
**Esteqlal Institute of Higher Education**

**Faculty of Computer Science**

**Research Title: password Authentication protocol**

---

**Author: Mohammad Milad Hassani**

---

**Year: Spring 2024**

# ABSTRACT

In this research various aspects of a password based authentication are described. First, methods of password storage are compared and evaluated from a security standpoint. In the next chapter, attacks on password hashes are evaluated and different methods of protection suggested. Then attacks on login mechanisms and diverse defense strategies are explored. Last part of the research describes a novel security testing tool that allows building massive Brute forcing networks of cooperating users.

But now we are exposed to many security threats: denial of service (DoS), scanning, password cracking, spoofing, eavesdropping, spamming, phishing, worms and others. As a result, many companies and organizations define their network security policy. It is a set of rules that should be followed by users to avoid or at least mitigate the security threats. Technically, the policy is often implemented by firewalls, intrusion detection and prevention systems (IDS, IPS) or a virtual private network (VPN). The firewall represents basic level of a defense. It inspects network traffic passing through it and denies or permits the passage based on a set of rules, a part of the network security policy. An intrusion detection and/or prevention should be performed to fulfill two basic requirements: to identify and/or protect host computer from security threats in the administered network connected to the Internet or other networks and vice versa. We point out that both requirements are important.

The network is exposed to attacks from outside as well as from inside. In addition, the second requirement is important due to the presence of botnets that exploit “zombie computers” in our network and use them to other malicious activities. In short, IDS and IPS are the “checkpoints” that supervise firewalls or other components dedicated to the network defense.

# Table of Contents

# LIST OF FIGURES

## Problem

This research analyses the performance and aspects of password authentication protocol, more specifically, show method for password security, vulnerabilities, performance factors and solutions for improving of securing of password with this method.

## Research Goals

The main goal of this research is to find out the different protocols of password authentication. Investigate methods that can improve security, compare between the security aspects, performances factors in different protocols that are used to implement and to find out which one is the most reliable and efficient in different scales of our system.

## Research Questions

This research is aimed to answer the following questions:

- What are the methods of password authentication?
- What is the performance of PAP protocol?

## Research Methodology

The methodology of this research is divided into a literature study part and a practical part. In the literature study part, the different security methods of the PAP methods are investigated, to find out security solutions, vulnerabilities and attacks in different PAP methods, compare between the functionality of these protocols and find out the performance elements of PAP in different PAP methods to improve the performance. In the practical part a PAP protocol is simulated to measure the performance and find out some of the security vulnerabilities and the solutions to make our system more secure.

## Research Structure

The rest of this research work is organized as follows: Chapter two explains about different method of password security and other technologies that is used in the storage area and make comparison between them, explain different protocols that are already used for implementing PAP and make a comparison between them about the working structure. Chapter three focuses more on the visualization and discusses the security risks, threats and vulnerabilities in PAP and different types of attacks in each of the protocols that already PAP implements on them, such as Chap and verifies the defense method for each one of them used to increase the level of security. Chapter four discusses about design of IDE and makes comparison between the performance elements of the PAP protocols

and the effects of some security issues on the performance of the system. Chapter five contributes with a set of conclusions from the research work and show the implementation of IDE.

# CHAPTER I
# INTRODUCTION

Ever since computers have started to be widely adopted and carriers for data requiring restricted access, the password based authentication has taken place as a most commonly used authentication method. Despite the long time that has passed since then, the password based authentication has not been dethroned by any other method. Even though these methods often offer better security, they usually come with a price that is too high to spur moving away from passwords. If this can be taken as a precedent, then it can be expected that passwords will account for largest part of authentication even in the near future. For this reason, it is worthwhile to explore strengths and weaknesses of the password authentication and strive for further improvements.

## 1.1 Comparison with other authentication schemes

There are many different methods of a user authentication, but there exist only three widely acknowledged categories in which every method should belong.1 These categories are based on the relationship between a user and an object that the user authenticates himself or herself with:

- **Something the user knows** Typical example of such category is the password based authentication, but it can be any question-answer scheme.
- **Something the user has** To this category falls every method requiring some kind of authentication token. Be it a smartcard or a simple user ID (like passport).

- **Something the user is** Mostly biometrics, but in some sense it can be for example the CAPTCHA (user proves being human, not certain human).

Methods from different categories can be combined together to create a multi-factor authentication that is considered much stronger than using method(s) only from one category. Also such combination is likely to add together advantages of said categories while reducing their disadvantages.

| | **Advantages** | **disadvantages** |
|---|---|---|
| Knows | easy to manage, cheap | easy to leak, lost is hard to notice |
| His | hard to copy, lost is spotted quickly | can be damaged, slow replacement if lost or damaged |
| Is | always available, hard to steal | Irreplaceable when compromised, false acceptance or false rejection |

Table 1: Authentication categories comparison

As can be seen from the table, the password based authentication is used and valued because of being easy and cheap. The user is not bothered with additional equipment or an uncomfortable procedure when gathering biometric information and administrators need very little equipment to both store and query the password.

But these features come with a price - users like to have weak passwords and tend to lose them. Unlike in other categories, such loss is recognized only from secondary effects (e.g. company rival knowing trade secrets) which can have devastating consequences.

## 1.2 Password storing

A password loss has to be prevented at two endpoints. The first one is the user. Probably no sensible technical measure can stop him or her from writing down the password on a little piece of paper titled ’my password’ and then leave it somewhere for adversary to grab. The only thing that can be done here is to force the user not to act like this.

The other endpoint however has much better prospects. Databases of user passwords can be shielded with layers of security measures to stop the attacker. These measures range from active defense elements like firewalls and IPSs to passive measures that serve as a last resort in case the attacker has breached in. One of such passive measures lies in a way how passwords are stored.

### 1.2.1 Plaintext passwords

Having a password on a disk in a plaintext form indicates more of a lack of security measures than the opposite, nevertheless, it is still a valid way to store passwords and as such is worth mentioning. The security flaw is obvious - when the attacker gets in, all passwords are waiting to be abused. But what could be the motivation to use such a weak measure? First, for a creator of the system it is the easiest possible way to get his job done. Second, a user who has forgotten a password can get it from an administrator without the need to change it. Needless to say that such arguments are not even remotely an excuse, and storing passwords in a plaintext should be avoided at all costs.

### 1.2.2 Encrypted passwords

Much better way to store a password is to cryptographically protect it. That way any attacker who manages to gain access to passwords is confronted with another, mostly computationally intensive task. Often this effort has to be spent on each

password separately leaving the others secure. Of many various ways of cryptographic protection of passwords one method is to employ block ciphers. This method is used for example in DES version of UNIX crypt (3) library function. To produce an output that is going to be stored instead of a password, following sequence of steps is executed:

- the password is truncated to eight characters
- each character is coerced to 7 bits, thus constituting a 56-bit DES key
- this key is then used to encrypt one block of all zeroes

- 
  the ciphertext is repeatedly encrypted 25 times with 12 bits of salt
- a result is stored as a base64 encoded string

These steps are repeated every time the password is supplied, and resulting strings are compared. It needs only a little modification to work with any other cipher. Passwords stored in this form are fairly resistant to attacks but do not enable lost password recovery for forgetful users. But this is just a minor inconvenience.

One specific method, that is somewhere in the middle between passwords in plaintext and passwords encrypted as described before, is encryption with a secret key. This approach has an advantage of allowing password recovery. Although it may appear to be as secure as is the strength of an encryption function, there is more to consider. Matter of a secrecy of the password was just shifted to matter of secrecy of that key. Once the key is leaked all passwords are vulnerable. Therefore, this method should be considered insecure.

### 1.2.3 Hashed passwords

The crypt (3) example in the previous section was not an encryption per se, because the password was used as a key, but the encrypted text was not intended to be hidden. It can be viewed as an example of a hash function constructed in an uncommon way, because it satisfies definition presented by Pernell One way hash function is a function h satisfying the following conditions:

- The description of h must be publicly known and should not require any secret information for its operation
- The argument X can be of arbitrary length and the result h(X) has a fixed length of n bits

- Given h and X, the computation of h(X) must be "easy"
  The hash function must be one-way in the sense that given a Y in the image of h, it is "hard" to find a message X such that h(X) = Y and given X and h(X) it is "hard" to find a message X1 different form X such that h(X1) = h(X)

Hash functions are probably the most prevalent method of a password transformation before storing. They offer a reasonably strong protection (especially when using functions producing hashes longer than 128 bits), are relatively easy to compute and protect even against insider attacks, which previous methods failed on.

During last few years there have been many researches on attacks on various hash functions, especially MD5 and SHA [1,2,6,7]. They have succeeded in discovering time-effective algorithms for finding collisions, but they are generally not a threat for password storage. Collision means that few bytes of plaintext are altered yet produce the same hash. And because the plaintext (password) is not known the collision itself is not going to help.

### 1.3 Password cracking

A security of a password is not equal to a security of a hash function that was used to hash it. On the contrary - no matter how strong the function is, a wrongly chosen password can render all security measures useless because password itself could be susceptible to various kinds of attacks.

Choosing the "right" password means balancing two interests - having a password that attacker will not be able to guess and having a password that user is going to remember without the need to write it down and eventually leave it in an

insecure place. This is by no means trivial task and following text aims to create guidelines for it with respect to various attacks that adversary may use.

### 1.4 Dictionary attacks

This category of attacks is by far the most popular for two reasons:

- **It is computationally least intensive.** To put things in perspective: a large wordlist with words 8 letters long can be between 100 - 250 MB in size. That is some 120 – 300 thousand words. Compared to about 209 trillion of different 8 character words that exist in lower-case alphabetical space it is a tiny fraction that an attacker has to try in comparison to simple brute force attack.
- **It is working in many cases.** Like many attacks based on a social engineering, it is working because of how humans tend to act. A user (especially not security educated), when given a choice to choose his own password, tends to stress only the memory aspect, i.e. "will I remember it in two days?". This stimulates using words found in his or her vocabulary with at most some cosmetic changes.

The dictionary attack is, however, as powerful as a dictionary itself. While there are many places where an adversary can get a dictionary (language corpus, etc.) such sources do not provide material "good enough" for most password cracking attempts because of aforementioned cosmetic changes. These changes constitute for example in appending a digit after a word chosen for password, which effectively nullifies the value of such dictionary in case of simple matching attack. Needless to say that over the years there emerged techniques of reinforcing such dictionaries to cover

changes ranging from a simple appending to various permutations. These techniques have to be taken into account.

### 1.4.1 Derivation of password from username

As was mentioned before, users tend to create passwords that are easily remembered. One way to reach such goal is to have a password that is somehow similar to a username which then serves as a memory hook for a forgetful mind. This is a very double-edged method as users are generally less creative than adversaries

wanting to steal their credentials. Here is a short list of derivations that are usually taken into account when adversary uses some password cracking program.

- Username duplication. User ’John’ with password ’John John’.
- Password as reversed username.
- For case sensitive passwords, converting username to upper/lowercase.
- Using N letter substring of username.
- Appending numbers and special characters to username.
- Prefixing username with numbers and special characters.
- Removing vowels from username.

### 1.4.2 Dictionary enhancing

This method is virtually the same as the previous one, with only few differences. First – there are much more words to apply a transformation to. Second - because of that not so many transformations are feasible on a current hardware. This method can grow a wordlist thousand fold and that is the reason why some transformations (like selective upper/lowercasing) have to be discarded.

### 1.4.3 Purpose-built dictionaries

Sometimes an overly strict password creation policy can itself because of a failing security. Suppose that there is an administrator responsible for a password policy who decides that all passwords should have the same form. All eight characters long, consisting of lowercase letters and numbers. And let’s say that this administrator knows about previously mentioned attacks. He or she decides that good countermeasure would be a more precise specification of password structure that would discourage ordinary users from choosing simple passwords.

Let’s say that this form has been chosen: NxxxNNNx, where N is number and x is lowercase letter. Such form is probably going to guard well against most of

transformations except the leetspeak conversion. However, once an adversary gets to know this pattern (and it has to be expected that adversary will find the pattern once) he or she can build a specific dictionary that will target this specific form. In this case the needed dictionary would have 264 _ 104 different words. That is a dictionary around 38 GB large, containing passwords for everyone abiding to this password policy. As the benchmarks of JTR have shown, it would take less than a day to crack every single password (DES) on a single computer. Not exactly secure...

### 1.5 Brute force attacks

Brute forcing a password is mostly a last-resort approach for any adversary. The nature of this attack dictates that all possible password combinations have to be tried until a password is found. Searching through an entire password space is very time consuming and the time needed to find a password varies greatly. For example, if an adversary was to commence simple brute forcing attack on an eight-character password starting from ’aaaaaaaa’ and ending at ’zzzzzzzz’ he would find ’clueless’ almost eight times faster than ’wiseguys’. If there were no other and more sophisticated ways of bruteforcing there would be only passwords with characters having high ordinal value. That is obviously not the case and following text will present some methods of speeding the brute forcing and increasing the chance of finding most of passwords in shortest time.

