Assessing the security of VoIP Services

Humberto Abdelnur, Radu State, Isabelle Chrisment, Cristian Popi

Assessing the security of VoIP Services

H. Abdelnur, R. State, I. Chrisment, C. Popi
LORIA - INRIA Lorraine
615, rue du jardin botanique
54602 Villers-les-Nancy, France
{abdelnur|state|chrisment|popi}@loria.fr

***Abstract*— VoIP networks are in a major deployment phase and are becoming widely spread out due to their extended functionality and cost efficiency. Meanwhile, as VoIP traffic is transported over the Internet, it is the target of a range of attacks that can jeopardize its proper functionality. In this paper we describe our work in a VoIP specific security assessment framework. Such an assessment is automated with integrated discovery actions, data management and security attacks allowing to perform VoIP specific penetration tests. These tests are important because they permit to search and detect existing vulnerabilities or misconfigured devices and services. Our main contributions consist in an elaborated network information model capable to be used in VoIP assessment, an extensible assessment architecture and its implementation, as well as in a comprehensive framework for defining and composing VoIP specific attacks.**

I. Introduction

VoIP network have recently become widely deployed. They provide several advantages which are not feasible in traditional telephony and in that way do become really appealing to service providers and regular end users. Although the current availability provided is still lower than in traditional telephony, we expect that in the short future the two services will become equal in terms of expected performance. Since no dedicated network is used for VoIP, the user is exposed to different attacks like Denial of Service (DoS), Eavesdropping, Hijacking, etc. Although VoIP technology is becoming more mature due to its enhanced functionality, many VoIP devices and services can present security threats if not properly configured. Our work is motivated by the lack of a comprehensive VoIP security solution integrating both the management plane and a security audit platform.

In this paper we present our approach for a VoIP security assessment platform. For this purpose, our paper is structured as follows. In section II we give a background overview of a VoIP network and highlight the security threat model. We next present the problematics for achieving an automated security assessment platform. In section III we describe the architecture of our tool focusing on its VoIP assessment functionality. In section IV we describe the underlying Network Information Model capable to represent the required data. In section V we describe methods to design a penetration test and how the assessment can be automated. We discuss relationships with existing research works in section VI and conclude the paper in section VII.

II. VoIP and Security

A. VoIP Architecture and Deployment

VoIP services are gaining popularity in the latest years because of their lower cost comparing to the traditional telephony, their facility to integrate other services like video conferences, instant messages and presence services. The major signalling protocols used in VoIP networks are the Session Initiation Protocol (SIP) [20] and H.323 [22].

The traditional telephony, known as the Public Switched Telephone Network (PSTN), is based on circuit-switched networks rather than the packet-switched oriented networks like the Internet. For every connection, the PSTN network reserves bandwidth independent of whether it is used or not. To perform a call from one end to the other, the Signalling System #7 protocol (SS7) [6] is used. This latter deals with call routing, call establishment and billing information.

An important difference between VoIP networks and the PSTN is that in the VoIP networks the intelligence is located in the end devices rather than in the core switching equipment. However in VoIP services, since all the data is supposed to be sent over public networks such as the Internet (in practice some VoIP service providers do use dedicated trunks), important security issues must be addressed. In consequence, a range of attacks has to be considered; for instance: Denial of Service (DoS), Spam over IP Telephony (SPIT), eavesdropping, hijacking, etc. For a complete and comprehensive list of the threats refer to VoIPSA [7].

In a SIP architecture three main entities are identified:

- **User Agents (UA):** These are the end devices. They are only used to initialize a session. A regular UA can be a soft or hard phone (i.e. software or hardware endpoints) as well as a gateway that connects to other VoIP protocols or the PSTN.
  UA can be divided in two logical entities: User Agent Client (UAC), which is the one in charge of initiating the request, and the User Agent Server (UAS) which is the one responsible for generating the responses to the received requests.
- **SIP Servers:** These services are not mandatory to establish a session between two SIP UA devices but they provide a vast range of extra functionalities to make it easier. According to their functionality, SIP servers can be subclassified as follows.
  - **SIP Registrar Server:** It is used from a UA in

1-4244-0799-0/07/$25.00 ©2007 IEEE

order to register in a SIP domain address. The Server obtains the UA's IP address as well as the associated user and stores them for future use.
  - **SIP Proxy Server:** It is used to forward requests on behalf of other SIP entities. It can not initiate a request by itself, but it can offer additional services like for instance security, authentication and authorization.
  - **SIP Redirect Server:** It is used to redirect the caller to the searched UA. The difference with respect to the proxy is that the Redirect Server tells the entity the contact address (of the UA) rather than forward requests itself. The redirect server is also able to retrieve multiple locations in order to fork the call.
- **SIP Location Sever:** It is used to keep a database of the users containing their URLs, IP address, features and other preferences. It is used by the SIP Servers to allow an application level mobility.

B. Assessment Functionality/Problems

Network security assessment provides the necessary knowledge to the administrators in order to estimate the degree of security that their networks reach. Different concepts of assessment range from inventory management and up to almost full fledged penetration tests [16]. The main objective of an assessment is to find potential vulnerabilities, while a penetration test will also exploit them. There is a fine borderline between penetration testing and assessment. An assessment test is not supposed to exploit vulnerability with a proof of concept code, but in practice in order to avoid false positives, this borderline is crossed.

