Intrusion Detection and Prevention to Enhance Security in VoIP Environments

Studienarbeit

Dimiter Vladimirov Milushev

November 9, 2006

Betreuer:
Professor Dieter Gollmann

Technische Universitaet Hamburg-Harburg
Sicherheit in verteilten Anwendungen
http://www.sva.tu-harburg.de/
Harburger Schlossstrasse 20
21079 Hamburg-Harburg
Deutschland

Contents

1 Introduction

Although more than 30 years have passed since the original proposal (in the early 70ies) to utilize data networks for voice transmission, Voice over IP has only been commercially feasible for the last several years and as a relatively new technology is inferior to the traditional digital voice telephony in terms of reliability, stability and security. Voice over Internet Protocol (VoIP), the real time transmission of voice over datagram networks, is becoming today a viable alternative to the traditional circuit-switched telephony. The benefits offered by VoIP are the reason for the popularity of the technology. Sometimes and for specific purposes the benefits can outweigh the drawbacks, but we still have to consider both carefully.

The major benefit of VoIP is the cost saving, due to the utilization of data networks for voice transmission and thus the lack of necessity for dedicated telephony equipment. Having a single infrastructure for both data and voice services means less wiring and also less management costs. Another advantage is the portability of a phone number in VoIP, i.e. it is not assigned to a telephone line, but instead configured in software to be mapped to a specific IP address of the device to be used. VoIP also allows convergence, the use of a single data network for voice, video and data communications [1]. Finally we have the concept of peripheral intelligence, characterizing VoIP telephony, which means the intelligence moves towards the user device and empowers users to be in control of the device usage and access value added services from different service providers [1].

Naturally, VoIP comes with some disadvantages: datagram networks offer a best effort service, thus latency and delay can be high, leading to an unacceptable Quality of Service. It is clear that the availability offered by a service based on IP networks can not be as high, as the one offered by traditional telephony. Last, but not least important, the security considerations of VoIP deployments came as an afterthought, similarly to the security of other information technologies at the time they emerged, such as TCP/IP, Wireless 802.11, Web Services.

Due to its nature (running on a packet-switched network), VoIP is plagued by all security issues that affect IP networks. Moreover, new specific VoIP security vulnerabilities come as a result of the merger of the IP and telephony worlds. The fact that signaling and data travel over the same network is new and specific to IP Telephony and can be the reason for severe security problems, since attacks on the signaling become easily feasible.

This Project Work aims at investigating the possibility to modify and enhance an existent network security technology - Intrusion Detection and Prevention, to solve specific security problems in a VoIP environment. The underlying logic for this investigation is the fact that in VoIP environments most of the threats are the same that we face in an IP network, with some additional VoIP specific vulnerabilities. Therefore our aim is to analyze specific VoIP attacks and look for respective countermeasures, based on Intrusion Detection and Prevention. The work is based on and can be described as an extension and analysis of SIP (session initiation protocol, dealing with the signaling) Intrusion Detection and

Prevention prototype software developed by NEC Labs, Europe. Additionally, we try to estimate the applicability and feasibility of utilizing Intrusion Detection and Prevention technology to enhance VoIP security.

This work is organized as follows: In Chapter 2 we present some background information about VoIP basics, communication scenarios, and relevant security threats. In Chapter 3 the Session Initiation Protocol is presented, before diving into SIP specific security issues in Chapter 4. Chapter 5 deals with Intrusion Detection and Prevention Systems in general, whereas Chapter 6 describes existent prototype SIP Intrusion Detection and Prevention software. In Chapter 7, we analyze some additional attacks, which could potentially be countered through the SIP IDS/IPS, as well as the potential problems coming with the solution. Chapter 8 summarizes the work and draws conclusions. Implementation of the ideas about countering additional attacks is presented in Appendix A.

2 Background

Voice over IP (VoIP), an emerging, promising technology with its advantages over traditional telephony (especially cost saving) has been a hot topic in the news for quite some time, but currently the hype is being switched to VoIP security issues, which threaten to hamper the further large-scale deployment and development of the technology. VoIP can be considered as a technology not only of the present, but also of the future, since substantial deployment is expected, especially considering the prognosis that the Next Generation Networks (all IP networks) will be a focal point of the telecommunications infrastructure in the near future. Unfortunately, the security of VoIP has not been of primary concern so far and thus there is a lot to be done. Thus a deeper investigation is needed to determine if this hype about VoIP security is well-grounded and what can potentially be done. One of the goals of this work will be to shed more light on this issue. In this chapter, I will set the background for my work, shortly introducing VoIP, VoIP security issues and Next Generation Networks, as a glance at present and at some future trends in telecommunications.

2.1 VoIP Basics

We start setting the background by explaining the principle of transmitting voice over an IP network [10], with the help of Figure 2.1. The voice is transformed into an analog electrical signal by a microphone (shown in 1). This is followed by the digitalization of the analog signal through an Analog to Digital converter (2), which could be found in any sound-card. Then the data is encoded and compressed. Now, the digitized audio signal will be divided into packets (3) and sent over the IP network (4). At the receiver's side the data received from the packets is reassembled in the correct order and decompressed (5). Then the audio data, which is in digital form, is turned back into analog electrical signal (through the Digital to Analog converter in the sound card). The analog signal can be heard by the receiver of a phone call through a speaker or earphone.

2.2 VoIP Communication Scenarios

Following the basic principle of voice over IP network transmission, several scenarios are possible for communication, utilizing the IP network and thus reducing costs. For completeness, we present these scenarios here [10].

2.2.1 Direct communication between 2 terminals over an IP network

Figure 2.1 shows the simplest scenario of VoIP, when two VoIP terminals, e.g. PCs with soundcards and SIP User Agent software connect directly over an IP network. Here, there are two cases, with or without a Call Server(CS). A CS, shown in Figure 2.2, is needed in the more general case (e.g. a SIP proxy server) for the connection establishment, i.e. signaling, connection and service monitoring. The

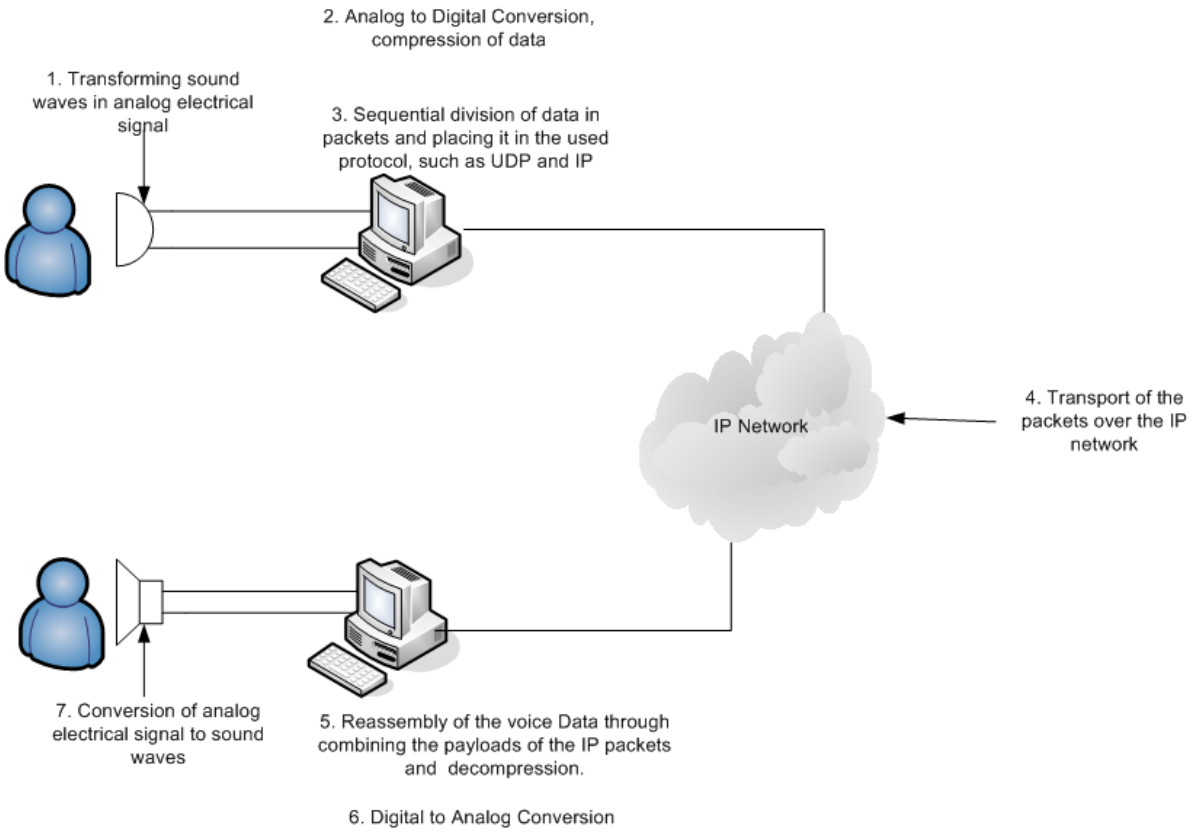

Figure 2.1: VoIP Basics [10]

VoIP terminals need to discover each other's IP address, for which the call server helps as well. In the second, special case (e.g. in a small network, where the VoIP Terminals know each other's IP address) the Call Server may not be needed. In this case the functionality of Mobility Management over an IP network is fully decentralized.

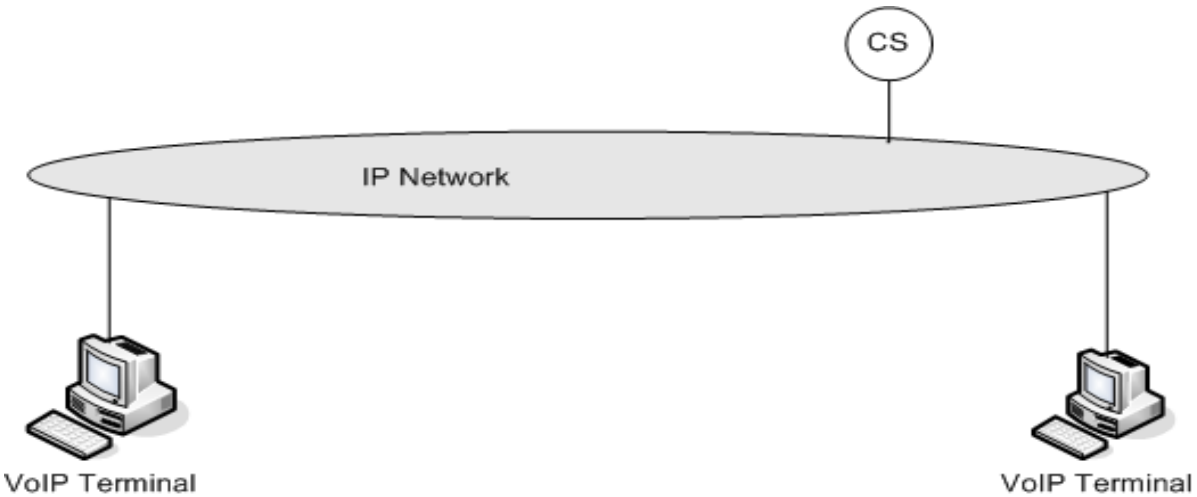

Figure 2.2: VoIP communication between 2 terminals, directly over an IP network [10]

2.2.2 Communication over IP/ISDN Gateway

In this scenario, shown in Figure 2.3, we see the interconnection of an IP network and an ISDN network. The communication between the VoIP Terminal and the ISDN-telephone is realized through a Gateway. The signaling between the gateway and the VoIP terminal may go directly or through a CS, which needs to have a Media Gateway Controller (MGC) functionality to communicate with the Media Gateway (GW in the figure). For the ISDN-telephone, the role of the Gateway and the Call Server play the role and are seen as another ISDN participant.

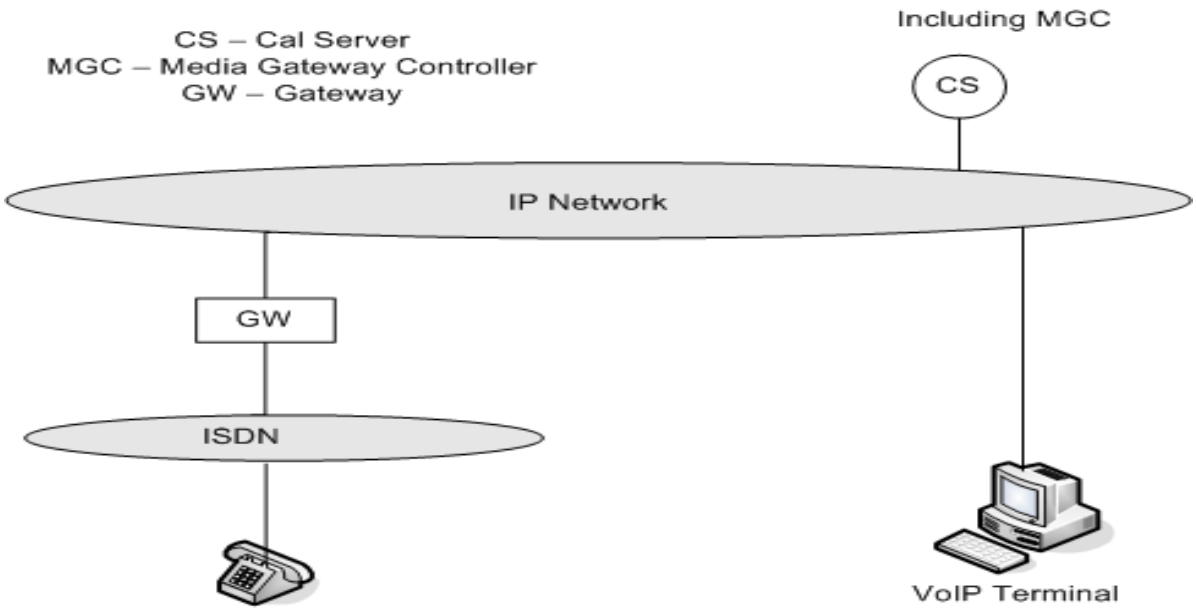

Figure 2.3: Communication between a VoIP-Terminal and an ISDN phone over IP/ISDN-Gateway [10]

2.2.3 Communication between two ISDN phones over IP network and IP/ISDN Gateways

This scenario,shown in Figure 2.4, is similar to the previous one, but here the VoIP terminal is fully eliminated. This scenario is used by telecom companies to drive down costs especially for long distance calls. The signaling can again be directly between the gateways, but that is not common. Rather, a Call Server with Media Gateway Controller functionality is used. This is needed when the Media Gateways in the GW are remotely controlled. For the ISDN telephones, the IP network is transparent.

2.3 VoIP security issues

Considering VoIP deployments, there are several important implications that should be noted [2]: a VoIP service provider needs to ensure privacy and access (e.g. for legal purposes) to information, maximize the service availability, keep the costs low, extend services to local, remote, mobile users, keep up with the legal issues. For all these concerns security is critical.

On a high enough level, the major security concerns for VoIP are confidentiality, integrity and availability. One of the main advantages of VoIP networks is utilizing a data network for transmitting voice and thus enjoying advantages in cost and ease of management. But on the other hand, such a deployment

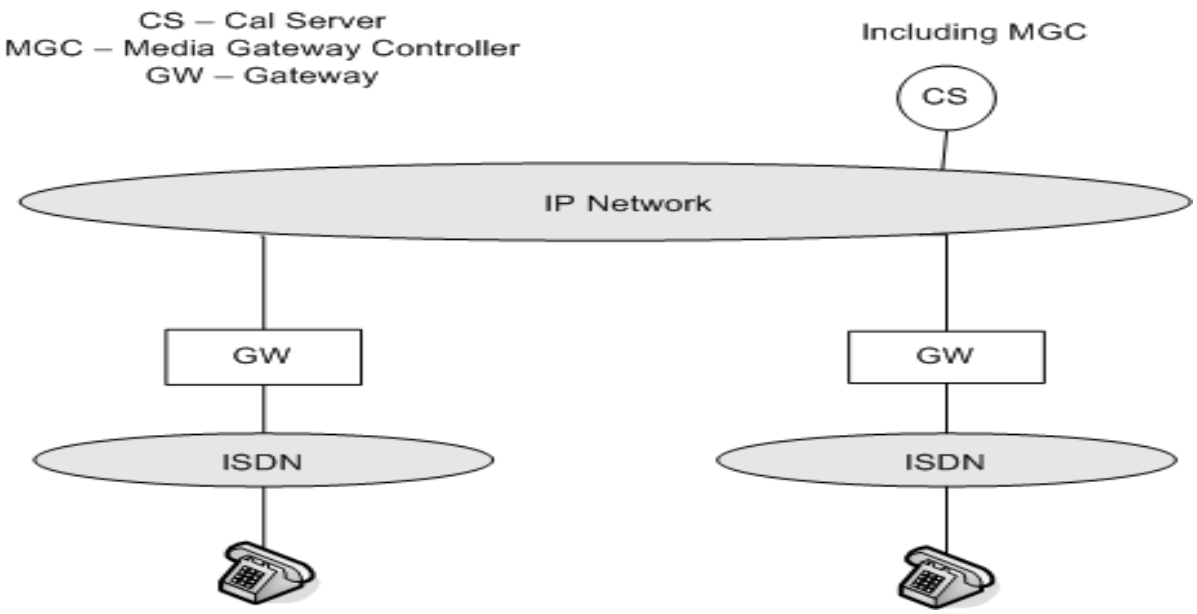

Figure 2.4: Communication between two ISDN phones over an IP network and IP/ISDN Gateways [10]

would make the voice network vulnerable to the same security threats and attacks that a data network usually suffers from. These include the well-known attacks such as DoS, DDoS, authentication attacks etc. Additionally, a voice network is generally vulnerable to toll fraud, privacy is extremely important and the offered QoS needs to be at an acceptable level. A major problem in terms of security for VoIP deployments lies in the VoIP specific protocols: there are many vulnerabilities, based on the signaling and media stream protocols (SIP and RTP respectively).

SIP, for example, transmits its packet headers and payloads in clear text, allowing attackers to observe and modify certain packets (breaching confidentiality and integrity). This can result in attacks like session tear down, call redirect, toll fraud. In addition RTP (Real Time Transport Protocol used for media transmission) does not provide encryption and authentication and thus an RTP flooding attack may result (raising an availability concern), by an attacker sending “malicious” packets to degrade conversation quality possibly to an unacceptable level. It is possible, for example, to bring down a terminal device by sending a carefully-crafted, malicious RTP packet. Most importantly for a voice service, confidentiality can easily be compromised, since an attacker sniffing the network and capturing RTP packets at an appropriate place in the network topology would be able to reconstruct a conversation. In addition, a conversation could be replayed or artificially constructed by combining words, which would breach integrity.

