Threat Prevention and Intrusion Detection in VoIP Infrastructures

Miroslav Voznak, Jakub Safarik, Filip Rezac

***Abstract*—**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 nowadays VoIP infrastructure, from emulating an end point device through SIP proxy to SSH terminal emulation. All these application called honeypots bring valuable data about hacker's activity with no threat to the running system. Grouping single honeypots in one cloud solution enables to gather more data on hacker activities and to provide results with higher information value. Analysis of a honeypot data is crucial for further improvement of existing security mechanisms in VoIP networks. The paper describes each honeypot used, brings an analysis of observed malicious activity and a design of the honeypot cloud concept.

***Keywords*—**Dionaea, Honeypot network, Kippo, VoIP attacks, VoIP honeypot

I. 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 (Spam over Internet Telephony) calls.

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? There is an option in security audit of whole VoIP network or using some penetration testing software [2]. Other way is in monitoring of malicious traffic with honeypots.

M. Voznak is an associate professor with Dpt. of Telecommunications, Technical University of Ostrava and he is also a researcher with Dpt. of Multimedia in CESNET (association of Czech universities and Czech Academy of Sciences), Zikova 4, 160 00 Prague 6, Czech Republic (corresponding author provides phone: +420- 603565965; e-mail: voznak@ieee.org).

J. Safarik is a PhD. student with Dpt. of Telecommunications, Technical University of Ostrava and he is also a researcher with Dpt. of Multimedia in CESNET, Czech Republic (e-mail: kuba.safarik@gmail.com).

F. Rezac is an assisstant professor with Dpt. of Telecommunications, Technical University of Ostrava and he is also a researcher with Dpt. of Multimedia in CESNET, Czech Republic (e-mail: filip@cesnet.cz).

Honeypots lure hackers through exposed security holes or vulnerabilities while emulating target services. Using honeypots we obtain real data about hacker activities and map actual attacks in the network, which is otherwise not possible [1], [3] and [4].

II. State of The Art

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 [5]. This article provides a close look on individual honeypot applications and its integration in honeypot cloud solution.

Most important parts of honeypot image are honeypot application, which log and monitor malicious activity. Nowadays exists many honeypot solutions emulating both single and multiple services. Because of our focus on IP telephony we decide to deploy VoIP oriented honeypots. Each honeypot emulates different aspects of VoIP network, its features are described below.

A. Artemisa 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 represented by the VoIP Honeypot application installed on PC, the situation is depicted on Fig. 1. The application connects to SIP proxy with the extensions defined in 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. 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, the situation is depicted on 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).

