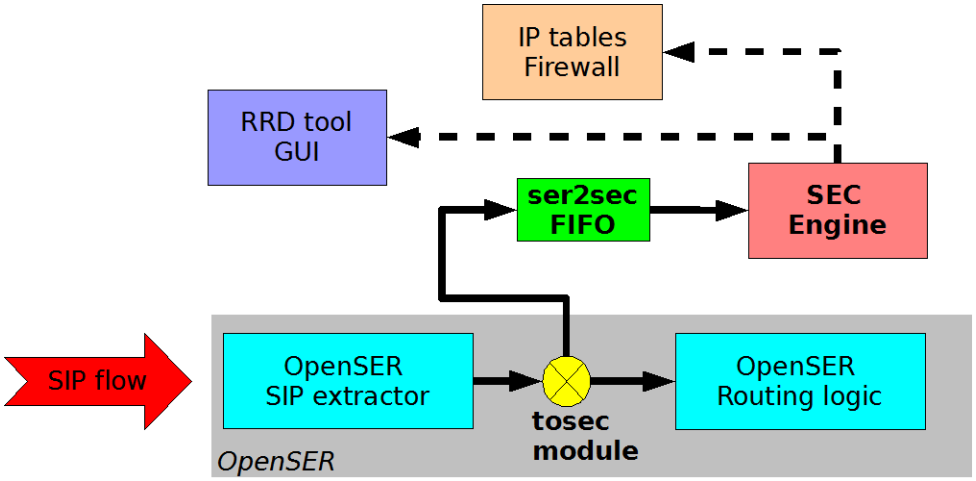


Figure 9.2: Diagram of a N_IDS based on SEC for OpenSER

9.4.2 Attack detection examples

We give here one example for each type of exploits. All these exploits have been discovered by the Madynes team using KIF [2]. In addition, we give examples of other attacks that can be detected like INVITE flooding and broken handshaking.

A malformed header signature (Type-1)

An example of a Type-1 signature is this message:

```
INVITE sip:Bob@192.168.1.3 SIP/2.0
Via:SIP/2.0/UDP\192.168.1.2;branch=00
From: Caripe<sip:caripe@192.168.1.2>;tag=00
To:<sip:Bob@192.168.1.3>;tag=00
Call-ID: caripe@192.168.1.2
CSeq: 2INVITE
```

The malformed header is the Via where a slash character is put instead of a space after the "SIP/2.0/UDP". This message performs a remote DoS on a Thomson ST 2030 SIP phone.

This exploit has to be detected in the parser by a Via parsing rule:

```
type=Single
ptype=RegExp
pattern=Via:\s?SIP/2.0/UDP\\.*
desc=$0
action=create malformed_Via_header
```

The created context allows another rule matching the end of the message to provide another information about the attack (e.g. the sender in the From header):

```
type=single
ptype=RegExp
pattern=^$
desc=$0
context= malformed_Via_header
action=write -  malformed Via header from %b; delete malformed_via_header
```

A malformed message signature (Type-2)

An example of a Type-2 signature is this message:

```
INVITE sip:Bob@$192.168.1.3 SIP/2.0\377\r
Via: SIP/2.0/UDP 192.168.1.2;rport;branch=00\377\r
Max-Forwards: 70\377\r
To: lynksys <sip:Bob@192.168.1.3>\377\r
From: <sip:tucuman@192.168.1.2>;tag=00\377\r
Call-ID: tucu@192.168.1.2\377\r
CSeq: 24865 INVITE\377\r
Contact: <sip:tucu@192.168.1.2>\377\r
Supported: 100rel\377\r
Content-Length: 0\377\r
```

Multiple headers are malformed where a meta character \377 (full byte in octal) is set before every carriage return. This message performs a remote reboot on a Linksys SPA941 phone. We consider that the attack happens when two or more headers carry the malformed character. In our detection scheme, when a header is infected it generates a “377” event, we create a “377” context if two headers are infected. At the end of the message, the existence of this context allows the parser to alert the attack like in the previous example.

```
type=Pair
ptype=RegExp
pattern=377
desc=$0
```

```
ptype2=RegExp
pattern2=377
desc2=$0
action2=create 377
```

A multiple malformed messages signature (Type-3)

An example of a Type-3 signature is this sequence of three messages:

```
1)INVITE sip:Bob@192.168.1.4 SIP/2.0
Via:SIP/2.0/UDP 192.168.1.2;rport;branch=00
From:<sip:gasparin@192.168.1.2>;tag=00
To:<sip:Bob@192.168.1.4>;tag=00
Call-ID: et@192.168.1.2
CSeq: 10 INVITE
Content-Length: 0

2)OPTIONS sip:Bob@192.168.1.4 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.2;rport;branch=01
From: <sip:gasparin@192.168.1.2>;tag=01
To: <sip:Bob@192.168.1.4>
Call-ID: et@192.168.1.2
CSeq: 11 OPTIONS
Content-Length: 0

3)OPTIONS sip:Bob@192.168.1.4 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.2;rport;branch=02
From: <sip:gasparin@192.168.1.2>;tag=02
To: <sip:Bob@192.168.1.4>
Call-ID: et@192.168.1.2
CSeq: 12 OPTIONS
Content-Length: 0
```