A solution to the previous problem is available with the proposal of SRTP (secure RTP) protocol, which is unfortunately not widely deployed yet. To make things worse, using encryption for VoIP will induce overhead and may cause a problem to supply the necessary QoS.

2.4 Next Generation Networks (NGN)

To additionally motivate our work, we take a look at Next Generation Networks, which may be considered the future of telecommunications, due to their capability to take into account new conditions,

such as open competition between operators due to market deregulation, huge amount of digital traffic, increasing demand from users for new multimedia services and for general mobility, etc.[11]. Securing VoIP is in a way a subset of securing voice over these NGN, which will probably be very important in the future.

A Next Generation Network is a packet-switched network able to provide services, such as telecommunication services, for example, and able to make use of "multiple broadband, QoS-enabled transport technologies and in which service-related functions and the underlying transport-related technologies are independent from each other" [11]. NGN offers unrestricted access by users to different service providers. It supports generalized mobility which will allow consistent and ubiquitous provision of services to users[11]. The NGN is characterized by the following fundamental aspects:

- Packet-based network
- Separation of control and carrier functions: call/session, and application/service
- Decoupling of service provision from network provision – can be offered separately and evolve independently
- Wide range of services, applications and mechanisms based on service building blocks (including real time/streaming/non-real time services and multi-media)
- Broadband capabilities, end-to-end QoS
- Open interfaces to interwork with legacy networks – PSTN, ISDN and GSM
- Generalized Mobility - the user is considered the same, no matter what access technology is used
- Unrestricted access by users to different service providers
- A variety of identification schemes which can be resolved to IP addresses for the purposes of routing in IP networks
- Unified service characteristics for the same service as perceived by the user
- Converged services between Fixed and Mobile
- Independence of service-related functions from underlying transport technologies
- Compliance with all regulatory requirements, for example concerning emergency communications and security/privacy, etc.

Finally, in the requirements for NGN, we notice the generalized requirements in terms of security, already presented in our discussion of VoIP security: the security mechanisms to protect the exchange of sensitive information over its infrastructure, to protect against the fraudulent use of the services provided by the Service Providers and to protect its own infrastructure from outside attacks.

3 Session Initiation Protocol (SIP)

The Session Initiation Protocol (SIP) was first defined in 1999, RFC 2543 and later updated in 2002 in RFC 3261. SIP together with H.323 (developed by the ITU) are the two signaling protocols used for VoIP, but SIP will most probably dominate the market in the future. SIP has been chosen by various standardization committees (IETF, ETSI, 3GPP), e.g. it is the standard for signaling in UMTS release 5, moreover most providers offering VoIP service to consumers offer SIP-based solutions.

SIP is an application-layer signaling and control protocol for multimedia communications over IP; it can be used to create, modify and terminate sessions with other participants. Additionally, SIP offers support for inviting new participants to already existing sessions, thus enabling conference calls. SIP allows participants to negotiate a session based on a session description and thus agreeing on a set of compatible media types. Based on SIP proxies, requests are routed to the user's current location; users can also be authenticated and authorized for receiving specific services, implement call routing policies, provide additional features. SIP also provides a registration function, allowing users to provide their current location to a global location service used by the proxies. SIP does not depend on the underlying transport protocol and can use TCP, UDP or SCTP.

3.1 SIP architecture

In the SIP specification five types of entities are defined: user agent (UA), SIP proxy, SIP registrar, SIP redirect server and SIP location server. The distinction between the servers is logical and not physical. Figure 3.1 [3] illustrates the types of SIP entities in a sample of SIP architecture.

Before discussing the entities, it would be useful to explain SIP addressing. SIP clients are identified through SIP addresses, which are based on uniform resource identifiers (URI). The long format is "sip:user:password@host:port;options", signifying as the names suggest the protocol, user, password, host at which the user is, port at which contact can be established and additional options, such as which transport protocol to use. Only the part "sip:user@host" is mandatory though and generally SIP URIs look like the latter.

A **User Agent** is a logical entity and it can be any terminal, which receives or sends requests, i.e. interacts with other entities through the SIP protocol. A User Agent can act as both User Agent Client (UAC) and User Agent Server (UAS). A UAC is the logical part of a UA which generates a new request and then sends it. Then for the duration of this transaction it acts as a UAC, but the UA could act as a UAS for a different transaction. A User Agent Sever is also a logical part of a UA, but it generates a response to a request. The response can possibly accept, reject or redirect the request. The role (UAC, UAS) lasts until the end of the transaction.

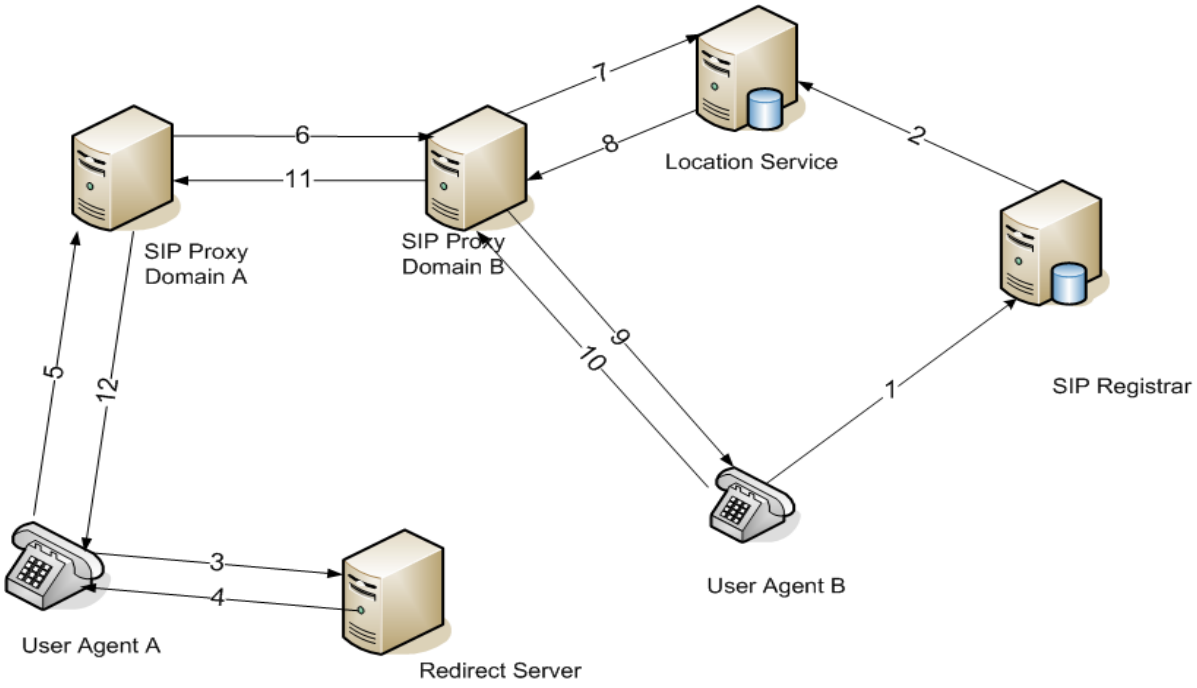

Figure 3.1: Possible SIP architecture [3]

A **SIP proxy** can receive both requests and responses. For a request, it must check the message validity and only if the massage is found valid, process routing information, determine targets, forward to these targets. On the other hand, a SIP proxy must process all responses.

A **SIP redirect server** returns to the sender of a message another address, to which the message should be sent instead of forwarding the message itself.

A **SIP registrar** in a specific domain provides registration service to the UAs belonging to that domain, thus mobility can be enabled. A user (identified through SIP-URI) reports his/her current IP address (location) to the registrar server in the domain. The registrar then updates this information in a *SIP location server* (global database), which stores the location of the users (maps a SIP URI with current IP address of the device being used by a user). Thus mobility is enabled and it is possible to redirect or forward the requests to the right user, when location changes.

3.2 SIP Messages

SIP is similar to HTTP, a client-server type of protocol based on UTF-8 text messages. Each message in SIP is either a request (“client” to “server”) or a response (“server” to “client”). A message consists of a message header (start line, header fields) and an optional message body. There are several request types defined in RFC 3261 and shown in Table 3.1. After a server (SIP entity) receives a request, it processes the request (if possible) and then returns a response to the request sender. Similarly to HTTP (v. 1.1), there are 6 ranges of response codes, designating success, client failure, server failure, redirection etc. as seen from Table 3.2.

SIP request	Purpose
INVITE	Initiate a signaling for conversation
BYE	Terminate connection between users in a session
ACK	Acknowledgement for an INVITE request
OPTIONS	Determine messages and codecs a UA or Server understands
CANCEL	Cancel a pending INVITE, does not influence completed request
REGISTER	Register location at a SIP registrar

Table 3.1: SIP Requests [3]

SIP Response Codes
1xx - informational
2xx - OK
3xx - redirection
4xx - client error
5xx - server error
6xx - global failure

Table 3.2: SIP response ranges [3]

3.3 Mandatory Fields

For a specific SIP request to be valid, the following mandatory fields must be present: To, From, Call-ID, CSeq, Max-Forwards and Via.

The **"To"** header field contains the logical recipient of the request or the address of record of the targeted user or resource.

The **"From"** header field contains the logical identity of the initiator of the request (usually user address-of-record), which may not be the initiator of the dialog. For example, a callee will put its own address in the From header field in a request sent by the callee to the caller.

The **"Call-ID"** header field is a unique identifier for a particular invitation or a registration of a client. It is used to set apart a group of messages, e.g. all messages from a dialog or all registration messages of a particular client will be characterized and identified by the same call-ID.

The **"CSeq"** header field serves as a way to order transactions within a dialog, uniquely identify transactions and differentiate between new requests and retransmissions. CSeq consists of a single, decimal sequence number (expressible as a 32-bit unsigned integer) and a method name part (case sensitive).

The **"Max-Forwards"** header field limits the number of hops a request can make on the way to the destination. It is an integer, which is decremented at every hop. When it reaches 0, the request is rejected with 483 error response – "too many hops".

The **"Via"** header field describes the transport used for the transaction and the location where the response is to be sent. The route of the request until a given time is recorded as well. In case more than one Via field is present, the message should be discarded, since it was probably misrouted.

3.4 Basic Signaling Steps for Session Establishment

To clarify how signaling with SIP and all the SIP entities actually works, we present a session establishment procedure referring again to Figure 3.1. The following steps occur:

1. User Agent B (UA B) registers with the Registrar in the domain. This is needed, so that UA B will be able to receive calls. The registration binds the SIP URI with the IP address of the device, on which B is present currently.
2. The SIP registrar updates the Location Service, which is a global database helping a proxy to resolve SIP URI's locations.
3. User Agent A (UA A) wants to establish a session with User Agent B and thus sends and INVITE message to user agent B. The server that receives the request though is a redirect server and not the proxy for the domain.
4. A redirect server does not forward a request, instead it returns a message with the "proper" address, where the message should be sent to, in this case the SIP proxy server of domain A.
5. The INVITE message is sent to the SIP proxy server of domain A.
6. The SIP proxy checks the validity of the message and forwards the message to the SIP proxy of domain B. In most cases the message is routed through several proxies, but here for simplicity we show direct forwarding. The other scenario would not be much different.
7. The SIP proxy of domain B receives the INVITE request with a specific SIP URI, but has to check the respective location (IP address) of the invited party. This is done thorough a message to the Location service.
8. The Location service returns the respective location.
9. The SIP proxy forwards the request to User Agent B.
10. A RINGING message is sent back to the SIP proxy in domain B.
11. A RINGING message is sent back to the SIP proxy in domain A.
12. A RINGING message is sent back to User Agent A.

To complete the session establishment, UA B sends a 200 OK message to UA A (when the phone is picked up), to which UA A sends an ACK message. The process of session establishment itself is described, based on a simple call flow in Figure 3.2. After the initial INVITE response, the proxy (proxies) returns the informational 100 TRYING response. After the UA B (party invited for a session) receives the INVITE, it sends back a RINGING message, as explained before to notify the UA. When the call is answered, a 200 OK message is sent back from the callee, which is answered with an ACK from the caller to the callee (directly, since the IP addresses are known). After that, the session is established and the conversation can begin, signified on the Figure by RTP flow (media flow). When User Agent B, for example, decides to hang up he phone, she sends a BYE request to User Agent A, which is acknowledged by 200 OK. The conversation ends.

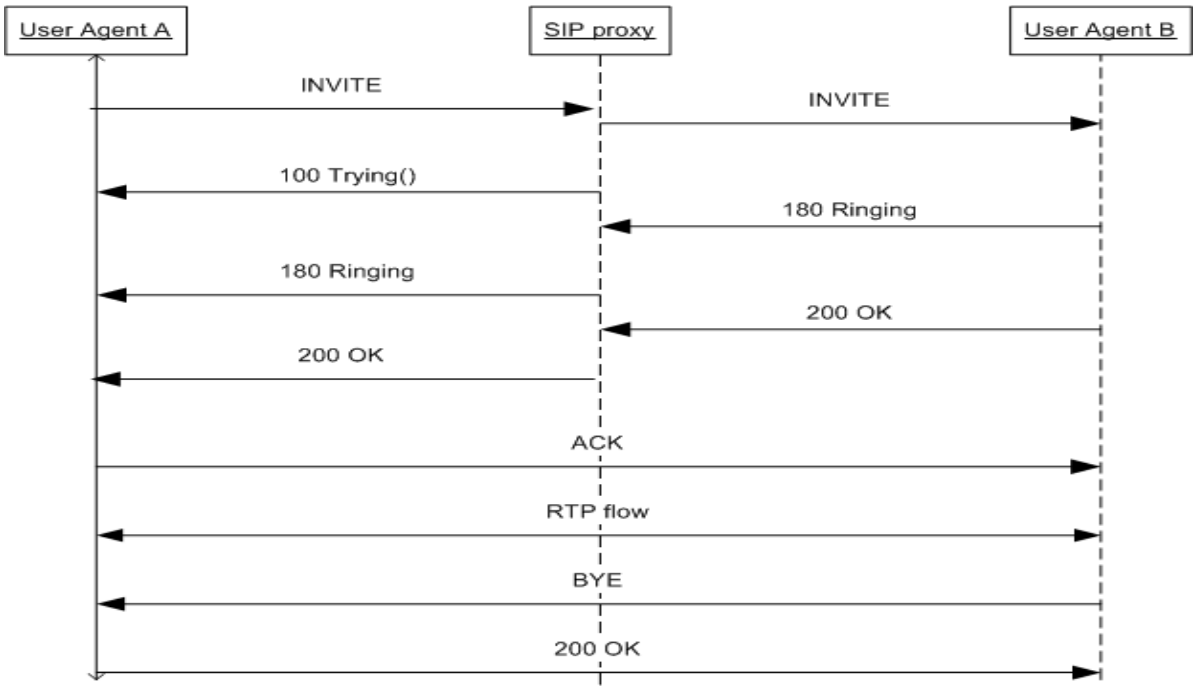

Figure 3.2: Sample SIP call flow

4 SIP Security Issues

As SIP is expected to be the prevailing signaling protocol of the (at least near) future and it is gradually turning into the de-facto standard, it is important to explore the main security implications, which are brought by the standard in the context of a taxonomy, developed by the VoIP Security Alliance (VOIPSA). VoIP provides a similar service to PSTN and is expected to achieve at least a similar level of availability, reliability, security and privacy. In terms of security and privacy, there are many complications compared to PSTN, where the problem is solved relatively well [1].

In PSTN, the service provided by an operator is based on controlling the routing in the network and switching lines between endpoints. The security is mainly physical, based on the restricted access to the dedicated telephony equipment. The parties, involved in a conversation are basically using sockets in private homes/institutions and having the sockets implies authorization to use the network. This works relatively well due to the need for physical access to explore vulnerabilities. The signaling channel is separated from the payload channel, which protects from possible manipulation on behalf of end devices [3].

Providing voice service over IP changes the situation drastically. The access to the communication channel is very easy and this is a major problem. Thus eavesdropping of unencrypted traffic (most deployments do not encrypt traffic) is a feasible task for an attacker at the right place in the network topology, with open-source tools like Ethereal, VOMIT etc. Moreover, SIP messages are text based, which gives opportunities for spoofing, hijacking, message tampering. Malicious SIP messages can be used for various DoS attacks, as well as for an SQL injection attack against a SIP Registrar. Additionally, SIP uses as a transport protocol TCP, UDP or SCTP, therefore it directly inherits the vulnerabilities in these protocols such as SYN flooding, TCP session hijacking etc. Other issues can be flaws in the software implementation of the SIP protocol, allowing specific messages to cause unauthorized access or crashes in the devices. Additional problems may occur when interacting between SIP and gateways to PSTN.

In this work, we focus on several types of attacks, classified by the VOIPSA (VoIP Security Alliance) taxonomy [4]. This Taxonomy divides the possible attacks into several categories: Social Threats, Eavesdropping, Interception and Modification, Service Abuse, Intentional Interruption of Service.

4.1 Social Threats

4.1.1 Unsolicited Calls - SPam over IP Telephony, SPIT

This is a yet non-existent attack, which has succeeded to draw a lot of attention both in research and in the news. A possible functioning would be the following: a script/tool appears that would iterate through SIP addresses (1001@myvoip.org, 1002@myvoip.org, 1003@myvoip.org . . .) and will start an audio stream, whenever a call is answered. This audio stream can come from a marketer

or could be a prerecorded message. It could supposedly be used by telemarketers, prank callers, script kiddies to initiate unsolicited and unwanted conversations, often with the mentioned prerecorded voice messages.

Today VoIP networks are generally small islands, e.g. within certain companies and are interconnected with PSTN through gateways, but in the future this trend will possibly change. Companies might want to directly connect IP-PBX systems to the Internet and thus allow incoming calls from any IP address. Until that time, telemarketers can only initiate unsolicited calls through the PSTN, which costs them and is thus not popular at all [2].

4.1.2 Theft of Services - Toll Fraud

This attack is not new in telephony and in a VoIP context would probably as well be a very popular one to exploit in the near future. The attack can be generally be described as an unauthorized user gaining access to a VoIP service, by impersonating a legitimate user. It can also be taking control over an IP phone and initiating outbound long distance calls. Thus, if authentication and authorization services of a VoIP operator are not at the proper level, this might lead to some losses from hackers, script kiddies etc. placing international (long-distance) calls from an IP phone through the PSTN connectivity. Losses for the service provider and/or legitimate users would result. Considering the SIP protocol, an attacker could use a SIP REGISTER message to impersonate a legitimate user and then take advantage of the respective user's services. To achieve this, a specific IP address needs to be spoofed and headers of the REGISTER request tampered.

4.1.3 Misrepresentation - identity, authorization, content

Misrepresentation is defined as delivering false information with respect to the identity, authority (rights) of another party or the content of information communicated [4].

Identity misrepresentation is intentional presentation of a false identity, as if it were true with the purpose to mislead. It can be a false caller id or number, false voice, name or organization in a voice/video mail, presentation of a false email, or false presence all with the goal to mislead.