The best approach to avoid intruders to get into the network is to have the latest software updates, restrict already known vulnerable services and the most important is to periodically perform comprehensive assessment tests. For these tests, network devices and hosted services must be automatically discovered by the assessment team and next the tests must be performed. This topic is also the main theme of our paper. Our work was challenged by providing an automated approach capable to be accurate in the discovery of the VoIP infrastructure, perform user driven attacks and allow dynamic extension with security plugins. The major challenges were posed by the numerous types of services that must be checked, the security constraints that exist, and additional firewalls and NAT devices that do increase the difficulty of an accurate fingerprinting and discovery phase.

For illustration, consider that a service by itself can provide a high level of security, but when it is in dependency or relationship with others a security breach might occur due to their interaction. Information about the type of entities, the services on which they are dependent as well as the hosting device is the starting point to discover flaws in the system.

For example, a simple SIP hard phone may depend on a DHCP service to obtain its IP address, on the DNS service to resolve other IP addresses, on a TFTP service to retrieve its configuration, and on a SIP Proxy and Registrar to make and receive calls. If one of those services is impersonated or compromised the overall functionality of the phone may become unsecured.

To network administrators, many pieces of information of the topology and the configuration of devices in the network are known. That information can be helpful to make the assessment more realistic and accurate. As described in this article, such information can be inferred by advanced TCP/IP and application level fingerprinting tools.

As a reminder, security assessment is less invasive than a real attack even though it has to be carefully planned in order not to disrupt services and compromise the whole network.

III. VoIP Assessment Architecture

The three main challenges that an automated assessment must meet are:

A. Discover as much as possible the topology, the services and the configuration running on the devices in the network,
B. Store and provide all the information gathered in an easily usable manner,
C. Launch different penetration tests, discovery and/or attack actions using the information acquired.

It is worth noting that this approach is not meant to be worked out at once, but rather progressively, in order to obtain all the information related to an assessment. One should be able to provide pre-defined scripts and the possibility to generate new ones, capable to launch actions using the common knowledge base, on the fly.

An architecture overview of the assessment tool is shown in Figure 1 where it is possible to differentiate the three tasks previously mentioned. A more detailed description of these concepts is given below. Note that it is not an exhaustive architecture design; many new capabilities such as plug-ins can be dynamically added to the corresponding levels of the tool.

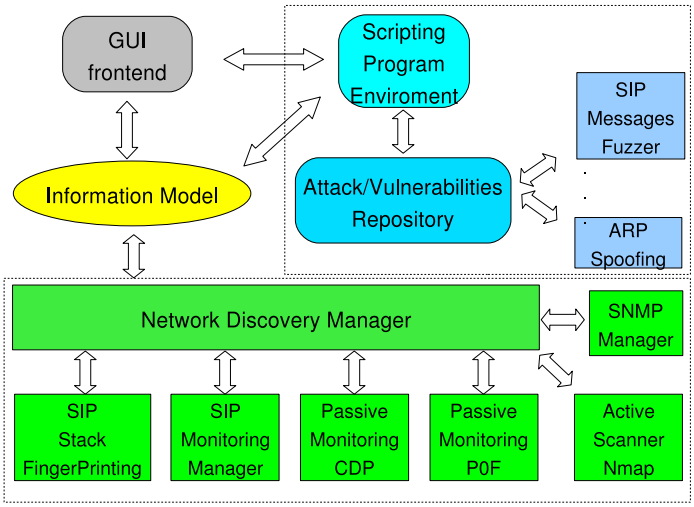

Fig. 1. Security Assessment Tool Architecture Overview

A. Discovery Actions

The ways to discover entities in a network, their configuration and their hosted services can be classified as either passive or active fingerprinting techniques. A passive fingerprinting tool is not meant to be aggressive, it only listens to all the traffic on the network and processes all this information to recognize the actual state and possible fingerprints of the device. At the opposite side, through the injection of crafted/normal packets, the active scanner tries many techniques to discover the operating system, version, open ports, etc. running in the target device. The examination of those replies provides a vast field of information that can be useful for detecting different versions of the implementations of a stack for a protocol. In the following we describe each tool that is reused by our platform and finally we conclude on how they complement each other in the overall system.

1) P0F Monitoring: This module integrates the discovery actions done by P0f [24], a passive OS fingerprinting detection tool, into the Information Model described in section IV. As a passive fingerprinting tool, it listens to the packets transiting the network across the scope of the hosting device. It analyzes the different behaviours and peculiar protocol level specifics presented by the target implementation. Using techniques that interpret the incoming and outcoming connection as well as the refused and established ones it is able to detect possible fingerprints in the messages. Thus, it will lead to know the software or/and version running in the target devices. P0F provides a mechanism to add new fingerprint signatures which depends on the combination of different values of the packet. The signature entry looks like: `wwww:ttt:D:ss:OOO...:QQ:OS:Details` as depicted in Table I:

wwww	window size. It can be *, %nnn, "Snn" (multiple of MSS) or "Tnn" (multiple of MTU) are allowed.
ttt	initial TTL
D	don't fragment bit (0 - not set, 1 - set)
ss	overall SYN packet size
OOO	options in the order they should appear in the packet.
QQ	list of oddities or bugs of this particular stack.
OS	OS genre (Linux, Solaris or Windows)
details	OS description (2.0.27 on x86, etc)

TABLE I
SIGNATURE ENTRY FOR P0F

Concerning this format, every packet captured by P0F will be compared with the list of signature entries and filtered to match the corresponding OS of the device in question.