While every message is correctly built if taken alone, the sequence of three messages with certain time restrictions provokes a remote reboot on Cisco 7940 phone. We consider that it is necessary for the attack to work that all messages have the same call-ID. We propose the following detection scheme:

```
type=Pair
ptype=RegExp
pattern=INVITE from (.*) to (.*) Call-ID (.*)
desc=$0
action= create $3
ptype2=RegExp
pattern2=OPTIONS from (.*) to (.*) Call-ID (.*)
```

```
desc2=$0
context2=$3
action2=event INVITE-OPTIONS Call-ID $3
window=2
continue2=TakeNext

type=Pair
ptype=RegExp
pattern=INVITE-OPTIONS Call-ID (.*)
desc=$0
context=$1
action= write - half-signature of INVITE-OPTIONS-OPTIONS
ptype2=RegExp
pattern2=OPTIONS from (.*) to (.*) Call-ID (.*)
desc2=$0
context2=$5
action2= write - INVITE-OPTIONS-OPTIONS signature is \-
detected call-ID $3; delete $3
window=2
```

Our signature is abstract: not every INVITE followed by two OPTIONS of the same call-ID may reboot the phone. This leads us to a very important question: If we have the trace of an attack, what exactly is the part that makes it work? For example, is it part of the attack the fact that the INVITE hasn't an SDP body? or that the From-tag is incremented by one by each message (it is the same to the transaction identifier in the branch parameter and the Cseq number). A good definition of the fields that cause the attack is essential to not throw false alarms.

Flooding detection

The following rule detects an INVITE flooding:

```
type=SingleWithThreshold
ptype=RegExp
pattern=INVITE from (.*) to (.*) Call-ID (.*)
desc=INVITE from $1
action= write - %t request flooding; write - URI: $1;
window=1
thresh=1000
```

The rule fires if more than 1000 messages are received by the OpenSER within less than one second (from the same user). The alarms contains the time of occurrence and the value in the From header. Using a SingleWith2Thresholds rule allows to recover to the normal condition when the flooding doesn't persist over time.

```
type=SingleWith2Thresholds
ptype=RegExp
```

```
pattern=INVITE from (.*) to (.*) Call-ID (.*)
desc=INVITE from $1
actionwrite - %t: request flooding; write - URI: $1;
window=1
thresh=1000
desc2=$0
action2=write - %t: request flooding falls down
window2=1
thresh2=100
```

Broken handshaking detection

We need two rules to detect a broken handshaking where a caller sends an INVITE and when receiving the 200 OK, doesn't acknowledge with an ACK:

```
type= Single
ptype=RegExp
pattern=200 OK from (.*) to (.*) Call-ID (.*) Cseq_method INVITE
desc=$0
action= create $3
continue=TakeNext

type= PairWithWindow
ptype=RegExp
pattern=200 OK from (.*) to (.*) Call-ID (.*) Cseq_method INVITE
desc=$0
action= write - broken handshaking from $1
ptype2=RegExp
pattern2=ACK from (.*) to (.*) Call-ID (.*)
desc2=$0
context2=$3
action2=delete $3
```

Actually, we need just one rule of type PairWithWindow. When an 200 OK is received and its Cseq_method is set to INVITE, we start a time window and wait for some ACK with the same Call-ID. The small problem is that we can not do any action before the time window is elapsed, in consequence we can not create a context for the Call-Id to match an incoming ACK. A second rule of type Single is needed to create a context for the Call-ID. We added a "continue=TakeNext" statement allowing the 200 OK to be matched by the two rules. If the transaction is completed, we delete the context, otherwise an alarm is generated containing the From value. This alarm is useless if the proxy doesn't put itself in the record-route because the ACK will be sent directly to the callee.

9.5 Conclusion

This is the last chapter of this manuscript. We presented a brief tutorial on the use of SEC to perform host and network monitoring on open source VoIP platforms like Asterisk and OpenSER. We presented a first prototype of a rule-based N_IDS and we demonstrated how appropriate event correlation rules are able to detect different types of malformed SIP messages attacks. The main difficulty using SEC is that SEC events haven't attributes. Therefore, we used some tricks based on rules and contexts in order to adapt it to our case. This is not a real weakness in the SEC design because it is not initially intended for network event recognition but as a system management tool. On the other hand, SEC has a real time performance which is convenient to VoIP. SEC correlation rules, by their types, fulfill network event correlation requirements like event threading and imposing temporal restrictions. Our final conclusion is that using a SEC-like design is efficient if coupled with suitable network event recognition tools such as the NERL [16] tools. Still, a rule-based approach remains weak if correlation rules are not carefully written by the human operator. Anomaly detection approaches (such as [79]) to detect malformed SIP messages should be addressed.

General Conclusion

As attacks on the emerging VoIP technology are popping-up in different forms with increasing impact on both users and infrastructure, more monitoring and security management is required. In this dissertation, we studied the VoIP threats and we investigated the best ways to ensure security and intrusion detection in VoIP domains.

In the first part of this manuscript, we introduced the VoIP protocols and highlighted their distributed and heterogeneous nature. We argued the specific security requirements by a brief survey of the most relevant threats. We then discussed the recent works on VoIP intrusion detection, traffic profiling, spamming and denial of service mitigation.

In the second part, we proposed three complementary contributions:

1. We proposed an online monitoring methodology based on machine learning techniques (Bayesian Networks and support vector machines). Our idea is to cut the ongoing signaling (SIP) traffic into small slices and to extract a vector of defined features characterizing each slice. Vectors are then pushed into a machine for classification based on a prediction model. We then use a deterministic event correlator to raise an alarm when suspicious and abnormal situations occur.