Authorization misrepresentation is basically lying about what one is allowed to do - intentional presentation of a false authority as if it were true with the goal to mislead. It can be presentation of a password, key or certificate of another, circumvention of conditional access, bypassing ordinary authentication, all of them with the intent to mislead. If one successfully succeeds in Identity misrepresentation, that automatically means success for Authorization misrepresentation.

Content misrepresentation is intentional presentation of false content, pretending that it were true. This includes impersonation of the voice or specific words of a user, modification of spoken, written or video content with the goal/intention to mislead. As illustration, we present two specific SIP attacks.

4.1.3.1 Registration Hijacking

The SIP REGISTER message allows a user to register a device and then receive specific services. The registrar checks the identity in the From header field to assess, if the request is allowed to modify the registration for the address-of-record in the To field. The problem is that lack of authentication

and integrity protection of the message (which is often the case, since these are not mandated in RFC 3261) allow the From field to be changed by a malicious party and thus that party could register unrightfully itself or a third party. This attack could only be performed by an attacker, who is able to observe signaling traffic and de-registers a legitimate user (Sending register message with “Expires” = 0, signifying the time interval after which the registration expires) and then registers his/her own device with the credentials of the legitimate user. Thus both impersonation and DoS against the legitimate user can be achieved. Calls for the legitimate user will be received by the attacker. The attack would succeed in the case of lack of integrity protection of messages and proper authentication. The attack is an example of identity and thus authorization misrepresentation, which can be the basis for content misrepresentation as well.

4.1.3.2 Server Impersonation

A redirect server can impersonate another SIP server, by redirecting calls by a UA to a specific, legitimate server to itself (malicious behavior) or to any other SIP server (again malicious). To prevent this kind of threat, a UA has to have a means to authenticate a server it communicates with. Considering the other attacks, mutual authentication between a UA and a SIP proxy server is highly necessary. Again, we have an example of identity and thus authorization misrepresentation.

4.2 Eavesdropping

Eavesdropping is a passive attack aiming at observing the signaling and/or media stream between two or more VoIP endpoints, without changing the transmitted data (the classical meaning of a passive attack). The attacker must be in the communication path. In VoIP, this is listening to a call or playing it back, or obtaining interesting packet fields (e.g. SIP header fields). This can be a basis for subsequent attacks of other types.

4.3 Interception and modification

This class of attacks assumes the attacker can see the entire signaling and media stream between two endpoints, and can modify the traffic as an intermediary (MITM) in the conversation or possibly even insert or replay “non-existent” traffic. The interception of conversations means lack of confidentiality. Conversation/signaling could be modified, which implies lack if integrity. Private information (user-names, passwords) may be learned.

4.3.1 Man-In-the-Middle Attack

An attacker successfully inserts himself/herself into the session between two parties, who are not aware that they do not communicated directly with each other, but the packets are routed through a third party, who has access and can possibly compromise the integrity of certain messages. Being able to read and possibly modify the correspondence is a crucial point for further attacks.

4.3.2 Message Tampering

A malicious proxy server on the route between two user agents can modify headers or message bodies of the signaling or the media streaming protocol. In SIP, this is viable, since often encryption and integrity protection are not used. This can also be a problem, if applied to SIP message bodies, communicating session encryption keys. The keys could be modified by an attacker acting as a MITM or the attacker could possibly change the security characteristics requested by an UA. Thus, it is generally a good idea to secure certain headers and message bodies end to end, so that integrity and confidentiality would not be compromised.

4.3.3 Tearing down Sessions

An attacker, who is capable of sniffing and sending packets on an IP network can also perform Session-Tear-Down attacks, meaning that the session is terminated by a request sent by the attacker and not by one of the legitimate participants. In SIP this can be achieved by sending BYE or CANCEL messages to a participating party. To accomplish this, the attacker needs to find out the parameters of a session, namely the To, From, CSeq and Call-ID fields. Then a BYE message can be inserted, tearing down the session. It's also possible to change session parameters by using a re-INVITE or UPDATE message, when the session is not established yet. This can lead to a broken session of a legitimate user by an attacker presenting a different IP address in a re-INVITE or UPDATE message. Obviously, strong authentication mechanisms of the sender of BYE, CANCEL, UPDATE and re-INVITE messages are needed to prevent this attack.

4.4 Service Abuse

Service abuse means improper use of services, unauthorized or unaccountable use of resources, fake identity, impersonation, session replay. This attack is much more general compared to toll fraud attacks, since this is any usage of service in a way it was not supposed to be used.

4.5 Intentional Interruption of Service

Denial of Service (DoS) and Distributed Denial of Service (DDoS) - the availability of a Call Server or a UA can be threatened by an extraordinary amount of artificially-generated requests by an attacker, which can lead to a lack of enough capacity to serve legitimate users. The attack is relevant for the SIP protocol as well.

4.5.1 Denial of Service

DoS prevents the normal use of the VoIP infrastructure on behalf of legitimate users that can be severe up to an extent where the service stops functioning completely. The target may be a specific server or a UA, but the goal could as well be degrading the usability of the entire network by overloading it. Specifically for the SIP protocol, an attacker could launch an attack from SIP servers by generating a large amount of RTP sessions toward clients. The attack is successful even if the device requires authentication, since the state is kept until the device waits for the authentication during the session establishment. However, the IP addresses of the victims have to be known for this attack to succeed

[1]. The SIP specific DoS attacks are several and they can be characterized by the name of the method they use.

4.5.1.1 CANCEL/DECLINE attack

The attacker is able to observe the signaling traffic, usually at a SIP proxy. After an INVITE message is sent to a callee, the attacker sends a DECLINE back to the caller. Potentially, the attacker could also send a CANCEL message to the callee (destination) to stop session establishment.

4.5.1.2 BYE attack

The attacker terminates a call at any moment, especially easy when authentication mechanisms are lacking or are weak.

4.5.1.3 REGISTER attack

The attacker deregisters a legitimate user's device and registers his/her own instead, thus impersonating the user and in fact executing DoS attack against the respective legitimate user.

4.5.1.4 SIP INVITE flooding attack

Attacker sends INVITE messages to a large number of users with spoofed IP address, which is basically the user target of the attack. The user (whose IP was provided as source IP) will receive response messages from all users who receive INVITE. Flooding attacks are possible with other messages as well, such as with REGISTER, OPTIONS etc.

4.5.2 Distributed Denial of Service

This is performed by launching simultaneous and coordinated DoS attacks from a large number of hosts (innocent public Internet hosts), which are under the control of an attacker. The first stage of such an attack is injecting hidden programs in hosts, making the hosts controllable (zombie hosts). The second part is the actual launching of an attack, which could be done automatically at a specified time or controlled by a master host, coordinating the attack.

In conclusion, to counter the attacks on SIP messaging discussed so far, several security requirements on the SIP messaging are needed [3]:

- Message Confidentiality – to prevent eavesdropping on message headers and bodies
- Message Integrity – to prevent alteration of message headers and bodies
- Authentication of SIP nodes – against spoofing, server impersonation, registration and call hijacking, tearing down sessions
- Mutual authentication between SIP node and SIP proxy server – against the ones discussed in the Authentication of SIP nodes and additionally against Server Impersonation attacks.
- Privacy of SIP traffic
- Availability – of UAs and SIP proxies

5 Intrusion Detection Systems

In this work, we explore the possibility of upgrading an IDS system to help against VoIP specific (mainly SIP signaling, but possibly RTP-specific as well) attacks and for the enhancement of the overall security in VoIP environments.The meaning of intrusion in computer and network security can be defined as a sequence of related events that deliberately try to cause harm, such as rendering a system or a network (also parts of it) practically unusable, providing an opportunity for unauthorized access to services or information, manipulation of confidential information etc. [1]. The goal of intrusion activity is generally to compromise confidentiality, integrity and/or availability of a system or a network segment, possibly by bypassing the implemented security mechanisms and/or increasing the range of privileges an attacker has on the system.

We can define an Intrusion Detection System as comprising the tools, methods, techniques and resources to identify, assess and report unauthorized or unapproved activity at the network or a host level [5]. An IDS can be software, hardware or a combination of both used to detect intruder activity in an automated way. Intrusion detection is based on the general assumption that the behaviors of an intruder and that of a legitimate user are substantially different from each other and this can be measured and determined. Nevertheless, it is impossible to have a crystal clear distinction between the behavior of an intruder and that of a legitimate user, but rather there is an overlap between them. Thus we can observe “false positives”, i.e. cases when an authorized usage of a system is identified as an intrusion. An attempt to limit these “false positives” by tighter definition and interpretation of what intruder behavior is would on its turn bring “false negatives”, i.e. intruder activities not recognized as such [6].

We can classify IDS based on several criteria [1]:

- Information sources – according to where the sources of event information come form – e.g. from a host, network or an application.
- Analysis – according to the main principle, used to analyze events so as to determine if intrusion is present or not. The most common types are knowledge (misuse) and anomaly (behavior) based detection.
- Response – according to the type of actions that the system takes, once intrusion is detected. Generally, the IDS can be either active or passive. Active means automated intervention, such as dropping packets, whereas passive are usually logging or sending message/alerts to someone (usually system administrators), who is responsible to take action.

5.1 IDS differentiated by Information Sources

Several categories of IDS exist, characterized by what exactly is being monitored in the specific approach to discover intrusion detection.

5.1.1 Network-based IDS

A Network-based IDS (NIDS) is usually placed inline on a network and analyzes raw network packets thus searching for attacks. An NIDS can protect an entire network or a subnet of the network. The Network Interface Card (NIC) of a NIDS must be in promiscuous mode, i.e. it accepts and forwards all packets, not only the ones which have the same destination address as the machine in which the NIDS runs. Thus a NIDS listens to the traffic in real time and analyzes it. Often times the architecture of a NIDS consists of sensors and hosts placed on different, strategic places in a network. They perform local analysis of the traffic and send reports about attacks to a central management console, where more sophisticated analysis of the gathered data is performed.

5.1.2 Host-based IDS

A Host-based IDS (HIDS)is installed as an agent on a single host. A HIDS protects only the machine, on which it resides. The NIC is not in promiscuous mode. It generally is software-based and looks into the operating system audit trails and system log files to detect any intruder activity. Whenever there is a change in these log files, a comparison is made with some predefined policy and possibly an alert is fired if a discrepancy is found. This can either be done in real time or the log files can be checked periodically. Operating system audit trails are generated at the kernel level, which implies more details and better chance to be authentic, thus these should be the preferred type of files to analyze. System logs can be easier to analyze, due to lack of so much detail, but may as well be modified by a skillful attacker. A subset of HIDS are Application based IDS. They analyze the events happening in a software application. The source for analysis is usually the application's transaction log files. The analysis is generally based on finding authorized users exceeding their authorization [1].

5.2 IDS differentiated by Events Analysis

The two basic ways for the analysis of events determine two types of intrusion detection: misuse and anomaly detection. Misuse detection looks for patterns of malicious activity which are predetermined to be "bad" and signify attacks. These are also called "signatures" of attacks. Most commercial products nowadays are based on misuse detection. Anomaly detection, on the other hand, looks for behavior that considerably deviates from the normal one to spot attacks. Currently, it is a matter of active research.

5.2.1 Misuse Detection (knowledge-based)

The main principle of misuse detection is to define a set of rules that can be used to decide that the given behavior is malicious. These rules are also called signatures. The rule-based system is used to indicate that known attacks are happening. The misuse detection IDS is only as good as its database with attack descriptions is. This means the database needs to be constantly maintained, finding and respectively updating the newest attacks' signatures. The problem is that often there are attacks, which are slight variations of each other and would be very difficult to detect by the exact signatures, on which this approach is based. Therefore, new attacks, based on this approach will be undetected for some time at least, until they are known, analyzed and understood, i.e. until the pattern (signature) is determined; this is a major disadvantage of knowledge-based detection. Since the patterns corresponding to the attacks are called signatures, this type of detection is often called "signature-based".

5.2.2 Anomaly Detection (behavior-based)

The principle of detection here is that abnormal or highly unusual behavior on a network or host signifies an attack. Thus the system needs to establish what normal behavior is, by constructing profiles for users, networks and hosts based on data collection and usually statistical techniques. Then again statistical tests are applied to determine with high level of confidence whether the new collected data represents legitimate behavior or not. The techniques used may include:

- Threshold detection – behavior of users, hosts, and networks can be counted and certain levels of activity can be established as permissible. The activity could be the number of failed logins per user, the number of allowed INVITE messages from a UA etc. This technique is used in some commercial systems.
- Statistical detection – it can be parametric and non-parametric. Parametric statistical detection is based on analysis of the observed distribution in comparison to a distribution that is assumed to be the normal one. Non-parametric statistical detection – the distribution is learned from historical data and significant changes are observed and noted as intrusion. This type of detection is also used in some commercial systems.
- Rule-based measures – observed data determines which patterns are acceptable and which are not, the patters are specified as rules and not as numbers, unlike non-parametric statistical detection.
- Neural networks, genetic algorithms, data mining, machine learning based algorithms – mainly in a research phase.

The anomaly detection approach is problematic, since a substantially larger number of false alarms are produced, compared to misuse detection. The reason is that user behavior can vary widely, change over time, or an attacker can gradually change what the system finds to be normal, by e.g. increasing the number of REGISTER messages sent to a registrar gradually. The major advantage of anomaly detection techniques is the fact that they supposedly discover new attacks, unlike misuse-based systems.

5.3 Intrusion Prevention Systems

An Intrusion Prevention System can be viewed as a combination between an IDS and access control (firewall or router). An IPS is a device, which exercises access control to protect networks and hosts from exploitation. The main advantage over IDS is that a network based IPS sits inline and drops malicious packets, thus protecting the network against intrusion more effectively. On the other hand, IDS can only detect malicious traffic and send alerts. Thus, when a system is exploited, the IDS may detect the exploit and fire alerts, but cannot block packets, reset connections etc. As a result, it might take a considerable amount of time until the administrators discover the exploit and take measures. Therefore the concept of prevention, an active response to an attack seems to be an attractive solution. The problem is that packets should be dropped only when we are sure that an attack is happening. That means that false positives need to be practically eliminated. In the opposite case, the IPS can turn out to be a sort of DoS attack against legitimate users. Moreover, since all packets are checked and the algorithms may be complex, the performance should be considered, the packets should not be delayed

excessively, otherwise we may run into QoS problems, especially in VoIP environments, where speed is crucial.

In summary, an IPS device generally sits inline, all the packets pass through it and are inspected for the presence of attacks. Usually several detection mechanisms need to be used for increased accuracy (e.g. both misuse and anomaly based). When a suspicious packet is detected, it must be dropped, additionally all other packets belonging to the same session can also be safely dropped. Similarly to IDS, IPS can be host-based and network-based. Network-based IPS for example, are a combination of IDS, firewall and some IPS features and are very similar to the next generation "deep inspection firewalls".

Generally, an IPS consists of 3 different modules [1]. These are protocol normalization and anomaly detection, signature detection engine and statistical anomaly detection engine. The following paragraphs quickly explain the modules.

5.3.1 Protocol Normalization and Anomaly Detection

Protocol normalization is assembling the packets, which arrive at the sensor and getting a view consistent with the one that the protected system would get if the packets are allowed. This is important, since many attacks span multiple packets and can only be discovered through looking at the whole packet sequence of a session. After that the packet is decoded and checked against specific rules of the respective protocol. Any deviation from the ordinary is flagged as protocol anomaly and forwarded to the detection correlation engine, where it is analyzed with other engines until a final decision is made.

5.3.2 Signature Detection Engine

This module matches bit patterns in packets with specific signatures. Supposedly not only specific, known attacks but also unknown ones exploring OS or application vulnerabilities are targeted. The sensor monitors the validity of a TCP/IP session and provides "stateful inspection", looking at state transitions for possible attacks. By correlating the signature and anomaly detection engine accuracy is added.

5.3.3 Statistical Anomaly Detection Engine

This module monitors and records statistics on all traffic going into and out of the protected system. A profile is usually built for network segments, describing the typical levels of activity. Thus attacks, such as DoS, DDoS can be detected as extreme and rapid fluctuations from the normal behavior for a network segment. Optimally, the IPS should be able to determine which packets belong to a certain attack and drop them.

IDS and IPS are different approaches to network security, the main difference between the approaches is determinism. IDS can use nondeterministic methods to discover and log potential threats with different probabilities [1]. This can include statistical analysis, volume shaping, and anomaly activity detection. The presence of false positives is less problematic. On the other hand, an IPS must be deterministic and should only drop packets when an attack is discovered with extremely high confidence, which is not a trivial task. Otherwise, an IPS can inflict a DoS attack on legitimate users.

6 SIP Intrusion Detection and Prevention

This section is based on the analysis and practical work (SIP NIDS prototype) of Niccolini et al. from NEC Labs, Europe. It explains the approach taken and sets the background for Chapter 7, where I describe my work based on the same prototype.

Some vulnerabilities for VoIP environments come from the fact that signaling and media can take different paths. Moreover, the access to the network is not restricted (unlike PSTN) and VoIP is standardized on open technologies, such as SIP, H.323, and RTP (whose specifications are available to the general public). The entities are implemented in software and often run on general purpose computing hardware. If we add the high number of systems, that an NIDS needs to protect, we see that the challenges are significant.

Many ideas have been presented in research about how a VoIP IDS should look like to be effective and efficient. In the context of thinking about SIP IDS, it would be important not to lose the big picture and be sure what our final goal is. Thus we start with a form of wish-list for VoIP Intrusion Detection Systems. The major ones are presented by Yu-Sung Wu et. al., they are namely cross-protocol intrusion detection and stateful intrusion detection [9]. Here are the points on the list:

- **Cross-protocol Intrusion Detection** The need to relate session establishment between entities with the actual media transfer is obvious for protecting against toll fraud/billing attacks, but it can also be useful for intelligent detection of other attacks, such as session tear-down, re-INVITE, REFER attacks etc. Thus cross-protocol intrusion detection and prevention is a main abstraction, which we need to consider in future work on VoIP IDS/IPS.
- **Stateful Intrusion Detection – assemble state from multiple packets** Stateful Intrusion Detection implies that a session's state needs to be monitored, usually through a state machine, in order not to allow malicious packets with messages that would be impossible to process according to the protocol specification. These messages, whether malicious or resulting from bad software implementation should better be dropped.
- **Knowledge-based and behavior-based techniques** Knowledge-based and behavior-based techniques can be combined and share information for more intelligent attack detection, especially considering the strict requirements on IPS, where false positives should practically not occur. For example, knowledge based IDS cannot discover new attacks, but they can get information about them from behavior based ones, which can. On the other hand, if the knowledge-based part has discovered an attack, that could help the work of the behavior-based system, especially in providing attacks data for statistical analysis.
- **Use a Network-based IDS – for scalability reasons** Network-based IDS should be preferable due to scalability reasons: an enormous number of systems needs to be protected. Moreover, due to the logging nature of HIDS, they are not suited to prevent DoS attacks.

