MASARYK UNIVERSITY
FACULTY OF INFORMATICS

Flow-based Intrusion Detection in Large and High-Speed Networks

PHD THESIS PROPOSAL

Jan Vykopal

Supervisor: doc. RNDr. Václav Račanský, CSc.

Brno, January, 2010 .....
supervisor signature

Contents

1 Introduction **3**
1.1 Dictionary attacks 4

2 State of the art **6**
2.1 Network behaviour analysis 6
2.1.1 Flow acquisition 6
2.1.2 Data storage 9
2.1.3 Data analysis 10
2.1.4 Reporting and prevention 12
2.2 Dictionary attack detection 12
2.2.1 SSH dictionary attack detection 13
2.3 Summary 14

3 Proposed research **15**
3.1 Schedule 16

References **17**

A Results of my study and research **22**
A.1 Publications 22
A.2 Presentations 22
A.3 Participation in projects 23
A.4 Teaching, supervising and reviewing 23
A.5 Passed courses 23

B Reprints of selected papers with original results **24**

1 Introduction

Similarly to electricity, computer networks are an essential part of a modern infrastructure. Many business and educational processes rely on available and *secure* networks. Hosts connected to the public Internet are under continuous attacks and private networks might be vulnerable to insider attacks. Computer and network security is also related to national security and *cyber warfare*. There are many cases [13] of cyber attacks such as denial of service or a website defacement in recent years. All of these attacks are possible mainly due to the following facts: (i) security aspects were omitted in the design phase of applications or protocols (such as in case of the TCP/IP suite), (ii) users often do not follow guidelines or best practices (for example, they choose weak passwords) and (iii) legal authorities are not adequately prepared for investigation of cyber crime.

Network attacks can be detected and even prevented by specialized systems deployed as a device in a network infrastructure or application running at networked hosts. The goal of a *network-based intrusion detection* is to identify attacks or malicious behaviour by observing network traffic, preferably in real time. In comparison to a *host-based detection* performed at specific hosts, this approach scales well and is transparent for users (it is not necessary to install any software on hosts). Next, it is the only possibility of intrusion detection in a large network without direct access to particular hosts (namely, customers of an Internet service provider or *eduroam* users at university). Traditional network intrusion detection systems (NIDS) inspect packet payload for known signatures of attacks. But this is not feasible in high-speed (multigigabit) networks. Other limitations of traditional NIDSs are: (i) a high rate of false positives that overwhelm security operators and (ii) an inability to process encrypted traffic.

In contrast, *network behaviour analysis* (*flow-based intrusion detection*) relies on information and statistics of network flows. A de facto standard for IP flow monitoring is *NetFlow* format. A network flow is commonly identified by a 5-tuple key consisting of source and destination IP addresses, source and destination ports and protocol of the network or transport layer. Statistics related to the netflows (such as numbers of transferred packets and bytes) are computed in predefined time windows. So the flow acquisition provides an aggregated view of network traffic and typically does not provide any information about packet payload. In addition, it significantly reduces the amount of data that need to be processed by methods of a network behaviour analysis.

A flow acquisition and storage, two essential parts of flow-based intrusion detection (FID in short), are satisfactorily addressed by various researches. But a flow data analysis is still in the early phase. Hence, our work is focused on analysis of IP flows, design and prototyping new methods of FID. My thesis will aim at a dictionary attack detection in large net-

works as an example of the FID method. Dictionary attacks against weak passwords is a serious security threat that is often omitted by vendors and developers of many existing applications. At present, a typical detection and prevention of this kind of attack is done in the login process of the given application, if at all. We are not aware of any *network-based* detection mechanism that addresses this type of attack in the context of large, high-speed and heterogenous networks. However, we believe that dictionary attacks against different network services have similar characteristics. Next, the network-based detection is capable to capture even *distributed attacks* which are becoming more and more popular in these days. Section 1.1 introduces the notion of *dictionary attack* and gives examples of its recent occurrences. Chapter 2 summarizes state of the art of a flow-based intrusion detection and dictionary attack detection and prevention.

Next, we will adapt some existing algorithms of NIDS that process whole packets to the flow-based approach. As a result, these methods will be able to process traffic in high-speed networks without any loss.

We will also study correlation with other detection methods and data sources about ongoing attacks. It appears as a promising way to lower false positives of various detection methods. Also, attack scenarios and attack typology will be examined as a side effect of the main work.

Initially, we will get closer to the main objective, a development of dictionary attack detection in large networks, by studying dictionary attacks against a specific network service (Secure Shell [38]). Then, we will generalize attack pattern to other network services.

Honeypots are "network traps" that produce, by definition, no false positives. We will use them in the following ways: (i) in the design phase of the attack pattern and for the pattern evaluation, (ii) as another data source containing information solely about attacks in data correlation and (iii) in studies of attackers' behaviour. All designed methods and algorithms will be evaluated in real network of the Masaryk University. Generally, we will employ "Unix approach": develop a suite of several methods that perform well-defined task instead of "the holy grail" that addresses all threats and attacks. My research plan is summarized in Chapter 3.

1.1 Dictionary attacks

These attacks are aimed at a wide-spread knowledge-based authentication. Adversaries suppose that users choose their passwords from a small domain (namely words – therefore *dictionary* attack). So the adversaries attempt to login to user accounts by trying all possible passwords, until they find the correct one. We focus on *online* dictionary attacks. That means attackers can verify whether a password is correct or not only by interacting with the login server. We assume that *offline* dictionary attacks – attacks

that enable the attacker to check all possible passwords without requiring any feedback from the server – are effectively prevented at present. [23].