2. We presented an innovative approach to design a VoIP specific honeypot (honeyphone). The honeyphone is supplied by an application program interface which controls a rich set of network tools. Rather than recording calls and dumping SIP packets to be studied as attack traces, the honeyphone is able to gather information in real time about the received messages. An important component in our architecture is the inference engine which can differentiate between a routing fault, a shooted message and a distrustable call. Simulation unrolling of examples showed promising results about the effectiveness of the information gathering tools and the correctness of the inference engine conclusions.

3. We presented a holistic approach for VoIP intrusion detection. The key components of our solution are host-based and network-based sensors as well as an event correlation framework.

The third part is intended to the assessment of our approaches. We started by a presentation of our assessment tool (the VoIP bot) and we depicted some attack scenarios. We experimented our monitoring approach using a data-set composed of real-world attack-free traces and attacks generated in a customized test-bed. We validated this approach through exhaustive experiments and demonstrated how to tune the detection parameters in order to achieve a high accuracy. We then compared our two proposed models (the

Bayesian networks model and the support vector machines one) for classification and anomaly detection in both short-term and long-term scales. Results demonstrated a real-time performance and a high accuracy for the detection of flooding and spamming attacks especially when efficient event correlation rules are enabled. Finally, we built a prototype of a rule-based event-driven network intrusion detection sensor as a component of our holistic security event correlation approach and we showed how it can be used to detect a series of malformed SIP messages. Our prototype is based on the Simple Event Correlation tool taking advantage from its strong points. Nevertheless, we highlighted its weakness especially in terms of event structure.

There are several important future directions:

- Feature selection: Our set of features is not comprehensive (only 38 features are defined). We should consider additional features such as statistics over the registration parameters (registration life-times, registration request inter-arrival times) and statistics over the SIP transactions. A specification-based denial-of-service detection approach employs statistics over a modified transaction state machine [26] that also should be studied. Some features have to be refined like for instance to differentiate between the traffic entering to the proxy and that exiting from it, and to differentiate the destinations of the calls from the destinations of the other signaling requests.

  We have used a simple but fast feature selection algorithm based on optimizing the support vectors machines classification accuracy and using the F-score filter [111]. This approach doesn't reveal the mutual information between features. Other feature selection algorithms should be considered: Wu et. al. [107] propose using a balanced information gain to measure the contribution of each feature in the classification, and the correlation between the features. Then, they present an algorithm to select the uncorrelated features having a large balanced information gain. Oh et. al. [68] propose a hybrid genetic algorithm for feature selection based on local search operations and show that its performance is superior to both simple genetic algorithm and sequential search algorithms. Fleuret [32] proposes a fast algorithm based on conditional mutual information to ensure the selection of features which are both individually informative and two-by-two weakly dependant. Yu et. al. [113] utilize the correlation-based feature selection (CFS) algorithm [40] - which is implemented as a component of the WEKA software (Waikato Environment for Knowledge Analysis) - in their network and transport flooding detection approach also based on support vector machines. The authors argue that the CFS algorithm has a widely accepted performance in contrast to [95] which suffers from its computational complexity.

- Unsupervised learning: Unsupervised learning techniques are attractive because they don't need a priori knowledge and are able to detect new and previously unknown attacks. More anomaly detection and unsupervised learning algorithms should be studied in the context of our VoIP monitoring system. Besides traditional algorithms like the k-means method (A survey can be found in [15]), we would like to take advantage from novel approaches which are essentially based on support

vector machines [93, 14, 109, 110].

- A single honeyphone has a limited field of view and therefore is able to detect a limited set of attacks. A high-interactivity honeypot (VoIP honeynet) should attract more attacks into a secured and observed environment by offering attractive services. A VoIP honeynet should be designed and implemented to simulate a whole SIP network, i.e. infrastructure (SIP proxies), services (PBXs) and subscribers (honeyphones). The project is still considered to be implemented and incorporated in the Madynes high security laboratory [13].

- The works on the security event correlation platform can be pursued in several directions: at the first layer, we should consider an appropriate network event recognition language such as NERL [16] in order to perform efficient network-based intrusion detection. At the second layer, we should consider alert correlation techniques (e.g. [101, 74]) in order to reveal attack strategies and predict attacker plans. Our work on the signatures of malformed SIP messages will be pursued in the context of the SECSIP firewall[23] currently under development in the MADYNES team. We aim to define a standard language for describing attack signatures that could be easily incorporated into VoIP-specific firewalls, application layer gateways and intrusion detection systems.

[23] http://secsip.gforge.inria.fr/

Bibliography

[1] H. Abdelnur, V. Cridlig, J. Bourdellon, R. State, and O. Festor. VoIP security management. In *The 18th meeting of the Network Management Research Group (NMRG)*, Jul 2005.

[2] H. Abdelnur, R. State, and O. Festor. KiF: a stateful SIP fuzzer. In *Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications (IPTComm '07)*, pages 47–56, New York, NY, USA, 2007. ACM.

[3] H. Abdelnur, R. State, and O. Festor. Advanced network fingerprinting. In *Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection (RAID '08)*, pages 311–330, London, UK, 2008. Springer-Verlag.

[4] H. Abdi. *Encyclopedia of Measurement and Statistics*, chapter Distance, pages 280–284. Thousand Oaks (CA): Sage, 2007.