- **Deployed at entry points of (SIP) networks – to observe all traffic** The IDS/IPS has to be deployed at the entry point of the networks, to be able to observe all traffic. This entry point, especially nowadays in commercial deployments is a Session Border Controller (can also be SIP enabled firewalls and SIP gateways), through which all the signaling and media traffic passes. Of course the solution is not elegant (not efficient, single point of failure etc.), but since it is a reality nowadays, we can assume that we place our IPS system in a SBC and thus try to improve the security of the VoIP deployment.

6.1 Snort

Snort is an open-source, libpcap-based packet sniffer and logger that can be used as a lightweight network IDS. It has a detection engine, based on rules, performing content pattern matching and thus detecting attacks such as buffer overflows, stealth port scans, CGI attacks etc. Snort also provides protocol analysis (e.g. for TCP), providing a so-called "stateful inspection" for the respective protocols [13].

Snort has the option to work inline and use iptables for packet capture, instead of libpcap. In inline mode, Snort basically acts as an Intrusion Prevention System, since it can perform modification and/or dropping of packets considered malicious. Besides, it works in bridge modality, meaning it is invisible to attackers.

The prototype of a SIP Intrusion Detection and Prevention System, developed by NEC Labs, is based on Snort and can work both in IDS or IPS mode, depending on the configuration. Here, we present quickly the Snort architecture (Figure 6.1) and describe its blocks, to give an impression about how the developed SIP preprocessor fits in the general Snort architecture.

Packet capture block: as the name suggests, this block captures raw packets from the NIC and passes them to the next block, the packet decoder. If configured to work in IDS mode, Snort uses the libpcap library to capture traffic from the NIC, but in IPS, inline mode, it acts as a transparent bridge, routing all traffic at the application level using iptables libraries and the ip_queue module [8].

Packet decoder: this block is organized around the layers of the protocol stack. It determines the protocol of each packet, then checks if some conditions are met for the specific protocol and generates alerts if problems exist. The problems reported could be malformed packet headers, too large packets, incorrect TCP options. A syntax check is performed, starting from the data link layer (layer 2) and then going through the network layer and transport layer (3 and 4). A specialized decoder exists for the analysis at each layer. Speed is emphasized, pointers are set in the packet data, so that it would be available for later analysis by the detection engine.

Preprocessors: these are plug-ins, which are written in C++ and generally allow Snort to arrange or modify packets before they are passed to the detection engine, where packets are matched against signatures. The block is extremely important, since it prepares the data packets to be analyzed against rules in the next block; if not done properly, attacks can be missed. As an example, in case a hacker sends to a web server hexadecimal or Unicode characters and they are not turned into the standard form, in which the signatures are represented (usually bit patterns) the detection engine would be tricked and miss the attack. Preprocessors can also defragment packets, decode HTTP URIs, re-assemble TCP streams etc. to safeguard against hackers' techniques (such as fragmentation) used to bypass IDS.

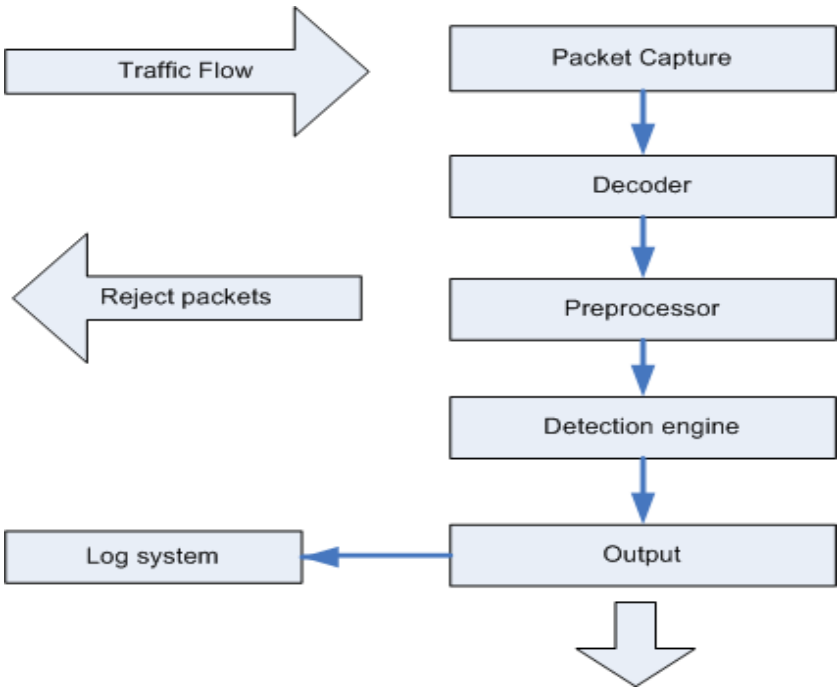

Figure 6.1: Snort Architecture [17]

These plug-ins can be very useful for analyzing protocols at layers above layer 4. The preprocessor is a very powerful block and important for us, since it can be used to implement both knowledge-based and behavior-based techniques, as it functions in stateful mode [1]. Note that the scheme in Fig. 6.1. is a bit misleading, since not all packets pass through the preprocessor mode, some packets go directly from the decoder to the detection engine block. The preprocessors are generally preconfigured and each one inspects packets from a specific protocol.

Detection engine: this block takes data from the packet decoder and enabled preprocessors and matches them with the rules in the rule-file. Here signatures and rules are checked to detect intrusion activity. If working in IDS mode, the "malicious" packets are simply logged, but in IPS mode they are dropped. Only knowledge-based intrusion detection is performed here.

Output block: manages the log output, it is configurable and can be set to text files or databases.

6.2 Importance of the Snort Preprocessors

The preprocessors give us the chance to not only utilize the pre-built tasks, such as packet reassembly, normalization, protocol-specific decoding, protocol anomaly detection, but also to add specific, user-defined functionalities and attempt detecting new attacks. Preprocessors are called once per packet of the specific protocol they try to analyze, and can call the detection engine directly with the processed data they have, if needed. Also statistics gathering and threshold monitoring could be performed here.

6.3 SIP Preprocessor

To analyze the syntax of protocols at levels higher than L4 (TCP, UDP), the Preprocessor block has to be utilized (until L4, this is done in the Decoder Block). In addition, the need to have cross-protocol detection and stateful inspection and detection [9], suggests the implementation of a SIP preprocessor and at a later stage an RTP preprocessor. This would enable us to perform syntax analysis with additional security checks for both protocols separately and eventually perform cross-protocol detection, allowing more sophisticated attack prevention. Moreover, the preprocessor gives the chance to also include behavior-based intrusion detection/prevention, starting with stateful analysis of transactions.

The purpose of the SIP preprocessor, as presented by its developers [1] is to detect DoS attacks, namely session teardown and flooding as well as detect SPIT (SPAM over IP Telephony) and do session surveillance. The preprocessor works in real-time and checks for attacks based on the SIP protocol vulnerabilities. It performs protocol analysis based on the specification in RFC 3261. Discovered problems are logged on standard output (console) and in a special log file, which can be viewed with a protocol analyzer, such as Ethereal. If configured to perform prevention, the suspicious packets are directly dropped.

The following parameters need to be specified in the configuration file before startup:

- ports: ports that the clients use, for SIP generally 5060, 5061
- calls: maximum number of calls a user can make or receive within a time period
- m_options: max number of OPTIONS requests a client can send or receive within a time period
- limit: max number of simultaneous connections that a proxy can handle
- period: duration of the observation time
- action-drop: a flag, if set to 1, it means that drop action is enabled, i.e. Snort runs in IPS mode
- proxy_ip: IP address of a SIP proxy

These parameters can be changed without recompiling the source code each time a change is done. Figure 6.2 shows the block structure of the Snort SIP preprocessor.

Only SIP packets go through the SIP preprocessor, the rest are directly passed from the Decoder to the Detection Engine (Figure 6.1). SIP packets are determined from predefined conditions on TCP/UDP port numbers, since RFC 3261 specifies SIP messages to be sent to port numbers 5060 and 5061. These parameters can be changed though, if and when a need arises. They are passed to the SIP preprocessor at start up time through a configuration file. This is done in the pre-filtered packets block. Then another filter comes, with much richer configurability options with respect to the previous one, but again with the basic idea to decide if we have a SIP packet, i.e. if it needs to be examined. After the filtering part, several types of checks are performed on the SIP packets to distinguish legitimate from illegitimate requests. They are the following:

- **SIP syntax analysis check:** the message is parsed through a function from the oSIP library called osip_message_parse. The function analyzes SIP messages, trying to verify if the message is conformant and well-formed (only certain problems of a message can be determined with certainty by this function). Since parts of the SIP messages are important and need to be kept for the further work of the preprocessor, the field data is extracted before the message is sent to the

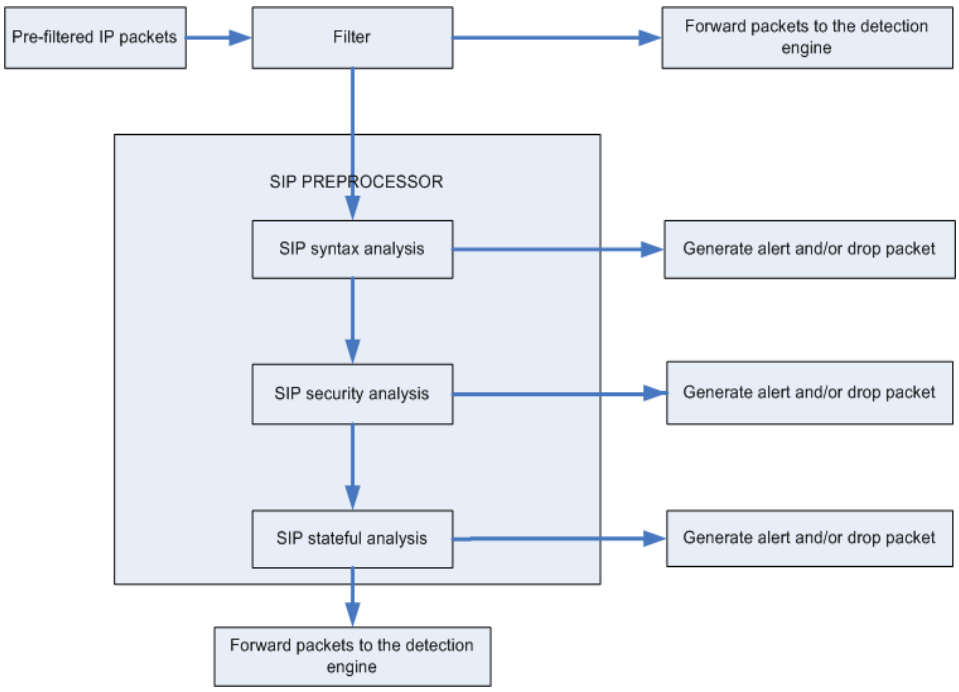

Figure 6.2: A block scheme of the SIP preprocessor [17]

osip_message_parse function. The extracted data is stored in a C structure – struct (table to hold the client connection) for further checks performed by the next blocks of the preprocessor. When the analysis results in an error, the event is logged and if the drop flag is set, the preprocessor actually drops the relative packet.

- **SIP security analysis check:** performed on mandatory fields, this is a deeper analysis after passing the initial check, checking the presence of SIP mandatory fields and their size. According to RFC 3261, the mandatory ones are Call-ID, CSeq, From, To and Via. Different checks are performed for each mandatory field.

  Call-ID: the format has to be localid@host and the preprocessor checks the presence and size of localid and host part.

  CSeq: check if the name of the method recorded here is the same as in the request line. This is done only for requests, since only they have a request line.

  To and *From*: check if the fields are conformant with the RFC 3261 specification.

  Via: check if the name of the protocol, its version and the parameter "branch" are present.

- **SIP stateful analysis check:** a state table, called "cnxmap" is kept in memory of the preprocessor if the message is INVITE, ACK or BYE. Each transaction started is kept in memory for a predefined, configurable amount of time. The idea here is to limit the number of transactions a user may initiate in a specific time period (against DoS attacks) and a stateful analysis to determine if the requests are following the predefined and expected patterns, which can be expressed by a state machine.

When a check fails, an alert is generated, that is Snort writes in a log file and if the action-drop flag is set at startup, then the packet is dropped. In conclusion we note that the preprocessor performs both

knowledge-based checks on the SIP syntax and can be a basis for further behavior checks based on transaction soft states being kept and analyzed.

6.4 Attacks the SIP preprocessor tries to detect

According to its developers [17], the IDS/IPS has been tested and found able to detect the following attacks: SPIT, DoS by flooding, session-tear-down, as well as perform session surveillance.

6.4.1 SPAM over Internet Telephony (SPIT)

A table is maintained by the preprocessor, where the necessary data to uniquely identify each transaction is kept. These are the To, From, Call-ID, and the IP source and destination addresses of the message. The preprocessor looks at each SIP message (Figure 6.3) and compares the From and To fields it finds (url1_from and url1_to), against all records in the table (url2_from and url2_to consecutively assigned the values from the table). If at least one match is found (either the url_from or the url_to or both) the check continues. This block is always executed for each SIP packet and finding a match is the initial condition, that needs to be fulfilled to enter any further attack detection/prevention. Now, supposedly a check on the respective IP addresses needs to be performed (for the URIs found to be equal). The general idea, to my assumption is, that an attacker needs to provide a valid IP address if she wants to perform a SPIT attack, since communication actually needs to be established, and providing a spoofed IP address does not make sense. On the other hand, an attacker may use several access technologies and have several Network Interface Cards with different IP addresses. This will have negative consequences, especially on a solution based on thresholds. The described check on respective IP addresses is not performed in the actual implementation, which is understandable due to the former reasons.

After the initial block common for all attacks, the actual check for a SPIT attack starts. If the message is an INVITE message and the two From fields are the same, then the counter for that specific UA is incremented and compared with a predefined threshold (the “calls” parameter in the configuration file sipcnf), defining the maximum number of connections a user may establish within a certain time interval (defined in the “period” parameter). This threshold is assigned to the variable “sipcnf.calls”. If the variable is surpassed, then there is some likelihood that a SPIT attack occurs, thus a log is generated and the packet may be dropped if configured to do so (action_drop flag is set). The logical operation of the preprocessor in trying to detect SPIT attacks is shown in Figure 6.3.

Explanation of the abbreviations in the figure:

- url1_from and url1_to are the From and To fields of the currently analyzed packet
- url2_from and url2_to are the From and To of the elements, kept in the table (of transactions)

6.4.2 DoS against an internal client, through INVITE messages

This attack detection is very similar to the previous one, the only difference being that the current To field of the packet is checked for equality against records in the table. Figure 6.4 presents the logical operation for preventing this attack. In case the threshold is surpassed, a log is generated and if configured to do so, the packet is dropped.

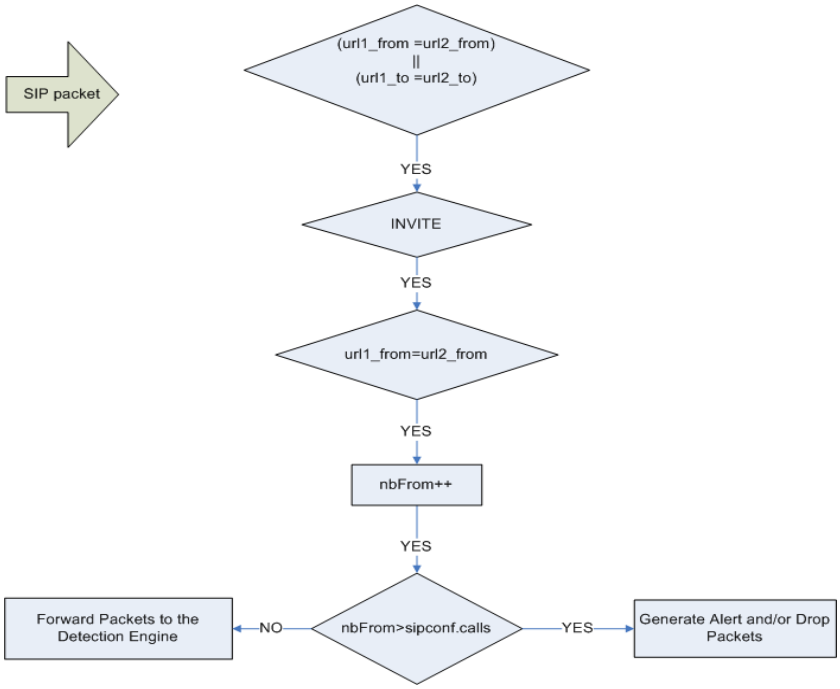

Figure 6.3: Logical Operation in attempt to prevent SPIT attacks [1]

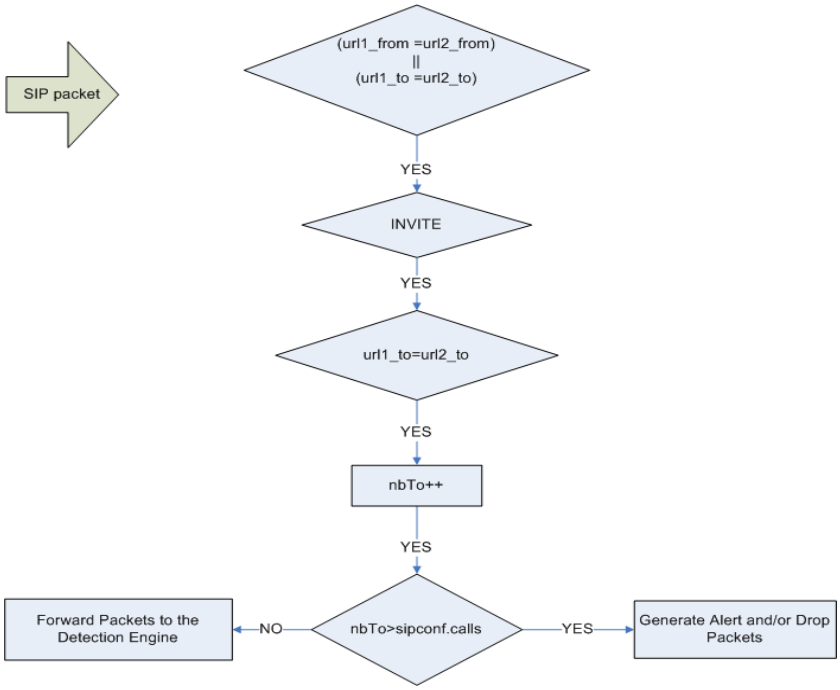

Figure 6.4: Logical Operation in attempt to prevent DoS attack against an internal client, utilizing INVITE messages [1]

6.4.3 Attacks by flooding with OPTIONS messages

These are two different attacks, one of them trying to detect a sender of a flood with OPTIONS messages and the other trying to detect flooding against an internal client. This type of attack is prevented similarly to the previously discussed “flooding with INVITE messages attacks”. The difference is that

here, we check if the message is OPTIONS (not INVITE) and the thresholds and counters are of course different. If the message is OPTIONS, we check the URLs it came from and then it went to. If they are the same, we increment the respective counter, here called “n_opt_From” or “n_opt_To” (or both of them if both coincide). Each of these counters are compared with the threshold “sipcnf.m_options”, the number of OPTIONS messages a user can send or receive (normally) in a period of time.