Dictionary attacks could be *simple* or *distributed*. In case of simple attacks, the adversary uses only one host that sends the authentication requests. Distributed attacks are stealthy and effective types of attacks, in which many attackers send relatively small numbers of requests at once. As a result, they are harder to detect than simple ones.

There are many studies (such as [28], [1], [30]) showing that dictionary attacks is a common type of network attacks. Another recent example[1] showed that dictionary attacks aim at various services and applications. We support this assertion by our own analysis of a new worm (botnet) exploiting default remote access to small office home office devices (e. g., ADSL routers, set-top boxes) around the world [40]. This worm is very malicious because it exploits vulnerabilities of devices that are not currently so secured like computers.

Last, but not least, a survey conducted by Yan et al. [37] showed that dictionary attacks were the most successful even if users had been forced to select non-trivial passwords.

[1] *Weak Password Brings 'Happiness' to Twitter Hacker*. Retrieved online January 3, 2010 at `http://www.wired.com/threatlevel/2009/01/professed-twitt/`

2 State of the art

In this chapter, we introduce a network behaviour analysis – one of the primary approaches to intrusion detection. All approaches are comprehensively described in [27]. Next, we summarize current state of the dictionary attack detection.

2.1 Network behaviour analysis

Here is the definition of *network behaviour analysis* proposed by Scarfone and Mell [27]:

> A network behavior analysis (NBA) system examines network traffic or statistics on network traffic to identify unusual traffic *flows*, such as distributed denial of service (DDoS) attacks, certain forms of malware (e.g., worms, backdoors), and policy violations (e.g., a client system providing network services to other systems).

The definition of a *flow* from RFC 3954 [8] follows:

> A flow is an 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.

A typical architecture of the NBA system consists of four layers that are described in the following sections. Figure 1 depicts the NBA system deployed in a network.

2.1.1 Flow acquisition

This layer acquires statistics about network flows and export them to a data storage for further analysis. NetFlow is a wide-spread format for IP flow monitoring and export. Originally, it was developed by Cisco Systems as a proprietary format (version 1–8) supported by their routers and switches. The Internet Engineering Task Force (IETF) standardized it as an open protocol (version 9) in 2006. The most common NetFlow versions are 5 and 9. Other versions are not widely used.

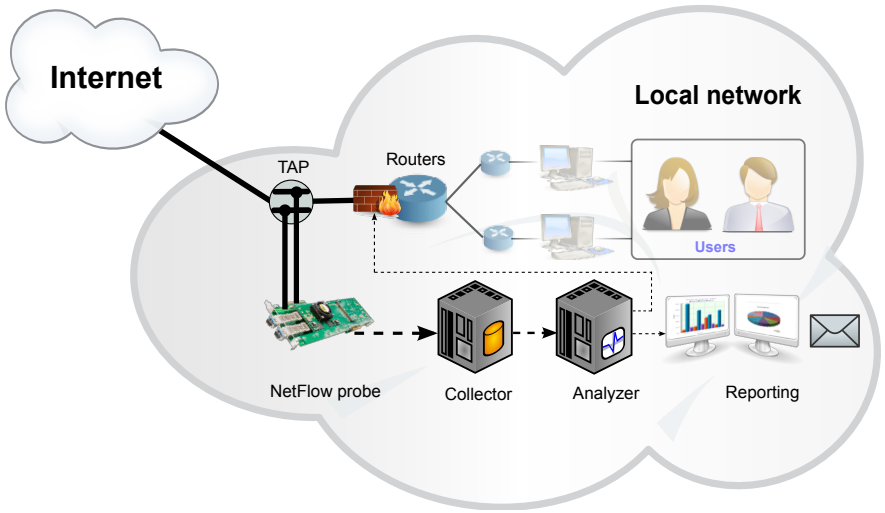

Figure 1: Architecture of a network behaviour analysis system and deployment in a network. A line width represents the amount of data transferred between particular layers.

NetFlow was followed by sFlow [22], another standard for flow monitoring that is supported by various vendors including 3Com, D-Link, Hitachi, Hewlett-Packard and NEC. The key difference between these standards is that sFlow provides only a sampled view of network flows (some packets are intentionally discarded and thus not considered in the flows acquisition process). Other vendors also develop their proprietary flow monitoring, namely Juniper Networks uses Jflow[2] and Huawei Technology has NetStream[3].

A NetFlow collection and expiration is driven by two values that can be set by user: the *active* and the *inactive timeout*. The active timeout is applied to regularly export information about 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 the collector.

Network flows are observed by routers, switches or specialized devices called *probes*. Originally, NetFlow export was implemented as an optional feature of routers and switches and acquired data were used for traffic accounting and billing. High-end devices that handle large amounts of data use *packet sampling* to reduce the flow acquisition overhead. However, packet sampling can negatively influence results of anomaly detection based on such flows. Namely, Mai et al. [19] demonstrated that "sampling

[2] `http://www.juniper.net/techpubs/software/erx/junose60/swconfig-routing-vol1/html/ip-jflow-stats-config2.html`

[3] Technical White Paper for NetStream: `http://www.huawei.com/products/datacomm/pdf/view.do?f=65`

distorts various traffic features and degrades the performance of three different port scan detection algorithms in terms of success detection ratio and false positives" and Brauckhoff et al. [3] concluded that sampling produces grossly inaccurate estimates of flow counts that makes the Blaster worm invisible at higher sampling rates. As opposed to this *deterministic sampling* – where exactly nth of every n packets is sampled – *random sampling* guarantees that each packet is sampled with a probability of $\frac{1}{n}$. This is secure against an eventual adversary that crafts packets to evade deterministic sampling [12]. Other types of sampling are motivated by decreasing data processing time. For example, Canini et al. [6] introduced *per flow packet sampling* to enable a traffic real-time classification of high-speed network traffic.