With regards to previously discussed methods - when an adversary decides to use brute forcing it can be expected that the password has at least one of the following characteristics:

- it is more than seven characters’ long
- it is not a variation of a username
- it is not derivable from a wordlist

In the first case, brute forcing is probably not going to help him that much, because the password space grows geometrically with a password length. Other cases are much more interesting

- a password with such characteristics is probably product of:
- a security educated user
- a random password generator

These two do not seem to have much in common, but as will be shown later in the text these two can behave similar enough to treat them as a one. Random password generators can be divided into number of categories, but in this text only two will be used, based on whether some conditions apply to generated passwords. The first category are simple generators. They are easy to program (password length times choose a letter from the character space) and are designed to exhaust the entire password space. Produced passwords are completely random (assuming that a good random function is used) and are a biggest obstacle to an adversary, because there is no way to reduce the password space and speed the brute forcing up. Although they may seem like an ideal source of passwords, they are not. And the reason is simple - users do not and often cannot remember them, which largely undermines this method. According to author's experience, most of services that used random generators did not produce passwords longer than seven characters. They have probably done it so that users could remember them, but in the same time they have opened a hole into system by having too small password space.

The second category consist of more sophisticated generators, that are trying to overcome the disadvantage of simple generators while remaining secure and random enough. Generally, it is done by using character permutations that are easy to remember. For example, common di- and trigrams present in a language that usually

do not solely consist of either vowels or consonants (which can easily happen with simple generators). Presence of character groups also determines a position of special characters (placed in between) that are inserted into a password for a bigger security. As an outcome, passwords can be said out loud by users without twisting their tongue, which adds to easier remembering. This is a quality sought probably by every user - therefore it can be expected that sophisticated generators and educated users are going to produce relatively similar passwords.

## 1.6 Rainbow tables

Let's imagine a skilled adversary - he has probably brute forced passwords many times. And with a limited number of hashing functions it is for sure that during his attempts he has created and used some hashes more than once. It is in his own interest to store already computed hashes with a corresponding password to use them later without the need for re computation. Luckily for him (and sadly from a security standpoint) there is a way to do it. However, it does not work exactly in such fashion that cracking attempts are recorded for future use. Instead adversary precomputes as much hashes as possible (preferably to fill entire hash space) and then only looks for matches. A method to do this effectively was devised by Hellman and improved by Rivest, Matsumoto, Kim, Kasuda and Oechslin.

## 1.7 Safeguarding passwords

The previous text has shown methods that are used for an effective password cracking. The rest of this chapter will present methods that can narrow options that attacker has and will suggest appropriate ways to construct safe passwords.

### 1.7.1 Salting

Should an adversary get hold of a hashed password list (e.g. /etc/shadow) it is necessary to make cracking of them as much difficult as possible. At best to disable

methods that cracks a password for sure in a short time (i.e. rainbow tables and its variants). One way to do so is a simple principle called salting.

Salt is a short (12 - 48 bits usually) random piece of data, that is concatenated with password before hashing takes place. It is then stored with the password as a public information. While it does not increase a password strength when adversary is trying to crack it with ordinary methods (dictionary, etc...), it effectively nullifies the power of rainbow tables, because a table precomputation must be done for each possible salt. 16 bits of salt then means that for cracking password of eight characters an adversary would need table for 10 characters. That would take a lot of time to precompute (salt is random, so the rainbow table must be built for all 224 ascii characters) and would be truly huge. The salting, when done wrong, can backfire though and these two examples from real life should show that the need for random salt is very reasonable.

_ In the first case, programmers obviously did not understand the concept of the salt very much and used one magic number as a salt for each password. Result - it was almost like they have not used the salt at all. For the first time, an adversary would have to re compute a table from scratch, but every other cracking attempt would be carried out in minutes.

_ Second example happened in a company which name should remain secret to save them (or at least their security department) from a shame. It was a company that decided to use random generated passwords, 8 letters long and alphanumeric. On top of such adequately secure scheme they have chosen to use first two letters of generated password as a salt. Thanks to this flash of security insight, they have effectively reduced their password space almost 13 hundred times and served their customer’s accounts to adversaries at silver platter.

# CHAPTER II
# METHODS FOR SECURITY

This chapter provides an introduction to the intrusion detection and modern methods for the network security analysis. We are mainly focused on the methods working at the IP layer. First of all, we explain basic terms related to the intrusion detection and traffic acquisition.

Then we describe and evaluate each method, especially according to the following criteria:

1. Coverage,
2. Effectiveness,
3. Performance,
4. Applicability for different types of data acquisition,
5. Ability of intrusion detection in encrypted traffic.

The first criterion is ability to detect security threats. The coverage is complete if the method detects both known and unknown threats. The second criterion stands for detection accuracy, the rate of false positives produced by the method. The speed of processing network traffic by the method, the third criterion, is crucial for a deployment in high-speed networks.

The fourth criterion determines whether packet capture and/or (sampled) flow-based data are suitable as the input of the evaluated method. Last criterion is more and more important in today's network. A basic classification of methods is taken from [12].

## 2.1 Intrusion Detection

We can divide intrusion detection systems (IDS) into two basic classes according to their position in the network: host-based intrusion detection systems

and network-based intrusion detection systems. Note there are other points of view of the IDS classification.

### 2.1.1 Host-based Intrusion Detection

This type of detection is performed on a host computer in a computer network. Host-based intrusion detection system (HIDS) usually monitors log files (e. g. firewall logs, web server logs and system logs) and the integrity of system files (e. g. the kernel integrity or opened ports).

### 2.1.2 Network-based Intrusion Detection

On the contrary, the network-based approach observes the whole network or its part. All inbound or outbound network traffic is inspected for suspicious patterns. The patterns can be represented as a signature, a string of characters that describes a certain attack. Another different approach is an anomaly-based detection. First, the model of a normal network behavior is created. Then the difference to the model is evaluated. If it is greater than predefined value (threshold), it can point out an attack.

Other network-based intrusion detection system (NIDS) use stateful protocol analysis to detect suspicious, unexpected or invalid sequences of packets in terms of a specific protocol. These methods are discussed in detail in relevant sections in this chapter. NIDS are passive systems: they are “invisible” to other hosts and mainly for the attackers.

In connection to IDS, there are frequently mentioned two following terms: false positive and false negative. The former denotes a false IDS alert: the system classifies benign traffic as malicious. On the contrary, the latter points to the malicious traffic that was not recognized by IDS. Of course, there is a tendency to minimize the numbers of both false positives and negatives. For example, if the IDS produces high false positive rate, it bothers the administrator about a subsequent

manual analysis of these alerts. In addition, there are some techniques, such as squealing, which exploit the vulnerability of IDSs to high false positive rates.

## 2.2 Intrusion Prevention

In comparison to IDS, an intrusion prevention system (IPS) is a reactive system in which ID is tightly coupled with firewall (and should be a part of the communication link). The main task of IPS is to mitigate (stop) the detected attack. IPS can be divided into three classes: host-based, network-based and distributed IPS.

## 2.3 Flow-Based Traffic Acquisition

The classic approach of many IDS or IPS to data collection is to capture all network packets that pass through the system, most frequently in pcap format1. In contrast, many routers and monitoring probes perform a flow-based data collection, typically in NetFlow format.

### 2.3.1 NetFlow and IPFIX

NetFlow was originally developed by Cisco Systems, the world leader in networking solutions. Many Cisco switches and routers are capable of exporting NetFlow records. There are two widely used versions: NetFlow version 5 and 9. The former is Cisco's proprietary format and the latter was standardized as an open protocol by IETF in 2006.

A flow is defined as a unidirectional sequence of packets with some common properties that pass through a network device. These collected flows are exported to an external device, the NetFlow collector. Network flows are highly granular; for example, flow records include details such as IP addresses, packet and byte counts, timestamps, Type of Service (ToS), application ports, input and output interfaces, etc. [34] Thus, the flow-based data collection provides an aggregated view of network traffic.

**IPFIX** The continuation of IETF effort leads to unification of protocols and applications that require flow-based IP traffic measurements. RFC 3917 defines requirements for exporting traffic flow information out of routers, middle boxes (e. g. firewalls, proxies, load balancers, NATs), or traffic measurement probes for further processing by applications located on other devices [33]. Consequently, Cisco's NetFlow version 9 was chosen as the basis of the IP Flow Information Export (IPFIX). There are no fixed properties (5-tuple) such as in NetFlow version 5. The user can flexibly define the properties used for flows distinction.

RFC 5101, published in January 2008, specifies the IPFIX protocol that serves for transmitting IP Traffic Flow information over the network [37]. Next, RFC 5102 defines an information model for the IPFIX protocol. It is used by the IPFIX protocol for encoding measured traffic information and information related to the whole process [38]. Thanks to the IPFIX flexibility, RFC 5103 can introduce the term Biflow, a bidirectional flow, and describe an efficient method for exporting Biflows information using the IPFIX protocol [39]. The bidirectional view of network traffic might be useful for security analysis. The development of IPFIX is not finished. The IPFIX working group is still working on a few Internet drafts that would be published as RFC. The most recent RFC was issued in April 2008. It provides guidelines for the implementation and use of the IPFIX protocol. [40] Packet sampling is performed (especially by routers) to save the NetFlow exporter resources.

We distinguish two basic types of sampling:

• Deterministic – exactly nth of every n packets is sampled,

• Random – each packet is sampled with a probability 1/n.

The constant n is called sampling rate. For example, if it is set to 4 and the device receive 100 packet, 25 packets are analyzed and 75 packets are dropped for the analysis. Only common packet header fields are recorded, not the whole payload.

The flow sampling is another type of aggregation.

Both the active and the inactive timeout values affect a flow creation. The active timeout is applied to long-lasting flows. If the flow has been inactive for the inactive timeout or the end of the flow is detected, flow statistics are exported from the probe to a collector. The collector is a server dedicated to collection, long-term storage and analysis of flow statistics.

### 2.3.2 Other Flow-based Technologies

Proprietary Cisco NetFlow or open IETF standards are not the only one flow-based solutions. Another industry standard was described in RFC 3176. SFlow is a technology for monitoring traffic in data networks containing switches and routers. In particular, it defines the sampling mechanisms implemented in the sFlow Agent for monitoring traffic, the SFlow MIB2 for controlling the sFlow Agent, and the format of sample data used by the sFlow Agent when forwarding data to a central data collector [32]. SFlow is supported by Alcatel-Lucent, D-Link, Hewlett-Packard, Hitachi and NEC. Other leaders in networking also develop their proprietary flow-based solutions: Juniper Networks use Flowy and Huawei Technology their NetStream.

## 2.4 Signature-based Detection

This of the oldest methods for security analysis. We mentioned it here because it is widely used by many commercial and open-source IDSs.is one

**Description** A signature is a pattern that corresponds to a known threat. Signature-based detection is the process of comparing signatures against observed events to identify possible incidents. It is the simplest detection method because it just compares the current unit of activity, such as a packet or a log entry, to a list of

signatures using string comparison operations. [12] In short, the detection works with “local” information.

**Evaluation** This method is very effective at detecting known threats, but largely ineffective at detecting previously unknown threats, threats disguised by the use of evasion techniques, and many variants of known threats. [12] For example, if the intruder use the Unicode representation of the slash character (%c0%af) and the signature contains the slash, signature-based detection is not successful (false negative). [1] Next, we describe an example of the signature. The following string is a simple rule for an open-source signature-based IDS Snort. [42]

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg: "WEB-ATTACKS kill command attempt"; flow:to_server, established; content:"/bin/kill"; nocase; classtype:web-application-attack; sid:1335; rev:5;) If Snort captures and recognizes a TCP packet with source IP address in the external network, any source port, destination address, destination port of HTTP server in the administered network and the payload contains string "/bin/kill", it alerts "WEBATTACKS kill command attempt" according to the rule. The rule contains variables $EXTERNAL_NET, $HTTP_SERVERS and $HTTP_PORTS. They must be set by the administrator, Snort (or any signature-based IDS) does not know particular values. Thus, the correct detection depends on up-to-date configuration.