Since this is a passive tool, no packet is injected into the network and thus the tool can be hardly detected. As a consequence it depends on the packets sent by the hosts, and this will make it less accurate than other active tools.

However, one important remark on P0F is that it can bypass a firewall, so it can be useful for detecting fingerprints through a connection allowed by the firewall, where active scanners will fail.

2) Active Scanner (NMAP): The Active Scanner module uses already existing software which allows the discovery of devices as well as their properties and running services. One such tool is NMAP [5] recognized as a very powerful tool for its remote OS detection and port scanning. Its discovery procedures are based on exact TCP/IP network stack fingerprinting. This is done by sending multiple packets to the target machine and examining specific fields in the answered TCP packets. Such packets belong to sample tests to discover supported options and vendor specific stack behaviour. Once those tests are answered, the received messages are analyzed and a fingerprint entry is created.

A signature entry is composed of many categories and is made of a list in the format `testname=value`. A value as well as a whole test can be missed depending on many factors as, for example, that the test is not supported by the system, no reply was received, etc. The result of the test includes categories like SEQ (sequence analysis of the probe packets), OPS (TCP option received from the probes), WIN (windows sizes of the probes), etc.

Once the fingerprint is created it's compared with an OS or service fingerprint signature in its database to identify the device.

This type of discovery is very aggressive since most of the sent packets violate the specifications of given network protocols. The information obtained by such a tool includes the running operating system, the open/closed/filtered ports as well as the services hosted by them, the version of the software, etc.

3) SNMP Manager: A few VoIP devices support SNMP and if the latter is not properly secured, important information can be leaked out by just interrogating the SNMP agent on the device. This entity is capable to do basic SNMP brute forcing and retrieve management related data to be fed into the assessment module.

4) Cisco Discovery Protocol (CDP) Manager: The Cisco Discovery Protocol [1] is a proprietary protocol used by Cisco devices to advertise themselves and discover other devices in the network. The objective of this protocol is network diagnosis and it is used to identify and troubleshoot the network. The packets generated by the entities supporting this protocol are distributed through multicast and contain relevant information. In the case of SIP phones, a CDP packet exposes a range of data: the actual device model, the firmware version, the device ID (which represents the name of the configuration file if this one is retrieved from a TFTP server), the MAC and IP addresses, the VLAN Domain, and some other less important information. In the functional context, every device periodically sends CDP messages to the multicast address. Cisco entities which support this functionality store the information received to be used as needed. Tools like Yersinia [10] and IRPAS [3] are instantiated in this module because of their ability to monitoring CDP packets and launch different attacks on this field.

5) SIP Stack Fingerprinting: Similar to P0F and NMAP, which use fingerprint signatures to discover the operating system and running services, additional VoIP specific fingerprinting is possible. One approach in this area of Hong Yan et al. [14] who use passive and active scanner techniques to find out properties about the SIP entities. The above mentioned work provided a table describing vendor and device specific particularities in their respective SIP stacks. This allows to identify different implementations and to classify them accordingly. For the active discovery part, the idea is to send "OPTION" messages and to check the returned set of capabilities. Depending on the answer to malformed messages, the order in the fields and/or the order in the capabilities this method is able to identify some fingerprints. Although a limited scope of devices can be fingerprinted now, an updated database of fingerprint signatures might be a very accurate method.

6) SIP Monitoring Manager: This module is in charge of classifying all the information related to SIP negotiation. Using intercepted packets it detects the current state of a phone (David Lee et al. in [15] demonstrated it using the OSPF protocol, but the same approach can be achieved for the SIP protocol). From isolated packets carrying information like the out-bound proxy, the SIP entities from which the packet has to traverse, etc. can also be recovered.

As a summary, P0F and NMap tools are found on the lower level of our discovery tools selection, where they identify different OS or applications which are running on specific devices. NMap, as an active scanner, provides more output, but it is easily detected by IDS. On the other hand, P0F is hard to discover and it is also able to detect signatures of devices behind a firewall or NATs. Meanwhile, the results offered by passive discovery are limited or incomplete and, overall, nothing sound can be passively determined, as O. Arkin describes in [9]. A third tool that allowed us to get even more information is the CDP monitor. This protocol is proprietary, but provides the most valuable source of device level information. Concerning the SIP protocol, a fingerprinting detector specific to this protocol, which complements the information gathered by the other tools, is included. Finally, the SIP monitoring manager is able to detect which is the current state of a SIP device. All the mentioned information from the different tools are merged together in order to provide a wider view of the current network.

B. Gathered Information

All the information obtained by the Discovery Actions previously mentioned, has to be represented in an organized structure which provides a simple interface to retrieve that data. This structure is defined in detail in section IV.

C. Attack Actions

Once some necessary information was obtained, different kind of scripts are launched. Those scripts basically exploit some known vulnerabilities to show the degree of exposure of the system. Also, other scripts retrieve new information that is added to the Information Model.

In order to lanuch different attacks, the tool provides a repository of scripts for which the Information Model is used to represent the necessary data. Among the scripts, attacks like ARP spoofing, Spanning tree attack, DoS, etc. can be launched derived from tools like [3], [10]. Other SIP related scripts are SIP-Registrar user enumeration, eavesdropping and RTP playout, which are instantiated in the fuzzy packet [13].

The Attack/Vulnerability repository also allows storing new scripts by simple addition of attacks files. On the fly dynamic testing is possible by our Scripting Program environment which offers an interface to create, try and modify those scripts.