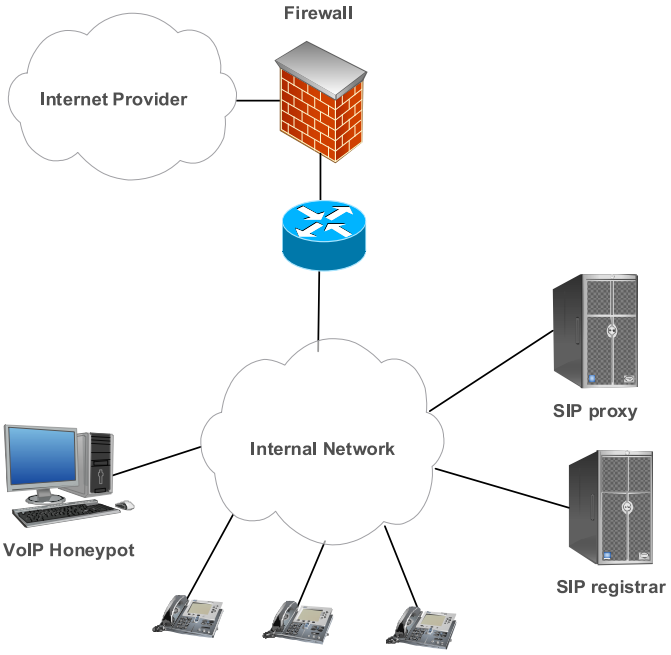

Fig. 1 A VoIP topology with a honeypot

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.

```
... 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

Once the call has been examined, series of bash scripts are executed. These scripts are executed with pre-defined arguments. Artemisa can launch some countermeasures against the incoming attacks.

B. Kippo features

The second used 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 I 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.

C. 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 are information from these multiple services only general but dionaea serves only 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 as a contrary to Kippo can also emulate code from these files.

Describing features of all this 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.

III. Designed Honeypot Network Concept

Running individual honeypot application on a single server brings valuable information. Exceeding numbers of running honeypots, especially if running in different networks on various geographic locations, causes unwanted overhead in data analysis. Without an automatic aggregation mechanism, this situation leads to decreasing profit from honeypot's data.

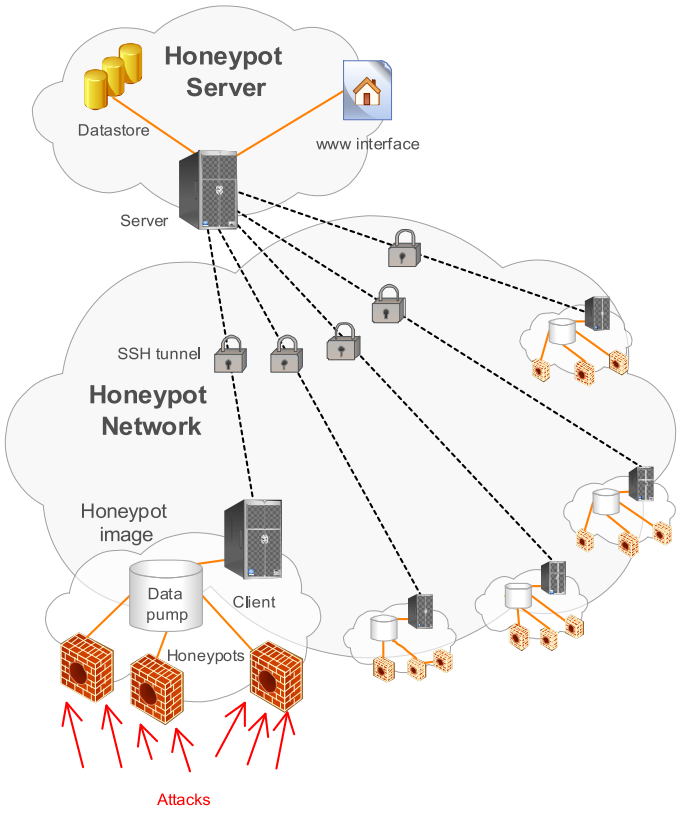

Fig. 3 Honeypot network concept

Fig. 3 illustrates a honeypot cloud concept with centralized point for data analysis. Whole concept is based on prepared honeypot image. The image can run on virtual or physical machine, and there is no need in installing software or configuration of the server. This image contains all software needed for correct honeypot functionality, data pump and client application for communication with server. Deployment of this honeypot image should be as easy as possible.

Data from all honeypot images pass through data pump and cleaning function. Clients send the prepared data to server application via encrypted SSH tunnels as is depicted on Fig. 3. Centralized server provides a data store for all honeypots data and esnures a monitoring all running honeypot images. Server side application transforms and integrates data from client to the data store. Results are available via web interface for further analysis and the security improvements. This architecture provides unified honeypots images with easy scalability for other honeypot modules. We focused our intent only on VoIP infrastructure but honeypot images can contain various honeypots or different honeypots in each image connected to the only one server. This solution gives us flexible platform for independent observation and evaluation of real threats.

IV. Honeypot Usability Tests

A. *Artemisa's usability tests*

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 is the active mode so that all SIP messages routed to the honeypot are analysed.

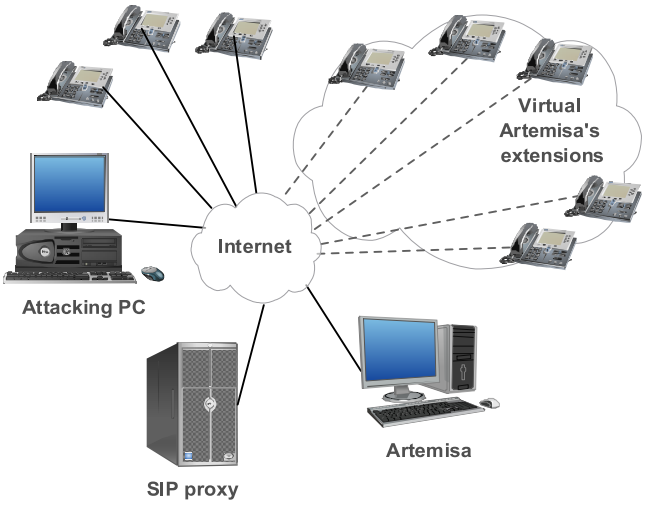

Fig. 4 Artemisa testing topology

To test the usefulness of an *Artemisa* honeypot, we prepared some tests in our testing topology as is depicted on Fig. 4. 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 (Intrusion prevention system), we have chosen to test the honeypot under test scenarios similar to that IPS system.

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*.

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 of such situations. With *svmap* we know about the running user-agent at honeypot's IP address as is listed below.

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