The rule above gives an example of possible false positive. Consider we provide an email service with web interface on our web servers. If someone sends an e-mail containing the searched string “/bin/kill”, Snort classifies such traffic as malicious. Particular network traffic and rules themselves influence the accuracy of detection. We can observe very low false positive rate on dedicated lines and vice versa. There are many ways to create rules and signatures. We could use any destination port and/or destination host instead of $HTTP_PORTS and/or

$HTTP_SERVERS in our example. Snort and other signature-based IDS allow specify the searched content as a regular expression too. E. g., it is useful when the signature differs only in used protocol (FTP, HTTP or HTTP). “General” rules are easy to manage, but can cause higher false positive rate. Snort was originally designed for small, lightly utilized networks. [42]

The core of the signature-based detection is generally expensive string matching. Every packet and its payload is inspected for searched signatures. Snort usually runs on COST (commercial off-the shelf) hardware and its performance is not satisfactory for this task in multi-gigabit networks.

This gap is fulfilled by hardware accelerators. Traffic Scanner, a hardwareaccelerated IDS based on Field-Programmable Gate Arrays (FPGAs). System uses an architecture based on non-deterministic finite automaton for fast pattern matching. Using this approach, throughput up to 3.2 Gbps is achieved on for all rules from Snort database. [46] Hardware acceleration is also interesting for commercial companies. [8]

Although signature-based detection handles mainly with packet payload, some signature consist of properties acquired by flow-based data collection. Then the limiting detection is possible. However, if sampling is used, some packets containing signatures can be lost and the effectivity is thus lower. We chose Snort as an implementation of the signature-based detection for evaluation.

We conclude the coverage is low, because only the known attacks specified by signatures are revealed by this method. The affectivity vary according to the quality of signatures, the risk of high false positives is high in common networks, even without the use of some IDS evasion techniques, e. g. squealing [43]. Performance is reasonable for high-speed networks only if it is supported by the hardware acceleration. The use of flow-based data as an input for this method is limiting.

Generally, signature-based IDS suffers from considerable latency in deployment of a brand-new rule (a signature) in such system. Last, but not least, the method cannot cope with encrypted payload.

## 2.5 Stateful Protocol Analysis

Another approach to intrusion detection is stateful protocol analysis that operate mainly on the higher layers of the TCP/IP network model. We mention it here for completeness and comparison.

**Description** Stateful protocol analysis (alternatively deep packet inspection) is the process of comparing predetermined profiles of generally accepted definitions of benign protocol activity for each protocol state against observed events to identify deviations. Unlike anomaly-based detection, it relies on vendor-developed universal profiles that specify how particular protocols should and should not be used. That means that the IDS is capable of understanding and tracking the state of network, transport, and application protocols that have a notion of state. [12]

For example, when a user starts a File Transfer Protocol (FTP) session, the session is initially in the unauthenticated state. Unauthenticated users should only perform a few commands in this state, such as viewing help information or providing usernames and passwords. An important part of understanding state is pairing requests with responses, so when an FTP authentication attempt occurs, the IDS can determine if it was successful by finding the status code in the corresponding response. Once the user has authenticated successfully, the session is in the authenticated state, and users are expected to perform any of several dozen commands. Performing most of these commands while in the unauthenticated state would be considered suspicious, but in the authenticated state performing most of them is considered benign. [12]

**Evaluation** Although there are some tools implementing basic stateful protocol analysis (such as stream43 in Snort), the method is not wide-spread (such as signature-based detection).

We identify the following reasons. Firstly, it is a very resource-intensive task, particularly in high-speed networks. The complexity of the analysis grows with the number of (simultaneous) sessions 4. Secondly, it relies on the "knowledge" of all analyzed protocols. Notice there are numerous differences between implementations by various vendors and definitions in RFC and other standards. In addition, only the analysis of known protocols is possible. Next, the attacks (e. g., denial of service attacks) that utilized well-formed packets and do not violate the normal behavior are not detected. Finally, the method is impuissant to encrypted packet payload too.

On the other hand, the method generally provides relatively high accuracy. In contrast to signature-based method that searches for known patterns in the packet payload, this method works with sessions. The method can correlate information obtained from the whole session together and provides better view inside the network traffic. Stateful protocol analysis can also reveals some threats that could be omitted by other methods that performs port-based traffic classification. Last, but not least, a limited subset of the analysis can process flows too.

## 2.6 Anomaly-based Detection

It is the process of comparing definitions of what activity is considered normal against observed events to identify significant deviations. An IDS using anomalybased detection has profiles that represent the normal behaviour of such things as users, hosts, network connections, or applications. The profiles are developed by monitoring the characteristics of a typical activity over a period of time. The major benefit of anomaly-based detection methods is that they can be very effective at detecting previously unknown threats. For example, suppose that a

computer becomes infected with a new type of malware. It will probably perform behaviour that would be significantly different from the established profiles for the computer.

[12]

### 2.6.1 Holt-Winters Method

This method, also known as triple exponential smoothing, has proven through the years to be very useful in many forecasting situations. It was first suggested by C. C. Holt in 1957 and was meant to be used for non-seasonal time series showing no trend. He later offered a procedure (1958) that does handle trends. Winters (1965) generalized the method to include seasonality, hence the name "Holt-Winters Method". [44]

**Description** Many service network variable time series exhibit the following regularities (characteristics) that should be accounted for by a model:

- A trend over time (i. e., a gradual increase in application daemon requests over a two month period due to increased subscriber load).
- A seasonal trend or cycle (i. e., every day bytes per second increases in the morning hours, peaks in the afternoon and declines late at night).
- Seasonal variability (i. e., application requests fluctuate wildly minute by minute during the peak hours of 4–8 pm, but at 1 am application requests hardly vary at all).
- Gradual evolution of regularities (1) through (3) over time (i. e., the daily cycle gradual shifts as the number of evening daylight hours increases from December to June). [3]

A simple mechanism to detect an anomaly is to check if an observed value of the time series falls outside the confidence band. A more robust mechanism is to use a moving window of a fixed number of observations. If the number of violations

(observations that fall outside the confidence band) exceeds a specified threshold, then trigger an alert for aberrant behaviour. Define a failure as exceeding a specified number of threshold violations within a window of a specified numbers of observations (the window length). See details in [3].

The author outlines this method for networking monitoring, but it can be useful for security analysis too. The continuation represents NfSen-HW, an experimental version of NfSen5 by Gabor Kiss from HUNGARNET6. He added a kind of aberrant behaviour detection based on the built-in Holt-Winters algorithm of RRDtool. [21]

**Evaluation** The settings of the parameters _, _, , confidence band and threshold are not clear. The model parameters need to be set and tuned for the model to work well. There is no single optimal set of values, even restricted to data for a single variable. This is due to the interplay between multiple parameters in the model. [3] The author also gives some suggestions and, more generally,
the authors of [14]. The fact that the fine tuning of Holt- Winters analysis is not a trivial task is confirmed by [11]. The settings could influence the false positives rate and consequently the accuracy of the method. The training period is also crucial: we naturally get false positives, if the malicious activity is considered normal.

A one-week training period usually gives satisfactory results. The coverage is quite good. Because some threats and attacks behave similarly, we can suggest the "sensitive" variable of network traffic and then detect even the previously unknown threats.

The question of performance is tightly coupled with the data acquisition. Theoretically, both packet capture and flow-based approach are possible. The latter provides aggregated information as input for the method. This means that the method itself does not work with a huge amount of data in (almost) real-time. That ensures the underlying layer: flow-based probes and collectors. The method "only" computes the forecast on the basis of the historical data (typically recent week [21]). The

flowbased approach was chosen for the deployment of this method in GEANT project [11] (NfSen-HW relies on NetFlow data).

### 2.6.2 Minnesota Intrusion Detection System (MINDS)

**Description** The core of MINDS is an anomaly detection technique that assigns a score to each network connection that reflects how anomalous the connection is, and an association pattern analysis based module that summarizes those network connections that are ranked highly anomalous by the anomaly detection module. The analysis is performed on 10-minute windows of NetFlow data8. Firstly, the feature extraction is done. MINDS introduces two types of features derived from standard NetFlow features: time window-based, connections with similar characteristics in the last T seconds and connection window-based, last N connections originating from (arriving at) distinct sources (destinations). The former obviously do not include malicious activities (such as stealthy port scans) which last more than T seconds. Hence, it is complemented by the latter.

The time window-based features are:

- Count-dest, number of flows to unique destination IP addresses inside the network in the last T seconds from the same source,
- Count-src, number of flows from unique source IP addresses inside the network in the last T seconds to the same destination,
- Count-serv-src, number of flows from the source IP to the same destination port in the last T seconds
- Count-serv-dest, number of flows to the destination IP address using same source port in the last T seconds.

**The connection window-based feature list follows:**

- count-dest-conn, number of flows to unique destination IP addresses inside the network in the last N flows from the same source

- count-src-conn, number of flows from unique source IP addresses inside the network in the last N flows to the same destination
- count-serv-src-conn, number of flows from the source IP to the same destination port in the last N flows
- count-serv-dest-conn, number of flows to the destination IP address using same source port in the last N flows. [27]

Secondly, the data is fed into the MINDS anomaly detection module that uses an outlier detection algorithm to assign the local outlier factor (LOF) [25], an anomaly score to each network connection. The outlier factor of a data point is local in the sense that it measures the degree of being an outlier with respect to its neighborhood. For each data example, the density of the neighborhood is first computed. The LOF of a specific data example represents the average of the ratios of the density of the example p and the density of its neighbors. [27]

Finally, the MINDS association pattern analysis module summarizes network connections that are ranked highly anomalous by the anomaly detection module. This module also uses some signature-based detection techniques. See section 3.5 in [27].

**Evaluation** MINDS was deployed at the University of Minnesota in August 20029. It has been successful in detecting many novel network attacks and emerging network behavior that could not be detected using signature based systems such as Snort. See section 3.4 in [27].

Input to MINDS is NetFlow version 5 data, because the authors admit they currently do not have the capacity to collect and store data in pcap (tcpdump) format. LOF requires the neighborhood around all data points be constructed. This involves calculating pairwise distances between all data points, which is an O(n2) process, which makes it computationally infeasible for millions of data points. The author suggests a sampling of a training set from the data and compare all data points to this

small set, which reduces the complexity to O(n × m) where n is the size of the data and m is the size of the sample. [27] On the other hand, the effectiveness can be decreased, namely because of the potential presence of threats in the training set. The coverage is good; the authors claim that the LOF technique also showed great promise in detecting novel intrusions on real network data. [27]

### 2.6.3 The Work of Xuetal.

Kuai Xu et al. developed a method that employs a combination of data mining and information theoretic techniques applied to flows, classify and build structural models to characterize host/service behavior's of similar patterns. [48]

**Description** The authors work with four-dimensional feature space consisting of srcIP, dstIP, srcPrt and dstPrt. Then clusters of significance along each dimension are extracted.

Each cluster consists of flows with the same feature value in the said dimension. This leads to four collections of interesting clusters: srcIP, dstIP, srcPrt and dstPrt clusters. The first two represent a collection of host behaviors while the last two represent a collection of service behaviours. Clusters with feature values that are distinct in terms of distribution are considered significant and extracted; this process is repeated until the remaining clusters appear indistinguishable from each other. This yields a cluster extraction algorithm that automatically adapts to the traffic mix and the feature in consideration. For example, the authors get 117 srcIP clusters from 89 261 distinct source IP addresses in trace file used in.

The second stage of the methodology is a behaviour classification scheme based on observed similarities/dissimilarities in communication patterns (e.g., does a given source communicate with a single destination or with a multitude of destinations?). For every cluster, an information-theoretic measure of the variability or relative

uncertainty RU of each dimension except the (fixed) cluster key dimension is computed:

Next, the dominant state analysis capture the most common or significant feature values and their interaction. The authors find clusters within a behaviour class have nearly identical forms of structural models ("simpler" subsets of values or constraints which approximate the original data in their probability distribution). This model can also help an analyst because it provide interpretive value for understanding the cluster behaviour.

Finally, the authors identified three "canonical" profiles: server/service behaviour (mostly providing well-known services), heavy-hitter host behaviour (predominantly associated with well-known services) and scan/exploit behaviour (frequently manifested by hosts infected with known worms). These profiles are characterized by BCs they belong to and their properties, frequency and stability of individual clusters, dominant states and additional attributes such as average flow size in terms of packet and byte counts and their variability.

**Evaluation** Firstly, there is no free available implementation (as opposed to Snort), hence the benchmarking is doubtful. However, we suppose satisfactory performance because the method was developed for saturated backbone links. In addition, it processes aggregated NetFlow data captured in 5-minute time slot10, not payload of each packet that go through in real-time.

Another advantage is that this method promises high coverage. The behaviour profiles are built without any presumption on what is normal or anomalous. The method dynamically extracts significant flows. There are no fixed rules applied to particular flow or packet.

The flow (cluster) is marked as exploit if it belongs to such profile. What is more, we can observe rare and interesting relationship between clusters and particular flow,

which can point out other (unknown) malicious behaviour (e. g., clusters in rare BC, behavioural changes for clusters and unusual profiles for popular service ports).