One of such scripts is the SIP Messages Fuzzer. Security testers of protocol implementations have showed that malformed message (containing for instance not protocol compliant values) fields are able to crash or exploit some entities. In a well known case, only one out of nine SIP phones was able to pass the tests performed by the "PROTOS Test-Suite: c07-sip" [23]. One single test may contain one or more exceptional elements which can range from exceptional IPv4 addresses to malformed tags. The tests are very simple and mostly probe the robustness of the parser, but they show the weakness in most implementations.

The assessment model provides a module to launch attacks in which messages are sent with a specified fuzziness. This module is instantiated from Fuzzy Packet [13] which was our first step through VoIP security assessment. Its architecture is based on an extensible plug-in system, where one main entity is in charge of delegating responsibilities. The functionality depends entirely on test scenarios and the associated plug-ins. In this way, new features can be easily added. A scenario is represented in XML and defines an active template as well as the binding with the responsible plug-ins. The scenario language expresses conditions that can be triggered in Python code.

Fuzzy Packet also provides scenarios which show how an existing call can be eavesdropped or even hijacked. By sniffing and injecting packets over the network it is possible to analyze the messages and generate rogue data.

Tools like the latter and Scapy [11] allow the user to read and inject packets, with all the desired custom options, in an extremely easy manner. The main difference between those two tools is that the first expresses the scenarios in a XML format, while the second is a low level packet injecting library.

IV. Information Model

The main objective of our Information Model is to represent, in a structured fashion, all the gathered information and to simplify access to it for any possible assessment. It represents the topology of a network, provided services, applications and any relevant information that can lead to find vulnerabilities.

Standards like the Common Information Model (CIM) [2] describe a sound and commonly used way to design an Information Model. Concretely, it defines a conceptual schema,

where it specifies how elements are represented. It is based on UML from which it leverages the object oriented modelling.

The gathered information is obtained in different ways and previous sections in this paper showed how discovery actions lead to such data. The design objectives must allow for an enhanced usability and capability to represent security related information as well as serve for future reuse. The main module contains a collection for every entity (i.e. devices, services or types of configuration) such that checking for one device in particular should be easily done. The navigation in the model should also allow starting from an entity and getting all the others pieces of information related to it. A simplified UML diagram of the model is shown in Figure 2.

The main module in the Information Model is the Device class, which contains information of the discovered entities: OS, version, etc. It also includes the settings and the running configuration, which are not necessarily the same. The first is related to the meta-configuration, i.e. the values defined to configure the device, while the second are the values used by the device. To illustrate such a case, we define an IP address received from a DHCP-Server as a setting while the actual IP address of the device belongs to the running configuration.

Each Device class is associated with a list of Service classes which represent the services hosted by it. The Service entities that are primarily concerned in this model are the ones related to SIP entities:

- DHCP Service: If present in a network, some entities depend on it to obtain their IP address. Also, extra information can be obtained, as for instance the routing gateway IP, the DNS server IP, a TFTP server IP, etc. If this service is compromised (i.e. fake IPs information is given by the service), the attacker can set up its own service and do malicious action as explained below.
- Routing Gateway: If this service is compromised, the attacker can act as a man in the middle and filter, modify or intercept the VoIP traffic.
- DNS Service: Entities rely on it to resolve IP addresses. Once the service is compromised, the attacker can resolve URL names to an IP address which is not the real one and for which he/she is providing a rogue service.
- TFTP Service: Many SIP phones request servers configuration information (i.e. files), that are used for configuration purposes, from TFTP. If this service is compromised, the devices relying on it will be under the control of the attacker.
- NAT Service: In a SIP environment it presents an important role because its existence will modify the payload transported at the application level. In order to fake information and to be as convincing as possible, this information has to be well known.

Concerning the Information Model, the data can be acquired by two different interfaces which provide more suitability for different tests and attacks.

- **Classical type:** This is the classical syntax of Object Oriented programming languages to navigate through the attributes of classes.
- **Filtering type:** This technique shows an approach focused on the use of filters. Each entity in a class has a function associated to it with the same name as the attribute. This function takes as argument the parameter "filter", which consists of a string that can be validated in each of the current entities. Only the entities satisfying the filter value will be returned in a collection instance. Such a collection provides the same methods and the same attributes declared by the items contained in it and its functionality is to map the methods and attributes to every one of its items.

To illustrate the basic concepts of both syntaxes, consider an example where one desires to retrieve all the TFTP service instances from the devices in the network (i.e. the ones identified by the discovery tools), which contain the specific file "SIP000B46D9CB86.cnf" in their repository. Supposing that the *Model* variable is an instance of the Information Model, the two possible ways to satisfy that request are shown below.

Classical type:

```
tftplist = []
for device in Model.Device:
   for service in device.hostedServices:
      if isinstance(service,TFTP_Service):
         if "SIP000B46D9CB86.cnf" in service.files:
         tftplist.append(service)
return tftplist
```

Filtering type:

```
Model.Device().hostedServices(
   filter="isinstance(self,TFTP_Service)
           and 'SIP000B46D9CB86.cnf' in self.files")
```

It is worth noting that those two techniques can be freely interlaced.