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 has been 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 [7], [9]. For this kind of attack we used a number of tools including *udpflood*, *rtpflood*, *inviteflood* and *sipp* [7], [10].

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 [6]. 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 application called SPITFILE for simulating these calls [8]. SPITFILE is an open-source SIP penetration tool which we developed for purposes of research and development of new protection alogorithms against spam in IP telephony. SPITFFILE puts much emphasis on the simplicity of using and generating SPIT attacks. SPITFILE was programmed in Python using wxPython GUI and the objective of the designed application is to generate phone calls and to replay a pre-recorded voice message. We adopted the SIPp application which focuses on testing and simulating SIP calls in VoIP infrastructure. SIPp is a open-source test tool or traffic generator for the SIP protocol and can read custom XML scenario files describing from very simple to complex call flows and also send media traffic through RTP [11], [12]. SPITFILE implements a graphic interface for SIPp and works with ready-made .xml diagrams. Thus, the simulation of a SPIT attack is much simpler.

Its control is very intuitive – the requested values are submitted into relevant fields and the SPIT attack is launched by clicking the SEND button. SPITFILE is available both for Linux and for MS Windows. SPITFILE can generate spam in two modes.

- Direct mode, it generates SPIT on IP phone directly without using a SIP Proxy.
- Proxy mode, it generates SPIT via SIP Proxy and thereupon can run against anything that is available behind the Proxy, theoretically involving not only IP phones but also ordinary phones and the whole telephone world. The proxy mode's menu is depicted on Fig. 5.

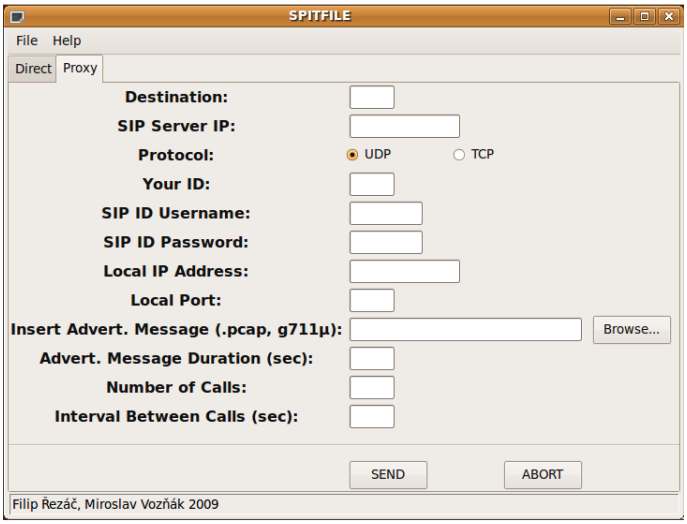

Fig. 5 SPITFILE interface in proxy mode

Before SPITFILE can be opened, preconfigured .xml diagrams should be imported into /etc/ directory. Afterwards we can launch SPITFILE and choose one of the two above mentioned attacks that we want to carry out. To run SPITFILE, just type the following command to the terminal: python <location of the SPITFILE.py file>.

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 a 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.

TABLE I
10 MOST FREQUENTLY USED PASSWORD COMBINATIONS

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. The in-depth analysis of the attacker's input is beyond the scope of this article.

Another interesting information acquired from the honeypot is the approximate location of the origin of the attacker's connection as is depicted on Fig. 6.

The Fig. 6 shows only the 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% of connection attempts were made from one of the first four countries.

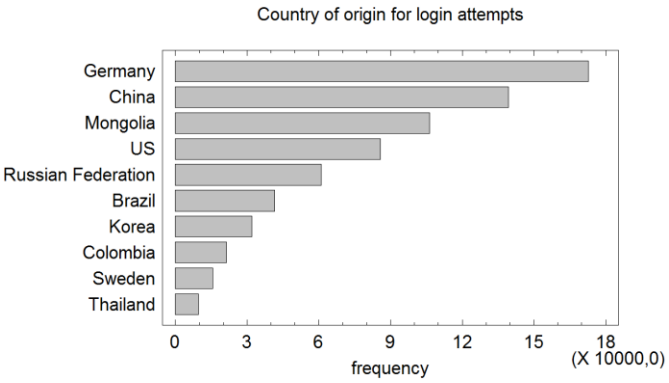

Fig. 6 Country of origin for logging attempts

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. Number of attack is not as high as in Kippo case but still high enough. Distribution of an attacking IP addresses is depicted on Fig. 7.

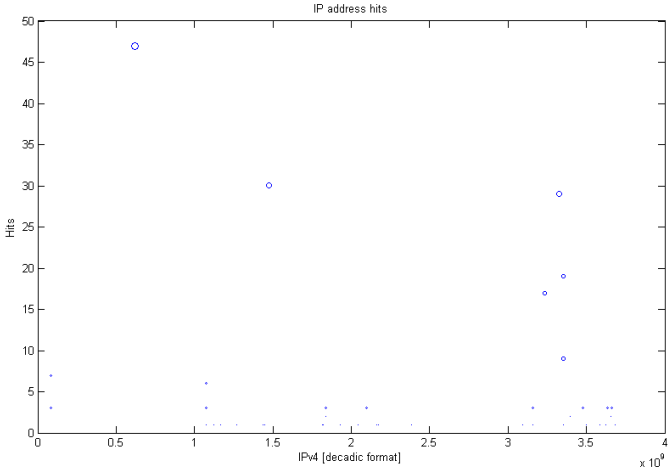

Figure 7: Hits on attacker IP address

Most attacking attempts come from Israel, first peak in Fig. 7 with IP address 37.8.54.135. The second most used IP was originated in Germany and third in Russia.

TABLE II
SIP MESSAGES TYPES ANALYSIS

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