6.4.4 Preventing a SIP client from receiving a call

This attack and its prevention are based on several assumptions. An attacker is able to sniff the network and thus sees the mandatory fields (To, From, Call-ID and CSeq) in an INVITE message, which can allow him/her to build a legitimate, well-formed CANCEL message. Another assumption is that the attacker would be at a different device, thus having a different IP address and would not be able to spoof the IP address.

The logic of attack detection is the following: after the standard initial block, the Call-ID of the currently inspected packet (cid1) is checked against the respective Call-ID from the sessions table. If the Call-IDs match, that means we have a request concerning an existent call. The following step is to check if the message is a CANCEL. If yes, we proceed by comparing the source IP address of the current packet (IP_src_1) with the one from the initial session establishment, recorded in the session table (IP_src_2). If they are different, then there is some likelihood that an attack has occurred. Then the preprocessor generates a log and may drop the packet, if configured to do so. This can be seen on Figure 6.5.

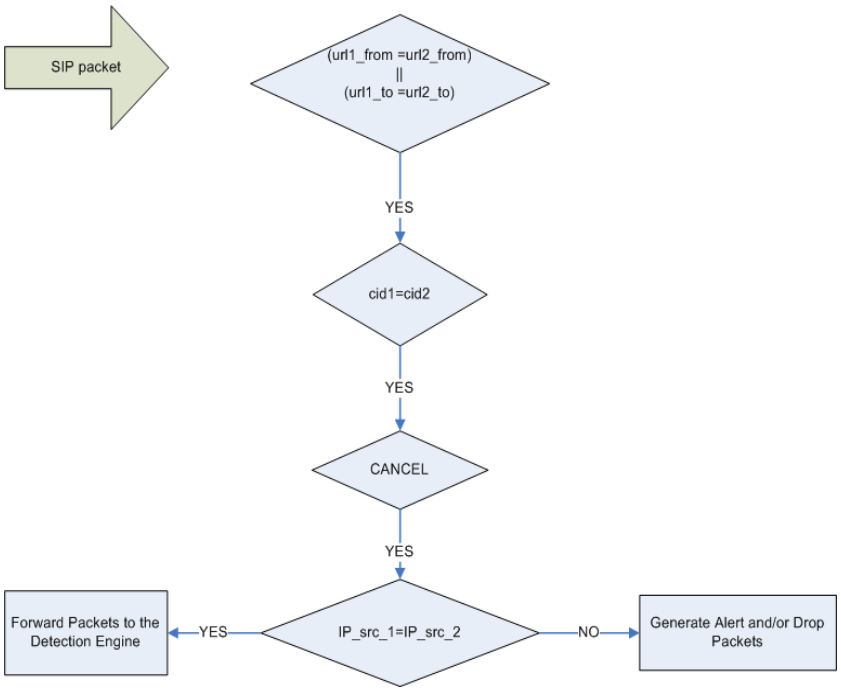

Figure 6.5: Logical Operation in attempt to detect an attacker preventing a SIP client from receiving a call [1]

Explanation of some abbreviations in the figure:

- cid1 the Call-ID of the currently analyzed packet

- cd2 the Call-Id of each element (taken sequentially) in the table that has the same From or To field as the currently analyzed packet

Obviously, the attack cannot be detected when an attacker is able to see and spoof the IP address. The authors of the preprocessor [1] suggest that this attack can be better handled by having a very short time slot between an INVITE and potential CANCEL message. If we have a sequence of INVITE and CANCEL messages and the time slot is less than the one specified for shortest possible, the packet may be blocked. This is an example of behavior-based detection. The assumption for the latter is that a legitimate user does not usually send INVITE and just very shortly a CANCEL. In any case, if the packet is dropped by mistake (false positive), the user could resend the CANCEL and no significant consequences would follow.

6.4.5 DoS with RANDOM messages

The preprocessor performs a security check (as explained before) based on oSIP libraries to ensure that the mandatory fields are present and well-formed. Mandatory fields are Request URI (only for request messages), Call-ID and its size, Via, From, To and CSeq. If not everything is fine, the packet is logged and possibly dropped.

6.4.6 Session surveillance

The session surveillance is modeled and implemented using a state machine and thus a protection against illegal or malicious transactions/transitions is performed. The state machine models only INVITE, ACK and BYE messages. Each transaction is recorded as a soft state and kept for a configurable amount of time. The stateful analysis just checks if the SIP transactions are following normal patterns. Also, the number of transactions per time period can be limited by a parameter in the configuration file sipcnf called “limit”.

The following Figure 6.6 shows the state machine used. For example, if the preprocessor receives an ACK message without receiving a respective INVITE before that, i.e. the current state is not “Wait ack”, then the ACK message can safely be logged as suspicious and dropped, if so configured.

6.5 Problems with the proposed solutions

In this part, we try to analyze specific problems, related to the detection of the proposed attacks. At a very high level, the main problems are NAT traversal, mobility and proxy issues and they are analyzed at the end of Chapter 7, since they are important for the enhancements to the SIP IDS prototype, proposed in that chapter as well.

6.5.1 Attacks whose detection is based on thresholds

These are basically flooding attacks, trying to exhaust resources or establish unauthorized/unsolicited contact. The idea is generally good, but highly problematic in practice, since it does not provide a way to find out reasonable values for the parameters. Moreover, as network and host behavior change over time, these values need to be properly updated. Improper values would definitely cause more harm than add value, especially if the prototype works in IPS mode. Nevertheless, the prototype presents a

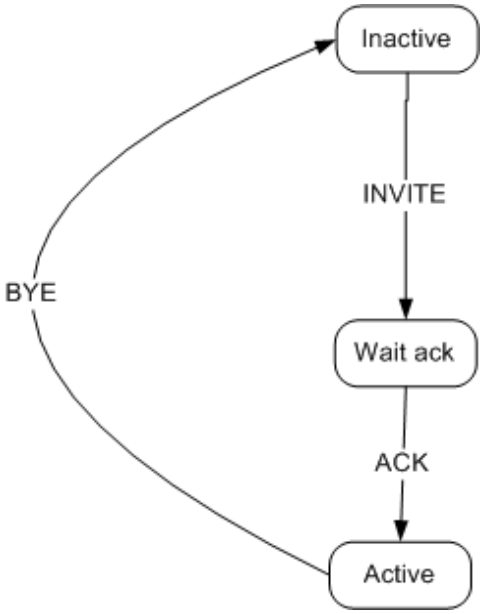

Figure 6.6: State Diagram for a SIP session [1]

good starting point and the problem of determining these values can be further investigated, possibly by trying to design a behavior-based algorithm, which would try to estimate normal behavior for the network and specific hosts.

6.5.2 Attacks whose detection is based on IP addresses

This is basically the call-teardown attack, implemented for a CANCEL message, for which it makes more sense, but possibly with similar success for a BYE message attack as well. The main problems with respect to attack detection based on IP are mobility, IP spoofing, NAT and proxy traversal, and having several NIC, respectively several different IP addresses (especially on a soft-phone). Mobility implies handoffs and possibly often changes of the assigned IP address of a terminal. The SIP IDS/IPS prototype will thus create false positives, resulting from this address change. Additionally, IP spoofing will enable an attacker to mask his real IP address and provide the expected one of the legitimate user, as seen by the IDS/IPS before. Finally, it is possible to access the network through different NIC and different access technologies, thus having several IP addresses on a terminal should be considered normal.

6.5.3 Session surveillance

This part of the prototype is very useful, since there are certain rules in SIP, describing allowable behavior and valid state transitions. Anything else can be considered a result of bad implementation and/or malicious activity and can safely be dropped. The messages considered in this session surveillance are only INVITE, BYE, ACK. It is worth considering how to possibly enhance this state machine for checking for valid transitions for other messages. We attempt that in the next chapter. Stateful detection is an important theoretical abstraction for designing a VoIP IDS.

7 SIP Intrusion Detection and Prevention Enhancement

In this chapter I will focus on describing and analyzing how some additional attacks can be approached by enhancing the SIP preprocessor, described in Chapter 6. In addition, problematic issues with the proposed IDS/IPS approach to the new attacks and to the ones described before will be considered. This chapter is central to my work and is based on my analysis of synthetic and real attacks, which threaten or are expected to threaten VoIP deployments in the future. In addition, possible solutions are proposed and implementation is demonstrated for some of the attacks, by modifying the code of the SIP preprocessor.

The following "new" to the SIP preprocessor attacks will be considered in this chapter:

- REFER attack
- Re-INVITE attack
- UPDATE attack
- REGISTER-related attacks
  - Attacks on the authentication mechanism: brute force password guessing
  - REGISTER flooding
  - Registration hijacking
- RTP flooding

7.1 REFER attack

The REFER extension (RFC 3515) of the SIP protocol (RFC 3261) provides a mechanism for a party (referrer) to give a second party (referee) an arbitrary URI to reference. When the URI is a SIP URI, the referee sends a SIP request, usually INVITE, to the provided URI, which is called the refer target (Figure 7.1). This method is generally used to enable applications, such as call transfer.

Figure 7.1 demonstrates how the REFER method works, according to RFC 3515. User Agent A (the referrer) sends a REFER request to User Agent B (the referee) with the address-of-record of User Agent C (the refer target). If the message is fine for UA B, then a 202 Accepted response message is returned and then a Notify (followed by respective 200 OK), just before sending invite to the refer target UA C. Then a normal session establishment happens between UA B and UA C. At the end, a Notify message is sent to UA A (followed by a respective 200 OK confirmation).

RFC 3892 extends the REFER method. It allows the referrer to send information about the request through the referee to the refer target. This information is used by the refer target to figure out whether

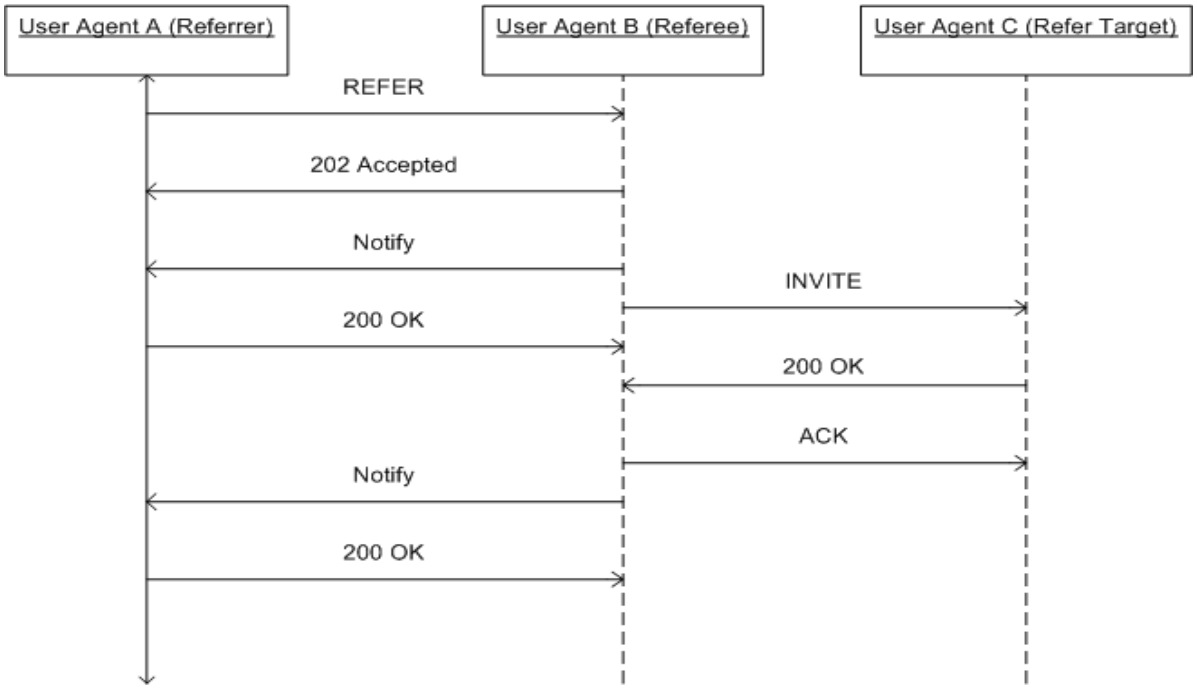

Figure 7.1: Refer Method, as described in RFC 3892

the request should be accepted or not. The information is usually the referrer's identity (SIP address-of-record) provided in a newly defined Referred-By header field, referred-to URI, time of the referral etc. Since the referee is in the perfect position to eavesdrop and/or launch a MITM attack against a refer target, for example using information of older sessions, an S/MIME mechanism for providing message integrity, authentication and non-repudiation of origin is available.

RFC 3892 introduces a Referred-By token, which is Authenticated Identity Body (in this case it MUST include only Refer-To, Referred-By and Date header fields) and must be identified with a MIME Content-ID field. In case this is combined with using S/MIME with digital signatures, we can achieve authentication, message integrity and non-repudiation of origin. A timestamp against replay attacks may also be added in the Authentication Identity Body. Unfortunately, the mechanism protecting the Referred-By token by using S/MIME is only optional, although requests not utilizing it may be rejected by a UA. In principle, it is a good practice for the recipient (refer target) to refuse to accept a request (it may do so according to the specification), if this S/MIME mechanism is missing.

A possible scenario for an Attack exists, when a Referred-By token is not S/MIME protected and is minimal (there is no integrity-protected To field, stating explicitly who the referee is). That means, the token asserts that the entity specified in the Referred-By Header (the referrer) asked at a certain time (as seen in the Date header) that the request be sent to the entity, indicated by the Refer-To header (refer target). But the assertion makes no claim who is asked to send the request, i.e. no claim is made who the referee is. Thus, there is the opportunity for cut-and-paste attack, i.e. an Attacker sees this minimal token and then can copy it in his/her own request. Then, it will appear to, e.g. Alice (in Figure 7.2), that the Attacker was refereed to her by Homer, when that was hardly the case. This attack is described in RFC 3892. This will enable the attacker to launch a SPAM over Internet Telephony (SPIT) attack against users.

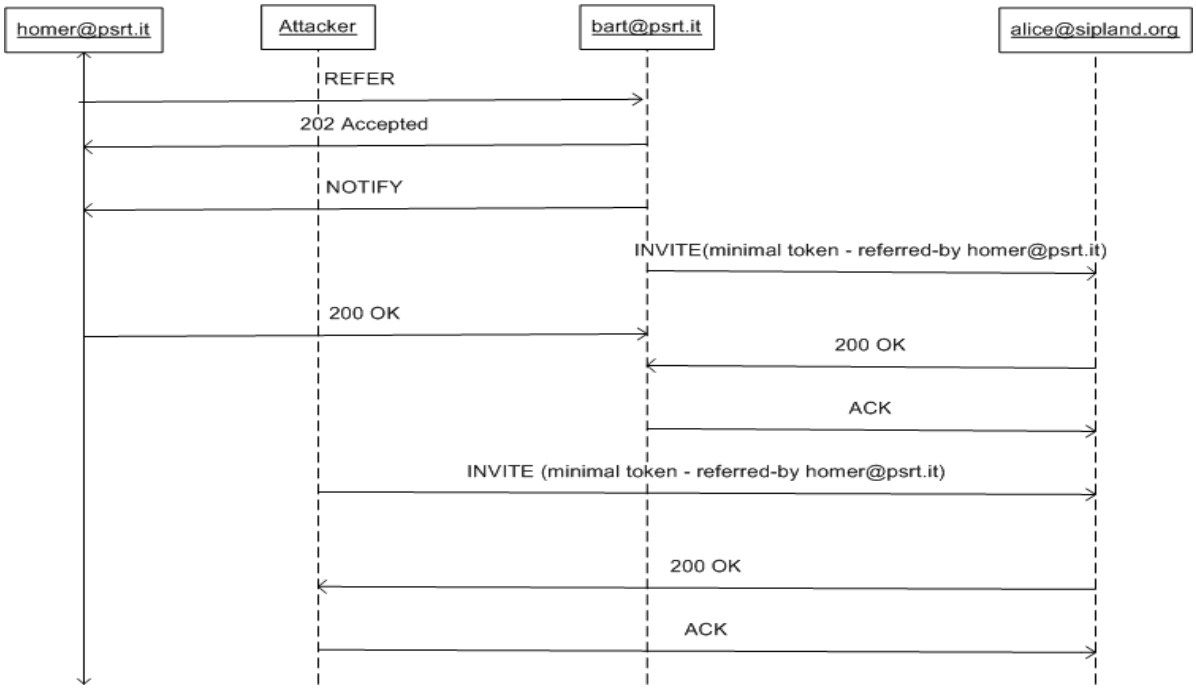

Figure 7.2: Cut-and-paste REFFER attack

It will be difficult to detect this attack by the SIP preprocessor, due to the fact that many connections need to be started to the same host (here alice@sipland.org) or from the same attacker so that the relative thresholds are surpassed and the attack is detected. In a case when an attacker listens for REFER messages to different UAs and sends them INVITE messages, as discussed above, an attack could be detected as SPIT. This would be difficult and setting the threshold should in the optimal case be done dynamically through some behavior-based algorithm. The reason is that the attacker can wait and send the INVITEs after some periods of time. The same reasoning is valid for setting the threshold for detecting a DoS attack against an internal client, through INVITE messages, when many INVITE messages are sent to a single client, such as here alice@sipland.org. In case the attackers are smart enough not to send too many messages to the same UA or from the same UA per specific period, the attack may work and be undetectable for the SIP preprocessor. The best countermeasure is to protect the REFER messages from eavesdropping, using S/MIME (including protection against replay) or TLS mechanisms.

As illustration of the fact that the attack can be feasible, we discuss two blind, also called unattended call transfer scenarios presented by Cisco. Blind call transfer is usually used by automated call transfer devices. The following 2 Figures are based on SIP - Call Transfer Enhancements Using the Refer method (Cisco implementation, [12]) in combination with a scenario described in [1], where an attacker has the chance to sniff traffic and construct a REFER request. The Attacker, sniffing the network constructs a REFER request, impersonating homer@psrt.it and thus can cause a Session Tear-down on the communication session between the two users Homer (homer@psrt.it) and Bart (bart@psrt.it). At the same time the Attacker could enable a SPIT (SPAM over IP Telephony) attack, by providing in the REFER request the SIP address-of-record of a VoIP Spammer, here called Spitter (spitter@doom.org). All this happens under the assumption that the Attacker can sniff the home network of Homer and learn the identity of the valid referrer (homer@psrt.it), the Call-ID and the other necessary parameters for a

REFER request. Another assumption: the optional S/MIME protected token, reflecting the identity of the referrer and especially the To field, supposed to explicitly state the referee, is not implemented.