V. Writing Assessment Tests

From the security point of view of a network administrator it is important to know the vulnerabilities of the managed system. Once these are detected, the administrator can take the necessary measures to make a tradeoff between the desired level of security and available flexibility to existing services. For this reason, an assessment test is helpful to detect any possible threats and it does so from an intruder's perspective. To design a penetration test in order to achieve a complete assessment, multiple conceptual solutions exist, among which the more important are Attack Trees, Attack Graphs, Petri's Network variants, etc. [12], [17], [19]. In the following section we describe the basic foundations of an Attack Tree design together with an instantiation related to an attack to SIP devices in a network. Our choice of the Attacks Trees design was made for the sake of simplicity, in order to illustrate possible attacks. Once the achievement of a possible attack explained, we proceed to show how it can be accomplished in our scripting environment.

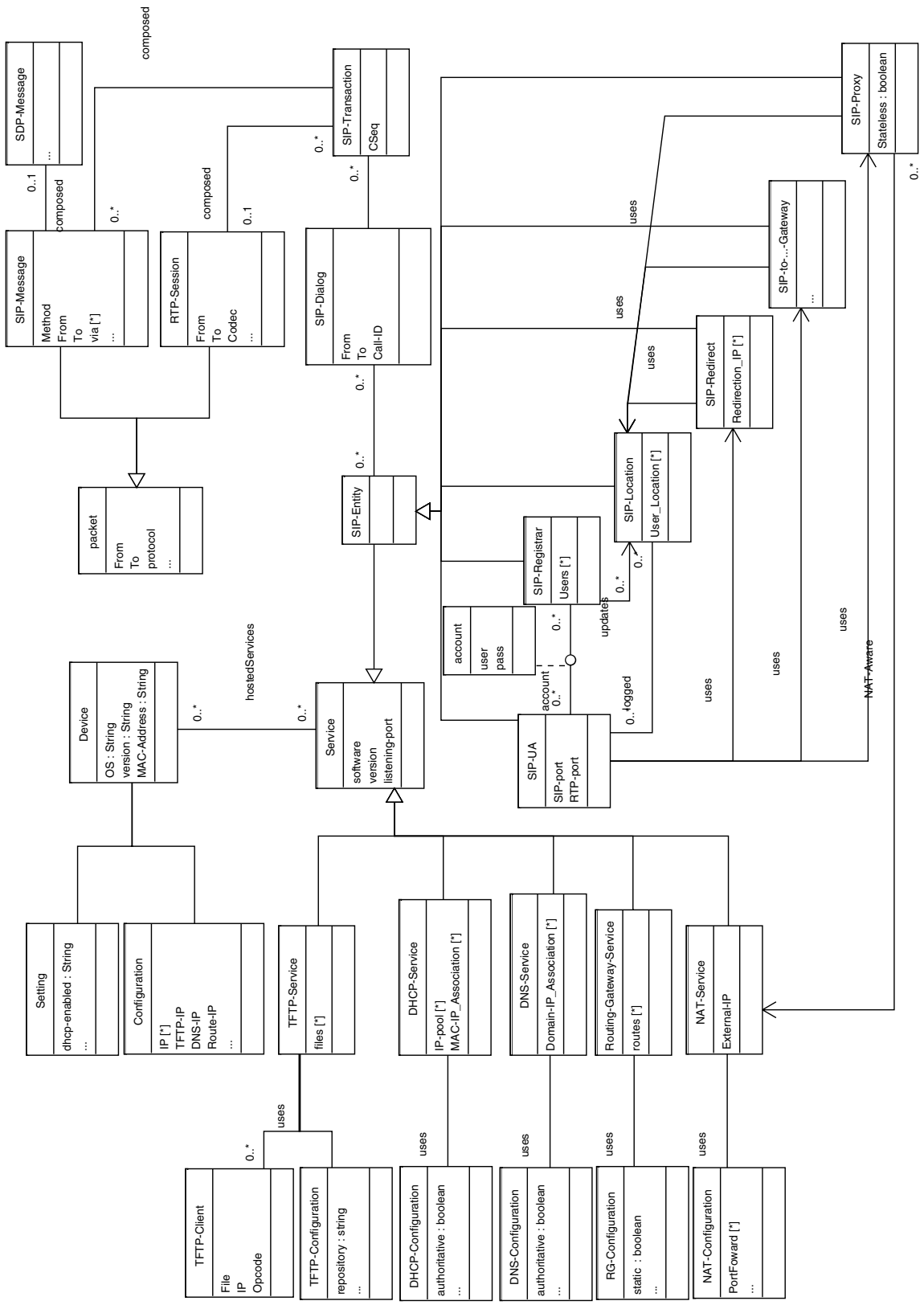

Fig. 2. Information Model UML diagram

A. Attack Trees

Attack trees [19] were designed to summarize possible actions or paths that must be taken in order to perform a penetration test. These trees represent each action as a node of a tree, where the root node represents the desired result actions. Each internal node is considered to be done when one or all of its children (depending if defined as a disjunction or conjunction respectively) are done, i.e. a bottom-up flow. The leaves define the actions that should be done. To differentiate a disjunction from a conjunction of actions in an internal node, the "and" string is attached to the bottom of the internal node if the action is a conjunction, and an empty string where there is a disjunction.

Attack trees are considered to be limited in their functionality, but they provide a good representation of the possible penetration test. In the following an attack tree will be exemplified and we will explain how to perform it using the Information Model described in section IV and the Scripting Environment described in section V-C.

B. VoIP Attack Tree Example

This example is intended to show some possible ways to intercept the SIP negotiation of a phone, and to go even further and intercept a VoIP call. Figure 3 shows the attack tree example with extra boxes that are used for the ease of the presentation.