Acquiring *non-sampled* flow data in high-speed networks is possible only with a standalone probe. The probe is built as a dedicated PC server with a specialized application that captures packets from network interface card, extracts key features of the packets, maintains flow cache and sends expired NetFlow records to another server (*NetFlow collector*). On one hand, some setups in small networks (with a load up to hundreds of megabits per second) do not need the dedicated server but the application (e. g., *nProbe*[4], *fprobe*[5] and *softflowd*[6]). On the other hand, our measurements [32] showed that it is necessary to deploy packet capture acceleration such as *PF_RING*[7] to capture all packets on fully loaded 1 gigabit networks without any loss. 10+ gigabit networks demand hardware-accelerated network interface cards (produced, e. g., by Endace[8], Napatech[9] and Invea-tech[10]) or cards specialized to the NetFlow acquisition (such as those developed in the Liberouter project[11]).

In general, there are two alternatives how to connect the standalone probes to network. Network traffic that should be monitored is: (i) forwarded via mirror port of a router/switch to the probe or (ii) copied by *test access port* (TAP) – an inline hardware device providing real-time copy of all data passing through the observed link. We can confirm claims of Zhang and Moore [39] that port mirroring performed by routers/switches impact collected traffic in terms of timing difference, packet reordering and losses.

The IETF IPFIX working group puts effort into unification of protocols and applications of IP flow monitoring. RFC 3917 [24] defines requirements for exporting traffic flow information out of routers, middleboxes (such as

[4] `http://www.ntop.org/nProbe.html`
[5] `http://fprobe.sourceforge.net/`
[6] `http://www.mindrot.org/projects/softflowd/`
[7] `http://www.ntop.org/PF_RING.html`
[8] `http://www.endace.com/dag-network-monitoring-cards.html`
[9] `http://www.napatech.com/products/network_adapters.html`
[10] `http://www.invea.cz/products`
[11] `http://www.liberouter.org`

firewalls) or traffic measurement probes for further processing by applications located on other devices. Next, candidate protocols for IP Flow Information Export (IPFIX) were evaluated in RFC 3955 [18]. As a result, NetFlow version 9 was chosen for extension to IPFIX. The main advantage over previous NetFlow versions is that users can flexibly define the flow key. They are not constrained by fixed properties (5-tuple) such as in case of NetFlow version 5. The IPFIX working group published suite of RFCs that specifies basics of this protocol as well as guidelines for implementation [9, 25, 2]. In addition, they extended an unidirectional definition of the flow to bidirectional and introduced the notion of *biflow* including description of an efficient method for exporting biflows information using the IPFIX protocol [31]. We believe that the bidirectional view of network traffic is a natural extension that is very promising for further analysis and intrusion detection. Last, but not least, IPFIX allows to set timestamps of the flow start and end with nanosecond resolution. This is very important for some detection methods that rely on accurate flow timestamps. We have already encountered this limitation of NetFlow format that provides only millisecond precision.

2.1.2 Data storage

The next layer is responsible for receiving NetFlow records from the previous layer and their storage for further analysis. A device dedicated for this task is called a *NetFlow collector*. The collector provide software tools for data querying and simple analysis. There are many implementations available, both open source and commercial applications. A comprehensive list of tools is maintained by SWITCH (Swiss national and research network) on its website[12].

We briefly introduce some of them in the following text. *NFDUMP*[13] is a CLI[14] suite of tools for receiving/replaying, storing and filtering NetFlow data (version 5, 7 and 9). Proprietary binary files organized in a directory structures serve as a data repository. *NfSen*[15] is a graphical web-based front-end for the NFDUMP tools. It periodically plots time series of aggregate statistics (numbers of flows, bytes and packets) in the scope of user-defined filters. Moreover, NfSen can be extended by external plug-ins that process collected NetFlow data in various ways. *flow-tools*[16] is a suite of tools similar to NFDUMP. *SiLK*[17], the System for Internet-Level Knowledge, consists of two parts: the *packing system* is similar to NFDUMP tools

[12] `http://www.switch.ch/network/projects/completed/TF-NGN/floma/software.html`
[13] `http://nfdump.sourceforge.net/`
[14] Command-line interface
[15] `http://nfsen.sourceforge.net/`
[16] `http://www.splintered.net/sw/flow-tools/`
[17] `http://tools.netsa.cert.org/silk/`

(supports NetFlow version 5 and 9 – however, SiLK supports even IPFIX from *YAF*[18]; stores NetFlow data as a compressed binary flat files), but the *analysis suite* is a very part of the next layer of NBA. Besides previously mentioned open source applications, there are also commercial tools such as *Caligare Flow Inspector*[19] or *IBM AURORA*[20] aimed at enterprise users demanding GUI[21] with data visualization, charts and reports.