The difference between the Figure 7.3 and Figure 7.4 is that in the first one, the BYE request is originated by the referee (Bart), whereas in the second one by the attacker (referrer). The session between Homer and Bart is broken by Bart, who thinks he received a genuine REFER message from Homer. In the other case, the attacker sends the BYE message, but the principle is the same. Additionally spitter@doom.org establishes a connection with the user Bart and can try to sell something or simply irritate the user.

The preprocessor may be used to try and detect the malicious BYE messages in both cases (by a similar mechanism, used to prevent a call-teardown by CANCEL messages), but that will not help much, even if those packets are identified and dropped and never make it to their destination. In case that the referrer does not send a BYE request, the referee will try to do that. So even if the session between the original participants cannot be broken, the SPIT attack still succeeds. The main task should be to catch the malicious REFER message and stop the attack earlier.

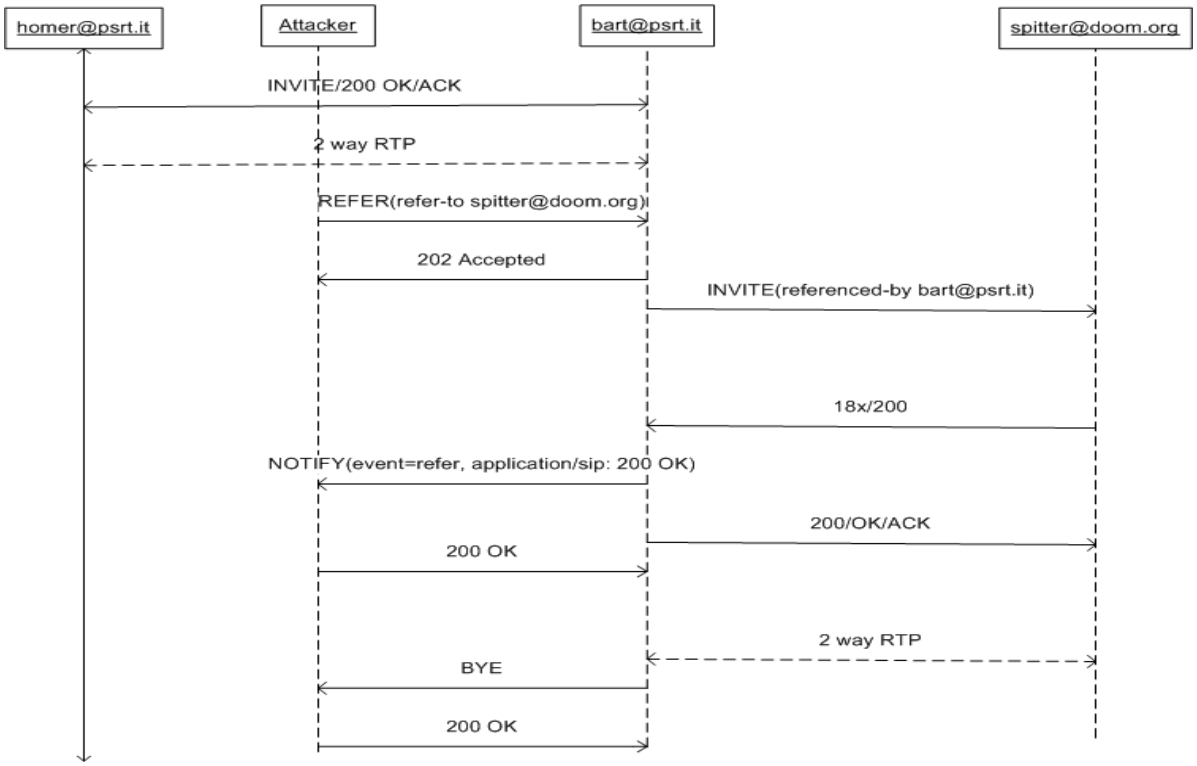

Figure 7.3: REFER attack with Referee originating the BYE request (Session Teardown and SPIT) [12]

If we only slightly modify the Logical operation of the SIP preprocessor from Figure 6.5, we could solve the issue, by capturing the REFER attack due to the fact that the Attacker and the attacked user homer@psrt.it have different IP addresses. This solution works only in the case when the attacker is not capable of spoofing the IP address of the party, which the attacker impersonates, in this case homer@psrt.it. In addition, mobility scenarios would be problematic, since the legitimate user can change the IP address quite often. The IDS/IPS will though consider that an attack, resulting in numerous false positives. The logical operation of this attack prevention is shown in Figure 7.5.

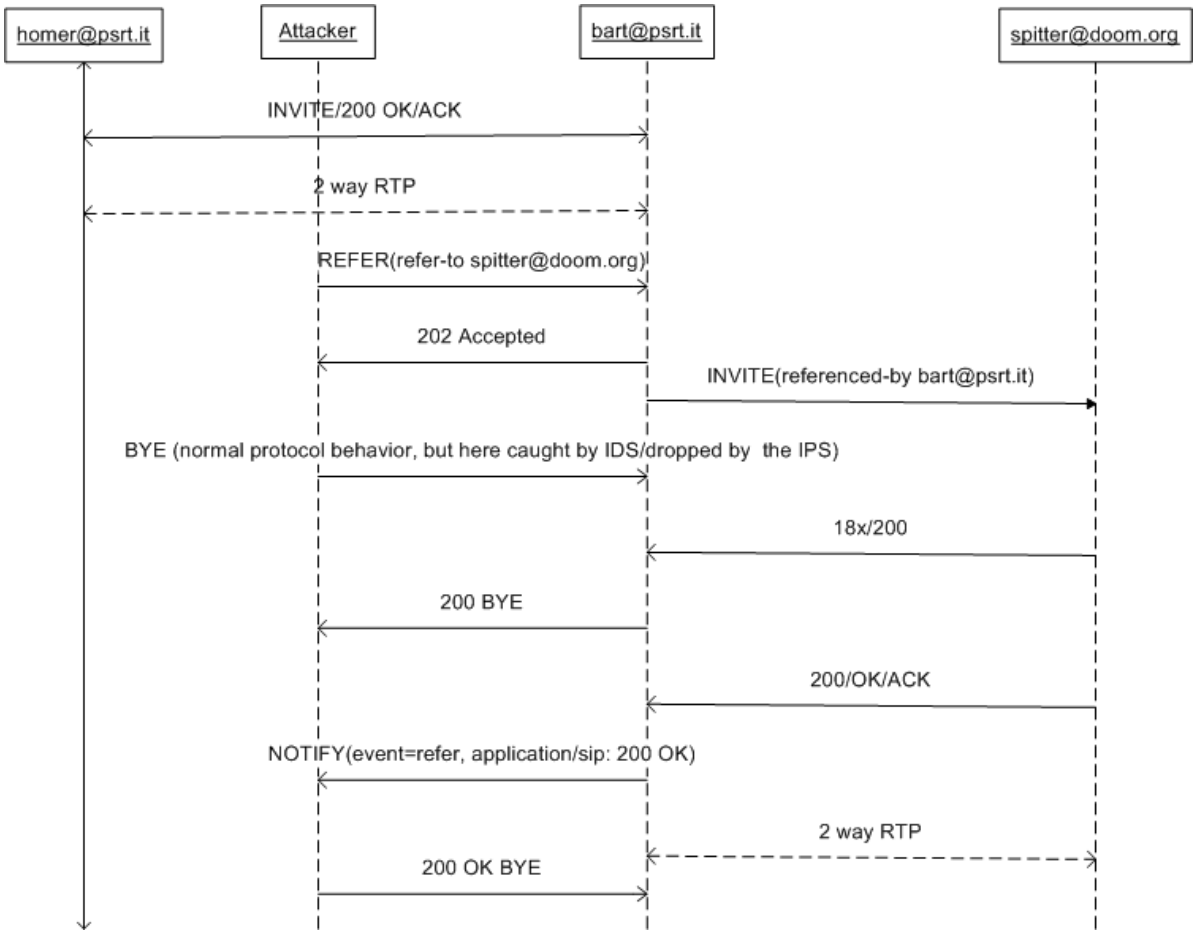

Figure 7.4: REFER attack with Referrer (Attacker) originating the BYE request (Session Teardown and SPIT) [12]

As discussed in Chapter 6, a table is maintained by the preprocessor, where the necessary data to uniquely identify each transaction is kept. These are the To, From, Call-ID, and the IP source and destination addresses of the message. The preprocessor looks at each SIP message and compares the From and To fields it finds, against all records in the database. If at least one match is found, the check continues and this is done as initial step for any attack detection. Now, the Call-Id is compared with the one of the record found. If there is a match again and the message sent is a REFER message, then we may assume that probably a user, who already established a connection want to refer the other party to a third party or resource. If that is the case, most probably (possibly quite a naive assumption) the user will be at the same device still, meaning the user will have the same IP address as before. Therefore, we check the IP address of the sender and the initial user. In case they are different, we may assume that an attack is occurring and thus the NO arrow is followed (on Figure 7.5). This implies the packet will be either logged as suspicious and dropped by the preprocessor, if configured to do so. If the source IP addresses are the same, as expected, the packet is passed to the detection engine, the next block in the Snort architecture. The code to implement this attack detection, as well as the other ones, is presented in Appendix A.

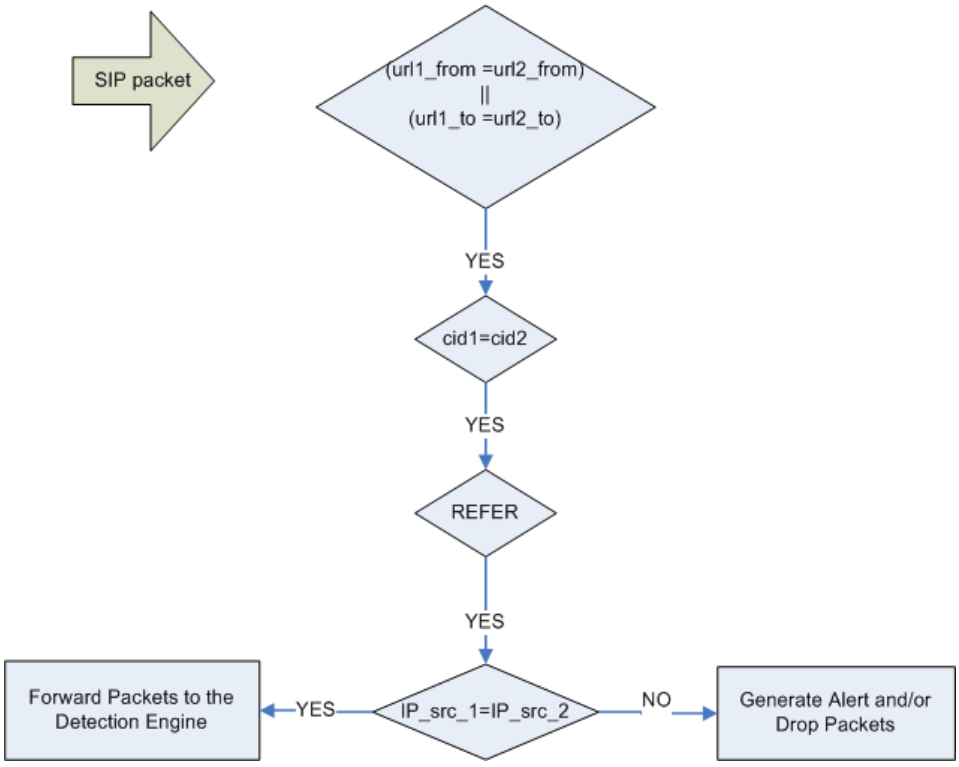

Figure 7.5: Proposed Logical operation of the SIP preprocessor against the REFER attack

7.1.1 Issues with the REFER proposed detection

Since the detection is based on IP address comparison, the same problems are relevant as discussed in Secion 6.5.2. The situation is somehow better, at least in terms of detecting the Cut-and-paste REFFER attack shown in Figure 7.2. The reason is the attacker wants to establish a connection with one of the parties, thus the attacker will not spoof the IP address, otherwise connection establishment will not be possible. The same mobility, NAT and proxy traversal, several NIC issues should be considered here, which make successful attack detection extremely difficult. Due to the probability for a large number of false positives, attack prevention (when the malicious packets should be dropped) seems to be out of the question. The mobility, proxy and NAT traversal issues are discussed at the end of this chapter, since they are important for most of the attack detection attempts.

7.2 Re-INVITE attack

A successful INVITE message establishes a dialog between 2 User Agents and a session. The established session can be modified by a second INVITE request within the same dialog that initially established the session. The further INVITE requests after establishing the session are called re-INVITE requests. The modification of a session can involve changing addresses or ports, adding a media stream, deleting a media stream etc.

An attacker, sending a re-INVITE to a user, already having established dialogue and session with a second user can launch a session teardown attack [14], i.e. DoS against both users (see Figure 7.6). Also, if voice emulation can be successfully used, this may compromise confidentiality and integrity. This attack can be successful when proper authentication and authorization for the re-INVITE message

is not required. The attack is similar in nature to a CANCEL/BYE attack (Figure 6.5), but if treated in the same way (checking the IP address of the sender), it will lead to plenty of false positives, since re-INVITE is actually used for IP address change, for instance due to mobility. From the point of view of Intrusion Prevention, the false positives make this type of solution useless in practice. The attack could also be caught (mitigated in a way) by the preprocessor as DoS against an internal client through INVITE messages, but that depends on the configuration of the threshold and is not very probable. Usually the first several re-INVITEs will be missed (false negatives), so generally no useful detection is implemented here. An option could be a behavior-based algorithm.

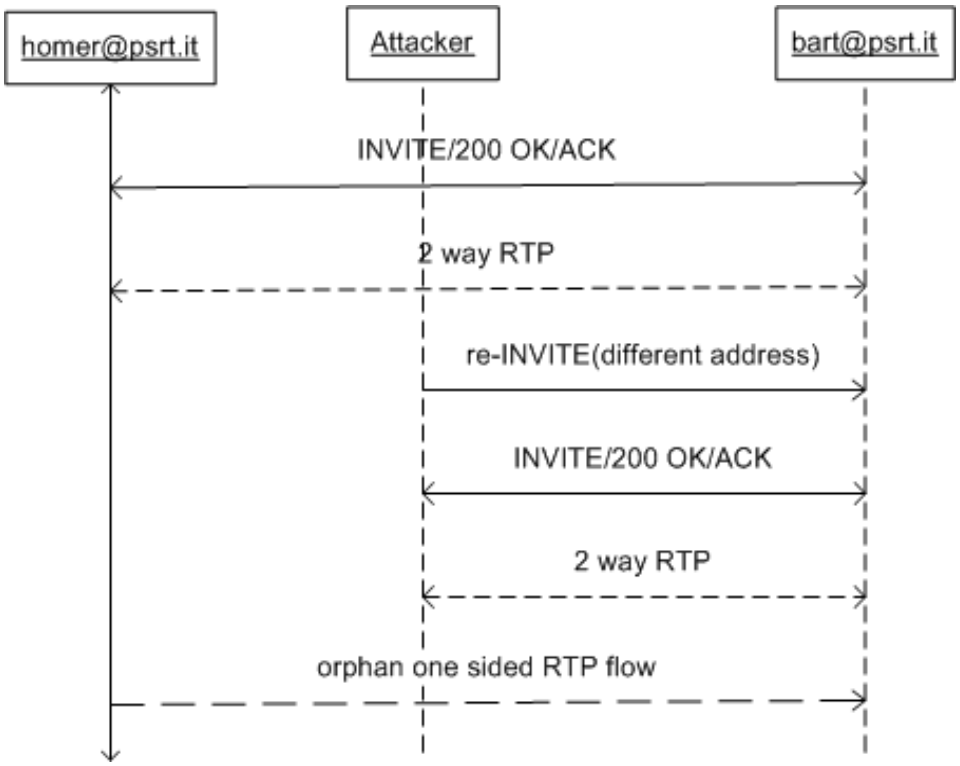

Figure 7.6: Session Teardown due to re-INVITE attack, based on [14]

The best solution for prevention seems to be, as against other DoS attacks (CANCEL/BYE), cross-protocol detection. This approach requires the existence of an RTP preprocessor as well, which communicates with the SIP preprocessor. Detection and prevention can be based on identifying the orphan RTP stream that is not expected to be present on behalf of the party, which supposedly sent re-INVITE (homer) at the old IP address, but is present instead. On the other hand, the Attacker could send homer@psrt.it a BYE message before the Attacker sends the re-INVITE, which if not caught would neutralize this defense strategy.

7.3 UPDATE attack

The UPDATE message is similar to re-INVITE in that it allows a client to update some parameters of a session. The difference is that an UPDATE request can be also sent before the initial INVITE has been completed. This part (before completing the initial session establishment) is the only part of the analysis, which would be different and we try to approach it. The only possibility here would be to

detect messages coming from a party (at an old IP address for example) after that party has supposedly sent an UPDATE message, stating that they are not at that old IP address any more. That would mean an attacker, impersonating a legitimate user is recognized. Of course, here the exact handoff mechanism needs to be additionally analyzed, since sometimes the mobile node may have several interfaces and be in coverage of different providers during handoff, which could be used at the same time for a while. In that case false positives are possible and the issue should be further analyzed. Additionally, as discussed before, a smart attacker can send a BYE message to the user, being impersonated. If that message is not recognized as attack and passed, the proposed solution detection, based on an orphan RTP flow will not work.

7.4 REGISTER related attacks

7.4.1 Attacks on the authentication mechanism: brute force password guessing and REGISTER flooding

As discussed in the VOIPSA forum (information given by John Tod), attacks like brute force password guessing utilizing the REGISTER request method are already being launched in reality. The extension ranges under attack were valid, only known to the administrators. The assumption is thus made by the author that the extensions were probably discovered by random scanning for UDP 5060, 5061 responses on public IP addresses. Since in the SIP specification - RFC 3261, REGISTER messages from an unknown user will result in the “404 error”, whereas bad credentials (but existing user) result in the “402 error”, it is possible to first discover valid users and then perform brute force or dictionary attacks on them. In the case described, the attack was coming from the same user agent and had about 10 queries per second.

This attack can be treated similarly to the way INVITE DoS is handled, i.e. a single UA should not have the right to send more than a certain number of REGISTER requests per certain time period. This will not work so well for a DDoS, where the IP addresses will be different, SIP URIs as well. A DDoS attack against a proxy (registrar) can be somehow alleviated by having a maximum number of REGISTER requests that it could have per time period, and any further request should be dropped. This could, unfortunately end up being a DoS against legitimate users, but at least will prevent the server from crashing and/or being totally useless due to the overload. This is a simple load monitoring process and only noting the number of REGISTER messages per time period from a specific user and to a specific user should be considered. Similar functionality could be implemented at a higher level in a behavior-based system, where of course the analysis of which behavior can be considered an anomaly is by far more complex. The simple, proposed logical detection of register flooding and brute force guessing is shown in Figure 7.7.