As already mentioned, the main objective of this attack tree is Intercepting a Call. This example shows two possible main branches to achieve this goal.

1) One approach in which the end-devices will not have to be exposed would be to intercept all the out-going SIP transit traffic. This can be possible if, for example, the out-going proxy of the network is impersonated. The next example describes this attack that requires two actions to be done together: 1) Set up a fake SIP-Proxy in order to provide the needed services and 2) alter the default route to reach the original proxy via the fake one. There are several ways to achieve the latter action, like for instance ARP poisoning, Spanning Tree attacks, DNS poisoning. We would recommend in general to go for layer 2 attacks because these are difficult to detect by an Intrusion Detection System (IDS).
2) The second approach aims at changing the configuration of the SIP User Agent. It also shows some possible ways to accomplish it.
    a) The first goal is to gain remote access to the device (if possible), which is not an easy task, but which can be done if the software is not up to date, for example. The steps described next aim at discovering as much information as possible, and the use of tools like Nmap (as described in section III-A) help to discover the open ports on the device as well as running services on each of them (including the version and software). Once all that information is processed, databases of known vulnerabilities as US-CERT [8], can be searched for the respective software and version, and one can then check for known vulnerabilities and exploit them.
    b) Secondly, the penetration tester can alter the services which a SIP UA is relying on (e.g. DHCP server, TFTP server, etc.). Thus, those services can be compromised in order that every UA retrieving its configuration from the TFTP server gets rogue information, which allows the attacker to intercept all its SIP traffic. This step, shown in the Figure 3, is framed in the *Box A*. Each of its sub-boxes (i.e. 1, 2 and 3) regroups related activities. Note that Box 3 has to be done in conjunction with Box 1 or 2 and only afterwards, not as standalone.

- **Box 1:** This attack shows how to redirect all traffic to a rogue service. The first thing that the attacker has to do is to set up a TFTP server. Once done, all the traffic directed to the target TFTP has to be transferred to the machine running this rogue TFTP service. One possible action to achieve this redirection is to poison the network with ARP spoofed packet. This poisoning has to be done just to the target devices or in switches close to the service. Other techniques exist to accomplish the same, as, for example, Spanning Tree attacks, but are out of the scope of this article. Another solution to redirect the traffic is to directly set up the IP address of the fake TFTP server on the target device. This can be easily done if the device is getting its TFTP IP address from the DHCP service. If that is the case, the attacker can run such a service with this fake information and overwrite the genuine data.
- **Box 2:** This approach shows a way where the files to be retrieved from the TFTP server are overwritten. First the attacker has to know which files he wants to overwrite or create. The identification can be done by a passive scanner which just reads traffic on the network and interprets messages under the TFTP protocol. This is possible, since no encryption is usually used in this step. Additionally, some devices use a file name that is built using well know prefixes and postfix attached to the MAC address of the device. Once the files are known, the TFTP server can be tested to see if write access is granted on it and if possible, these files are modified with rogue configuration data.
- **Box 3:** Finally the objective of this sub-tree framed in the box is to identify which are the devices that retrieve their configuration from the TFTP servers and make them reload the new configuration. The identification can be done as