Generally, we identified and discuss three issues related to collectors: (i) NetFlow data are exported via unreliable UDP protocol that is also easier to forge (and distort real monitored data) – hence, TCP and STCP [29] are introduced as other transport protocols in the IPFIX protocol specification, (ii) there is some kind of "feedback" when the exported NetFlow data are transmitted via monitored links – it can be eliminated by dedicated links between the exporter and the collector and (iii) due to flow aggregation, acquired data are available in *near* real-time – there is a trade-off between delay and the desired aggregation.

2.1.3 Data analysis

This layer is represented by methods that process stored flow data in terms of intrusion detection and provide output to the *reporting and prevention layer*. If any attack is detected, output usually contains details of a suspicious flow (at least timestamp and attacker's IP address) and optionally attack details.

Although many collectors provide some basic flow data analysis functionality (namely, simple visualization, lists of top talkers and conversations, port scan detection or other basic statistics computed from raw NetFlow data), there is still a lack of advanced flow-based intrusion detection methods and algorithms. In recent years, new methods for both generic and even very specialized intrusion detection emerged. Complex methods are usually proposed and evaluated by research community on packet traces captured from real networks. Only some of them address incorporation into the whole system. They often do not consider other layers under and above the data analysis layer.

CAMNEP [26] is a collaborative agent-based system that adopts main ideas of current statistical methods of NBA: (i) MINDS [11] (which is, in fact, another example of flow-based intrusion detection system) employs an outlier detection algorithm to assign the *local outlier factor* (LOF) [4] (an anomaly score) to each network connection, (ii) behaviour models for profiling Internet backbone traffic [36], (iii) *Origin-Destination flow analysis* aimed at volume anomalies [16] that use matrices and *Principal Compo-*

[18] Yet Another Flow sensor: `http://aircert.sourceforge.net/yaf/`

[19] `http://www.caligare.com/netflow/caligare_flow_inspector.php`

[20] `http://www.zurich.ibm.com/aurora/`

[21] Graphical user interface

nent Analysis to separate the space of traffic measurements into normal and anomalous subspaces and (iv) detection based on *feature entropy* [17] – observing changes in distributional aspects of packet header fields (namely, source and destination addresses and ports). Besides these methods, there are many other statistical detection algorithms based on feature entropy. Nychis et al. [21] evaluated some of them and conclude that: (i) we should look beyond port and address distributions for fine-grained anomaly detection; (ii) consider distributions that complement each other in their detection capabilities and use bidirectional flow abstractions for computing traffic distributions.

A time series analysis is another statistical approach to anomaly detection. The *Holt-Winters method* alias *triple exponential smoothing* has been used for forecasting since 1960s. It considers *linear* and *seasonal trends* of the time series so it has many applications in various branches (e. g., economics and healthcare). Brutlag [5] first showed that it is possible to apply this method even on volume characteristics of network traffic and automatically detect changes between current and forecasted values.

Denial-of-service (DoS) attacks are a very serious threat to today's Internet. Therefore, much effort is also spent on a detection of these attacks and their distributed form [7].

Similarly to string pattern matching used in deep packet inspection, there are heuristics that are crafted to detect a very specific attack, an attack class or a misuse pattern. A typical example is a port scan detection: the data analysis is limited only to find flows with defined combination of TCP flags and/or packet size. If a number of such flows exceeds a predefined threshold, the port scan is detected. Other heuristics try to classify network traffic (including attack classes) using connection patterns (such as [14]).

Generally, specialized heuristics are not so powerful as statistical methods, which find outliers in the data sample or time series, and they cannot detect *zero-day attacks*[22]. But they are very efficient and has a lower false positives rate. While statistical methods provide better outputs for flow data from large networks or backbone links, heuristics can accurately detect attacks even in a smaller network (e. g., feature entropy of traffic of several hosts can highly vary and thus "hide" eventual attack).

Although the data analysis processes flow data aggregates, computational complexity is still the key limitation of many advanced methods. Fortunately, there are also examples of simpler but efficient methods: entropy estimation by common data compression algorithms [35] or the port scan detection. Some applications performing flow data analysis (e. g.,

[22] A zero-day attack exploits a security vulnerability that is unknown to others or there is not still any fix or patch available.

Stager[23]) try to improve speed of the data loading and processing by using more effective data organizations such as SQL databases.

2.1.4 Reporting and prevention

Finally, the highest layer of the NBA system presents outputs of the data analysis layer to user or sends them to other devices. If the analysis layer contains more detection methods, this layer should aggregate and even correlate their outputs to avoid overwhelming user with too many events.

An effective attack event presentation and visualization is still a big challenge. It is traditionally implemented as a virtual *dashboard* that summarizes detected attack in various time perspectives and forms (tables, charts, graphs, listings etc.).

Other ways of reporting are sending a message to: (i) a request tracker of CSIRT[24] or (ii) to other devices of the (network) infrastructure. Although there is a standard for exchanging such messages, the Intrusion Detection Message Exchange Format [10], most of the messages are still sent via e-mail or published as new rules or signatures for the specific IDS. An automated early warning among different sites is in the very beginning of development.

The accurate detection is an important prerequisite of a (semi)automated *intrusion prevention*. The NBA system, as described, processes a copy of the network traffic. It is not an in-line device so it cannot directly block malicious connections. So the only way to mitigate detected attacks is to update access lists of devices that can block traffic (e. g., firewalls and routers with access control lists).