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 send SIP messages. All attacks occur in typical sequences. Table II

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 average number of messages in each connection group.

Most of observed attacks can be divided into 2 groups. First represents various types of a PBX scanning & probing. Attackers send OPTION message and wait for a response 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 we observed several attacks which could not be simply classified into groups or generally categorized.

V. Client Structure

Client part of our honeypot image consists of several components, the crucial part is a honeypot application. The presented usability tests in previous chapter demonstrate pros and cons of selected applications. In order to prepare efficient tool we decided to include two honeypots in final image, Dionaea and Kippo. Artemisa is not included for several factors. Most limiting is the necessity of running existing PBX. Artemisa itself only emulates an end-point device and without PBX is not able to work. Preparing a special PBX on the image only for Artemisa honeypot is counterproductive. Dionaea comprises also ability to answer INVITE messages therefore the part of call investigation is not affected. Information about attacks from Artemisa are only general and our testbed proved their low usability. Last flaw of Artemisa is a lack of database support. All malicious traffic information are in plain-text formatted files, no connectors to databases exist and we would have to start from scratch. The final structure of client is illustrated in Fig. 8.

Both Dionaea and Kippo support logging information into a database. Each application uses a different type of database and different database scheme. There is a data pump for data mining and information are gathered from both databases. The data pump is based JDBC connector (Java Database Connectivity) and data are also cleaned and checked for errors.

Communication with server handles another part of application which we developed for honeypot image. Whole communication between client and server application is encrypted via SSH tunnelling. This feature also brings a possibility to easily authenticate client nodes with certificates. SSH is a good way for ensuring secure communication without a necessity of developing a new protocol or handling an encryption and an authentication for the particular purpose [13].

Data gathered through data pump are converted into serialized objects. These objects are periodically sent to server for further operations and analysis. The honeypot client includes so-called dead man switch mechanism. With this functionality is possible to monitor honeypot status easily and there is no necessity of developing a specialized monitoring application. Clients send a simple UDP packet to server each 5 minutes, the dispached information are checked on server by timers which are individually configurable for each image.

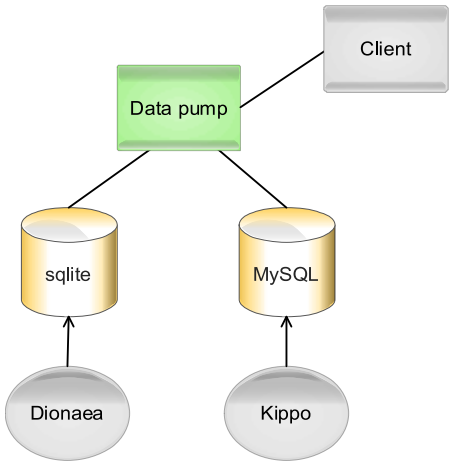

Fig. 8 Honeypot image – client structure

The server identifies a non-active honeypot after three undelivered messages and creates a system warning which is reported in standard log file and eventually an administrator is notified by e-mail as well.