The authors did not mention (and they actually could not mention) accuracy evaluation, because they used live network traffic without any knowledge of structure (mainly portion of malicious traffic). Last, but not least, we note the method stands on a port-based traffic classification.

### 2.6.4 Origin Destination Flow Analysis

Anukool Lakhina et al. have introduced Origin-Destination (OD) flow as a basic unit of network traffic. It is the collection of all traffic that enters the network from a common ingress point and departs from a common egress point. [24] They believe that a thorough understanding of OD flows is essential for anomaly detection too. Lakhina et al. distinguish from other authors because they perform wholenetwork traffic analysis: modeling the traffic on all links simultaneously. OD flow is often high-dimensional structure (it depends on the size of the network), hence the authors utilize a technique called Principal Component Analysis (PCA) to reduce the "dimensionality". They found that the hundreds of OD flows can be accurately described in time using a few independent dimensions. The following description and evaluation is relevant to two chosen methods based on OD flow analysis (not to the only one as in previous sections).

**Description** Volume anomalies detection is based on a separation of the space of traffic measurements into normal and anomalous subspaces, by means of PCA. [23] The authors suppose that a typical backbone network is composed of nodes (also called Points of Presence, or PoPs) that are connected by links. The path followed by each OD flow is determined by the routing tables. The authors use the term volume anomaly to refer to a sudden (with respect to time step used) positive or negative change in an OD flow's traffic. Because such an anomaly originates outside

the network, it will propagate from the origin PoP to the destination PoP. OD flow based anomalies are identified by observing link counts. The mapping of the data to principal axis i with normalization to unit length follows.

Such vectors capture the temporal variation common to the entire ensemble of link traffic time series along principal axis i. Since the principal axes are in order of contribution to overall variance, the first vector captures the strongest temporal trend common to all link traffic, the second captures the next strongest, and so on. Next, the vectors (components) are separated by a simple threshold-based method. As soon as a projection is found that exceeds the threshold (e.g., contains a 3_ deviation from the mean), that principal axis and all subsequent axes are assigned to the anomalous subspace. All previous principal axes then are assigned to the normal subspace. [23] Then we can decompose a set of traffic measurements at a particular point in time into normal and residual components. The size of the residual component is a measure of the degree to which the particular measurement is anomalous. Statistical tests can then be formulated to test for unusually large size, based on setting a desired false alarm rate. See details in [23], section 5.1. Another method, feature entropy detection is presented in [22]. The method comes from observation of a change in distributional aspects of packet header fields, features. In contrast to previous method, it can also capture some anomalies that have a minor effect on the traffic volume: worms spreading, stealthy scans or small denial of service attacks. Thus, traffic feature distributions are used there instead of traffic volume. Entropy captures in a single value the distributional changes in traffic features, and observing the time series of entropy on multiple features exposes unusual traffic behaviour. [22]

### 2.6.5 Cooperative Adaptive Mechanism for Network Protection (CAMNEP)

CAMNEP is an agent-based network IDS. It is not the only one method, but the whole system based on a few method described above. In spite of the fact, we

mention it here, because it is an interesting concept of an incorporation of modern detection method that profits from the synergy effect.

**Description** The architecture consists of several layers (see Figure 2.1) with varying requirements on on-line processing characteristics, level of reasoning and responsiveness. While the low-level layers need to be optimized to match the high wire-speed during the network traffic acquisition and preprocessing, the higher layers use the preprocessed data to infer the conclusions regarding the degree of anomaly and consecutively also the maliciousness of the particular flow or a group of flows.

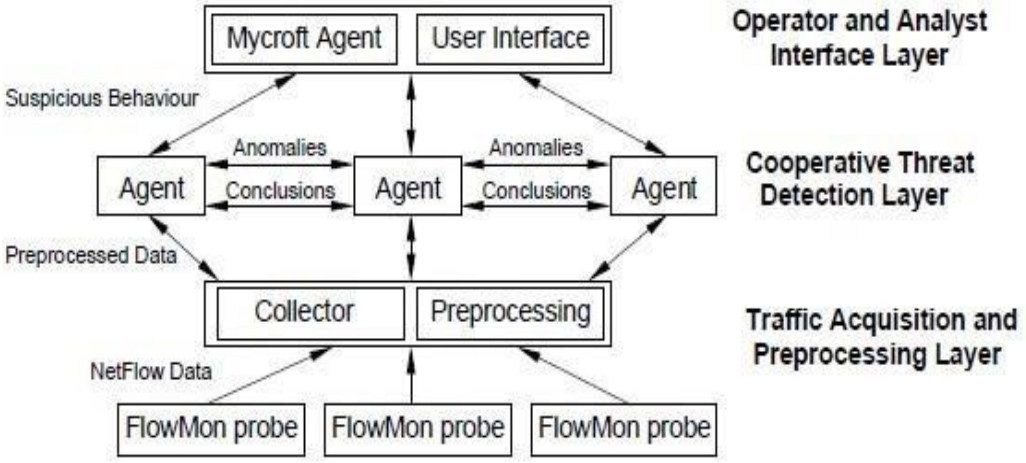


Figure 1: CAMNEP Architecture

Traffic acquisition and preprocessing layer acquires the data from the network using the hardware-accelerated NetFlow probes and perform their preprocessing. This approach provides the real-time overview of all connections on the observed link. The preprocessing layer aggregates global and flow statistics to speed-up the analysis of the data.

Cooperative threat detection layer consists of specialized, heterogeneous agents that seek to identify the anomalies in the preprocessed traffic data by means

of their extended trust models. There are four agents that employ detection methods based on MINDS, work of. and work of Lakhina et al. Note the agents are not complete implementation of the methods described in previous sections. The authors chose only these features and ideas that are computationally efficient in nearrealtime and even is possible to integrate them into the whole agent platform. For example, MINDS agent performs only simplified observation of time windowdefined features and compares them with history data to determine the anomaly of each flow.

As a result, each agent determines the anomaly of each flow as a value in the [0, 1] interval, where 1 represents the maximal anomaly, and 0 no anomaly. The values are shared with other agents. Each agent integrate these values into its trust model. To preserve the computational feasibility, these models work with significant flow samples and their trustfulness in the identity-context space.

Trustfulness is also determined in the [0, 1] interval, where 0 corresponds to complete distrust and 1 to complete trust. Hence, low trustfulness means that the flow is considered as a part of an attack. The identity of each flow is defined by the features we can observe directly on the flow: srcIP, dstIP, srcPrt, dstPrt, protocol, number of bytes and packets. If two flows in a data set share the same values of these parameters, they are assumed to be identical. The context of each flow is defined by the features that are observed on the other flows in the same data set, such as the number of similar flows from the same srcIP, or entropy of the dstPrt of all requests from the same host as the evaluated flow. The identities are the same for all agents, but the contexts are “agent-specific”.

The anomaly of each flow is used to update the trustfulness of flow samples in its vicinity in the identity-context space. Each agent uses a distinct distance function, because it has a different insight into the problem. The cross correlation function is implemented to eliminate random anomalies.

Finally, each agent determines the trustfulness of each flow and all agents provide their trustfulness assessment to the aggregation and visualization agents, and the aggregated values can then be used for traffic filtering. The authors can define the common misclassifications errors using the trustfulness and maliciousness of the flow. The flows that are malicious and trusted are denoted as false negatives, and the flows that are untrusted, but legitimate are denoted as false positives.

The higher level is operator and analyst interface layer. The main component is an intelligent visualization agent that helps the operator to analyze the output of the detection layer, by putting the processed anomaly information in context of other relevant information. When the detection layer detects suspicious behaviour on the network, it is reported to visualization.

**Evaluation** First of all, note that CAMNEP as a whole stands on the incorporated detection methods. One advantage is that the architecture is modular. The agent platform can be widened by other agents, other (new) anomaly-based detection methods. The authors argue that the use of trust model for integration of several anomaly detection methods and efficient representation of history data shall reduce the high rate of false positives which limits the effectiveness of current intrusion detection systems. We participated on the evaluation and testing of the system. Results are also described in [4].

In a nutshell, the attacks with more than several hundreds flows are consistently discovered by all agents. The slower attacks, using lower number of flows (300 and less) are more tricky. Note that the evaluation was performed in a campus network loaded with thousands of flows per second. On the other hand, CAMNEP is not able to detect attacks consist of few packets, e. g. buffer overflow attack.

## 2.7 Summary

In this chapter, we studied a few detection methods for security analysis of a computer network. Definitely, this is not an exhaustive list of known methods, but a selection of widespread and as well as interesting methods and approaches. We started with the commonly used signature-based method. Although, it operates at higher layers than we are focused on, it is good for a comparison with other methods. Then we briefly described and evaluated stateful protocol analysis that extends previous method in a particular way. Both methods inspect packets even their payload. Note this approach also can interfere with law issues.

In contrast, the anomaly-based detection methods generally process flows, namely 5- tuple (srcIP, srcP ort, dstIP, dstP ort, protocol) constructed from packet headers. It is more efficient, particularly in multi-gigabit networks. On the other hand, the flow acquisition is not a simple task, especially for non-dedicated devices such as routers. Due to that fact, packet sampling is used. Unfortunately, it can introduce some inaccuracy. The impact of packet sampling on anomaly detection is discussed in [15].We think that future work could be aimed at other key features that form the flow. Thus, the 5-tuple could be changed and/or extended.

Another significant contrast between statistical methods and the others is that statistical methods build behaviour profiles at host and service levels using traffic communication patterns without any presumption on what is normal or anomalous. However, the “level of presumption” differs. While Holt-Winters algorithm builds a model for normal traffic based on parameter settings and a priori knowledge of the periodic structure in traffic, the methods proposed by Xu et al. and Lakhina et al. do not rely on any parameter settings and normal traffic behaviour is captured directly in the data.

Next, the statistical anomaly-based methods have to cope with three basic steps that were outlined in [23]:

• Detection,

• Identification,

• Quantification.

In fact, there is only one step in case of the other methods. They simply "know" what they find (e. g., in terms of signature or protocol definition), hence we a priori identify a searched anomaly and quantify its relevance.

Finally, there are a few existing IDSs based on the mentioned methods. Snort is a leading representative of signature-based IDS and the de facto standard for intrusion detection. It is wide-spread because it is an open-source software. Currently, we did not find any network-based toolset that implements anomaly based detection methods. The one exception to this conclusion is most likely CAMNEP that validated the selected methods in distinct environment to the authors' environment.

# CHAPTER III
# VISUALIZATION

The key problem of the analysis is to comprehend the results of the whole process. We can acquire data that (truly) picture the network traffic and process them by various methods.

However, if we do not use any data-mining technique, we still have to interpret the results manually. It is throughout feasible in small network, but absolutely inconceivable in high speed networks because the human being does not manage to evaluate the large amount of information. The visualization should help us and present significant information in different and more comfortable view. For example, tcpdump is the most used tool for network monitoring and data acquisition.

It is a command-line tool that can read packets from network interface or data file and display each packet on a new line on output. In contrast, a network packet analyzer Wireshark1 utilizes graphical user interface (GUI) and, for instance, "colourizes" packet display based on filters. Actually, the tool processes classification and results are presented as various color's. We also can interactively browse the capture data, view summary and detail information for each packet. We confirm such (small) improvements ease the analysis.

However, not only the color's usage is the visualization. In this chapter, we discuss the visualization as an integral part of modern security analysis. We outline some ways of visualization in current software tools and evaluate their contribution to the analysis acceleration.

We mainly focus on open-source software that visualize captured network traffic in pcap or NetFlow format. Meanwhile data in pcap format contain packet headers and the payload, NetFlow records intentionally omit the payload.

### 3.1 Charts

The basic visualization instrument is a chart. There are many tools extending basic software that perform only data acquisition. These tools often plot twodimensional charts that depict time series of monitored values or their aggregations. It is a simple and thus widespread method of visualization. Namely, NfSen [28] integrates nfdump outputs with various charts that show time series of total number of packets, flows and traffic volume. See Figure 3.1.

The charts are also used in other tools such as FlowScan2, Java Netflow CollectAnalyzer3, ntop4, nfstat5, NetFlow Monitor6, Caligare Flow Inspector7 or Stager8. Charts are also used in network monitoring. A network administrator can easily look at the appropriate chart and immediately make a decision if a network or

security anomaly occurred. In such cases, the relevant curve used to grows or drops sharply.