[5] J. P. Anderson. Computer security threat monitoring and surveillance. Technical Report Contract 79F26400, James P. Anderson Co., Box 42, Fort Washington, PA, 19034, USA, 1980.

[6] M. Arango, A. Dugan, I. Elliott, C. Huitema, and S. Pickett. RFC2705: Media Gateway Control Protocol (MGCP) version 1.0, 1999.

[7] I. Arce. Voices, I hear voices. *IEEE Security and Privacy*, 4(4):80–83, 2006.

[8] I. Arce and E. Levy. An analysis of the Slapper worm. *IEEE Security and Privacy*, 1(1):82–87, 2003.

[9] V. A. Balasubramaniyan, M. Ahamad, and H. Park. CallRank: Combating SPIT using call duration, social networks and global reputation. In *Fourth Conference on Email and Anti-Spam (CEAS2007)*, Mountain View, California USA, 2007.

[10] D. Barbara. *Applications of Data Mining in Computer Security*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[11] D. Barbará, N. Wu, and S. Jajodia. Detecting novel network intrusions using Bayes estimators. In *Proceedings of the First SIAM Conference on Data Mining*, April 2001.

[12] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman. RFC3711: The Secure Real-time Transport Protocol (SRTP), 2004.

[13] F. Beck, O. Festor, and R. State. High security laboratory - network telescope. Technical Report 9999, Centre de recherche INRIA Nancy -Grand Est, Projet MADYNES, March 2007. `http://hal.inria.fr/inria-00337568/en/`.

[14] A. Ben-hur, D. Horn, H. T. Siegelmann, and V. Vapnik. A support vector method for clustering. In *Advances in Neural Information Processing Systems 13*, pages 367–373. MIT Press, 2001.

[15] P. Berkhin. Survey of clustering data mining techniques. Technical Report Berkhin:02, Accrue Software, San Jose, CA, 2002. `http://www.accrue.com/products/rp_cluster_review.pdf`.

[16] K. Bhargavan. *Network event recognition*. PhD thesis, University of Pennsylvania, Philadelphia, PA, USA, 2003. Supervisor-Carl A. Gunter.

[17] N. Bonelli, S. Giordano, G. Procissi, and R. Secchi. BRUTE: A high performance and extensible traffic generator. In *Proceedings of International Symposium on Performance and Extensible Traffic Generator (SPECTS'05)*, PA, USA, July 2005.

[18] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[19] E. Y. Chen. Detecting DoS attacks on SIP systems. In *Proceedings of 1st IEEE Workshop on VoIP Management and Security*, pages 53–58, San Diego, CA, USA, apr 2006.

[20] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: a distributed anonymous information storage and retrieval system. In *International workshop on Designing privacy enhancing technologies*, pages 46–66, New York, NY, USA, 2001. Springer-Verlag New York, Inc.

[21] L. Dang, C. Jennings, and D. Kelly. *Practical VoIP using Vocal*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, first edition, July 2002.

[22] D. E. Denning. An intrusion-detection model. In *IEEE Symposium on Security and Privacy*, pages 118–133. IEEE Computer Society Press, Apr 1986.

[23] T. Dierks and C. Allen. RFC2246: The TLS protocol version 1.0, 1999.

[24] Y. Ding and G. Su. Intrusion detection system for signal based SIP attacks through timed HCPN. In *Proceedings of the Second International Conference on Availability, Reliability and Security (ARES'07)*. IEEE Computer Society, 2007.

[25] R. Droms. RFC2131: Dynamic host configuration protocol, 1997.

[26] S. Ehlert, C. Wang, T. Magedanz, and D. Sisalem. Specification-based denial-of-service detection for SIP Voice-over-IP networks. In *The Third International Conference on Internet Monitoring and Protection (ICIMP)*, pages 59–66, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[27] D. Endler and M. Collier. *Hacking Exposed VoIP: Voice Over IP Security Secrets and Solutions*. McGraw-Hill Professional Publishing, 2007.

[28] P. Faltstrom. RFC2916: E.164 number and DNS, 2000.

[29] R. Fan, P. Chen, and C. Lin. Working set selection using second order information for training support vector machines. *J. Mach. Learn. Res.*, 6:1889–1918, 2005.

[30] J. Fiedler, t. Kupka, S. Ehlert, T. Magedanz, and D. Sisalem. VoIP defender: Highly scalable SIP-based security architecture. In *Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications (IPTComm '07)*, New York, NY, USA, 2007. ACM.

[31] C. Fleizach, M. Liljenstam, P. Johansson, G. M. Voelker, and A. Mehes. Can you infect me now?: malware propagation in mobile phone networks. In *Proceedings of the 2007 ACM workshop on Recurring malcode (WORM '07)*, pages 61–68, New York, NY, USA, 2007. ACM.

[32] F. Fleuret. Fast binary feature selection with conditional mutual information. *J. Mach. Learn. Res.*, 5:1531–1555, 2004.

[33] O. Francois and P. Leray. Bnt structure learning package: documentation and experiments. Technical report, FRE CNRS 2645. Laboratoire PSI, INSA Rouen, 2004.

[34] E. Gabber, P. Gibbons, D. Kristol, Y. Matias, and A. Mayer. Consistent, yet anonymous, web access with LPWA. *Commun. ACM*, 42(2):42–47, 1999.