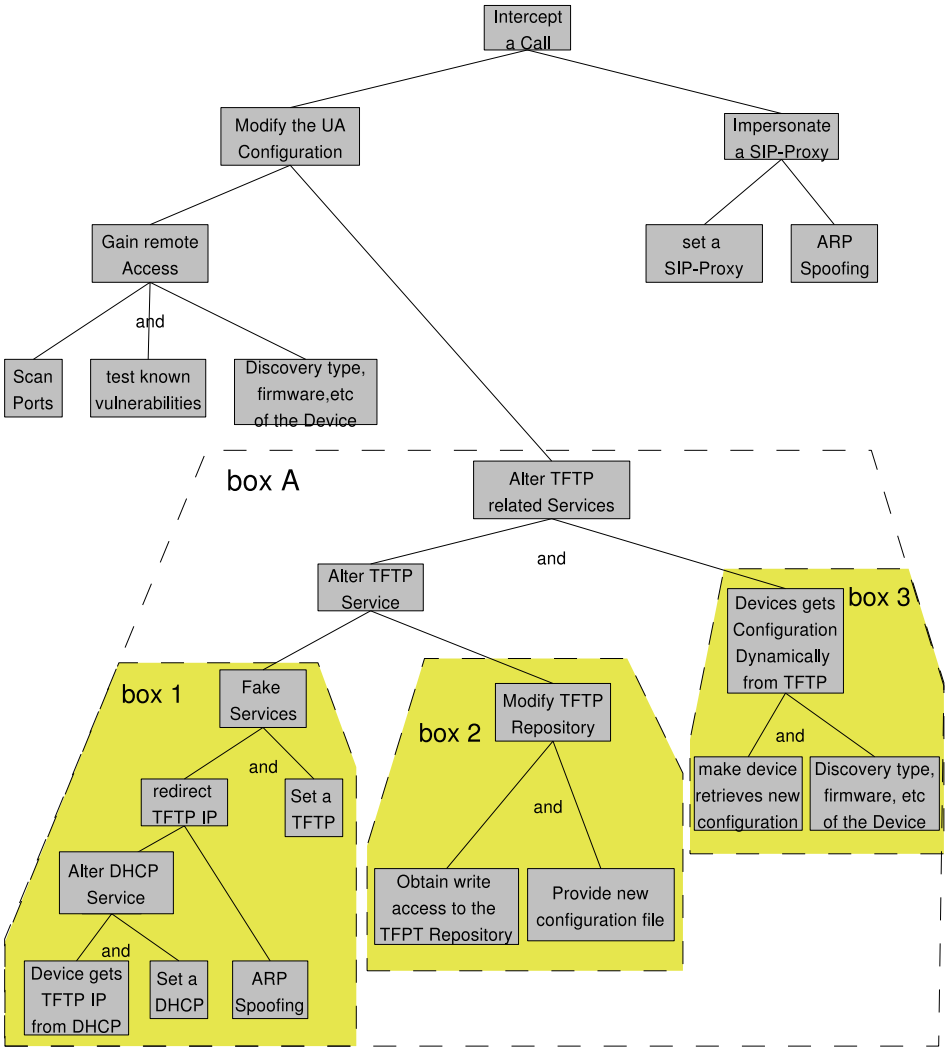

Fig. 3. Attack Tree Example

explained in the Box 2 by a passive scanner which is listening to the TFTP protocol. Once the TFTP dependent devices are identified, information concerning their OS, software, version, etc. will help to find vulnerabilities that will force the device to retrieve the new configuration (e.g. reboot the device).

It is worth to note that those attacks are very naive and if they accomplish the objective, it is very probable that they will be detected by an IDS. The contribution of this example is to show how automation of these tasks can be done.

C. Scripting Environment

This extension represents the active assessment part of the architecture. It provides an environment where pre-defined tests can be retrieved from a repository and where new ones can be generated on the fly, with the possibility of storing the procedure in the repository for later use.

As the core implementation and the wrappers (i.e. wrappers for the external tools used) are coded in Python, all tests are specified as a Python script and the interactive environment is a derivation of the one from Python.

1) Python Language Embedding: Lately, tools like Scapy [11] (a Packet Manipulator Program) are gaining popularity, mostly because their usability and flexibility of programming. Our approach was designed with the same goals in mind, to include functionalities developed by these tools and complement them with information from application levels. Thus, the information included is the one described in the Information Model which will provide all the data obtained by the discovery actions. Meanwhile, the interactive environment corresponds to an extension of the Python interactive environment which provides functionalities such as a low level packets generator and pre-defined high level attacks.

2) Example script: Figure 3 in box 1, shows a sub-attack that, as explained previously, tries to redirect the real TFTP server IP to a rogue one.

To alter the DHCP Service, first a rogue server must be

running in a vulnerable machine, which will try to distribute the IP of the fake TFTP server.

To create an ARP spoofing attack the repository provides a predefined script, which may require some information like the target IP, the IP of the device that has to be impersonated and the MAC to redirect the packets. For such attacks, the application has to be launched inside the LAN. The script attack should proceed like shown below.

```
## Retrieves all the Service instances of the devi-  ##
## ces which provides TFTP                            ##
>>> TFTPServices = Model.Services(filter=
      "isinstance(self,TFTP_Service)")

## Start ARP spoofing for each Server                 ##
>>> for tftp_server in TFTPServices:
...   tftpIP = tftp_server.Device.Configuration.IP

## For the IP of the clients of this TFTP             ##
...   for clientIP in tftp_server.TFTP_Client().IP:

## Launch ARP poisoning attack                        ##
>>>     Attack.ARP_Poisoning( targetIP= clientIP,
                              impersonateIP= tftpIP,
                              fakeMAC= fakeMAC)
```

The second alternative to alter a TFTP Service described as Box 2 (Figure 3), i.e. for discovery if writing access is granted, should proceed as follow.

```
>>> from scapy import srl,IP,UDP
>>> from scapy.extras import TFTP
>>> grantedAccess = []

## Retrieves all the Service instances of the devi-  ##
## ces which provides TFTP                            ##
>>> TFTPServices = Model.Services(filter=
                    "isinstance(self,TFTP_Service)")

## Check writing access for each Server               ##
>>> for tftp_server in TFTPServices:
...   tftpIP = tftp_server.Device.Configuration.IP
...   for file in tftp_server.files:

## Use scapy to inject TFTP packets                   ##
...     p = srl(
            IP(dst=tftpIP)/
            UDP()/
            TFPT(dst_file=file,Opcode=WRITE_REQUEST))

## Check the answer for writing access                ##
...     if p[3].Opcode != ERROR_CODE:
...       grantedAccess.append((tftpIP,file))
```

Finally, the third box shown in Figure 3, which should work in conjunction with the box 1 or 2, requires that the device requests the new rogue configuration from the TFTP server. Some SIP hard phones reload the dial plan by sending a NOTIFY message with the option event "check-cfg" and depending on the current configuration it is possible to make them reboot and reload it.

```
>>> from scapy import srl,IP,UDP
>>> from scapy.extras import SIP

## Retrieves all the SIP-UA with the                  ##
## version which can be rebooted                      ##
>>> sipUAs = Model.Services(
                    filter="isinstance(self,SIP-UA)
                    and self.software = 'XXXX'
                    and self.version <= 'X.XX'")

## Use scapy to inject the NOTIFY packets             ##
>>> for UA in sipUAs :
...   UA_IP = UA.Device.Configuration.IP
...   p = srl(
            IP(dst=UA_IP)/
            UDP()/
            SIP(Method="NOTIFY",Event="check-cfg"))
```

VI. Related Work

Security assessment for VoIP is rather immature at the moment. Both a methodology as well as supporting tools are missing. Some recent books like, for instance, Practical VoIP Security [21] and VoIP Security [18] describe the functioning, best practices and recommendations for securing VoIP network and services without, however, providing guidelines on how to test them.