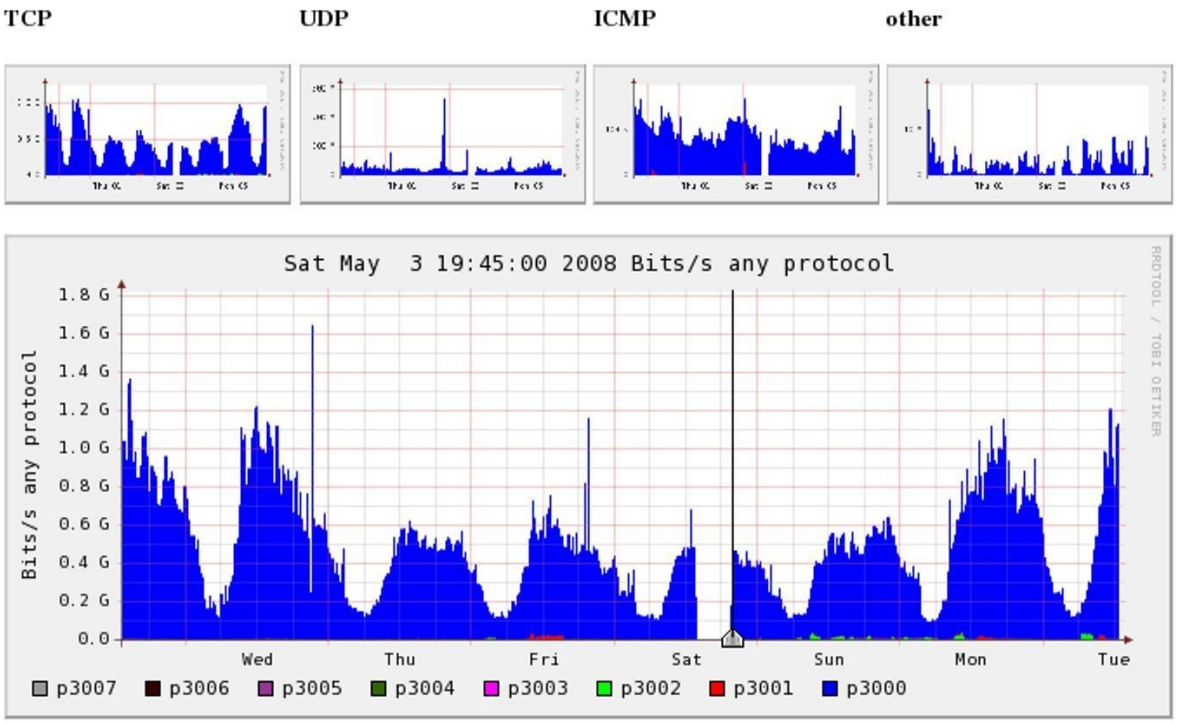


Figure 2: A chart of network traffic volume in NfSen

## 3.2 Mapping in Space

This visualization technique draws points in two or quasi three-dimensional space that is displayed on a screen. It makes use of the human stereoscopic vision and "convert" patterns in the captured data into graphic patterns in defined space. For instance, The Spinning Cube of Potential Doom is an animated visual display of network traffic. Each axis of cube represents a different component of a TCP connection: X is the local IP address space, Z is the global IP addresses space and Y is the port numbers used in connections to locate services and coordinate communication (such as 22 for SSH and 80 for HTTP). TCP connections, both attempted and successful, are displayed as single points for each connection. Successful TCP connections are shown as white dots. Incomplete TCP connections

are shown as colored dots. Incomplete connections are attempts to communicate with nonexistent systems or systems no longer listening on that particular port number.

The Cube colours incomplete connections using a rainbow colour map with colour varying by port number; colour mapping assists viewers in locating the point in 3D space. [6] For example, a port scan in captured data creates a line in the cube (see Figure 3.29). It is more useful and efficient view on such event comparing to a manual examination of a tcpdump or even Wireshark output.

An extension of the Cube is InetVis [16]. Similar approach is also used by Flamingo [9]. PortVis [29] and rather use two-dimensional space.

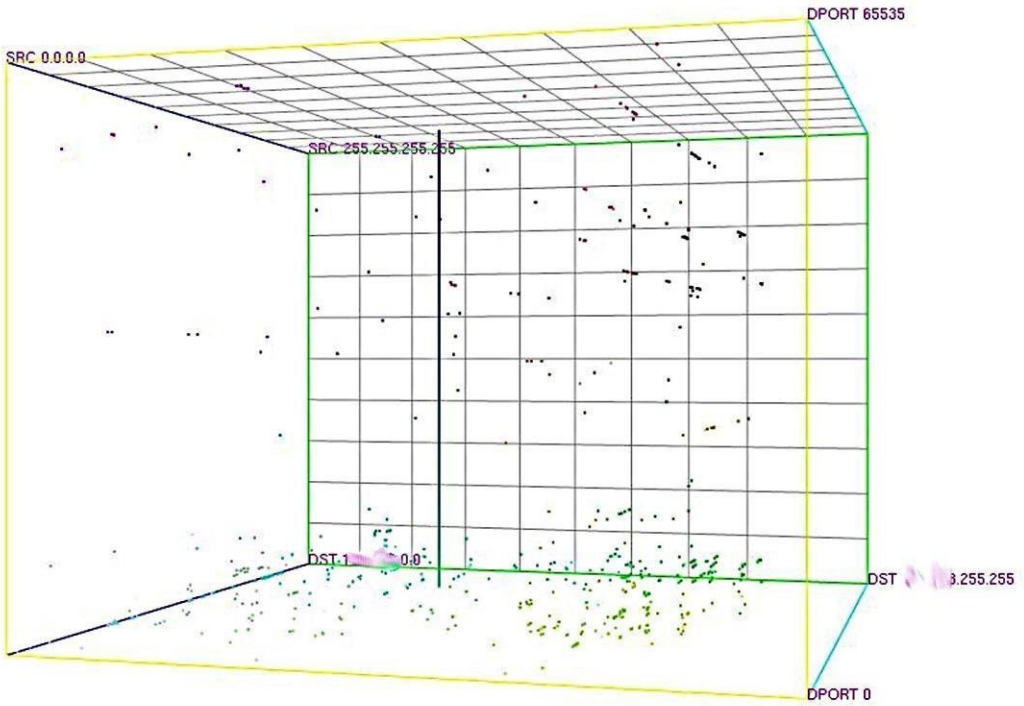


Figure 3: port scan the GPL

## 3.3 Graphs

A natural representation of the network traffic is a graph where vertices correspond to hosts and (oriented) edges correspond to the communication (flows)

captured between the hosts (see Figure 3.3). This structure digestedly depicts who communicates with whom. For a comparison, classic output is on Figure 3.4.

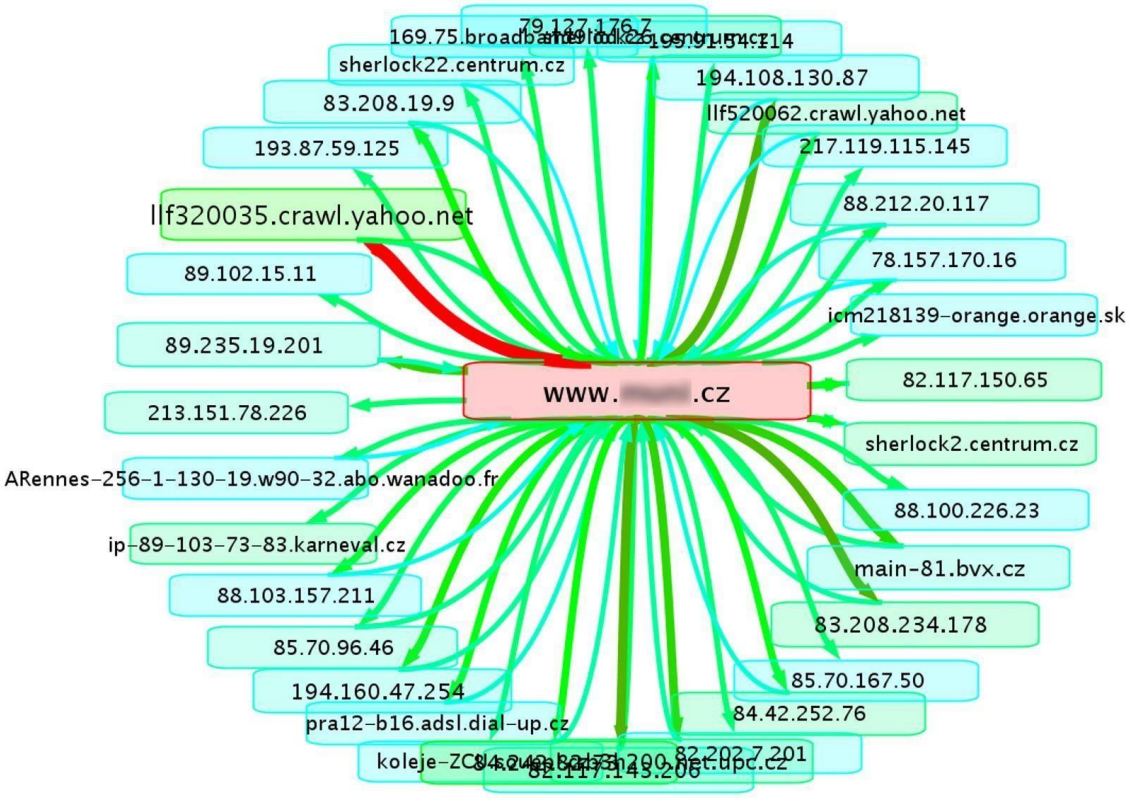


Figure 4: Network traffic as a graph

NfVis10 stands for NetFlow Visualizer and it is a proof of concept tool based on the perfuse visualization toolkit11. The graph-based traffic representation is enhanced with several significant features. The user can list the flows and traffic statistics associated with each edge/host. The traffic can be filtered and aggregated according to many relevant features.

The visual attributes of the display (such as node/edge size and colour) can also adapt to these characteristics, making the user's orientation easier. The information provided by "third parties" (DNS) is seamlessly integrated into the visualization. As current network traffic is a scale-free network, it is particularly important to handle the visualization of super nodes, i.e. the nodes with a high

number of connections. These nodes are typical for many attack scenarios, as well as for high-value targets. Visualizer therefore replaces the one-shot connections to/from these hosts by a special representation of a “cloud” of traffic, and only singles out the nodes that also connect to other nodes in the observed network.

MyNetScope12 is a network visual analytics platform based on the standard NetFlow data and heterogeneous data sources. It evolves NfV is in two important ways. Firstly, it can incorporate other external data sources such as DNS resolution, who is response, outputs of

| | | | | | | |
|---|---|---|---|---|---|---|
| 117.50.86.71 | 148.228.205.218 | TCP | 315 | 37 | 80 | 66796 |
| 107.252.174.21 | 148.228.205.218 | TCP | 17 | 1 | 80 | 1135 |
| 148.228.205.218 | 107.252.174.21 | TCP | 16 | 1 | 42587 | 17382 |
| 96.207.181.5 | 148.228.205.218 | TCP | 6 | 1 | 80 | 684 |
| 148.228.205.218 | 96.207.181.5 | TCP | 6 | 1 | 1933 | 737 |
| 123.64.213.194 | 148.228.205.218 | TCP | 1 | 1 | 80 | 40 |
| 148.228.205.218 | 117.50.86.71 | TCP | 7 | 1 | 4632 | 990 |
| 148.228.205.195 | 121.47.105.162 | TCP | 196 | 2 | 1346 | 274595 |
| 148.228.205.218 | 117.50.86.71 | TCP | 18 | 1 | 4684 | 19235 |
| 148.228.205.218 | 117.50.86.71 | TCP | 6 | 1 | 4685 | 4335 |
| 148.228.205.218 | 117.50.86.71 | TCP | 1 | 1 | 4635 | 40 |
| 121.47.105.162 | 148.228.205.195 | TCP | 169 | 4 | 443 | 6834 |
| 148.228.205.195 | 121.47.105.162 | TCP | 79 | 2 | 1352 | 107215 |
| 148.228.205.218 | 117.50.86.71 | TCP | 9 | 1 | 4687 | 6867 |
| 148.228.205.218 | 117.50.86.71 | TCP | 6 | 1 | 4690 | 1792 |
| 148.228.205.218 | 117.50.86.71 | TCP | 8 | 1 | 4686 | 1666 |
| 148.228.205.218 | 117.50.86.71 | TCP | 18 | 1 | 4689 | 15058 |
| 148.228.205.218 | 117.50.86.71 | TCP | 8 | 1 | 4691 | 1804 |
| 148.228.205.218 | 117.50.86.71 | TCP | 5 | 1 | 4694 | 1868 |
| 121.37.81.102 | 148.228.205.195 | TCP | 93 | 9 | 443 | 8030 |
| 148.228.205.218 | 117.50.86.71 | TCP | 22 | 2 | 4695 | 21238 |
| 148.228.205.218 | 117.50.86.71 | TCP | 11 | 2 | 4696 | 8111 |
| 148.228.205.218 | 117.50.86.71 | TCP | 16 | 2 | 4692 | 5753 |
| 148.228.205.218 | 117.50.86.71 | TCP | 8 | 1 | 4688 | 4400 |
| 237.238.8.219 | 148.228.205.218 | TCP | 226 | 16 | 80 | 18888 |
| 148.228.205.218 | 116.204.159.253 | TCP | 14 | 1 | 2118 | 18957 |
| 148.228.205.218 | 116.204.159.253 | TCP | 4 | 1 | 2117 | 3762 |
| 116.204.159.253 | 148.228.205.218 | TCP | 18 | 4 | 80 | 3450 |
| 101.249.217.238 | 148.228.205.218 | TCP | 13 | 1 | 80 | 941 |
| 148.228.205.218 | 101.249.217.238 | TCP | 12 | 1 | 44531 | 13927 |
| 231.39.29.116 | 148.228.205.218 | TCP | 32 | 3 | 80 | 2148 |
| 148.228.205.218 | 116.135.87.175 | TCP | 13 | 2 | 4842 | 16840 |
| 148.228.205.218 | 254.166.192.97 | TCP | 26 | 1 | 1575 | 31400 |
| 254.166.192.97 | 148.228.205.218 | TCP | 15 | 8 | 80 | 4090 |
| 116.135.87.175 | 148.228.205.218 | TCP | 7 | 1 | 80 | 1026 |
| 107.252.188.190 | 148.228.205.218 | TCP | 1525 | 46 | 80 | 111012 |
| 148.228.205.218 | 107.252.188.190 | TCP | 15 | 1 | 59975 | 16354 |
| 148.228.205.195 | 121.37.81.102 | TCP | 64 | 2 | 1882 | 78651 |
| 148.228.205.218 | 117.50.86.71 | TCP | 7 | 3 | 4636 | 280 |