2.2 Dictionary attack detection

Currently, dictionary attack detection and prevention is done at host level. Besides common host-based intrusion detection systems, there are other applications available (mainly for Unix operating systems) that parse application log files and detect dictionary attacks against network services (such as *fail2ban*[25] that provides even prevention capabilities; *LogSurfer*[26] and *logwatch*[27] with specific rules). The attack is reported if a number of unsuccessful login attempts exceeds a predefined threshold. However, this host-based detection does not scale well in large networks (maintenance costs grow linearly with the number of hosts). Next, it cannot detect distributed and stealthy attacks (the number of attempts from one attacking

[23] `http://www.nanog.org/meetings/nanog35/presentations/oslebo.pdf`
[24] Computer Security Incident Response Team
[25] `http://www.fail2ban.org/`
[26] `http://www.crypt.gen.nz/logsurfer/`
[27] `http://www.logwatch.org/`

host does not reach the threshold within the detection time window). Recently, several independent sources reported[28] an increased appearance of distributed attacks against Secure Shell [38] (SSH), one of the most popular network services at present (even among attackers). We believe that this trend will be followed by attacks against other services.

In the following text, we summarize state of the art of the dictionary attack detection and prevention against SSH. Besides SSH, dictionary attacks often aim at web applications, namely their login interface. We noticed that some web applications prevent to break-in attempts by constraints on user passwords (e. g., Twitter does not allow users to choose the most popular weak passwords at registration[29]).

2.2.1 SSH dictionary attack detection

Although there are many techniques for a prevention of dictionary attacks against SSH, sometimes it is impossible or impractical to implement them. It is commonly recommended to disable password authentication (and use host keys) or implement an access control list that limits allowed source IP addresses[30] and/or a connection rate. *Port-knocking* [31] and changing the default SSH listening port can be considered as examples of the "security through obscurity" principle. Public (remote) available blacklists, such as *sshbl*[32], are intended for sharing IP addresses of attackers that appeared in other networks around the world. Their credibility is at least questionable. *DenyHosts*[33] is a popular SSH log analyzer that can block incoming attacks and report attackers to a central blacklist. It can block even attackers that were reported by other DenyHosts instances deployed in other networks. Unfortunately, remote blacklists are generally vulnerable to a remote log injection.

Concerning network-based SSH dictionary detection, these attacks can be revealed by some generic statistical methods of NBA if attacker attempts hundreds of logins and passwords in the detection time window. However, these methods cannot distinguish between successful and unsuccessful attacks due to their generic nature. In a nutshell, we are not aware of any detection method specialized to the dictionary attack detection.

[28] A summary is available at `http://asert.arbornetworks.com/2008/12/distributed-ssh-brute-force-attacks/`

[29] `http://www.whatsmypass.com/370-banned-twitter-passwords`

[30] This is ineffective if the allowed address becomes compromised.

[31] `http://www.portknocking.org/`

[32] `http://www.sshbl.org`

[33] `http://denyhosts.sourceforge.net/`

2.3 Summary

To conclude, the network behaviour analysis (NBA) is a near real-time, passive flow-based intrusion detection at the network and transport layer of the TCP/IP network model. It is capable of detecting active attacks such as (D)DoS, worm spreading, port scanning, dictionary attacks. The flow aggregation lowers the amount of processed data as well as processing time that is crucial for high-speed networks. Acquired and stored flows can be analyzed by various methods including simple specialized heuristics, attack pattern matching and complex generic statistical methods developed for both backbone and local networks.

We described the architecture of a NBA system that consists of four interfacing components. But there are also systems (mainly commercial ones) where is difficult to clearly distinguish separate layers (such as *Arbor PeakFlow*[34] or *Plixer Scrutinizer*[35]). Some parts of NBA are present in *Security Information and Event Managers* (namely, *QRadar SIEM*[36]) as well.

Dictionary attack detection is currently done at host level, if at all. Log analyzers, both generic and specialized on a particular network service, parse log files and count number of attempts from single IP address. If a predefined threshold is exceeded in the defined time window, the attack is reported. These applications detect effectively simple attacks but cannot capture stealthy distributed attacks because the detection is done separately for each source IP address. Network-based dictionary attack detection is still in the early stage, namely we are not aware of any detection method specialized to this type of attacks.

[34] `http://www.arbornetworks.com/`

[35] `http://www.plixer.com/products/netflow-sflow/scrutinizer-netflow-sflow.php`

[36] `http://www.q1labs.com/products/407/qradar-nsm/`

3 Proposed research

The aim of my research is to contribute to flow-based intrusion detection, particularly to the flow data analysis in large and multigigabit networks dealing with a huge amount of data. We identified that although the dictionary attack and its distributed form are frequent types of attacks, there is not any specialized network- or flow-based method available. Hence we focus on this type of attack and will design such detection method, a generic heuristic for dictionary attack detection. In addition, the method should even determine whether the attack was successful or not. This is very important for its putting into daily operation. Attack scenarios and attack typology will be examined as a side effect of the main work.

As opposed to many already published detection methods, this one will be incorporated into existing flow-based intrusion detection system developed at the Masaryk University and used by its Computer Security Incident Response Team. The whole system is being developed and extended step by step by a few PhD and undergraduate students – my research is primarily focused on the flow data analysis. All designed methods and algorithms will be evaluated in real network of Masaryk University, which is an ideal test bed for our purposes: it is a large, heterogeneous and high-speed network with tens of thousands hosts. An evaluated prototype of the dictionary attack detection method will contribute to overall security of the university network by publishing a blacklist of detected attackers. The results will be also presented at international workshops and in peer-reviewed publications.