The preprocessor looks at each SIP message and compares the From and To fields it finds, against all records in the “database” as before. If at least one match is found the check continues, which is the usual first step. Now, we check if the message is a REGISTER request. If the source SIP URL (and IP address) coincides with one already in the table, it means that a certain user who registered recently, already registers again. In that case the counter for that specific UA is incremented and compared with a predefined threshold, defining the maximum number of REGISTER requests a user may establish (try to establish) within certain time interval. This threshold is assigned to the variable “sipcnf.register_from”.

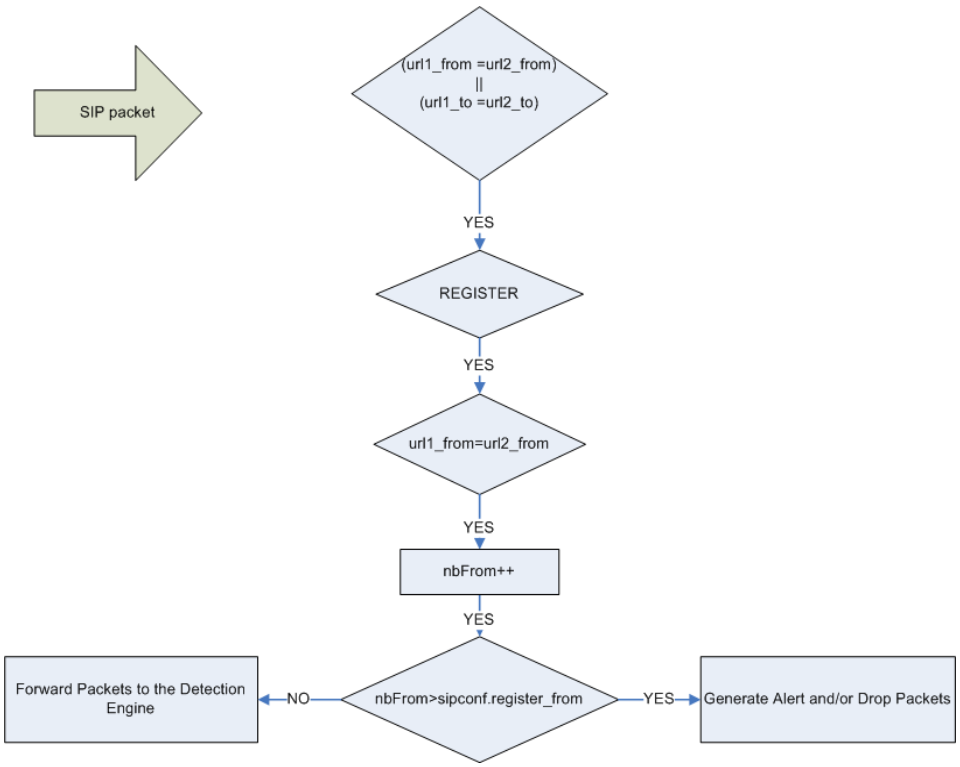

Figure 7.7: Logical Operation in attempt to detect register flooding and brute force password guessing

If the variable is surpassed, then there is some likelihood that a REGISTER flooding or brute force password guessing occurred. In this case a log is generated and the packet may be dropped if configured to do so (action_drop flag is set). Else the packet is passed to the Detection Engine, the next block in the Snort architecture. Similarly for a SIP registrar, we may want to limit the number of REGISTER requests sent per time period, to prevent an overload and a possible crash of the registrar. In that case, exactly the same is done, but we define another counter regTo and compare it to another threshold parameter (sipconf.register_to). In addition, the criteria is that the URL, that the message comes from should be compared to previous ones in the time period (i.e. url1_to and url2_to need to be compared, as shown in Figure 7.8). The code to implement the attack detection from Figures 7.7 and 7.8 is presented in Appendix A.

The problems with the detection of these two flooding attacks are generally discussed in 6.5.1. The general idea is that more work is needed for developing a behavior-based algorithm, which would be able to periodically calculate and dynamically update the threshold parameters.

7.4.2 Registration hijacking

The SIP specification (RFC 3261) does not mandate the usage of authentication mechanism for authenticating the User Agent to the Registrar ("A registrar SHOULD authenticate the UAC"). This gives the opportunity to an attacker, having the ability to sniff the network in general or a specific user's communication (the REGISTER request), to perform registration hijacking, i.e. de-register the legitimate user's device and register his/her own device instead (in the case when no proper authentication is used). As a result the legitimate user's traffic will be received by the attacker. This can be not only a DoS attack

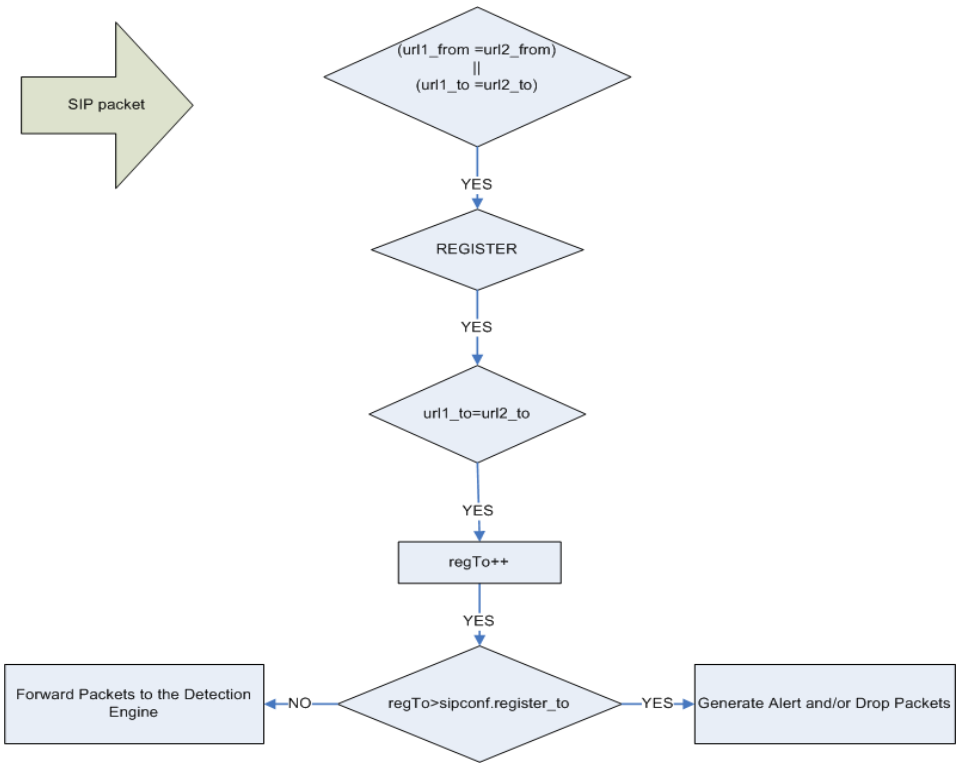

Figure 7.8: Logical Operation in attempt to detect register flooding on a SIP registrar

against a legitimate user, but also in the presence of voice-emulation facility can breach confidentiality and integrity, by allowing a party to masquerade as somebody else. The fields Request-URI (the domain of the location service for which the registration is meant), To(address of record whose registration is to be created, queried, or modified), From (address-of-record of the person responsible for the registration – usually the same as the To field), Call-ID (all registrations from a UAC SHOULD use the same Call-ID header field value for registrations sent to a particular registrar), CSeq (UA MUST increment the CSeq value by one for each REGISTER request with the same Call-ID(changed in a next boot cycle)) are mandatory and the attacker can see them in a rightful registration. Observing these fields, the Attacker can create another REGISTER request, with expiration interval set to 0 and incrementing the CSeq value as expected. This will in effect deregister the user and allow the Attacker to register his own device to match the user's address of record. Figure 7.9 based on RFC 3261 shows the attack.

This kind of behavior can be very suspicious if done within a small time period. Again, normal and abnormal behavior of REGISTER messages could be potentially analyzed in terms of when are they sent, where a user is expected to be and where he is in practice. Beyond some mobility scenarios, in a rather naive approach, we can use a scheme similar to the one for blocking call tear-down attacks, i.e. by IP address. The following scheme explains the idea better.

If two or more REGISTER messages came from the same SIP URI within specific time period, we should compare the source IP addresses. If they match, everything seems to be fine. If they do not, the message may be logged and possibly dropped. The code to implement this attack detection is presented in Appendix A.

Again, we face the same difficulties as in trying to detect other attacks by IP, these are described in

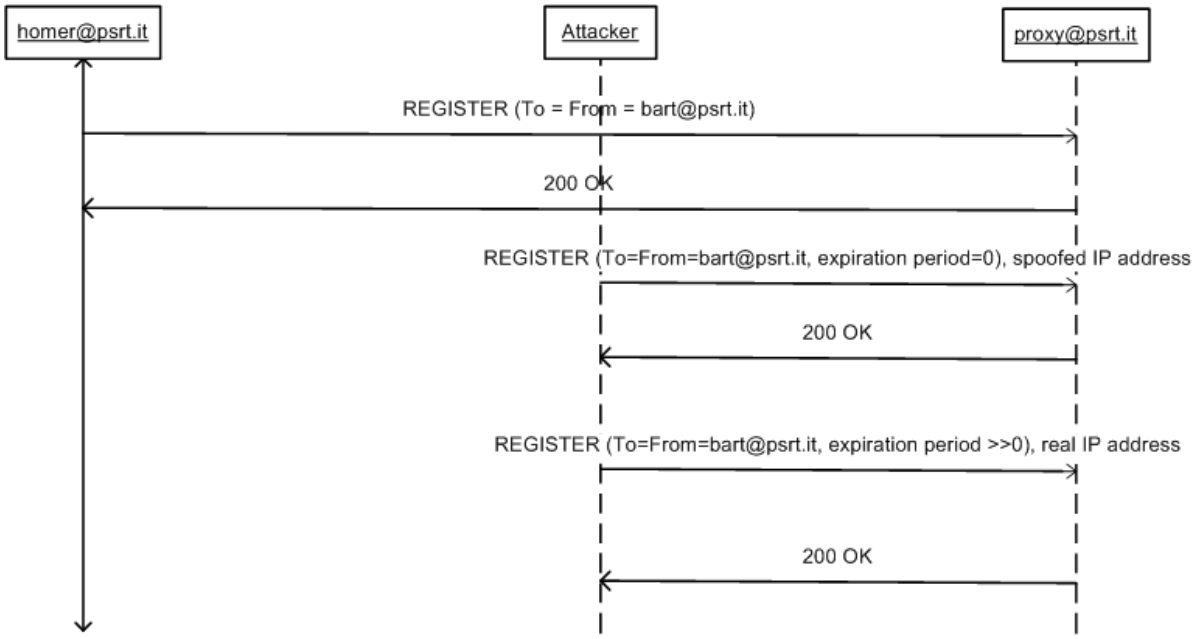

Figure 7.9: REGISTER attack, Registration Hijacking sequence diagram

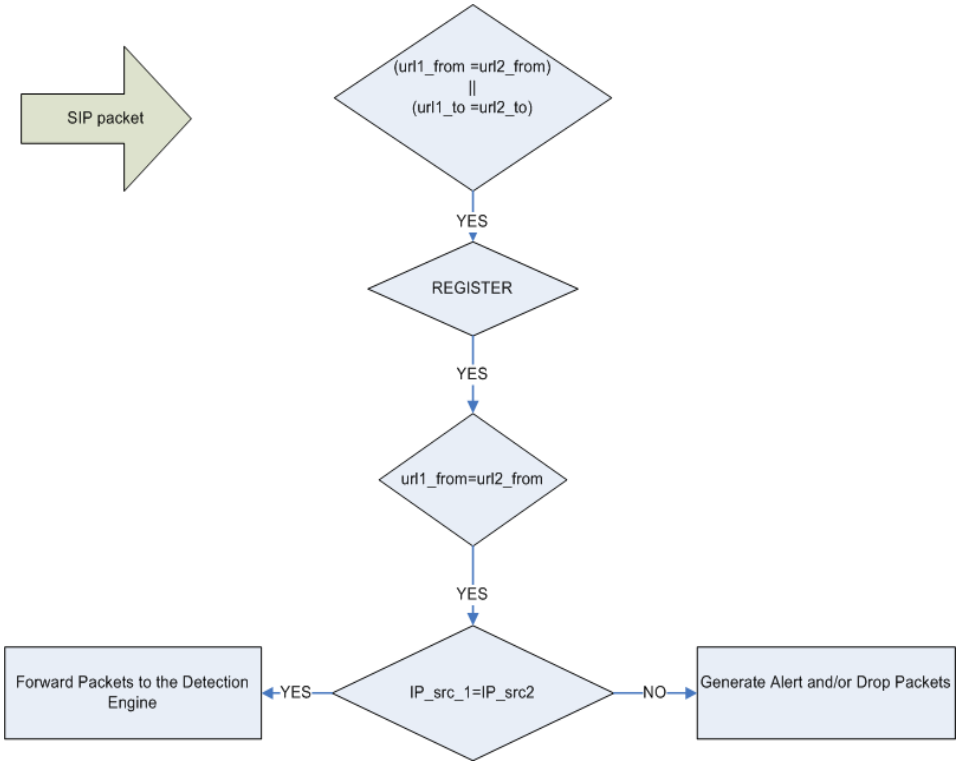

Figure 7.10: Logical Operation in attempt to detect Registration Hijacking

6.5.2. Another problem here could be inserting by an attacker of a message with a bogus From header, which if successful will deceive the attack detection mechanism.

7.5 Modifying the Flow Chart to include UPDATE and re-INVITE

It can be quite useful, if the Flow Chart of a SIP session (picture 6.12 in [1]) is updated to include more messages and state transitions. For example, it does not take into account the possibility of receiving an INVITE message in the Active state (this happens when a re-INVITE is sent). This means that such re-INVITE messages will be dropped, if the preprocessor is configured to drop packets. We also include the UPDATE message, which possibly can be received in the Wait Ack and Active state. As part of enhancing the SIP IDS prototype, those two messages are also included in the state machine as shown in Figure 7.11. The code to modify the flow chart is also presented in Appendix A.

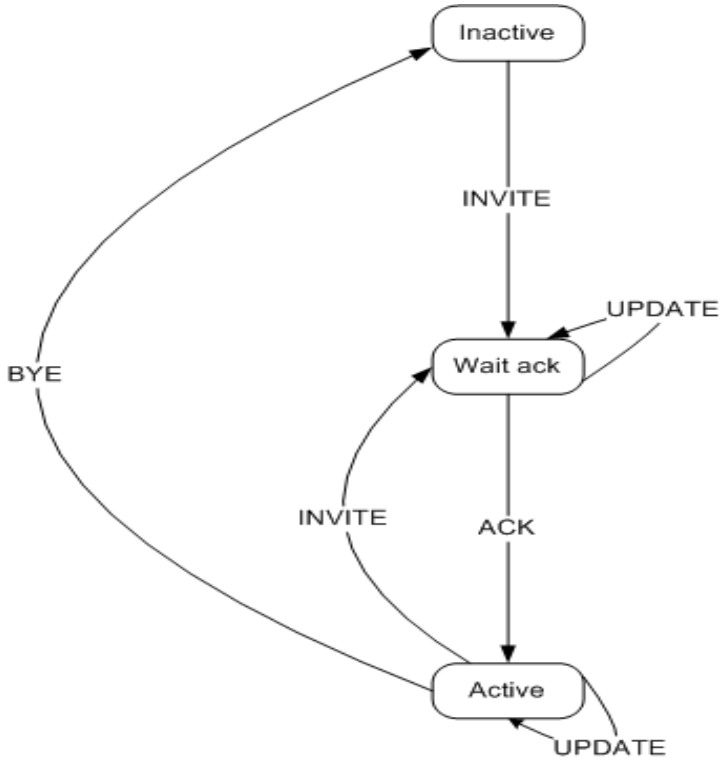

Figure 7.11: Modified Session Flow Chart to include UPDATE and re-INVITE

7.6 RTP flooding

This attack is not SIP related, but is discussed here since its potential solution has to do with cross-protocol detection and mainly with the development of an RTP preprocessor. The attack is targeted against media processing VoIP components, e.g. media gateways, IP phones etc., which are bombarded with “garbage” packets to degrade service quality. The RTP packets are characterized by random sequence RTP numbers and spoofed IP and MAC addresses. Attack tests were performed against IP phones by SecureLogix. In most cases, the attacks (with RTP packets having random numbers) resulted in severe or total degradation of audio quality. Attacks with sending large, crafted RTP packets (about 1500 bytes) often caused a crash, as reported by SecureLogix.

The problem could be solved through the development of an RTP preprocessor, which would discard packets with sequence numbers, indicating suspicion of an attack (i.e. difference than expected numbers is too large and thus outside some limits). These packets, even if not being malicious, most

probably came too late anyway, so dropping them would cause no harm. Packets, which are larger than normal should and can be easily discarded as well.

7.7 Problems with the proposed solutions

7.7.1 NAT Traversal

For this analysis, we assume that we have a network with private IP addresses, NAT and then a Session Border Controller (where the IDS/IPS prototype is situated). The NAT maps the private IP address and port to the corresponding public IP address of the NAT and a certain port.

The REGISTER message is actually supposed to convey to and save the IP address and the port of the corresponding SIP URI in the registrar, so that the user can be contacted from outside. This port is relevant for the session and can be used to distinguish an attacker from a legitimate user, e.g. for the CANCEL/BYE session teardown when an attacker and attacked are behind the same NAT and the IP address seen at the SBC is the same.

The SIP preprocessor sees the registration in the REGISTER message and from there can record additionally to the IP address, the port number. Thus, if the registration is successful, the IDS/IPS can detect attacks, based on the port number. In case of proxies between the NAT and the SBC, the port will still be known, since it must be kept in the "rport" tag, used to facilitate NAT traversal (also the "received" tag is set). The only problem here is that the "rport" could be removed by the first proxy that sees the message and this would mean having two messages (rightful and malicious) that are identical and cannot be differentiated from each other. Thus, the attacker can send the REGISTER message successfully and the attack will be not detected. Against this, there is nothing we could do with the proposed approach.