Figure 5: Network traffic as a listing of flows

Various anomaly detection methods and the network topology information. Secondly, it is a scalable solution even for wide networks. We participate on its development and testing, hence we can confirm these statements. The integration of external data sources is very welcome because it is not common that a security analyst works only with primary data such as tcpdump outputs or NetFlow records. He or she generally has to gather additional information from other available sources. Otherwise, the complete inspection of the security incident is not possible.

We also mention other graph-based visualization tools. VisFlowConnect-IP visualizes network traffic as a parallel axes graph with hosts as nodes and traffic flows as lines connecting these nodes. These graphs can then be animated over time to reveal trends. Cooperative Association for Internet Data Analysis (CAIDA) develops two interesting tools.

LibSea13 is both a file format and a Java library for representing large directed graphs on disk and in memory. Scalability to graphs with as many as one million nodes has been the primary goal. Additional goals have been expressiveness, compactness, and support for application-specific conventions and policies.Walrus14 is a tool for interactively visualizing large directed graphs15 in three-dimensional space. By employing a fisheye-like distortion, it provides a display that simultaneously shows local detail and the global context. Although, they are not specialized application for network traffic visualization, it would be useful to combine them for this purpose if there was a tool that provides output in LibSea format.

### 3.4 Summary

We explained why the visualization is important in the security analysis and introduced three techniques and tools that they utilize. The common used charts were

subsequently complemented by methods that use mapping in space and graph representation of network traffic. We also summarized their contribution to the analysis.

Naturally, the progress of visualization tools is connected with development of tools that acquire and/or process network data. E. g., both tcpdump and Wireshark stand on libpcap a system-independent interface for user-level packet capture16. Similarly, NfSen is a graphical web-based front end for the nfdump NetFlow tools and Walrus stands on LibSea.

We hope that a good visualization tool should display a complex picture of the network traffic, ideally with marked up-to-date security incidents. However, all available details of hosts and their communication should be displayed in wellarranged tables, charts and listings on demand too.

# CHAPTER IV
# DESIGN OF THE IDS

We described and evaluated several approaches to the intrusion detection as well as visualization techniques of network traffic. In this chapter, we take into account our conclusions and discuss the design of the intrusion detection system for large networks. First, we identify and give reasons for the requirements on such IDS and then we design a solution that meet these requirements.

First of all, notice that we decided for intrusion detection system. In contrast to intrusion prevention system (IPS), it "only" monitors the network traffic and alerts an operator in case of a security incident. Consequently, he or she analyses the incident and eventually ensures its mitigation. If we deployed IPS and it alerted false positive, it would immediately block a legitimate network connection. Another reason is that IPS must be in-line (a part of the link).

When the IPS fails, the whole network may fail as well. Hence, we are conservative because of the occurrence of false positive alarms and system failure. These are the main reasons for the IDS deployment.

## 4.1 Requirements on the IDS

### 4.1.1 Accuracy

Accuracy is a fundamental requirement on any IDS. However, it is very difficult to meet this requirement for current systems. They suffer from high rate of false positives. In addition, there are some IDS evasion techniques such as squealing. Due to these facts IDSs are not widely accepted and deployed by network administrators. High false positive rate overwhelms the administrators that are busy anyway. On the contrary, false negatives are undetectable in routine operation. So IDS creates a "false sense of security".

### 4.1.2 Detection of Novel Threats

Now, there are many IDS capable of detection of known threats, especially signature-based

IDS such as Snort. Their drawback is that the rule base of such IDS has to be maintained by the network or security administrator. Moreover, novel threats are included in the rule base manually, often by third-party vendors. Finally, it is obvious that these systems are forceless to novel threats. Therefore, the proposed IDS should detect even novel threats by some more efficient detection mechanism.

### 4.1.3 Operating in a High-speed Networks

We request a solution that will operate in multi-gigabit networks. In case of the data link layer is Ethernet, the IDS should support 1 and even 10 Gigabit Ethernet at wire speed. Note that IEEE is developing 40 and 100 Gigabit Ethernet now.

### 4.1.4 Early Detection

IDS should begin with the detection as soon as possible a network packet passes through its sensor. The results should be available to the security administrator in (near) real-time because some security incidents last only a few minutes, even a few seconds.

### 4.1.5 Long-term Data Storage

Besides the early detection, the IDS should also provide records of mid-term and long-term data. This is important when a Computer Security Incident Response Team (CSIRT) outside our organization reports a security incident that originated from our network before some time. If the IDS stores appropriate records, the security analysis is then easier.

### 4.1.6 IPv6 support

Although, wide IPv6 [19] deployment is not as fast as it was expected1, we require its support. Nowadays, there are many well-secured IPv4 networks and the

administrators work on IPv6 deployment. However, they often “forget” about IPv6 network security. Thus, the IDS should operate on both IPv4 and IPv6.

#### 4.1.7 Scalability

IDS should monitor a network consisting of hundreds as well as thousands of computers. IDS should be scalable and should not require any additional maintenance when a new host is connected to the network or another host is disconnected or replaced. Again, the additional maintenance annoys network administrators.

#### 4.1.8 Easy Maintaining

This requirement is closely connected with scalability. Moreover, we expect the IDS maintenance will not consume too much time of a system administrator after its deployment. Technically, all hardware components should be rack-mountable into a standard 19" rack.

#### 4.1.9 Transparency

The notion of transparency actually comprises two requirements. First, the IDS should be “invisible” at the IP layer. That means we should not assign any IP to the IDS (except a management module). This is required to avoid some attacks such as (distributed) denial of service (DDOS and DOS) where attacker floods the network with packets destined for the IP address of IDS. Second, the IDS should not markedly influence network topology and network traffic in any way. Namely, latency should be preserved and the IDS should not load network links uselessly.

#### 4.1.10 Security Robustness

It is clear that IDSs attract attackers’ attention. The IDS itself should be invulnerable and robust to security threats. We can prevent some attacks if we meet the previous requirement of transparency at the IP layer. Next, the IDS integrity should be intact. For instance, if the IDS is composed of several components, their

communication could be invaded or eavesdropped. At all events, the security administrator must receive true results of the detection.

#### 4.1.11 Anomaly Detection in Encrypted Traffic

Many current IDSs fail in the detection of threats in encrypted network traffic. Such systems rely on the payload inspection. The proposed IDS should recognize anomalies even in the encrypted traffic because more and more network services use encryption.

#### 4.1.12 User-friendly Interface and Well-arranged Visualization

Last, but not least requirement is on the user interface. If the IDS meets all the previous requirements, but the presentation of the results is not well-arranged, the IDS is not usable.

On the one hand, the interface should be helpful to the user and should offer all available views of the data. On the other hand, it should provide support for repetitive transactions and detailed view. The interface should be personalized by the user.

### 4.2 Solution

We decided for Network-based IDS (NIDS) to meet the following requirements:

• Scalability,

• Easy Maintaining,

• Security Robustness.

In contrast to Host-based IDS (HIDS), the deployment of a new host in network does not demand more effort to monitor the network activity of the new host. There is no need to install any specialized software on the host. Note that the network may consist of some specialized hosts (besides common servers or workstations). So, the HIDS installation is impossible in such a case. Next, NIDSs are passive devices, “invisible” for the attackers. On the contrary, HIDSs rely on processes that running

in the operating system of the host. We also consider the deployment, testing and possible upgrade of IDS. Generally, it is easier to update one component of NIDS than many components of HIDS on hosts.

We propose the solution that is consisted of several components and layers. Network probes are “eyes and ears” of the proposed intrusion detection system. Collectors are the “memory”, MyNetScope with data sources is the “brain and heart” and MyNetScope analyst console acts as the “mouth” of the IDS. The “nervous system and blood circulation” is represented by network links that connect all parts together. After all, the architecture (Figure 4.1) is similar to the CAMNEP architecture depicted in Figure 2.1.

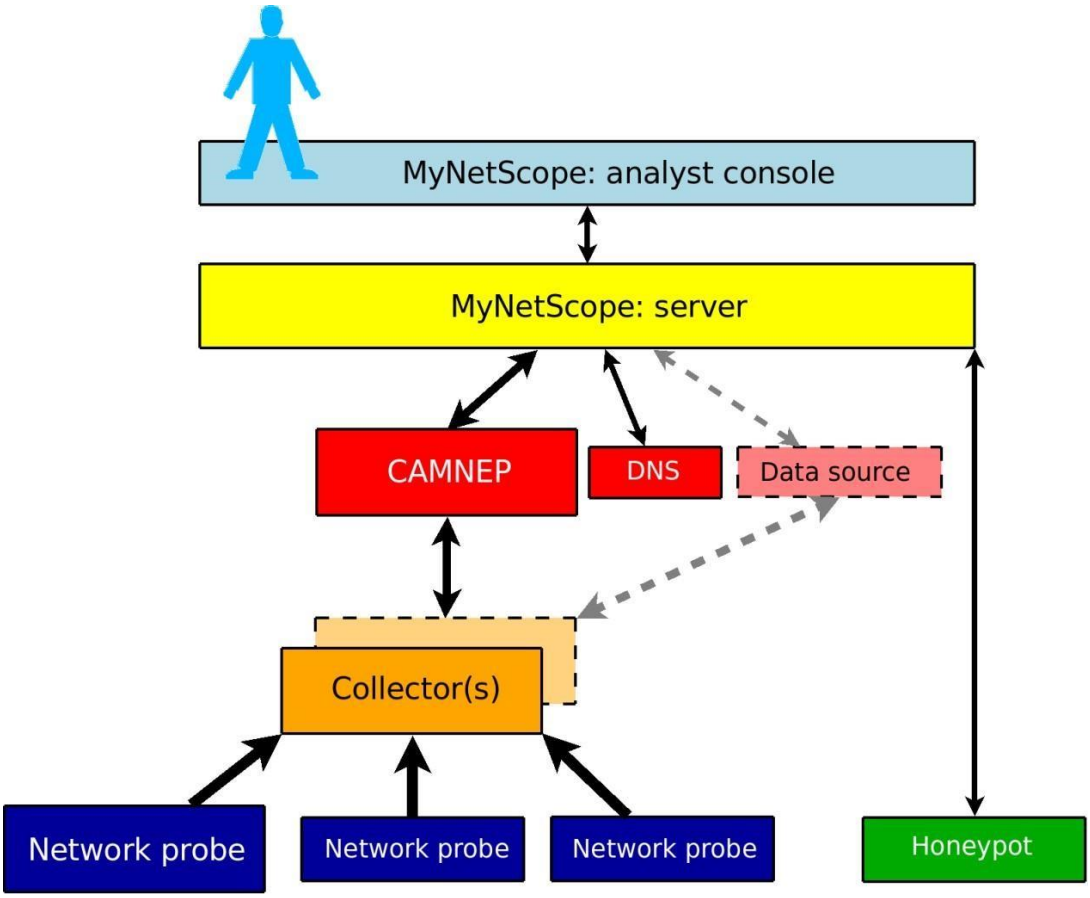


Figure 6: The Architecture of the proposed system

### 4.2.1 Network Probes

Probes create the bottom layer of our system. They acquire network traffic and serve collectors with captured data. This section discusses probe features and probe deployment in the administered network.