We have already studied dictionary attacks against one of the most popular network services, Secure Shell, to learn attack scenarios and attackers' behaviour. As a result, we defined [33] and evaluated [34] an SSH dictionary attack pattern. Further, we propose an extension of the pattern to other network services because dictionary attacks against different services have similar characteristics from the view of network flows. Observing a change of the ongoing attack, end host profiling or bidirectional flow analysis [20] could help with distinguishing a successful and unsuccessful attack.

Besides the main objective described above, we will adapt existing network-based intrusion detection methods that inspect packet payload to process network traffic in high-speed networks. We will benefit from traffic aggregation that enables processing in near real-time even in busy networks. We have already developed and published [15] a prototype of a Network Address Translation (NAT) detection[37] as an example.

Our attention will be also paid to lowering a false positive rate of various detection methods by output correlation among them and with other

[37] An unauthorized NAT device can introduce serious security issues such as misuse of the network resources.

information about attacks and attackers, such as honeypots and public blacklists. This is inspired by our experiment described in [34]: we found out that the majority of SSH dictionary attacks were preceded by earlier port scanning. The idea of a reputation system will be investigated in the context of network security and intrusion detection.

3.1 Schedule

My detailed study and research plan follows:

- An extension of the prototype implementation described in [33] by detection of a distributed dictionary attack against SSH. State doctoral exam and defence of this thesis proposal. Spring 2010.
- Design, implementation and evaluation of a detection method aimed at dictionary attacks against web services and applications. Autumn 2010.
- Analysis of similarity of already designed methods for dictionary attack detection and their generalization for other common network services. Spring 2011.
- Design of incorporation other data sources about attacks (such as honeypots and blacklists) to the detection process. Study of reputation systems. Submission of the thesis. Autumn 2011.
- Deployment of network-based dictionary attack detection in the Masaryk University network. The thesis defence. Spring 2012.

References

[1] E. Alata, V. Nicomette, M. Kaaniche, M. Dacier, and M. Herrb. Lessons learned from the deployment of a high-interaction honeypot. In *EDCC '06: Proceedings of the Sixth European Dependable Computing Conference*, pages 39–46, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2648-9. doi: http://dx.doi.org/10.1109/EDCC.2006.17.

[2] E. Boschi, L. Mark, J. Quittek, M. Stiemerling, and P. Aitken. IP Flow Information Export (IPFIX) Implementation Guidelines. RFC 5153 (Informational), April 2008. URL `http://www.ietf.org/rfc/rfc5153.txt`.

[3] Daniela Brauckhoff, Bernhard Tellenbach, Arno Wagner, Martin May, and Anukool Lakhina. Impact of packet sampling on anomaly detection metrics. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 159–164, New York, NY, USA, 2006. ACM. ISBN 1-59593-561-4. doi: http://doi.acm.org/10.1145/1177080.1177101.

[4] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: Identifying Density-Based Local Outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 93–104, 2000.

[5] Jake D. Brutlag. Aberrant Behavior Detection in Time Series for Network Monitoring. In *LISA '00: Proceedings of the 14th USENIX conference on System administration*, pages 139–146, Berkeley, CA, USA, 2000. USENIX Association.

[6] Marco Canini, Damien Fay, David J. Miller, Andrew W. Moore, and Raffaele Bolla. Per Flow Packet Sampling for High-Speed Network Monitoring. In *Proceedings of the First International Conference on Communication Systems and Networks (COMSNETS'09)*, January 2009.

[7] Glenn Carl, George Kesidis, Richard R. Brooks, and Suresh Rai. Denial-of-Service Attack-Detection Techniques. *IEEE Internet Computing*, 10:82–89, 2006. ISSN 1089-7801. doi: http://doi.ieeecomputersociety.org/10.1109/MIC.2006.5.

[8] B. Claise. Cisco Systems NetFlow Services Export Version 9. RFC 3954 (Informational), October 2004. URL `http://www.ietf.org/rfc/rfc3954.txt`.

[9] B. Claise. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. RFC 5101

(Proposed Standard), January 2008. URL http://www.ietf.org/rfc/rfc5101.txt.

[10] H. Debar, D. Curry, and B. Feinstein. The Intrusion Detection Message Exchange Format (IDMEF). RFC 4765 (Experimental), March 2007. URL http://www.ietf.org/rfc/rfc4765.txt.

[11] L. Ertöz, E. Eilertson, A. Lazarevic, P. Tan, V. Kumar, J. Srivastava, and P. Dokas. *Next Generation Data Mining*, chapter The MINDS — Minnesota Intrusion Detection System. MIT Press, Boston, USA, 2004.

[12] Sharon Goldberg and Jennifer Rexford. Security vulnerabilities and solutions for packet sampling. In *Proceedings of the IEEE Sarnoff Symposium*, 2007. Invited paper.

[13] Shane Harris. The Cyberwar Plan. *National Journal*, 2009. Retrieved online December 26, 2009 at http://www.nationaljournal.com/njmagazine/cs_20091114_3145.php.

[14] Wolfgang John and Sven Tafvelin. Heuristics to Classify Internet Backbone Traffic based on Connection Patterns. In *ICOIN '08: Proceedings of the 22nd International Conference on Information Networking*, pages 1–5, 2008.

[15] Vojtech Krmicek, Jan Vykopal, and Radek Krejci. Netflow Based System for NAT Detection. In *Co-Next Student Workshop '09: Proceedings of the 5th international student workshop on Emerging networking experiments and technologies*, pages 23–24, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-751-6. doi: http://doi.acm.org/10.1145/1658997.1659010.

