https://doi.org/10.26552/com.C.2013.2A.191-196

Jakub Safarik – Miroslav Voznak – Filip Rezac – Lukas Macura *

IP TELEPHONY SERVER EMULATION FOR MONITORING AND ANALYSIS OF MALICIOUS ACTIVITY IN VOIP NETWORK

The paper aims at gathering information about attacks from real internet infrastructure and their analysis. For this purpose, we prepared a set of honeypots monitoring various aspects of VoIP infrastructure including SIP endpoint and SSH terminal emulation. SIP endpoints are registered with real SIP registrar and the incoming calls are routed to a honeypot according the rules in dialplan. The honeypot gathers valuable data about hacker's activity with no threat to production systems. Analysis of the honeypot data is crucial for further improvement of existing security mechanisms in VoIP networks. The paper describes the honeypot's behaviour and brings an analysis of a detected malicious activity as well.

Keywords: *Artemisa, Dionaea, Kippo, VoIP attacks, VoIP honeypot.*

1. Introduction

The paper describes the use of honeypots in a VoIP infrastructure. These systems become increasingly necessary as the number of IP-based telephony solutions rises. Nearly all large companies today rely on some kind of IP telephony in their internal communication. This situation only induces greater hacker interest in these services. Nowadays, many companies have experienced abuse such as social engineering or spit calls [1].

The way to protect this infrastructure is to keep up with hackers and constantly improve security mechanisms. But achieving this simple goal is not easy at all. The basic rule is to keep all systems and their versions up to date, with at least access policies properly set and encryption of all crucial data. But this is not always possible in VoIP systems. The question is how do we find the system's bottleneck? Each of us needs to turn into a hacker in his own system to find system weaknesses. Although this gives a better understanding of the whole infrastructure, the number of security holes found depends on the skills of security auditor. Even if the auditor fixes all existing weaknesses there can still be security holes which can be exploited by either a foreign or inner attacker.

Another option is to create honeypots which lure hackers. Using these honeypots, we can obtain real data about hacker activities and map actual attacks in the network, which is otherwise not possible [2, 3].

The main purpose of a honeypot is to simulate the real system and interact with anyone in the same way as the production system would. It watches the behaviour of anyone who interacts with it [4].This article provides a close look on honeypots referred to as Artemisa and Kippo.

2. Honeypot features

An Artemisa honeypot can be deployed in any VoIP infrastructure which uses a SIP protocol. In this infrastructure it plays a role of a regular SIP phone (see Fig. 1)

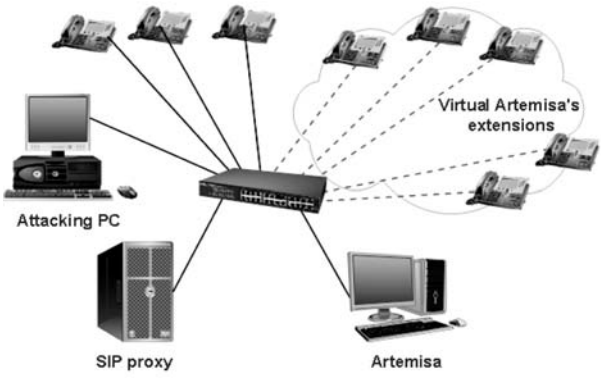

Fig. 1 A VoIP topology with a honeypot

The honeypot itself can run on a physical or virtual machine. The program connects to SIP proxy with the extensions defined in

* **Jakub Safarik, Miroslav Voznak, Filip Rezac, Lukas Macura**
Department of Telecommunications, Faculty of Electrical Engineering and Computer Science, VSB-TU Ostrava, Ostrava-Poruba, Czech Republic, E-mail: jakub.safarik@vsb.cz

a configuration file. The extensions should be within the range which is typically used for real accounts. The main purpose is to establish a better masking against the potential attackers [5]. Artemisa itself does not simulate PBX but rather an active end point device.

Once the call is established on one of Artemisa extensions, the honeypot simply answers the call. At the same time, it starts to examine the incoming SIP message. Artemisa then classifies the call and saves the result for a further review by the security administrator (Fig. 2).