[35] D. Geneiatakis, G. Kambourakis, T. Dagiuklas, C. Lambrinoudakis, and S. Gritzalis. SIP security mechanisms: A state-of-the-art review. In *Proceedings of the Fifth International Network Conference (INC 2005)*, pages 147–155, Samos, Greece, July 2005.

[36] D. Geneiatakis, G. Kambourakis, C. Lambrinoudakis, T. Dagiuklas, and S. Gritzalis. A framework for protecting a SIP-based infrastructure against malformed message attacks. *Comput. Netw.*, 51(10):2580–2593, 2007.

[37] A. Gulbrandsen, P. Vixie, and L. Esibov. RFC2782: A DNS RR for specifying the location of services (DNS SRV), 2000.

[38] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Mach. Learn.*, 46(1-3):389–422, 2002.

[39] ITU Recommendation H.323. Packet-based multimedia communications systems, 2000.

[40] M. A. Hall. *Correlation-based Feature Selection for Machine Learning.* PhD thesis, The University of Waikato, 1999.

[41] M. Handley and V. Jacobson. RFC2327: SDP: Session Description Protocol, 1998.

[42] M. Handley, C. Perkins, and E. Whelan. RFC2974: Session Announcement Protocol, 2000.

[43] The honeynet project. *Know Your Enemy: Learning about Security Threats (2nd Edition).* Pearson Education, 2004.

[44] H. Javitz and A. Valdes. The SRI IDES statistical anomaly detector. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 316–326. IEEE computer society, May 1991.

[45] A. Johnston. *SIP: Understanding the Session Initiation Protocol, Second Edition.* Artech House, Inc., Norwood, MA, USA, 2003.

[46] H. Kang, Z. Zhang, S. Ranjan, and A. Nucci. SIP-based VoIP traffic behavior profiling and its applications. In *Proceedings of the 3rd annual ACM workshop on Mining network data (MineNet '07)*, pages 39–44, New York, NY, USA, 2007. ACM.

[47] S. Kent and R. Atkinson. RFC2401: Security architecture for the Internet protocol, 1998.

[48] P. Kolan and R. Dantu. Socio-technical defense against voice spamming. *ACM Trans. Auton. Adapt. Syst.*, 2(1):Article 2, 2007.

[49] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur. Bayesian event classification for intrusion detection. In *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC '03)*, page 14, Washington, DC, USA, 2003. IEEE Computer Society.

[50] C. Kruegel, G. Vigna, and W. Robertson. A multi-model approach to the detection of web-based attacks. *Computer Networks*, 48(5):717–738, August 2005.

[51] C. Krügel, T. Toth, and E. Kirda. Service specific anomaly detection for network intrusion detection. In *Proceedings of the 2002 ACM symposium on Applied computing (SAC '02)*, pages 201–208, New York, NY, USA, 2002. ACM Press.

[52] D. Richard Kuhn, Thomas J. Walsh, and S. Fries. Security considerations for Voice over IP systems. National Institute of Standards and Technology (NIST), Special Publication, 800-58, January 2005.

[53] J. F. Kurose and K. W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet Package.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[54] H. Lee, J. Song, and D. Park. Intrusion detection system based on multi-class SVM. In *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, volume 3642 of *Lecture Notes in Computer Science*, pages 511–519. Springer Berlin / Heidelberg, 2005.

[55] Kunlun Li and Guifa Teng. Unsupervised SVM based on p-kernels for anomaly detection. In *Proceedings of the First International Conference on Innovative Computing, Information and Control (ICICIC '06)*, pages 59–62, Washington, DC, USA, 2006. IEEE Computer Society.

[56] M. Luo, T. Peng, and C. Leckie. CPU-based DoS attacks against SIP servers. In *IEEE/IFIP Network Operations and Management Symposium (NOMS 2008)*. IEEE, April 2008.

[57] M. A. Maloof. *Machine Learning and Data Mining for Computer Security: Methods and Applications (Advanced Information and Knowledge Processing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[58] D. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja1, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-Peer computing, July 2003. HPL-2002-57 (R1).

[59] S. Mukkamala, G. Janoski, and A. Sung. Intrusion detection: Support vector machines and neural networks. *The IEEE Computer Society Student Magazine*, 10(2), 2002.

[60] K. P. Murphy. The Bayes Net Toolbox for MATLAB. *Computing Science and Statistics*, 33:1–20, 2001.

[61] P. Naïm, P. Wuillemin, P. Leray, O. Pourret, and A. Becker. *Réseaux Bayésiens*. Eyrolles, 2004.

[62] M. Nassar, S. Niccolini, R. State, and T. Ewald. Holistic voip intrusion detection and prevention system. In *Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications (IPTComm '07)*, pages 1–9, New York, NY, USA, 2007. ACM.

[63] M. Nassar, R. State, and O. Festor. Intrusion detections mechanisms for VoIP applications. In *Third annual security workshop (VSW'06)*. ACM Press, Jun 2006.

[64] M. Nassar, R. State, and O. Festor. VoIP honeypot architecture. In *Integrated Network Management*, pages 109–118. IEEE, 2007.

[65] M. Nassar, R. State, and O. Festor. Monitoring SIP traffic using support vector machines. In *Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection (RAID '08)*, pages 311–330, London, UK, 2008. Springer-Verlag.

[66] S. Niccolini, R.G. Garroppo, S. Giordano, G. Risi, and S. Ventura. SIP intrusion detection and prevention: recommendations and prototype implementation. In *VoIP Management and Security, 2006. 1st IEEE Workshop on*, pages 47–52, April 2006.