In the research community, recent work on formal descriptions on how penetration tests can be accomplished ranges from modelling attack trees [19], Colored Petri Nets [12] to Attacks Graphs [17]. The major differences consist in their power of expressiveness as well as in their complexity. However, all of them do address in a formal way how to combine attack actions and show how a main goal can be achieved from a sequence of steps that will jointly form a global plan.

In practice, Nessus [4] is a vulnerability scanner among the most widely used by network security teams. It is able to obtain from its scans or by interacting with NMAP [5], a large variety of essential functionalities that are highly relevant in an assessment test. It also provides a scripting language NASL (Nessus Attack Scripting Language) to add new functionalities, but in fact this language was meant to be secure rather than powerfull which limits the capability to write advanced tests. Meanwhile, Scapy [11] embeds itself in Python providing a real language environment lacking however a common Knowledge database to show the current topology and status of the network. Our approach took as starting point the provision of a common Knowledge database obtained from different applications and it is embedded in a programming language (Python) to freely define the steps to take in the assessment process.

VII. Conclusion

Our current research, performed in the MADYNES research group, addresses the secure management of VoIP networks. One of our main work direction is an integrated system able to retrieve as much as possible the information of the environment. Such data can be used by penetration tests and can show scenarios allowing to identify where the flaws of the system are located.

In this article we mainly focus on VoIP networks but we consider that this approach as well as the architecture model can be extended to several other types of networks. We show how accurate information from the network can be gathered, and next provide an Information Model capable to represent it in an appropriate way for assessment methods.

Our paper has three main contributions: we propose a network information model capable to represent the information required to perform VoIP assessment, we describe a VoIP assessment architecture and its implementation and we build a framework based on attack tree modelling in order to represent and write VoIP attacks. Our future work consists in a smart VoIP infrastructure capable to perform automated intrusion prevention and active response.

REFERENCES

[1] CDP: Cisco Discovery Protocol. http://www.cisco.com/ : Configuring the Cisco Discovery Protocol.

[2] "Common Information Model (CIM)". http://www.dmtf.org/standards/cim/.

[3] "IRPAS: Internetwork Routing Protocol Attack Suite". http://www.phenoelit.de/irpas/.

[4] "Nessus: Vulnerability Scanner". http://www.nessus.org/.

[5] "Nmap: Network Mapper ". http://www.insecure.org/nmap/.

[6] Signalling System #7 (SS7). http://www.itu.int/ITU-T/.

[7] "The Voice over IP Security Alliance (VOIPSA)". http://www.voipsa.org/Activities/taxonomy.php.

[8] "US-CERT Vulnerability Notes". http://www.kb.cert.org/vuls/.

[9] O. Arkin. "Demystifying Passive Network Discovery and Monitoring Systems". http://www.usenix.org/publications/login/2005-06/pdfs/arkin0506.pdf, June 2005. ;login: The USENIX Magazine.

[10] D. Barroso and A. Andrs. "Yersinia: Framework for layer 2 attacks".

[11] Philippe Biondi. "Scapy: Packet Manipulation Program".

[12] O. Dahl and S. Wolthusen. Modeling and execution of complex attack scenarios using interval timed colored petri nets. In *IWIA '06: Proceedings of the Fourth IEEE International Workshop on Information Assurance (IWIA'06)*, pages 157–168, Washington, DC, USA, 2006. IEEE Computer Society.

[13] R. State H. Abdelnur, V. Cridlig and O. Festor. "VoIP Security Assessment: Methods and Tools". In *The 1st IEEE workshop on VoIP Management and Security*, April 2006.

[14] H. Zhang Z. Shae H. Yan, K. Sripanidkulchai and D. Saha. "Incorporating Active Fingerprinting into SPIT Prevention Systems". *Third Annual VoIP Security Workshop (VSW'06)*, July 2006.

[15] D. Lee, D. Chen, R. Hao, R. Miller, J. Wu, and X. Yin. A formal approach for passive testing of protocol data portions. In *ICNP '02: Proceedings of the 10th IEEE International Conference on Network Protocols*, pages 122–131. IEEE Computer Society, 2002.

[16] C. McNab. *Network Security Assessment*. O'Reilly; 1 edition, March 2004).

[17] S. Jha R. Lippmann O. Sheyner, J. Haines and J. Wing. Automated generation and analysis of attack graphs. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 273, Washington, DC, USA, 2002. IEEE Computer Society.

[18] J. Ransome and J. Rittinghouse. *VoIP Security*. Elsevier, 2005.

[19] B. Schneier. "Attack Trees, Modeling Security Threats,". *Dr. Dobb's Journal*, December 1999.

[20] H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. http://www.ietf.org/rfc/rfc3261.txt, June 2002.

[21] A. Zmolek A. Rosela M. Cross L. Chaffin B. Baskin T. Porter, J. Kanclirz and C. Shim. *Practical VoIP Security*. Andrew Williams, 2006.

[22] International Telecommunications Union. H.323 Standards. http://www.itu.int/home/index.html.

[23] Oulu University. PROTOS Test-Suite: c07-sip. http://www.ee.oulu.fi/research/ouspg/protos/testing/c07/sip, 2005.

[24] Michal Zalewski. "P0f: a versatile passive OS fingerprinting tool".