The message is classified in the following steps. First of all, Artemisa looks for fingerprints of well-known attack tools. If the attacker uses some popular hacking tool, the fingerprint of this tool can easily reveal the malicious intentions. Then it checks domain names and SIP ports on the attacker side (provided they are really opened). There is also a similar check for media ports. Requested URI are also checked, as well as the ACK message received from the user. Finally, Artemisa checks the received RTP stream – provided a RTP stream was established (the audio trail of the received call can be stored in a WAV format).

```
... output ommitted ...

| | Category: Interactive attack

+ Checking if media port is opened...
|
| No RTP info delivered.
|
| Category: Spoofed message

... output ommitted ...

+ The message is classified as:
| Attack tool
| Spoofed message
| Interactive attack
| Dial plan fault
| Scanning
| Ringing

************************************* Correlation **************

Artemisa concludes that the arrived message is likely to be:

* The attack was created employing the tool SIPVicious.
* A flooding attack.

... output ommitted ...
```

Fig. 2 An example of the output file

This sequence of procedures helps Artemisa to classify the call. The result is then shown in a console. The results can be saved into a pre-defined folder or they can be sent as a notification by e-mail. Once the call has been examined, a series of bash scripts is executed. These scripts are executed with pre-defined arguments. Artemisa can launch some countermeasures against the incoming attacks [6].

A. Kippo features

The second tested honeypot is based on different foundations. It is not VoIP oriented as Artemisa. It simulates a SSH server. When someone tries to connect to a server with a honeypot running on it, the twistd application redirects this user to the honeypot. This happens where the user IP address is not included in the list of permitted IP address.

Once the connection with the honeypot is established, the attacker must enter correct username and password. These are set to the most used username *root* and password is the second most common combination of numbers *123456* (Table 1 lists Top 10 most frequently used passwords). Other combinations for the root access can be added to `data/pass.db` file.

Kippo logs every login attempt. Where the entered combination is valid, the intruder is granted access to a fake filesystem. Every command entered into the honeypot is logged and behaviour typical for a particular command is emulated (for the most common commands only). If the user tries to download something from the Internet, Kippo saves this file into a secure folder for further examination.

All logs made by *Kippo* are saved in a *MySQL* database which facilitates the subsequent analysis.

B. Dionaea features

All previously mentioned honeypots were single service oriented ones. Dionaea belongs to a multi-service oriented honeypot which can simulate many services at a time. Typically is information from these multiple services only general but dionaea serves only a small number of them like SMB (Microsoft's printers, files, serial ports sharing protocol), HTTP, FTP, TFTP, MSSQL (Microsoft SQL server), SIP protocols. Attackers abuse these protocols in most cases. Dionaea has also ability to save malicious content needed by hackers securely, but contrary to Kippo, it can also emulate code from these files.

Describing features of all these protocols is beyond the scope of this article and further features focus only on the SIP protocol. Dionaea works in a different way as Artemisa. There is no need for connecting to an external (or production) VoIP server. It simply waits for any SIP message and tries to answer it. It supports all SIP requests from RFC 3261 (REGISTER, INVITE, ACK, CANCEL, BYE, OPTIONS). Dionaea supports multiple SIP sessions and RTP audio streams (data from stream can be recorded). For better simulation of a real IP telephony system, it is possible to configure different user agent phone mimics with custom username, password combinations. There is functionality for a different pickup time on simulated phones via pickup delay feature. All traffic is monitored, and logs are saved in plain-text files and in sqlite database.

3. Monitoring traffic using artemisa

As mentioned before, Artemisa investigates all traffic which is routed to its extensions. That's not the whole truth. Artemisa can run in three different modes depending on the settings in the `behavior.conf` file. These modes are called passive, active and aggressive.

In the passive mode, Artemisa only takes the incoming calls and answers them. Using the active mode, we can achieve the same functionality as in the passive mode. In addition, Artemisa starts to examine the incoming SIP messages as described above. The last mode – the aggressive mode – attacks the intruder with its own bash script (the scripts are located under the `/scripts` directory). Typically, we run Artemisa in the active mode so that all SIP messages routed to the honeypot are analysed.

To test the usefulness of an *Artemisa* honeypot, we prepared some tests in our testing topology. We built a simple VoIP network with asterisk running as PBX and some end-point hardware and software phones. The honeypot was installed on a machine inside our network with five extensions. These extensions were running in the active mode. Since we are developing our own IPS system, we have chosen to test the honeypot under test scenarios similar to that IPS system.

4. Results

A. Artemisa's usability tests

First of all, we should start scanning the whole network from the point of view of a typical intruder. Many applications can be used for this purpose. We used two such applications – *nmap and SIPVicious* [7].