[16] Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffic anomalies. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 219–230, New York, NY, USA, 2004. ACM. ISBN 1-58113-862-8. doi: http://doi.acm.org/10.1145/1015467.1015492.

[17] Anukool Lakhina, Mark Crovella, and Christophe Diot. Mining anomalies using traffic feature distributions. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 217–228, New York, NY, USA, 2005. ACM. ISBN 1-59593-009-4. doi: http://doi.acm.org/10.1145/1080091.1080118.

[18] S. Leinen. Evaluation of Candidate Protocols for IP Flow Information Export (IPFIX). RFC 3955 (Informational), October 2004. URL http://www.ietf.org/rfc/rfc3955.txt.

[19] Jianning Mai, Ashwin Sridharan, Chen nee Chuah, Hui Zang, and Tao Ye. Impact of packet sampling on portscan detection. *IEEE Journal on Selected Areas in Communications*, 24:2285–2298, December 2006. ISSN 0733-8716.

[20] Pavel Minarik, Jan Vykopal, and Vojtech Krmicek. Improving Host Profiling with Bidirectional Flows. In *CSE '09: Proceedings of the 2009 International Conference on Computational Science and Engineering*, pages 231–237, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3823-5. doi: http://dx.doi.org/10.1109/CSE.2009.23.

[21] George Nychis, Vyas Sekar, David G. Andersen, Hyong Kim, and Hui Zhang. An empirical evaluation of entropy-based traffic anomaly detection. In *IMC '08: Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 151–156, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-334-1. doi: http://doi.acm.org/10.1145/1452520.1452539.

[22] P. Phaal, S. Panchen, and N. McKee. InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks. RFC 3176 (Informational), September 2001. URL `http://www.ietf.org/rfc/rfc3176.txt`.

[23] Benny Pinkas and Tomas Sander. Securing passwords against dictionary attacks. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 161–170, New York, NY, USA, 2002. ACM. ISBN 1-58113-612-9. doi: http://doi.acm.org/10.1145/586110.586133.

[24] J. Quittek, T. Zseby, B. Claise, and S. Zander. Requirements for IP Flow Information Export (IPFIX). RFC 3917 (Informational), October 2004. URL `http://www.ietf.org/rfc/rfc3917.txt`.

[25] J. Quittek, S. Bryant, B. Claise, P. Aitken, and J. Meyer. Information Model for IP Flow Information Export. RFC 5102 (Proposed Standard), January 2008. URL `http://www.ietf.org/rfc/rfc5102.txt`.

[26] Martin Rehak, Michal Pechoucek, Karel Bartos, Martin Grill, Pavel Celeda, and Vojtech Krmicek. CAMNEP: An intrusion detection system for high-speed networks. *Progress in Informatics*, (5):65–74, March 2008. ISSN 1349-8614. URL `http://www.nii.ac.jp/pi/n5/5_65.pdf`.

[27] Karen Scarfone and Peter Mell. Guide to Intrusion Detection and Prevention Systems (IDPS), 2007. Recommendations of the National Institute of Standards and Technology. Retrieved online December 26, 2009 at `http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf`.

[28] C. Seifert. Analyzing Malicious SSH Login Attempts, 2006. Retrieved online January 3, 2010 at http://www.securityfocus.com/infocus/1876.

[29] R. Stewart. Stream Control Transmission Protocol. RFC 4960 (Proposed Standard), September 2007. URL http://www.ietf.org/rfc/rfc4960.txt.

[30] J. L. Thames, R. Abler, and D. Keeling. A Distributed Active Response Architecture for Preventing SSH Dictionary Attacks. In *IEEE Southeastcon 2008*, pages 84–89, 2008.

[31] B. Trammell and E. Boschi. Bidirectional Flow Export Using IP Flow Information Export (IPFIX). RFC 5103 (Proposed Standard), January 2008. URL http://www.ietf.org/rfc/rfc5103.txt.

[32] Jan Vykopal and Tomáš Mrázek. Packet Capture Benchmark on 1 GE, 2008. CESNET technical report 22/2008. Retrieved online December 26, 2009 at http://www.cesnet.cz/doc/techzpravy/2008/packet-capture-benchmark/.

[33] Jan Vykopal, Tomas Plesnik, and Pavel Minarik. Network-Based Dictionary Attack Detection. In *International Conference on Future Networks*, pages 23–27, Los Alamitos, CA, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3567-8. doi: http://doi.ieeecomputersociety.org/10.1109/ICFN.2009.36.

[34] Jan Vykopal, Tomáš Plesník, and Pavel Minařík. Validation of the Network-based Dictionary Attack Detection. In *Security and Protection of Information 2009, Proceeding of the Conference*, pages 128–136, Brno, Czech Republic, 2009. University of Defence. ISBN 978-80-7231-641-0.

[35] Arno Wagner and Bernhard Plattner. Entropy Based Worm and Anomaly Detection in Fast IP Networks. In *WETICE '05: Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*, pages 172–177, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2362-5. doi: http://dx.doi.org/10.1109/WETICE.2005.35.

[36] Kuai Xu, Zhi-Li Zhang, and Supratik Bhattacharyya. Profiling internet backbone traffic: behavior models and applications. *SIGCOMM Comput. Commun. Rev.*, 35(4):169–180, 2005. ISSN 0146-4833. doi: http://doi.acm.org/10.1145/1090191.1080112.