**Data acquisition** Network probes monitor the link and export captured data in the Net- Flow format. We decided for this format to meet the requirement on operating in multigigabit networks. We reject the use of SNMP counters and packet traces. The former gives coarse-grained data and the latter is very difficult. It is practically infeasible to capture and store packet at wire speed even with specialized hardware.

We emphasis we do not rely on NetFlow data that export some (edge) Cisco routers that may exist in present network. Not only have our measurements revealed that Cisco's routers do not export NetFlow correctly in all circumstances. [26] Obviously, the main task of the router is to route network traffic. We must take into account that NetFlow export is additional feature.

On the other hand, the NetFlow data from routers can be supplemental data source for our system.

Next, we rather avoid the packet sampling due to possible distortion of acquired data.

Our decision is supported by [15]. We recommend to use probes based on COST (commercial off-the-shelf) computers because of their cost. There are two alternatives of network interface cards (NIC) used in the probes. The former utilizes common NIC (such as Intel) and the latter rely on the COMBO technology developed in the Liberouter project2. The software probes that capture network traffic by NIC (such as nprobe) is not sufficiently efficient. [20] Hence, we deploy Flow-

Mon, a hardware-accelerated passive network monitoring probe. [10] Generally, the software probes are satisfactory for small networks, the hardware-accelerated probes for large, multi-gigabit networks. Both types of probes meet the requirement on transparency since they are "invisible" at the IP layer. There is no IP address assigned to the interface performing packet capturing. IPv6 is supported thanks to the use of NetFlow version 9.

**Location** A network probe monitors traffic passing through a certain node of the network. Thus, the location of the network probe determines what is monitored. This is very important because the proposed system is based on data provided by network probes. Ideally, each packet that ingresses or egresses the administered network should pass through the place where the probe is located. We discuss this with network administrators of the campus network of the Masaryk University. We identify that the probes should be located "in the neighborhood" of the edge router considering the network traffic from/to the Internet. Figure 4.2 shows the location of the main probe. We were choosing between two alternatives.

We suppose that the edge router acts as a firewall too. If we placed the probe in front of the router/firewall, we would also monitor the traffic that would not enter the administered network. We chose the second alternative. The main probe is located in the administered network, behind the router/firewall. This ensures that the probe "see" only the traffic that passed through the firewall. The firewall usually implements (a part of) the security policy of the organization.

As discussed above, we will not insert the probe into the network link, but only a network tap. It is a hardware device which provides a way to access the data flowing across a computer network3. Thus, we actually delegate the responsibility for the continuous operating to the tap. If we use the tap that requires power supply, we

should connect it to the uninterruptible power supply (UPS). Also we should choose tap with dual power supply unit in case of failure.

The main probe is capable to capture only the attacks that originate from or are destined for outside the network. Concerning attacks by insiders, we propose to deploy other probes

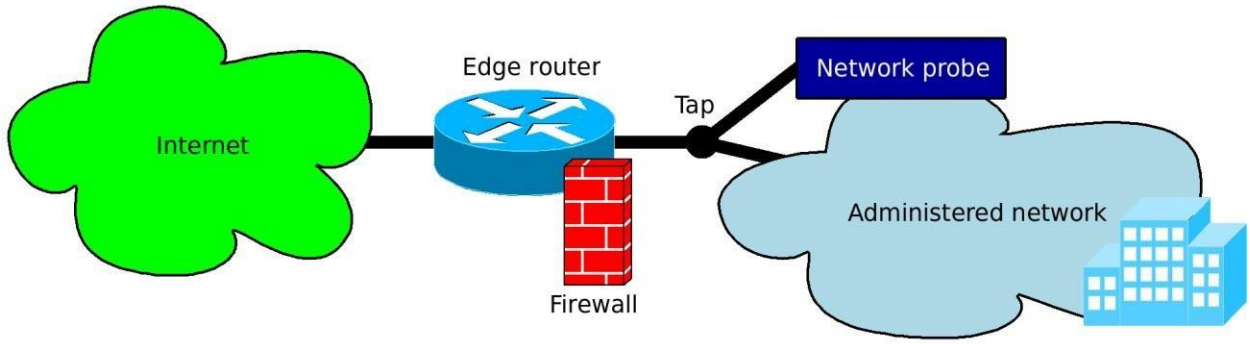


Figure 7: Network probe location

Inside our network, especially in front of/behind the firewalls that protect particular network segments. Then we can reveal possible malicious activities of hosts in our network. For instance, Figure 4.3 depicts deployment of one main probe and three inside the administered network. It can be demanded in campus or corporate networks. There is one segment consisting of more sensitive servers than the others or the organization is large enough to monitor network traffic inside the organization.

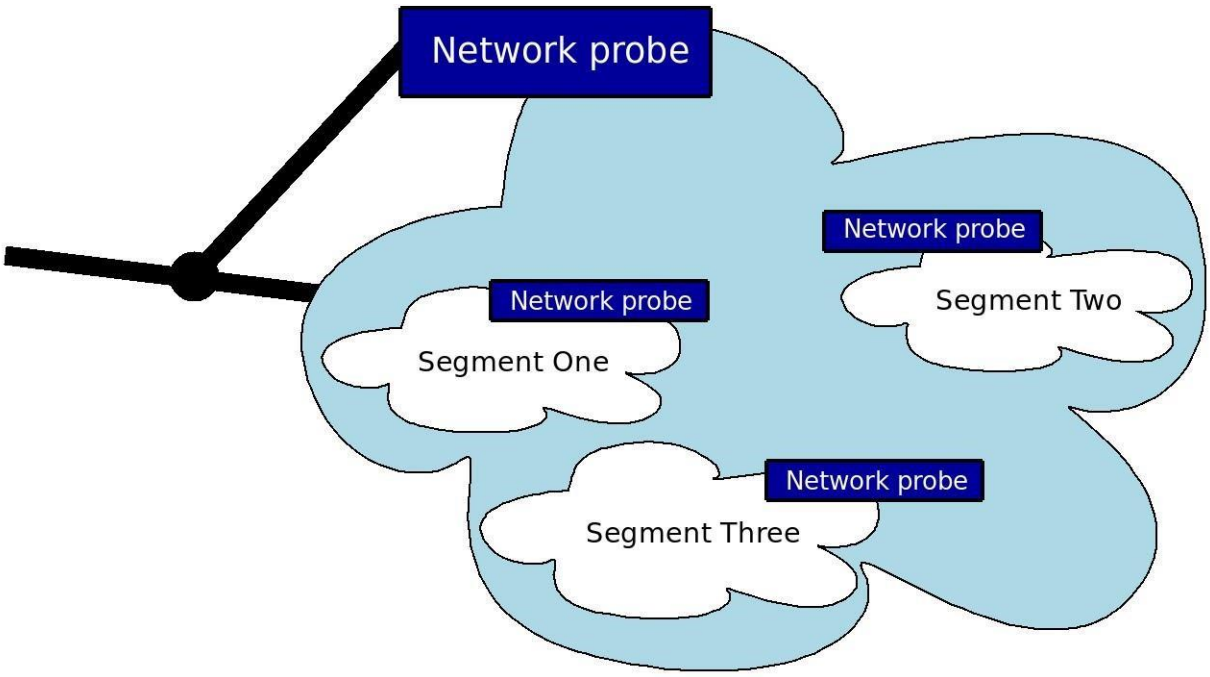


Figure 8: probes inside the network

**Honeypots** beside the NetFlow probes, we propose to deploy honeypots to complement the probes functionality. It is an information system resource whose value lies in unauthorized or illicit use of that resource. [13] We chose a lowinteraction honeypot because we want to perform passive rather than active detection. The output of a honeypot should be a list of hosts (from outside and even inside the network) that try to communicate with imaginary hosts in the administered network. Typically, we reserve several unassigned IP addresses (or the whole subnet) for the honeypot. If it observes a connection attempt to such address, it logs the host that originated the connection. However, we ought to avoid premature conclusions. For example, consider an user who type an incorrect IP address, misconfigured host and so on.

**Security:** Security robustness is very important for such devices as network probes. The probe itself is controlled via management interface. We use secure channel (namely SSH) and the access is granted only from specified IP addresses.

We employ identity management system such as RADIUS [31]. It is advantageous to distributed systems because it eliminates synchronization issues. Last, but not least, we use NTP4 to synchronize the clocks of computers over a network. Since the probes timestamp the flows using the host time it is necessary to set the precise time.

**Maintenance and Management** Generally, the probes are easy to maintain devices. If we place them in network and set up, they will work and fulfill their task. However, if they do not send any data to the collector, we cannot determine whether the monitored link or the probe fails. Hence, we employ NETCONF Configuration Protocol [36] over SSH to monitor a probe status.

### 4.2.2 Collectors

A NetFlow collector is responsible for correct reception and storing NetFlow data that are exported by network probes. To prevent reinventing the wheel, we use existing tools and software that is well tested and wide-spread. In case of NetFlow collectors, we rely on nfdump and NfSen toolset [28]. Our collectors receive and store NetFlow records but also perform some preprocessing tasks such as periodically execution of scripts that monitor policy violation. Collectors comply with requirements described above as well as other parts of the proposed IDS.

**Security** To meet security requirements, we specify IP addresses of probes that are authorized to send the NetFlow data to the particular collector. Notice that the collector itself does not restrict the reception of NetFlow records. It can be considered to be a security threat since the NetFlow records are transmitted in UDP packets that can be easily forged. If we do not want to transmit NetFlow records via the same network, we can connect the collectors directly to the probes through local network and thus considerably intensify the security. In addition, this could lighten the loaded network links.

**Long-term data storage** Although NetFlow records are already aggregated (in terms of network flows), they occupy relatively a lot of disk space. For example, the records that cover one month of network traffic of large campus network occupy about 240 GB of disk space5. If we do not deploy more probes, we could utilize only one collector. Nevertheless, long-term data storage requires enough space on disk drives.

### 4.2.3 MyNetScope and Data Sources

In this section, we describe the core of our intrusion detection system. This layer requires data from collectors and other sources for its operation.

**MyNetScope** We employ MyNetScope platform that was briefly described in Section 3.3.

It is not a standalone application, it is designed as client/server architecture. The server reads NetFlow records from collectors, performs some preprocessing tasks on the flows and replies to analyst's queries that are submitted by client application (analyst console). Again, the entire communication between all parts is encrypted. We use SSH tunnels.

**CAMNEP** MyNetScope itself does not perform intrusion detection. It is very useful visualization tool that meets the requirements in Section 4.1.12. Its power is in integration of external data sources. We decided to deploy part of the CAMNEP project (described and evaluated in Section 2.6.5) as the "brain" of our intrusion detection system. Thus, we can meet following requirements:

• Accuracy,

• Detection of Novel Threats,

• Operating in a High-speed Networks,

• Early Detection,

• Anomaly Detection in Encrypted Traffic.

We use mainly the CAMNEP Cooperative Threat Detection Layer that combines modern intrusion detection methods. In summary, we get better accuracy than we would deploy particular anomaly detection methods separately. The methods are able to detect novel threats and anomalies in case of the security anomaly is captured as network traffic anomaly too.

For instance, a worm spreading or denial of service attack is “visible” in network flows. On the contrary, single packet that causes buffer overflow on a host computer does not represent the network traffic anomaly. Next, the methods were designed for high-speed networks from the very beginning or they were modified to meet this requirement. The detection is performed in 5-minute time windows. This is a reasonable interval due to flow aggregation, commonly used in connection with NetFlow. Finally, since the methods work purely with packet headers, the anomaly detection is possible even in case of the encrypted payload.

CAMNEP Detection Layer computes for each network flow its trustfulness. This value is then imparted to MyNetScope and the user can view the suspicious flows and query the MyNetScope for other relevant information.

**Other data sources** Apart from CAMNEP, we also utilize other data sources such as DNS server, who is service or specific scripts that periodically check for policy violation. Their output is then included in MyNetScope too. These scripts are discussed in the next chapter.

### 4.3 Summary

We identified and explained twelve fundamental requirements on an intrusion detection system for large networks. Then we design a distributed system that meet these requirements.

The system consists of several layers and components:

- NetFlow probes and honeypots,
- Collectors,
- CAMNEP and other data sources, MyNetScope platform: server and client (analyst console).

# CHAPTER V
# DEVOLOPMENT OF IDS

We have already begun with system deployment and testing in the large campus network. This chapter summarizes our present experience in using the designed system. First, we describe in detail the system deployment status. We structure the description according to Section 4.2. Then we outline a use case and compare a security analysis performed with the help of the designed system with the classic approach.