Both these applications yielded useful information. Yet neither *nmap* nor *svmap* was detected by the honeypot. In case of *svmap* there was information about the incoming SIP message, but this message was not analysed. No results were created after the network was scanned. This behaviour was quite surprising as typically, each attack starts by scanning the network. *Artemisa* should take account such situations.

With *svmap* we know about the running user-agent at honeypot's IP address (Fig. 3)

```
158.196.244.241:5060  |  Twinkle/1.4.2  |
T-Com  Speedport  W500V  /  Firmware  v1.37
MxSF/v3.2.6.26
```

Fig. 3 Svmap application output

Using this information we began looking for extensions running in the testing network. Direct scanning of the SIP proxy server was not detected by the honeypot, but when we use the *svwar* tool directly against the IP address on which a honeypot is running, we get information about all active extensions. This scan was recognized by the honeypot and an adequate result file was created. *Artemisa* correctly concludes that messages received came from a *SIPVicious* scanner. On the other hand we know from the *SIPVicious* output that these extensions do not behave as normal clients. This can stir up more caution on the side of the intruder.

The aim of other attacks was to flood the client's device with various types of SIP messages. Using some of these attacks, the intruder can achieve a DoS attack on a closed group of end-point devices [8]. For this kind of attack we used a number of tools including *udpflood, rtpflood, inviteflood* and *sipp* [9].

Each of these applications can launch a simple DoS attack. As *Artemisa* is a mere VoIP honeypot, it only detects attacks using the SIP protocol. Accordingly, only flood attacks from *inviteflood* and *sipp* were detected. In case of *inviteflood*, the application was successfully recognized thanks to its well-known fingerprint.

The *Sipp* application was not designed for hacking or penetration testing but this functionality can be achieved easily. We used specific call scenarios with a similar impact as the above mentioned flooding tools. Using sipp we can generate a high number of SIP messages which was immediately detected as a flood attack by the honeypot. In this situation, the whole honeypot stopped responding shortly and no result was recorded for the attack at all. Mere 250 SIP messages per second caused this situation. If we use lower sending rates, the attack was recognized well and the output file was successfully created.

Identifying a spit call is one of the most important features of the honeypot. We used the application called Spitfile for simulating these calls. Spitfile is an open-source SIP penetration tool. Using this tool, we can easily generate arbitrary calls. All of these calls were successfully detected by Artemisa and the appropriate output files were generated.

At last we tried to make a call to the honeypot extensions with hardware and software phone. Calls were marked as a scanning and ringing attack in both cases. So it seems that Artemisa evaluated almost every SIP message aiming at its extensions as some kind of attack.

The results from the detected attacks are stored in the `results`/directory inside the Artemisa folder. The output is in a simple text format and in html format, both of them containing the same data.

All functionalities mentioned before concern honeypots running in the active mode. The aggressive mode looks more interesting with its ability to counterattack the intruder. Artemisa contains three bash scripts to stop malicious activity. However, a close look at these scripts was surprising.

Let's start with the last script `on_spit.sh`. Inside this script, there is only one comment. This comment may activate a firewall rule, but the command is not included. This script is totally useless unless we rewrite it. Even the remaining scripts do not contain anything but comments inside. A simple condition (commented) is included in the `on_scanning.sh` script, which runs a python script to crash the scanning by the *SIPVicious* application (but only this particular application). The `on_flood.sh` script has a commented command inside to apply an iptables rule on the IP address and port. These are given by a parameter. This solution is not bad but if we want to block some traffic, there is a chance that a false positive attack will be blocked, so some automatic recovery

mechanism should also be included. This can be easily solved by adding another script. This script will remove the rule after a certain interval. The main issue in blocking traffic using iptables is that the command applies the rule on a local machine. However, it only blocks the consequences on the honeypot, not on the main firewall which protects the whole infrastructure. This feature makes the aggressive mode useless against the attack of any intruder.

Using the honeypot in the passive mode is worthless because *Artemisa* only answers the call with no further analysis and without results being saved to a file.

B. Kippo data analysis

We use a *Kippo* honeypot to analyse SSH traffic in a real network with seven active monitoring sensors. The honeypot has been active and gathering data for a month. During this period, 873342 connection attempts were observed.

Only a small part of these connections was successful. Table 1 lists ten most frequently used password combinations enabling connections.