All honeypot images are prepared for final deployment. There is already configured firewall allowing only traffic specific for each honeypot application and internal client communication with server. The image contains also automatic NTP (Network time protocol) checking mechanism for the proper time settings. It is possible to administrate the client images through SSH with proper certificate but there is almost no reason for this action except for initial settings. This connection is active only 10 minutes after image boot and can be reopened from the server application.

VI. Server Structure

Some features of server side applications were already described in the previous section. Server's main task is collection of received data and their analysis. Due to this functionality is whole application implemented as a ROLAP (Relational online analytical processing) and the concept is depicted on Fig. 9.

Otherwise from a client (honeypot image), server is considered as enterprise application and not as an image. Before ROLAP receives data from nodes and stores the data to database, several last steps are performed. Nodes send already cleaned and fixed data structures, so ROLAP can proceed to data transformation and integration according to data store scheme. All data are divided in two groups – on the dimensions (date and time, honeypot, locations, …) and the facts. Data warehouse consists of a star or a constellation design. Internally run data store on the relational database MySQL 5.5 release. Built-in aggregation function ensure the aggregation of all stored data in groups by hours, days, months and years. This function is implemented for higher performance of database. The server also comprises a self-monitoring mechanism for logging important information like

system logins and analysis of actions or failures. System log contains details information about each honeypot node outage.

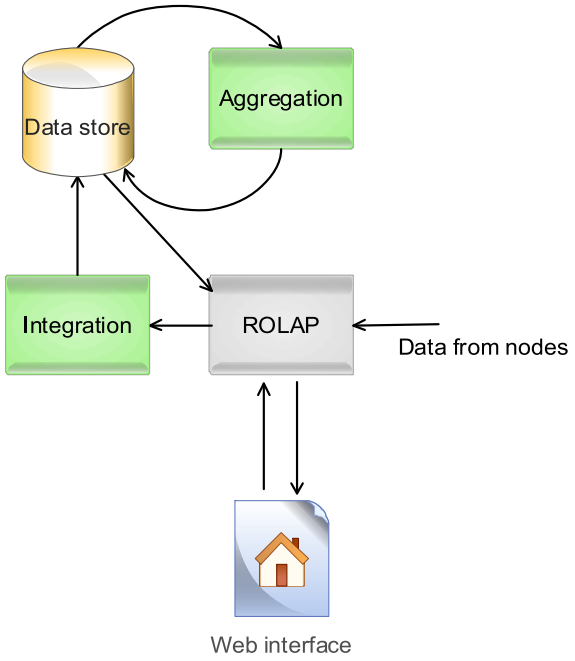

Fig. 9 Server side application structure

Each honeypot node must be activated and authorized before establishing a full connection between server and client. Node authentication is included in SSH tunnel assembly, and there is no other mechanism. After successful authentication and authorization start client periodical notification and data transfers. The gathered data from honeypot node are sent periodically each day. Server operates with data which are only one day old in the worst case. It is possible to change this interval to a minimum period of an hour. System contains a deactivation function for case of removing honeypot.

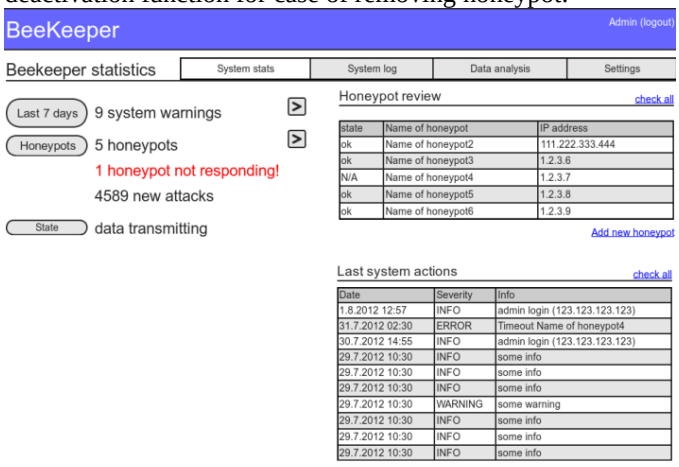

Fig. 10 Web user interface – main screen

Each attack detected on any honeypot is checked for source location. This feature works via IP address localization. System automatically downloads lists of IP address range for all states. Final file runs through parsing function and the system operates with information from RIR (regional Internet registry) databases only one day old. Using IP address localization brings another information value to already collected information.

Whole system is controlled via built-in web user interface running on secured HTTPS protocol. Access to the system is protected with user accounts, in the future is planned implementation of other login options using Shibboleth or OpenID. A wireframe of web user interface, after user login, is shown in following Fig. 10.

The design of the interface is involved as an intuitive for users enabling various data analysis. The data analysis functions are based on user input and illustrated in Fig. 11. The displayed information can be exported in a single CSV (Comma-separated values) format file for even further analysis with an advanced statistical or mathematical software.