## 5.1 Deployment status

### 5.1.1 Network Probes

First of all, we started with the probe deployment and testing. As discussed in Section 4.2.1, we considered various probe locations. We were discussing with network administrators and we were testing selected locations. Finally, we decided for the main probe located behind the edge router/firewall and the other probes located in front of the firewall that protects selected subnets (typically faculty subnets). It arises from the organization structure of the university. Institute of Computer Science (ICS) is responsible for the development of information and communication technologies at the university. Although the faculties and other departments are to a certain degree autonomous units, they must adhere to rules1 and cooperate with ICS. Therefore, it is useful that such an arrangement of probes can capture a policy violation inside the network. The main probe is temporarily connected to the SPAN port of the edge router (Cisco Catalyst 7609). We chose

hardware-accelerated FlowMon probe with 10 Gigabit Ethernet interface. Since we use the SPAN port2, we have to enable packet filtering at the probe.

Thus, only packets from/to the Masaryk University are acquired by the FlowMon probe.

We were also testing traffic acquisition of all packets from SPAN port, but the router serves other international links that are heavy loaded. It required several times more disk space on the collector. In addition, the probe cannot determine the correct AS3 because the traffic contains packets from all interfaces of the router. Now, the probe processes every weekday about 6 TB of data (1.2 Gbps spikes) in 200 million of flows (4 000 flows per second spikes).

We have recently deployed the second probe. It is located in front of two routers that connect the Faculty of Informatics with the university backbone. There are two network taps between the backbone routers and the routers of the Faculty. The probe is connected to the taps. We employ a four-port software probe FlowMon there. According to our measurement, we decided for non-accelerated version of the probe. Although, we have not yet acquired any data from this probe, we expect the link usage will be lower than in case of the main probe. Both probes export data in NetFlow version 9 format.

A honeypot deployment is being prepared. It is a small daemon that creates virtual hosts on a network. The hosts can be configured to run arbitrary services, and their personality can be adapted so that they appear to be running certain operating systems. Honeyed enables a single host to claim multiple addresses.4 Network administrators have already assigned the address space for honeypots. We dispose of 254 IPv4 addresses for a honeypot operation. Although the address space is unused, we can observe numerous requests for the connection originated outside the administered network. So we expect the honeypot will help us with security analysis.

We also plan to assign some IPv6 addresses to the honeypot.

In this phase of the deployment, we decided to assign public IP addresses to the probe management interfaces due to an easier access and maintenance.

### 5.1.2 Collectors

We still use only one PC5 equipped with 1TB hard drive. We estimate that this is sufficient to store NetFlow record from the main probe for about 4 months. We will consider the usage of some data thinning technique, compression or other collectors dedicated to each probe.

Results obtained from the data acquisition by the second probe can answer this question. Nevertheless, we have to cope with the trade-off between the long-term data storage and the completeness of the records.

The collector is also utilized for preprocessing. There is the crone daemon6 periodically executing scripts that check the policy violation. The scripts are described in detail in the next subsection. They usually perform tasks that load the collector and their evaluation last some time (typically a few minutes). It is not surprising, because they typically process all day data (up to 17 GB). So, the scheduling and planning has become more important in case of many scripts. Similarly to the probes, the collector is protected by firewall and communicates via assigned public IP address.

### 5.1.3 MyNetScope and Data Sources

We have designed the use of the CAMNEP project as the main data source. The probes and the collector is prepared to CAMNEP deployment in the next phase. Now, we are focused on MyNetScope. We are responsible for MyNetScope analyst console testing and development of the scripts, the additional data sources. The MyNetScope analyst console is still under development and in alpha testing phase.

We are currently reporting bugs and suggest improvements of the system to MyNetScope developers. We have already deployed two scripts that check the selected rules of the security policy.

These scripts are in routine operation and their output helps with a security analysis. The scripts are periodically executed every night7 on the collector and provide output in two formats. First, plain text files at the web server that is running on the collector are useful for network administrator. Second, files with rules for MyNetScope platform access these external data sources in MyNetScope analyst console. According to the rules, the nodes (hosts) that violate the security policy are "colorized". In addition, we plan that user will be able to filter the hosts that violate the particular policy.

**Reverse DNS entry policy** the first script checks if all hosts (IPv4 addresses) from the

Masaryk University network that were communicating previous day have a valid DNS reverse entry. Every Internet-reachable host should have a name. Many services available on the Internet will not talk to you if you are not correctly registered in the DNS. For every address, there should be a matching PTR record in the in-addr.arpa domain. [30] Examples of effects of missing reverse mapping are described in [7].

The script utilizes nfdump tool. It filters all communicating hosts from the Masaryk University network and save the output to a temporary text file. Next, all IP addresses are passed as a parameter to a DNS lookup utility host8. If the lookup fails, the relevant IP address is logged. Network administrators can then inform appropriate administrators who are responsible for such hosts. In spite of the fact that the script execution time differs, it takes approximately up to 10 minutes in a weekday.

**SMTP traffic policy** the second script checks for anomalies in the SMTP traffic on TCP port 25. It is not permitted to send e-mails to SMTP servers outside the Masaryk University network excepting several well-known servers. We also monitor which hosts in the administered network behave as SMTP servers due to possible participation in spam campaigns.

The script logs all host inside the network that were communicating via TCP port 25 excepted replies to port scanning attempts. We take into account only the flows that contain packets with TCP flags SYN, ACK and FIN. That means we are interested in TCP connections where the 3-way and 4-way handshake occurred. The former is used for the connection establishment and the latter for its termination.

Again, we use nfdump with a relevant filter to obtain interesting hosts. The script execution takes about 5 minutes. We point out it processes all-day data.

## 5.2 Use Case

In spite of the fact all parts of the system have not been deployed yet, we can use some its components for security analyses of the network, namely the main NetFlow probe and the NetFlow collector. We mention the system use case in this section. In April 2008, the Masaryk University received a warning on a phishing scam from Security Incident Response Team (SIRT) of Internet Identity9. Phishing is an attempt to criminally and fraudulently acquire sensitive information, such as usernames, passwords and credit card details, by masquerading as a trustworthy entity in an electronic communication10. Network administrators had confirmed this. Consequently, they disconnected the host from the network and informed us.

We had to investigate this security incident in three ways:

1. To validate the findings of SIRT,
2. To determine whether the phishing attack was successful,
3. To find out who was responsible for the attack.

Apart from information provided by a host administrator, we were inspected NetFlow records. First of all, we identified a host profile. We set a filter for the destination IP address of the host and filtered out all TCP flows that contained only SYN TCP flag. We found out the host was used via secure shell. The administrator confirmed that the host had been reserved for development and the presence of the web server was very suspicious. They also reported that the attacker had changed the super user password. Second, we validated that the host acted as a web server: it had replied to requests on TCP port 80 that is reserved for web traffic. In addition, we could exactly determine when the server had replied for the first time. In total, we observed 54 distinct hosts (IP addresses) that communicated with the attacked host. Hence, we fulfilled the first and the second point.

Finally, we were investigating the origin of the forged website. We supposed that the host had been exposed to a SSH brute force attack. Consequently, we inspected the network traffic on TCP port 22 that is reserved for SSH before the web server had been set up. We found an extreme growth of number of flows in short time. This could point out just SSH brute force attack. Since each attempt to log in is performed on a new port, it is considered to be a new flow in terms of NetFlow. We identified a host that was responsible for too many flows. So we fulfilled even the third point and closed the investigation of the security incident. We enclose a CD-ROM containing all relevant data to this incident (see Appendix B for the CD contents).

After some time, the administrator provided us a disk image of the entire drive of the attacked host. We found in log files some entries that confirmed our findings. Of course, we could investigate the incident without our system. We could only inspect the system log files. However, the logs or the whole host are not always available. For instance, consider the advanced attacker who deletes the log files. We

emphasis that we used only two (lower) layer of the designed system: FlowMon probe and NfSen collector. After the CAMNEP deployment the system will automatically determine a list of hosts (flows) with low trustfulness. In addition, MyNetScope platform visualizes the traffic as a graph, a natural picture of a network traffic.

### 5.3 Summary

We described the status of the development of the designed system. We were focused on our work: system component testing, development and integration of other data sources into the whole system (e. g., scripts that check the organization security policy). Although some parts of the system are still under development, we could use it to investigate the security incident with satisfactory results.

# CONCLUSION

The goal of this research was to design a system that simplifies a security analysis of large networks.

First of all, we studied the state of the art in intrusion detection and prevention. We focused on modern methods that operate at the IP layer since they are efficient in high-speed gigabit networks. On the contrary, stateful protocol analysis or signaturebased detection performed at higher levels of the TCP/IP model are both resource demanding tasks. Hence, some statistical methods do not inspect the whole packet but only the packet headers. They operate on NetFlow data acquired from routers

(typically from Cisco devices) or the packet traces that are later "converted" into network flows. Although these methods work only with the packet headers, they are able to detect some anomalies in the network behaviour. Next, we identified and explained essential requirements on the intrusion detection system.

Then we designed a distributed system that meets the requirements. The system consists of several various components. We combined some existing subsystems and have been developing an integration platform. We employed hardware-accelerated NetFlow probes, honeypots, NetFlow collectors, MyNetScope platform and other data sources such as DNS, who is and the output of other scripts that (pre)process acquired data. We note there are about fifteen people involved in this long-term and dynamic project.

We contributed to the system development by testing the particular components and examples of scripts that check some organization's security rules. These scripts are in routine operation and we can easily validate the adherence to the rules. We also tested a part of the system on the investigation of a security incident that was reported by a third-party. As a result, we identified a host that had attacked

computer from the Masaryk University. The host changed the super user password and ran a forged website to acquire usernames and passwords of clients of a bank. Finally, we suggest future work could be aimed at developing a new detection method based on new directions in data acquisition. Namely, the use of IPFIX format would “access” interesting feature in the packet payload for the anomaly-based detection methods. Currently, we are bounded by 5-tuple of NetFlow format. Also a closer integration of other data sources such as honeypots would be valuable.

# REFERENCES


[1] Northcutt, S. and Frederick, K. and Winters, S. and Zeltser, L. and Ritchey, R.: Inside Network Perimeter Security: The Definitive Guide to Firewalls, VPNs, Routers, and Intrusion Detection Systems, New Rider's Publishing, 2003, 978-0735712324. 2.1, 2.4

[2] Paxson, V.: Bro: A System for Detecting Network Intruders in Real-Time, 1999, <http://www.icir.org/vern/papers/bro-CN99.html> . 2.7

[3] Brutlag, J.: Aberrant behaviour Detection in Time Series for Network Monitoring, 2000, <http://www.usenix.org/events/lisa00/full_papers/brutlag/ brutlag_html/index.html> . 2.6.1

[4] Rehák,M. and Peˇchoucˇek,M. and Bartoš, K. and Grill,M. and Cˇ eleda, P. and Krmícˇek, V.: CAMNEP: An intrusion detection system for high-speed networks, 2008, <http: //www.nii.ac.jp/pi/n5/5_65.pdf> . 2.6.5, 2.1, 2.6.5

[5] Rehák,M. and Peˇchoucˇek,M. and Cˇ eleda, P. and Krmícˇek, V. and Novotný, J. andMina ˇrík, P.: CAMNEP: Agent-Based Network Intrusion Detection System (Short Paper), 2008. 3.3

[6] Lau, S.: The Spinning Cube of Potential Doom, 2004. 3.2

[7] Senie, D. and Sullivan, A.: Considerations for the use of DNS Reverse Mapping , 2008, <http://www.ietf.org/internet-drafts/ draft-ietf-dnsop-reverse-mapping-considerations-06.txt> . 5.1.3

[8] Graham, I.: Achieving Zero-loss Multi-gigabit IDS Results from Testing Snort on Endace Accelerated Multi-CPU Platforms, 2006, <http://www.touchbriefings. com/pdf/2259/graham.pdf> . 2.4

[9] Oberheide, J. and Goff, M. and Karir, M.: Flamingo: Visualizing Internet Traffic, 2006. 3.2

[10] Cˇ eleda, P. and Kovácˇik, M. and Konírˇ, T. and Krmícˇek, V. and Žádník, M.: CESNET technical report number 31/2006: FlowMon Probe, 2006, <http://www.cesnet. cz/doc/techzpravy/2006/flowmon-probe/flowmon-probe.pdf> . 4.2.1

[11] Malagon, C. and Molina, M. and Schuurman, J.: Deliverable DJ2.2.4: Findings of the Advanced Anomaly Detection Pilot, 6. 9. 2007, <http://www.geant2. net/upload/pdf/GN2-07-218v2-DJ2-2-4_Findings_of_the_Advanced_ Anomaly_Detetion_Pilot.pdf> . 2.6.1