10 Most frequently udes password combinations Table 1

Password	count
	28146
123456	17625
password	6325
1234	5663
12345	5501
123	5342
1qa2ws3ed	5278
a	5121
test	4743
qwerty	4601

As we can see from the table, password *123456* was used 17625 times. The correct username root was used only in 2551 cases. These 2551 cases led to a connection to a fake filesystem. An intruder then typically uploads some kind of a script which should be used for a DoS attack on an outside server [10]. The in-depth analysis of the attacker's input is beyond the scope of this article.

Another interesting fact acquired from the honeypot is the approximate location of the origin of the attacker's connection (see Fig. 4).

The figure only shows first ten positions. Attacks from Germany account for 25.21% of the total connection attempts. China came second with 20.33% and Mongolia third with 15.52%. Almost 75%

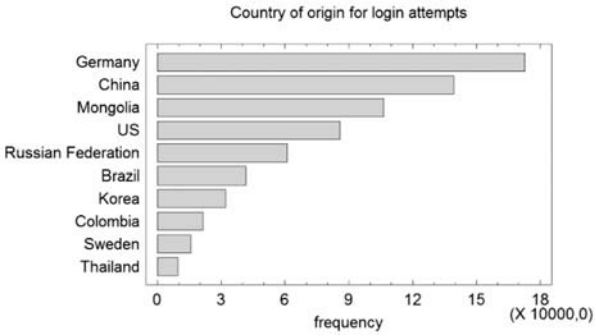

Fig. 4 Country of origin for logging attempts

of connection attempts were made from one of the first four countries.

As mentioned above, we have seven different sensors. With nearly a million attempts, each sensor was tested every 23 seconds.

C. Dionaea data analysis

Dionaea was monitoring malicious traffic for 18 days. The number of attacks is not as high as in Kippo case but still high enough. Fig. 5 shows the distribution of attacking IP addresses.

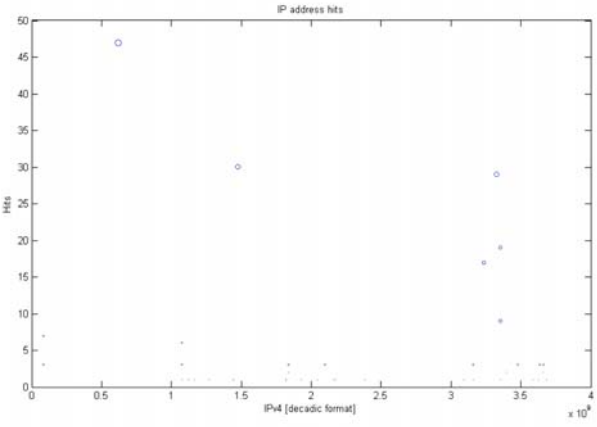

Fig. 5 Hits on attacker IP address

Most attacking attempts come from Israel (first peak in Fig. 5) with IP address 37.8.54.135. The second most used IP was originated in Germany and third in Russia.

Dionaea provides also additional information about attacks like used SIP messages, SIP header information, SDP and RTP statistics. Interesting is typical attack behaviour based on sent SIP messages. All attacks occur in typical sequences. Table 2 shows gathered SIP message data. In column groups is the number of SIP message's grouped by different connections. One connection is a single session with emulated honeypot's SIP proxy. Ratio simply

illustrates the average number of messages in each connection group.

SIP message type analysis Table 2

SIP message	Groups	Count	Ratio
ACK	40	303	7,575
BYE	4	4	1,000
CANCEL	1	11	11
INVITE	18	85	4,722
OPTIONS	76	76	1,000
REGISTER	28	1745	62,321

Most of attacks can be divided into 2 groups. First represents various types of a PBX scanning & probing. Attacker send OPTION message and wait for an answer or simply try to place a call with immediate cancelation (it means INVITE message followed by CANCEL message).

Other group represents flood attacks. Our previous tests confirmed an extraordinary vulnerability of SIP proxy against OPTIONS floods, but attackers still use only REGISTER message flooding.

At last there are few attacks which cannot be simply placed in groups or generally categorized [11, 12].

5. Conclusion

All the tests which we carried out on a VoIP honeypot gave us a solid look on its features. The main goal of this article is to test the honeypot functionality and analyse gathered data. Another goal is to consider its deploying in our real IP telephony infrastructure.