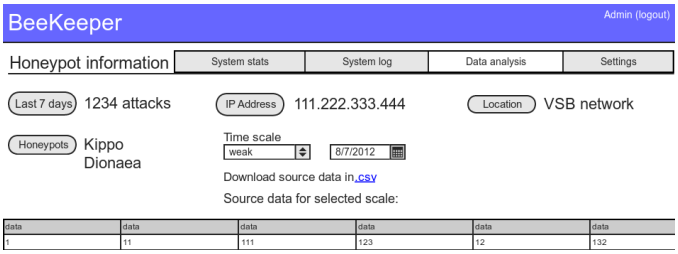

Fig. 11 Data analysis of collected data from honeypots

We would like to import a MATLAB feature in a future version for better and deeper analysis with advanced functions. Last part of server side architecture is a VoIP attack classification engine. The performed experiments with Dionaea proved the existence of several attack groups. These groups are essential in classification of threats. The Most harmful type of observed attacks is DoS on application layer which are based on flooding with SIP messages, these floods can be easily carried out and mostly result in a depletion of sources at victim side. Other group of attacks is scanning, probing, ringing and SPIT calls. The attacks are grouped according to SIP sessions and source IP addresses. Classification is based on types, numbers and ratio of received SIP messages.

VII. Conclusion

All experiments with honeypots which were carried out in our testebed gave us a solid look on its features. The main goal of our work is to design VoIP honeypot cloud and to contribute to the improvement of security in IP telephony. In order to realize our aim we examined functionalities several existing honeypots, after that we analysed gathered data which we could exploit for security improvements, finally we designed our honeypot network concept.

The testbed 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. It is freezing while analysing a flooding attack by the *sipp* application was also quite surprising. It was a result of the flooding at higher rates. Accordingly, Artemisa

can not handle such a big mass of SIP messages.

The default Artemisa mode is the active mode for call inspection, the mode is configurable and we recommend the agressive mode to exploit all possibilities. 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. Similar functionality includes also Dionaea honeypot without typical Artemisa flaws. Reasons for not including Artemisa into final image are discussed in foregoing section.

SSH honeypot Kippo gives us a good idea about connection attempts on a standard SSH port. The information acquired were used to gain a better understanding of the attacker's behaviour, the collected data represent a valuable source of malicious IP addresses. The database containing the used username and password combinations is also very interesting. After some time, the rate of connection attempts decreases, it happens once if the honeypot has been discovered. 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 was flooding attacks using only REGISTER messages. We found no relations between IP addresses used for attacks on both Kippo and Dionaea.

Our previous research was 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 data store for the following analysis and other improvement of security mechanisms. Whole honeypot network concept is based on centralized architecture with modular design, providing a flexible platform for monitoring attacks not only for IP telephony infrastructure. Future possibilities of honeypot data analysis lie in involving a distributed system for communication with IPS systems and immediate reaction on attacks on one system to improving security against detected attacks on other connected systems. This automatic defence mechanism should detect real threats and raise security in systems without malicious activity proactively.

ACKNOWLEDGMENT

This work has been supported by the Ministry of Education of the Czech Republic within the project LM2010005.

REFERENCES