7.7.2 Proxy Issues

If the messages are relayed through a proxy (or several proxies before the SBC, where our IDS stays), the IP address to consider is the one of the originator of the request. To make this, we need to examine and record the appropriate Via field in the SIP packet. The IP address we are interested in, is the one at the "bottom of the stack" of Via headers (i.e. the first one chronologically). If the "received" field is present, that would mean the UA client is behind a NAT, so we would be interested in the IP address in the received field (IP of the NAT device) and additionally in the port number. This approach though, is rather naive, since an attacker can change the IP addresses in the Via header by spoofing them. In the case of a DoS attack, this is perfectly fine with the attacker, he/she does not need to be reachable on the return path in order to send packets successfully. In case of SPam over Internet Telephony instead, the attacker needs to advertise its real address, or else he will not get any reply and will not be able to initiate the session and deliver the SPIT. In this case it would also be possible, that the attacker tries to mask himself and does not send all the Via fields. Yet another problem comes from the fact, that masking of the Via field may be performed by a real proxy on the route, which does not want to disclose the network addresses and topology to the outside. Thus, the only way we could rely on this approach would be a scenario, where the Via field is initially end to end encrypted, integrity protected, and cannot be removed on the way.

7.7.3 Mobility

It seems to be nonsense to implement mobility without proper authentication of User Agents. Also cryptography should be used when needed. This implies that the REGISTER, re-INVITE, UPDATE, REFER attack detection described before, since based on IP addresses cannot be applied in mobility scenarios; they assume lack of authentication (highly non-realistic) and will produce an enormous number of false positives. The direction to go in the research for solving the issue would be cross-protocol and stateful detection, combined with new behavior-based intrusion detection algorithms.

In case of a successful brute force attack against the authentication mechanism though, we run into problems. The registration hijacking would then not be a problem and all the rest of the discussed REGISTER-related attacks would be able to succeed. On the other hand, we might and have to (that is, if we want to provide mobility service at all) be confident in the secure implementation of the REGISTER request (authentication and maybe encryption of some parts). Assuming that a mobile node registers with the registrar in the home network even when roaming, we can make a check between where the device is according to the registrar (IP address) and where it seems to be (where a message originated from – IP address). Of course NAT issues should be also addressed here. If there is a mismatch, obviously something went wrong and it might be good to either fire an alert or drop the packet. This is in case a client sends an updated REGISTER in the home network, as the first thing to be done when entering a new network. If that is not the case, but the client tries to re-INVITE the correspondent terminal, then the only chance to solve the problem is for the registrar to challenge the message.

8 Conclusions

In this work we have explored the possibility to utilize and enhance an existent, developing technology – Intrusion Detection and Prevention in an attempt to solve VoIP security-related problems. Central to the work has been a SIP IDS/IPS preprocessor prototype, developed by NEC Labs, Europe, based on the open-source Network IDS Snort. Therefore we conclude this work by summarizing the analysis of this prototype and its current characteristics, as well as exploring the possibility for its enhancement (and that of IDS in general), to solve more problems and thus become a viable option for solving security issues in a VoIP environment.

First, this prototype is not and most probably was not aimed as a complete solution, but should be considered as a part of it, as a step in the right direction to solve security issues in VoIP deployments. To be more concrete, although it is efficient, it does not realistically help against the described attacks. There are too many assumptions, and too many conditions, which should be fulfilled in order for the detection/prevention to work. Thus it would work properly only in some special cases, namely when the attacks are not very intelligent.

There are several blocks of the preprocessor though, which are very useful and should be present in an optimal IDS solution for VoIP: these are the syntax analysis check, the security analysis check and the stateful analysis check. These checks respectively analyze the SIP message to verify that it is well-formed and with a SIP-conformant syntax; check the presence of mandatory fields; and perform stateful analysis on transactions to determine if they are to be allowed, as well as limit their number per time period. If problems are found in these blocks, the packets can be rightfully dropped and the SIP preprocessor can successfully work as an IPS.

Not very promising is the suggested approach for dealing with the detection and prevention of specific attacks. The ones based on thresholds can be highly problematic, since we have no reasonable way to discover the values for the parameters (e.g. allowed calls per time period). Moreover, as network and host behavior change over time, these values need to be properly updated. Improper values would definitely cause more harm than add value, especially if the prototype works in IPS mode. Nevertheless, the prototype presents a good starting point and the problem of determining these values should be further investigated through designing a behavior-based IDS algorithm, which would try to define what normal behavior for the network and specific hosts is.

The attacks, whose detection is based on an IP address of an attacker being different than that of a legitimate user are even more problematic and it is almost impossible to make them function. The main problems with respect to these attacks are mobility, IP spoofing, NAT and proxy traversal, and having several NIC, respectively several different IP addresses. These problems provide many ways for an attacker to evade detection, but moreover are a basis for generation of a large number of false positives, which is unacceptable for a good IPS. As a result we may say, that this way of thinking for detection/prevention is quite weak.

Moreover, most of the attacks being detected based on thresholds and IP address, are based on the assumption that necessary security measures and “best practices”, available in SIP are not implemented, which is often the case since they are not mandatory. An IDS/IPS should not be a substitute for these best practices. It should be able to detect problems based on some of these “bad practices”, but this should only be a minimal part of it. Message confidentiality and integrity, authentication of SIP nodes, mutual authentication between SIP nodes and proxies, privacy of SIP traffic can and should be implemented by developers. It is not possible to rely solely on an IDS/IPS to solve all security issues by some sort of magic.

On the other hand an IDS/IPS should have a similar preprocessor for RTP, which will analyze the syntax in similar ways, but will also enable cross-protocol detection together with the SIP preprocessor. Thus, more intelligent attack detection/prevention will be possible. Optimally, the preprocessors should also include a behavior-based part, which would share attack information with the knowledge-based one and thus improve the quality of detection and prevention.

In conclusion, we should remember the most important items on our wish-list for a functional VoIP IDS/IPS: Cross-protocol Intrusion Detection, Stateful Intrusion Detection and combination of knowledge-based and behavior-based techniques. Having this in mind, we understand why the characteristics of the SIP preprocessor IDS/IPS software are far from our expectations of a complete solution. It is based only on the signaling protocol (no cross protocol detection, which is powerful), has some stateful session inspection, which should probably be enhanced and there is a lot to be done in the field of integrating behavior-based techniques in the IDS. Therefore, IDS/IPS technology seems a promising, applicable, worth exploring approach for solving security issues in VoIP environments, but all the requirements need to be implemented. Then we would be able to give a more definitive answer about how good the approach exactly is.

List of Figures

List of Tables

Bibliography

[1] Paola Viscarelli. *Design and Development of a Session Border Controller for securing SIP-based services*

[2] Dan York, *VoIP Security: How Secure is Your IP Phone?*. VoIP Security Presentation to IP Telephony for Government Conference - April 18, 2006

[3] Joachim Posegga, Jan Seedorf.*Voice over IP: Unsafe at any Bandwidth?*

[4] VoIP Security Alliance (VOIPSA).*VoIP Security and Privacy Threat Taxonomy*. Public Release 1.0, 24 October 2005

[5] Carl Endorf, Eugene Schultz, Jim Mellander.*Intrusion Detection and Prevention*, McGrawHill

[6] William Stallings.*Cryptography and Network Security*. 3rd edition

[7] Martin Roesch.*Snort - Lightweight Intrusion Detection for Networks*

[8] http://www.netfilter.org

[9] Yu-Sung Wu et. al.*SCIDIVE: A Stateful and Cross Protocol Intrusion Detection Architecture for Voice-over-IP Environments*

[10] U. Trick, F. Weber.*SIP, TCP/IP und Telekommunikationsnetze*

[11] *NGN 2004 Project description*. Version 3, 12 February, 2004

[12] http://www.cisco.com/en/US/products/sw/iosswrel/ps1839/products_ feature_guide09186a0080110c01.html

[13] http://www.snort.org

[14] Dorgham Sisalem et. al.*Low Cost Tools for Secure and Highly Available VoIP Communication Services (SNOCER)*. Deliverable D2.1

[15] Dorgham Sisalem et. al.*Low Cost Tools for Secure and Highly Available VoIP Communication Services (SNOCER)*. Deliverable D2.2

[16] Stefano Salsano, Luca Veltri, Gianluca Martiniello. *Seamless vertical handover of VoIP calls based on SIP Session Border Controllers*

[17] Saverio Niccolini et. al.*SIP Intrusion Detection and Prevention: Recommendation and Prototype Implementation*

[18] J. Rosenberg et. al.*SIP: Session Initiation Protocol*. RFC 3261, June 2002

[19] M. Handley et. al.*SIP: Session Initiation Protocol*. RFC 2543, March 1999

[20] R. Sparks.*The Session Initiation Protocol (SIP) Referred-By Mechanism*. RFC 3892, September 2004

[21] R. Sparks.*The Session Initiation Protocol (SIP) Refer Method*. RFC 3515, April 2003

[22] Kerry Cox, Christopher Gerg.*Managing Security with Snort and IDS Tools* 1st edition

A Code Listings

Here, I will present the code I have written, coming with each of the figures, showing logical attack detection from Chapter 7.

Figure 7.5 Proposed Logical operation of the SIP preprocessor against the REFER attack

```
1  /* the block with the REFER attack detection */
2  if(compare_callid(cid1,cid2) == STS_SUCCESS)
3  {
4          if (IP_source_1 == IP_source_2)
5          LogMessage(``IP source are egual \n'');
6          else LogMessage("IP source are differents \n");
7
8          if (cancel == STS_TRUE)
9          /* it is a CANCEL request */
10         {
11                 if (cnxmap[i].invite_rec == STS_TRUE) /*connection was established
                       */
12                 {
13                         if (IP_source_1 != IP_source_2)
14                         /* DoS Attack */
15                         {
16                         LogMessage("preventing SIP client from making a call \n");
17                         DEBUG_WRAP(DebugMessage(DEBUG_SIP,"Preventing SIP client
                               from making a call \n"));
18                         GenerateSnortEvent(p, GENERATOR_SPP_SIPDECODE,
                               PREVENTING_ERROR, 1, 0, 0,
19                         PREVENTING_ERROR_STR);
20
21                                 /* if configured so, drop packet */
22                                 if (sipcnf.drop)
23                                 InlineDrop();
24                                 return;
25                         }
26                 }
27         }
28
29
30         /* NEW proposed solution against the REFER attack
31         /* if it is a refer message */
32
33         if (refer == STS_TRUE)
```

```
34          {
35                  if (cnxmap[i].invite_rec == STS_TRUE)
36                  {
37                  LogMessage("preventing REFER attack \n");
38                  DEBUG_WRAP(DebugMessage(DEBUG_SIP,"Preventing REFER attack \n"));
39                  GenerateSnortEvent(p, GENERATOR_SPP_SIPDECODE, REFER_ERROR, 1, 0,
                        0, REFER_ERROR_STR);
40
41                      /* if configured so, drop *packet */
42                      if (sipcnf.drop)
43                      InlineDrop();
44                      return;
45                  }
46          }
47  }
```

Figure 7.7 Logical Operation in attempt to detect register flooding and brute force guessing

```
1  /*if the message is REGISTER */
2  /* This is a way to detect and protect from REGISTER flooding and brute force
3  *password guessing*/
4
5  if (msg_reg == STS_TRUE)
6  {
7  LogMessage("this message is a REGISTER \n");
8  LogMessage(" but REGISTER messages may be restricted \n");
9
10         /* count how many registrations this caller
11         * has already made in a period */
12         if ((compare_url(url1_from,url2_from) == STS_SUCCESS) &&
               (cnxmap[i].expires > time_now))
13         regFrom += 1;
14
15 LogMessage("this is registration number %d",regFrom);
16 LogMessage(" made from this caller \n");
17 DEBUG_WRAP(DebugMessage(DEBUG_SIP,"Registrations made by %s are %d\n",
18 url1_from->username,regFrom));
19
20         /*check if this caller can make another registration */
21         /* Checking against REGISTER flooding and brute force *password guessing
               attack*/
22
23         if (regFrom > sipcnf.register_from)
24         {
25         event_id = GenerateSnortEvent(p, GENERATOR_SPP_SIPDECODE, REGISTER_ERROR,
26         1, 0, 0, REGISTER_ERROR_STR);
27         LogMessage("this caller can not make MORE REGISTRATIONS \n");
28
```

```
                /*if configured so, drop packet */
                if (sipcnf.drop)
                {
                InlineDrop();
                return;
                }
        }
}
```

Figure 7.8 Logical Operation in attempt to detect register flooding on a SIP registrar (similar to Figure 7.7)

```
/*if the message is REGISTER */
/* This is a way to detect and protect from REGISTER flooding  on a SIP
    registrar*/

if (msg_reg == STS_TRUE)
{
LogMessage("this message is a REGISTER \n");
LogMessage(" but REGISTER messages to be sent may be restricted  \n");


        /* count how many registrations this caller
        * has already made in a period */
        if ((compare_url(url1_to,url2_to) == STS_SUCCESS) && (cnxmap[i].expires >
            time_now))
        regTo += 1;

LogMessage("this is registration number %d",regFrom);
LogMessage(" made to this  SIP proxy/registrar \n");
DEBUG_WRAP(DebugMessage(DEBUG_SIP,"Registrations made by %s are %d\n",
url1_to->username,regTo));

/*check if this caller can make another registration */
/* Checking against REGISTER flooding and brute force *password guessing attack*/
        if (regTo > sipcnf.register_to)
        {
        event_id = GenerateSnortEvent(p, GENERATOR_SPP_SIPDECODE, REGISTER_ERROR,
        1, 0, 0, REGISTER_ERROR_STR);
        LogMessage("this registrar can not receive MORE REGISTRATIONS \n");

                /*if configured so, drop packet */
                if (sipcnf.drop)
                {
                InlineDrop();
                return;
                }
        }
```

```
35 }
```

Figure 7.10 Logical Operation in attempt to detect Registration Hijacking

```
1  /*if the message is REGISTER */
2  /* attempt to detect and prevent Registration Hijacking - rather naive *
       approach*/
3
4  if (msg_reg == STS_TRUE)
5  {
6  LogMessage("this message is a REGISTER \n");
7  LogMessage(" but REGISTER messages may be restricted if not coming from expected
       IP \n");
8
9          if (IP_source_1 == IP_source_2)
10         LogMessage("IP source are egual \n");
11         else
12         LogMessage("IP source are differents \n");
13
14         if(compare_url(url1_from,url2_from) == STS_SUCCESS)
15         {
16                 if (IP_source_1 != IP_source_2)
17                 {
18                 LogMessage("Preventing a naive Registration Hijacking attack \n");
19                 DEBUG_WRAP(DebugMessage(DEBUG_SIP,"Preventing a naive Registration
                        Hijacking attack \n"));
20                 GenerateSnortEvent(p, GENERATOR_SPP_SIPDECODE,
                       REGISTER_HIJACK_ERROR, 1, 0, 0,
21                 REGISTER_HIJACK_ERROR_STR);
22
23                         /* if configured so, drop packet */
24                         if (sipcnf.drop)
25                         InlineDrop();
26                 return;
27                 }
28         }
29 }
```

Figure 7.11 Modified Session Flow Chart to include UPDATE and re-INVITE

```
1  /* The re-INVITE and UPDATE state machine enhancement */
2  /*checking for a correct re-INVITE,i.e. when the call id is the same as
3  * the one recorded before this implies that the state cannot be
4  *Inactive, has to be Active; if not, drop the packet extra added code:
5  *Dimiter */
6
7  if (invite == STS_TRUE)
8  {
9  LogMessage("INVITE request received \n");
```

```
10 DEBUG_WRAP(DebugMessage(DEBUG_SIP, "INVITE request received
11 (cnxmap[%d].active = 5d)\n", i, cnxmap[i].active));
12
13         if (cnxmap[i].active == STS_TRUE)
14         /* since *re_INVITE only from Active state should *be true */
15         {
16         cnxmap[i].active=STS_FALSE;
17         cnxmap[i].wait_ack=STS_TRUE;
18         DEBUG_WRAP(DebugMessage(DEBUG_SIP, "re_INVITE message received \n"));
19         }
20         else
21         /*this packet is malicious or result of *wrong implementation , better
               dropped */
22         {
23         LogMessage("re-INVITe but for inactive connection, unaccpetable \n");
24         GenerateSnortEvent (p, GENERATOR_SPP_SIPDECODE, RE_INVITE_ERROR,
25         1, 0, 0, RE_INVITE_ERROR_STR);
26
27                 /* if configured so, drop packet */
28                 if (sipcnf.drop) InlineDrop();
29                 return;
30         }
31
32 }
33
34
35
36 /*Checking an UPDATE message against the modified session flow *chart
37 * (state machine) extra added code: Dimiter*/
38
39 if (msg_update == STS_TRUE)
40 {
41 LogMessage("UPDATE request received \n");
42
43         if (cnxmap[i].wait_ack)
44         {
45         DEBUG_WRAP(DebugMessage(DEBUG_SIP,
46         "UPDATE request received (cnxmap[%d].active = 5d)\n", i,
               cnxmap[i].wait_ack));
47         }
48         else if (cnxmap[i].active)
49         {
50         DEBUG_WRAP(DebugMessage(DEBUG_SIP,
51         "UPDATE request received (cnxmap[%d].active = 5d)\n", i,
               cnxmap[i].active));
52         }
53         else
```

```
54          {
55          LogMessage("UPDATE message is not supposed to be received in this state
                (Inactive) \n");
56          GenerateSnortEvent (p, GENERATOR_SPP_SIPDECODE, UPDATE_ERROR, 1, 0, 0,
                UPDATE_ERROR_STR);
57                  /* if configured so, drop packet */
58                  if (sipcnf.drop) InlineDrop();
59          return;
60          }
61  }
```

B Acronyms

CGI Common Gateway Interface

CS Call Server

DoS Denial of Service

DDos Distributed Denial of Service

ETSI European Telecommunications Standards Institute

GSM Global System for Mobile Communications

GW Gateway

HIDS Host-based Intrusion Detection System

HTTP Hypertext Transfer Protocol

IDS Intrusion Detection System

IETF Internet Engineering Task Force

IP Internet Protocol

IPS Intrusion Prevention System

ISDN Integrated Services Digital Network

ITU International Telecommunication Union

MAC Media Access Control

MGC Media Gateway Controller

MGW Media Gateway

MIME Multipurpose Internet Mail Extensions

MITM Man-in-the-middle

MM Mobility Management

NAT Network Address Translation

NGN Next Generation Networks

NIC Network Interface Card

NIDS Network-based Intrusion Detection System

OS Operating System

PBX Private Branch Exchange

PC Personal Computer

PSTN Public Switched Telephone Network

QoS Quality of Service

RTP Real time Transport Protocol

SBC Session Border Controller

SCTP Stream Control Transmission Protocol

SIP Session Initiation Protocol

S/MIME Secure/ Multipurpose Internet Mail Extensions

SP Service Provider

SPIT Spam over IP Telephony

TCP Transmission Control Protocol

UA User Agent

UAC User Agent Client

UAS User Agent Server

UDP User Datagram Protocol

URI Uniform Resource Identifier

URL Uniform Resource Locator

UTF-8 Unicode Transformation Format

VoIP Voice over Internet Protocol

VOIPSA Voice over IP Security Alliance

3GPP Third Generation Partnership Project