[67] P. O'Doherty and M. Ranganathan. JAIN SIP Tutorial: Serving the developer community. `http://www-x.antd.nist.gov/proj/iptel/tutorial/JAIN-SIP-Tutorialv2.pdf`.

[68] I. Oh, J. Lee, and B. Moon. Hybrid genetic algorithms for feature selection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(11):1424–1437, 2004.

[69] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[70] J. Peterson and C. Jennings. RFC4774: Enhancements for authenticated identity management in the Session Initiation Protocol (SIP), 2006.

[71] J. C. Platt. *Fast training of support vector machines using sequential minimal optimization*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

[72] D. C. Plummer. RFC826: An Ethernet Address Resolution Protocol, 1982.

[73] F. Pouget and M. Dacier. Honeypot-based forensics. In *Asia pacific information technology security conference (AusCERT '04)*, May 2004.

[74] X. Qin. *A probabilistic-based framework for INFOSEC alert correlation*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, USA, 2005.

[75] J. Quittek, S. Niccolini, S. Tartarelli, M. Stiemerling, M. Brunner, and T. Ewald. Detecting SPIT calls by checking human communication patterns. In *IEEE International Conference on Communications (ICC 2007)*, Jun 2007.

[76] B. Ramsdell. RFC2633: S/MIME version 3 message specification, 1999.

[77] J. Ransome and J. Rittinghouse. *Voice over Internet Protocol (VoIP) Security*. Digital Press, Newton, MA, USA, 2004.

[78] B. Reynolds and D. Ghosal. Secure IP telephony using multi-layered protection. In *Proceedings of The 10th Annual Network and Distributed System Security Symposium*, San Diego, CA, USA, feb 2003.

[79] K. Rieck, S. Wahl, P. Laskov, P. Domschitz, and K.-R. Müller. A self-learning system for detection of anomalous SIP messages. In *Principles, Systems and Applications of IP Telecommunications (IPTCOMM), Second International Conference*, Jul 2008.

[80] R. Romano, C. Aragon, and C. Ding. Supernova recognition using support vector machines. In *Proceedings of the 5th International Conference on Machine Learning and Applications (ICMLA '06)*, pages 77–82, Washington, DC, USA, 2006. IEEE Computer Society.

[81] J. Rosenberg, G. Camarillo, and D. Willis. RFC5360: A framework for consent-based communications in the Session Initiation Protocol (SIP), October 2008.

[82] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. RFC3261: SIP: Session initiation protocol, 2002.

[83] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. RFC3489: STUN - Simple Traversal of User Datagram Protocol (UDP) through Network Address Translators (NATs), 2003.

[84] R. Schlegel, S. Niccolini, S. Tartarelli, and M. Brunner. SPam over Internet Telephony (SPIT) prevention framework. In *GLOBECOM*, 2006.

[85] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RFC3550: RTP: A Transport Protocol for Real-time applications, 2003.

[86] A. A. Sebyala, T. Olukemi, and L. Sacks. Active platform security through intrusion detection using naïve Bayesian network for anomaly detection. In *London Communications Symposium*, 2002.

[87] H. Sengar, R. Dantu, and D. Wijesekera. Securing VoIP and PSTN from integrated signaling network vulnerabilities. In *1st IEEE workshop on VoIP Management and Security (VoIP MaSe)*, Vancouver, Canada, April 2006.

[88] H. Sengar, H. Wang, D. Wijesekera, and S. Jajodia. Detecting VoIP floods using the Hellinger distance. *IEEE Trans. Parallel Distrib. Syst.*, 19(6):794–805, 2008.

[89] H. Sengar, D. Wijesekera, H. Wang, and S. Jajodia. VoIP intrusion detection through interacting protocol state machines. In *Proceedings of the 38th IEEE International Conference on Dependable Systems and Networks (DSN'2006)*. IEEE Computer Society, 2006.

[90] D. Shin and C. Shim. Progressive multi gray-leveling: A voice Spam protection algorithm. *IEEE Network*, 20(5):18–24, Sep/Oct 2006.

[91] H. Sinnreich and A. B. Johnston. *Internet communications using SIP: delivering VoIP and multimedia services with Session Initiation Protocol*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[92] K. Sollins. RFC1350: The TFTP protocol (revision 2), 1992.

[93] Z. Songfeng, L. Xiaofeng, Z. Nanning, and X. Weipu. Unsupervised clustering based reduced support vector machines. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03)*, April 2003.

[94] S. J. Stolfo, S. Hershkop, K. Wang, O. Nimeskern, and W. Hu. A behavior-based approach to securing email systems. In *Mathematical Methods, Models and Architectures for Computer Networks Security*, pages 57–81. Springer Verlag, 2003.

[95] Y. Sun and J. Li. Iterative RELIEF for feature weighting. In *Proceedings of the 23rd international conference on Machine learning (ICML '06)*, pages 913–920, New York, NY, USA, 2006. ACM.

[96] SUN Microsystem. Java Media Framework API Guide, november 1999. `http://java.sun.com/products/java-media/jmf/2.1.1/guide/index.html`.

[97] P. Truong, D. Nieh, and M. Moh. Specification-based intrusion detection for H.323-based Voice over IP. In *Proccedings of the IEEE International Symposium on Signal Processing and Information Technology*. IEEE Computer Society, 2005.