[37] Jeff Yan, Alan Blackwell, Ross Anderson, and Alasdair Grant. Password Memorability and Security: Empirical Results. *IEEE Security and Privacy*, 2(5):25–31, 2004. ISSN 1540-7993. doi: http://dx.doi.org/10.1109/MSP.2004.81.

[38] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Authentication Protocol. RFC 4252 (Proposed Standard), January 2006. URL `http://www.ietf.org/rfc/rfc4252.txt`.

[39] Jian Zhang and Andrew W. Moore. Traffic Trace Artifacts due to Monitoring Via Port Mirroring. In *Proceedings of the Fifth IEEE/IFIP E2EMON*, pages 1–8, 2007. ISBN 1-4244-1289-7. doi: http://www.cl.cam.ac.uk/~awm22/publications/zhang2007traffic.pdf.

[40] Pavel Čeleda, Radek Krejčí, Jan Vykopal, and Martin Drašar. Chuck Norris Botnet – in nome di Chuck Norris, 2009. Submitted for publication.

A Results of my study and research

A.1 Publications

- Jan Vykopal, Tomáš Plesník, and Pavel Minařík. Network-based Dictionary Attack Detection. In *Proceedings of International Conference on Future Networks (ICFN 2009)*, pages 23–27, Los Alamitos, CA, USA: IEEE Computer Society, 2009. ISBN 978-0-7695-3567-8.

- Jan Vykopal, Tomáš Plesník, and Pavel Minařík. Validation of the Network-based Dictionary Attack Detection. In *Security and Protection of Information 2009, Proceeding of the Conference*, pages 128–136, Brno, Czech Republic, 2009. University of Defence. ISBN 978-80-7231-641-0.

- Vojtech Krmicek, Jan Vykopal, and Radek Krejci. NetFlow Based System for NAT Detection. In *Co-Next Student Workshop '09: Proceedings of the 5th international student workshop on Emerging networking experiments and technologies*, pages 23–24, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-751-6.

- Pavel Minarik, Jan Vykopal, and Vojtech Krmicek. Improving Host Profiling with Bidirectional Flows. In *CSE '09: Proceedings of the 2009 International Conference on Computational Science and Engineering*, pages 231–237, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3823-5.

- Jan Vykopal. NetFlow, monitorování IP toků a bezpečnost sítě. In *Sborník příspěvků z 35. konference EurOpen.CZ, 4.–7. října 2009*, pages 63–70, Plzeň, Czech Republic, 2009. EurOpen.CZ. ISBN 978-80-86583-17-4. (in Czech)

- Jan Vykopal and Tomáš Mrázek. Packet Capture Benchmark on 1 GE, 2008. CESNET technical report 22/2008. URL: `http://www.cesnet.cz/doc/techzpravy/2008/packet-capture-benchmark/`.

A.2 Presentations

- Network monitoring workshop for GN3/NA3/T4, October 21st, 2009, Belgrade, Serbia.

- TF-CSIRT meeting, September 25th, 2009, Tallinn, Estonia.

- Several presentations at Faculty of Informatics, Masaryk University, Brno, (*Seminar on Informatics*, *Postgraduate seminar on IT security and cryptography* and *English for Academic Purposes*) and *Liberouter* (CESNET) workshops.

A.3 Participation in projects

- **Security of Czech army information and communication systems – On-line monitoring, Visualization and Packet Filtration. Computer Incident Response Capability Development in the Cyber Defence Environment.** Researcher. Grant no. OVMASUN200801, 2008–2012.
- **CSIRT-MU**. Leader and founder of Computer Security and Incident Response Team at the Masaryk University.
- **Liberouter**, CESNET. Optical National Research Network and Its New Applications. Testing group leader. Grant no. MSM6383917201, 2004–2010.
- **Intelligent Logging Server**, CESNET. Investigator and researcher. Grant no. 291R1/2009, 2009–2010.

A.4 Teaching, supervising and reviewing

During the autumn of 2009 I taught an elective course called *PV210 Security analysis of network traffic*. Next, I supervised two and reviewed four bachelor theses.

A.5 Passed courses

- PA168 Postgraduate Seminar on IT Security and Cryptography
- VV041 English for Academic Purposes
- VV043 Academic Writing in English
- IA067 Informatics Colloquium
- IA068 Seminar on Informatics
- VV045 Photography III

B Reprints of selected papers with original results

- Jan Vykopal, Tomáš Plesník, and Pavel Minařík. Network-based Dictionary Attack Detection. In *Proceedings of International Conference on Future Networks (ICFN 2009)*, pages 23–27, Los Alamitos, CA, USA: IEEE Computer Society, 2009. ISBN 978-0-7695-3567-8.

- Jan Vykopal, Tomáš Plesník, and Pavel Minařík. Validation of the Network-based Dictionary Attack Detection. In *Security and Protection of Information 2009, Proceeding of the Conference*, pages 128–136, Brno, Czech Republic, 2009. University of Defence. ISBN 978-80-7231-641-0.

- Pavel Minarik, Jan Vykopal, and Vojtech Krmicek. Improving Host Profiling with Bidirectional Flows. In *CSE '09: Proceedings of the 2009 International Conference on Computational Science and Engineering*, pages 231–237, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3823-5.

- Vojtech Krmicek, Jan Vykopal, and Radek Krejci. NetFlow Based System for NAT Detection. In *Co-Next Student Workshop '09: Proceedings of the 5th international student workshop on Emerging networking experiments and technologies*, pages 23–24, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-751-6.