The test revealed that Artemisa is not the silver bullet solution for discovering all security threats. Its main disadvantage is that it does not recognize a scanning attempt into the infrastructure. There is also a performance issue while analysing a flooding attack. It was a result of the flooding at higher rates. Accordingly, Artemisa cannot handle such a big mass of SIP messages.

But the biggest disappointment was the behaviour of *Artemisa* in the aggressive mode. It is a good idea to implement some mechanisms that can proactively block malicious activity but in case of this honeypot it was done weakly. The passive mode can only be used for testing purposes. We have not seen any option for using it in a real network. The active mode is the only applicable mode, and it should be a default setting for our honeypot.

On the other hand, Artemisa does exactly what we want it to do. An Artemisa honeypot is not a simulation of PBX, but rather of an endpoint device. Despite the fact that it freezes at high flooding rates, its principal utility is to detect suspicious call activity and spit attacks. This is by no means an easy task for many other tools.

SSH honeypot Kippo gives us a good idea about connection attempts on a standard SSH port. The information acquired was used to gain a better understanding of the attacker's behaviour. It is also a good source of malicious IP addresses. Another gathered information contains combinations of username and password used for connection (by attacker), command line activity history or samples of downloaded malicious software. After some time, the rate of connection attempts decreases. This happens once the honeypot has been discovered. It is necessary to change the honeypot configuration each time it is used.

Dionaea honeypot is great in simulating a SIP proxy. The data gathered show real attacks and gave us valuable feedback for improving existing security mechanisms. Quite surprising were flooding attacks using only REGISTER messages. We found no connection between IP addresses used for attacks on both kippo and dionaea.

Our previous research focused on creating an open-source IPS to protect VoIP PBX. Using honeypots to gather information about hacker's actions was extremely helpful. Now we orient our further research on creation of a honeypot network which should gather information on various locations. All this information would be stored in one datastore for the following analysis and other improvement of security mechanisms.

Acknowledgement

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 218086.

References

[1] SPITZNER, L.: *Honeypots: Tracking Hackers,* Addison-Wesley Professional, 2002.
[2] KLIMO, M., KOVACIKOVA, M., SEGEC, P.: Selected Issues of IP Telephony, *Communications - Scientific Letters of the University of Zilina,* vol. 6, No. 4, pp. 63–70, 2004.
[3] DUHA, J., DADO, M., JARINA, R.: Communication Technologies and Services, *Communications - Scientific Letters of the University of Zilina,* vol. 5, No. 3, pp. 33–35, 2003.
[4] SISALEM, D., KUTHAN, J., ELHERT, T. S., FRAUNHOFER, F.: *Denial of Service Attacks Targeting SIP VoIP Infrastructure: Attack Scenarios and Prevention Mechanisms.* IEEE Network, 2006.
[5] PROVOS, N., HOLZ, T.: *Virtual Honeypots,* Addison-Wesley Professional, 2007.

[6] REZAC, F., VOZNAK, M., TOMALA, K., ROZHON, J., VYCHODIL, J.: Security Analysis System to Detect Threats on a SIP VoIP Infrastructure Elements, *Advances in Electrical and Electronic Engineering,* vol. 9, No. 5, pp. 225–23, 2011.
[7] VOZNAK, M., REZAC, F.: Web-based IP Telephony Penetration System Evaluating Level of Protection from Attacks and Threats, *WSEAS Transactions on Communications,* vol. 10, No. 2, pp. 66–76, February 2011.
[8] JOSHI, R. C., SARDANA, A.: *Honeypots: A New Paradigm to Information Security,* Science Publishers, 2011.
[9] ENDLER, D., COLLIER, M.: *Hacking Exposed VoIP,* McGraw-Hill Osborne Media, 2009.
[10] SAFARIK, J., VOZNAK, M., REZAC, F., MACURA, L.: *Malicious Traffic Monitoring and its Evaluation in VoIP Infrastructure,* 35th Intern. Conference on Telecommunications and Signal Processing, TSP, Prague, pp. 259-262, 2012.
[11] VOZNAK, M., SAFARIK, J.: DoS Attacks Targeting SIP Server and Improvements of Robustness, *Intern. J. of Mathematics and Computers in Simulation,* vol. 6, No. 1, pp. 177–184, 2012.
[12] VOZNAK, M., REZAC, F.: Threats to Voice over IP Communications Systems, *WSEAS Transactions on Computers,* vol. 9, No. 11, pp. 1348–1358, November 2010.