[98] P. Uppuluri and R. Sekar. Experiences with specification-based intrusion detection. In *RAID '00: Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, pages 172–189, London, UK, 2001. Springer-Verlag.

[99] R. Vaarandi. SEC - a lightweight event correlation tool. In *Proceedings of the 2002 IEEE Workshop on IP operations and Management*, pages 111–115, Oct 2002.

[100] A. Valdes and K. Skinner. Adaptive, model-based monitoring for cyber attack detection. In *Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection (RAID '00)*, pages 80–92, London, UK, 2000. Springer-Verlag.

[101] A. Valdes and K. Skinner. Probabilistic alert correlation. In *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID '00)*, pages 54–68, London, UK, 2001. Springer-Verlag.

[102] V. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[103] V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, 1998.

[104] VoIPSA. VoIP security and privacy threat taxonomy. Public Realease 1.0, Oct 2005. `http://www.voipsa.org/Activities/VOIPSA_Threat_Taxonomy_0.1.pdf`.

[105] T. J. Walsh and R. Kuhn. Challenges in securing Voice over IP. *IEEE Security and Privacy*, 3(3):44–49, 2005.

[106] Y. Wu, S. Bagchi, S. Garg, N. Singh, and T. K. Tsai. SCIDIVE: A stateful and cross protocol intrusion detection architecture for Voice-over-IP environments. In *International Conference on Dependable Systems and Networks (DSN 2004)*, pages 433–442. IEEE Computer Society, Jun 2004.

[107] Y. Wu and A. Zhang. Feature selection for classifying high-dimensional numerical data. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:251–258, 2004.

[108] B. Xiao, W. Chen, Y. He, and E. H.-M. Sha. An active detecting method against SYN flooding attack. In *Proceedings of the 11th International Conference on Parallel and Distributed Systems (ICPADS'05)*, pages 709–715, Washington, DC, USA, 2005. IEEE Computer Society.

[109] L. Xu and D. Schuurmans. Unsupervised and semi-supervised multi-class support vector machines. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2005.

[110] L. Xu, D. F. Wilkinson, F. Southey, and D. Schuurmans. Discriminative unsupervised learning of structured predictors. In *ICML*, pages 1057–1064, 2006.

[111] C. Lin Y. Chen. *Feature Extraction, Foundations and Applications*, chapter Combining SVMs with various feature selection strategies. Springer, 2006.

[112] H. Yan, K. Sripanidkulchai, H. Zhang, Z.-Y. Shae, and D. Saha. Incorporating active fingerprinting into SPIT prevention systems. In *Third annual security workshop (VSW'06)*. ACM Press, Jun 2006.

[113] J. Yu, H. Lee, M. Kim, and D. Park. Traffic flooding attack detection with SNMP MIB using SVM. *Computer Communications*, 31(17):4212 – 4219, 2008.

[114] G. Zhang, S. Ehlert, T. Magedanz, and D. Sisalem. Denial of service attack and prevention on SIP VoIP infrastructures using DNS flooding. In *Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications (IPTComm '07)*, pages 57–66, New York, NY, USA, 2007. ACM.

[115] R. Zhang, X. Wang, X. Yang, and X. Jiang. Billing attacks on SIP-based VoIP systems. In *Proceedings of the first USENIX workshop on Offensive Technologies (WOOT '07)*, pages 1–8, Berkeley, CA, USA, 2007. USENIX Association.

[116] Y. Zhang. *Bayesian graphical models for adaptive filtering*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2005.

Note: *All URLs are last visited and verified on 8/12/2008.*

Glossary

This glossary defines the important abbreviations used along this manuscript

3GPP : 3rd Generation Partnership Project: is a collaboration between groups of telecommunications associations, to make a globally applicable third generation mobile phone system specification.

B2BUA : Back-to-Back User Agent: is a type of SIP device that receives a SIP request, then reformulates the request and sends it out as a new request. Responses to the request are also reformulated and sent back in the opposite direction.

CAS : Channel Associated Signaling: is the in-band type of signaling used within the PSTN.

DES : Data Encryption Standard: is a widely used method for encrypting information based on a symmetric-key algorithm using a 56-bit key.

DHCP : Dynamic Host Configuration Protocol: A protocol for dynamically assigning IP addresses to devices on a network.

DNS : Domain Name Service: a network service used to translate between domain name and IP addresses.

H.323 : Recommendation from the ITU Telecommunication Standardization Sector (ITU-T) that defines the protocols umbrella to provide packet-based multimedia communication.

IAX : Inter Asterisk eXchange: is native to Asterisk PBX and supported by a number of other softswitches and PBXs. IAX carries both signalling and media on the same path.

IETF : Internet Engineering Task Force: is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet.

IMS : Internet Multimedia Subsystem: originally designed to evolve mobile networks to deliver IP multimedia to mobile users, IMS has become the core component within 3G, cable TV and NGNs.

IPsec : Internet Protocol Security: is a suite of protocols for securing IP communications by authenticating and encrypting each IP packet of a data stream.

ISDN : Integrated Services Digital Networks: The key feature of the ISDN is that it integrates speech and data on the same lines, adding features that were not available in the classic telephone system.

ISUP : ISDN User-Part: is the application layer of the SS7 signaling.

ITU : International Telecommunication Union: is the leading United Nations agency for information and communication technologies. ITU's role in helping the world communicate spans 3 core sectors: radiocommunication, standardization and development.

LAN : Local Area Network: a high-speed computer network covering a small physical area.