[1] N. Provos, T. Holz, *Virtual honeypots*, Addison-Wesley Professional, 2007.

[2] M. Voznak, F. Rezac, "Web-based IP telephony penetration system evaluating level of protection from attacks and threats," *WSEAS Transactions on Communications*, Volume 10, Issue 2, February 2011, pp. 66-76.

[3] N. Quynh, Y. Takefuji, "A Novel Stealthy Data Capture Tool for Honeypot System," *WSEAS TRANSACTIONS on COMPUTERS*, Issue 1, Volume 5, 2006, pp 209-215.

[4] R.C. Joshi, A. Sardana , *Honeypots: A New Paradigm to Information Security*, Science Publishers, 2011.

[5] L. Spitzner, *Honeypots: Tracking Hackers*, Addison-Wesley Proffesional, 2002.

[6] M. Voznak, F. Rezac, "Threats to voice over IP communications systems," *WSEAS Transactions on Computers*, Volume 9, Issue 11, November 2010, pp. 1348-1358.

[7] D. Endler, M. Collier, *Hacking Exposed VoIP*, McGraw-Hill Osborne Media, 2009.

[8] M. Voznak, F. Rezac, "VoIP SPAM and a defence against this type of threat," in Proc. *14th WSEAS International Conference on Communications*, 2010, pp. 172-177.

[9] M. Voznak, J. Safarik, "DoS attacks targeting SIP server and improvements of robustness," *International Journal of Mathematics and Computers in Simulation*, Volume 6, Issue 1, 2012, pp. 177-184.

[10] D. Sisalem, J. Kuthan, T.S. Elhert, F. Fraunhofer, "Denial of Service Attacks Targeting SIP VoIP Infrastructure: Attack Scenarios and Prevention Mechanisms." *IEEE Network*, 2006.

[11] M. Voznak, J. Rozhon,"SIP infrastructure performance testing," in Proc. *9th WSEAS International Conference on Telecommunications and Informatics*, 2010, pp. 153-158.

[12] M. Voznak, J. Rozhon,"Methodology for SIP infrastructure performance testing," *WSEAS Transactions on Computers*, Volume 9, Issue 9, September 2010, pp. 1012-1021.

[13] W. Yang, R. Jan,"Requirements for security protocols," *WSEAS Transactions on Computers*, Volume 5, Issue 7, July 2006, pp. 1576-1581.

[14] R. Chochelinski, I. Baronak, "Private Telecommunication Network Based on NGN " in Proc. *32nd International Conference on Telecommunications and Signal Processing*, Dunakiliti, 2009, pp. 162-167.

Miroslav Voznak is an Associate Professor with Dpt. of Telecommunications, Technical University of Ostrava. He is also a researcher with Dpt. of Multimedia in CESNET (association of Czech universities and Czech Academy of Sciences). He received his M.S. and Ph.D. degrees in telecommunications, dissertation thesis "Voice traffic optimization with regard to speech quality in network with VoIP technology" from the Technical University of Ostrava, in 1995 and 2002, respectively. Topics of his research interests are Next Generation Networks, IP telephony, speech quality and network security. He was involved in several FP EU projects.

Filip Rezac was born in 1985 and his M.S. received from Technical University of Ostrava in 2009. Since 2009, he has continued in PhD. study in Dpt. of Telecommunications and he gained position of assistant professor with the same Dpt. in 2010. His research interests are focused on Voice over IP technology, Network Security and Speech Quality. Since 2008, he has become a researcher with Dpt. of Multimedia in CESNET.

Jakub Safarik received his M.S. degree in telecommunications from Technical University of Ostrava, Czech Republic, in 2011 and he continues in studying Ph.D. degree at the same university. His research is focused on IP telephony, computer networks and network security. He has been involved in the research activities of CESNET association since 2011.