MGCP : Media Gateway Control Protocol: used between elements of a decomposed multimedia gateway which consists of a Call Agent, and a Media Gateway.

NAT : Network Address Translator: is the process of modifying network address information in datagram packet headers while in transit across a traffic routing device for the purpose of remapping a given address space into another.

NGN : Next Generation Networks: The general idea behind NGN is that one network transports all information and services (voice, data, and media) using the IP protocol. Therefore the term all-IP is used to describe the transformation towards NGN.

PSTN : Public Switched Telephony Network: is the network of the world's public circuit-switched telephone networks.

QoS : Quality of Service: is the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance.

RTCP : Real-time Control Transport Protocol: RTCP is used to control aspects of RTP sessions.

RTP : Real-time Transport Protocol: RTP is used to exchange media information such as voice or video.

S/MIME : Secure / Multipurpose Internet Mail Extensions: is a standard for public key encryption and signing of e-mail encapsulated in MIME, can also be used for SIP.

SDP : Session Description Protocol: is intended for describing multimedia communication sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation.

SIP : Session Initiation Protocol: Application layer signaling protocol for the establishing, modifying and terminating of multimedia sessions or calls.

SIPS : Secure SIP URI: allows resources to specify that they should be reached securely. It mandates that each hop over which the request is forwarded up to the target domain must be secured with TLS. The last hop from the proxy of the target domain to the user agent has to be secured according to local policies.

SS7 : Signaling System7: is the out-band type of signaling used within the PSTN.

TCP : Transmission Control Protocol: is an Internet transport layer (layer 4) protocol for reliable, connection-oriented data delivery. Connection-oriented means that a connection must be set up before communicating entities can send protocol data units.

TFTP : Trivial File Transport Protocol: a simple protocol that uses UDP to transfer files.

TGREP : Telephony Gateway REgistration Protocol: is used for registration of telephony prefixes and resource information supported by telephony gateways. The prefix and resource information are passed on to a TRIP Location Server.

TLS : Transport Layer Security: is a cryptographic protocol that provides security and data integrity for communications over TCP.

TRIP : Telephony Routing over IP: is a policy driven dynamic routing protocol for advertising the reachability of telephony destinations, and for advertising attributes of the routes to those destinations.

URI : Uniform Resource Identifier: is a compact string of characters used to identify or name a resource on the Internet.

URL : Uniform Resource Locator: is a type of URI that specifies where an identified resource is available and the mechanism for retrieving it.

VoIP : Voice over Internet Protocol: The set of functions, capabilities or features of a network including the ability to initiate, accept or refuse, identify, authenticate, connect, maintain, route, process, store, retrieve, and disconnect any voice or video content with any number of parties desired by the User.

Résumé

La Voix sur IP (VoIP) est devenue un paradigme majeur pour fournir des services de télécommunications flexibles tout en réduisant les coûts opérationnels. Le déploiement à large échelle de la VoIP est soutenu par l'accès haut débit à l'Internet et par la standardisation des protocoles dédiés. Cependant, la VoIP doit également faire face à plusieurs risques comprenant des vulnérabilités héritées de la couche IP auxquelles s'ajoutent des vulnérabilités spécifiques. Notre objectif est de concevoir, implanter et valider de nouveaux modèles et architectures pour assurer une défense préventive, permettre le monitorage et la détection d'intrusion dans les réseaux VoIP.

Notre travail combine deux domaines: celui de la sécurité des réseaux et celui de l'intelligence artificielle. Nous renforcons les mécanismes de sécurité existants en apportant des contributions sur trois axes : Une approche basée sur des mécanismes d'apprentissage pour le monitorage de trafic de signalisation VoIP, un pot de miel spécifique, et un modèle de corrélation des évenements pour la détection d'intrusion. Pour l'évaluation de nos solutions, nous avons développés des agents VoIP distribués et gérés par une entité centrale. Nous avons développé un outil d'analyse des traces réseaux de la signalisation que nous avons utilisé pour expérimenter avec des traces de monde réel. Enfin, nous avons implanté un prototype de détection d'intrusion basé sur des règles de corrélation des événements.

Mots-clés: Voix sur IP, Session Initiation Protocol (SIP), Corrélation des événements, Détection d'intrusion, Pot de miel, Réseaux Bayesiens, Machines à vecteurs de support (SVM), Réseau zombie.

Abstract

Voice over IP (VoIP) has become a major paradigm for providing flexible telecommunication services and reducing operational costs. The large-scale deployment of VoIP has been leveraged by the high-speed broadband access to the Internet and the standardization of dedicated protocols. However, VoIP faces multiple security issues including vulnerabilities inherited from the IP layer as well as specific ones. Our objective is to design, implement and validate new models and architectures for performing proactive defense, monitoring and intrusion detection in VoIP networks.

Our work combines two domains: network security and artificial intelligence. We reinforce existent security mechanisms by working on three axes: a machine learning approach for VoIP signaling traffic monitoring, a VoIP specific honeypot and a security event correlation model for intrusion detection. In order to experiment our solutions, we have developed VoIP agents which are distributed and managed by a central entity. We have developed an analyzer of signaling network traces and we used it to analyze real-world traces. Finally, we have implemented a prototype of a rule-based event-driven intrusion detection system.

Keywords: Voice over IP, Session Initiation Protocol (SIP), Event correlation, Intrusion detection, Honeypot, Bayesian networks, Support vector machines(SVM), Botnet