Flow-based Detection of SSH Intrusion Attempts

Laurens Hellemons
University of Twente
P.O. Box 217, 7500 AE Enschede
The Netherlands
l.a.j.hellemons@student.utwente.nl

ABSTRACT

In the face of increasing network growth, security practices require scalable monitoring techniques. Flow analysis, the practice of capturing and analyzing aggregated traffic information, is one such technique. This paper demonstrates the practical viability of using flow analysis for security purposes by implementing and testing an algorithm for the detection and analysis of SSH intrusion attacks.

Keywords

NetFlow, SSH, Intrusion detection

1. INTRODUCTION

In recent years, computer networks have experienced substantial growth in terms of both size and speed. This situation has caused a strain on the capabilities of traditional network security monitoring systems, where hardware requirements for basic monitoring capabilities increase sharply with network size due to the sheer amount of data that must be analyzed. Consequently, research has increased into developing alternative monitoring techniques. One such technique is called flow analysis.

Flow analysis relies on the monitoring and inspection of data flows instead of packets. A flow is a set of packets with certain common properties, such as source and destination IP addresses and ports [2]. By collecting and analyzing only flow information, the amount of data to be processed lies several orders of magnitude below that associated with packet-level analysis, which improves scalability drastically.

Research is now increasingly examining the extent to which flow analysis can be used to supplement or even substitute package-based security methods. One of the open issues in this area has been the actual implementation of a proof-of-concept for an entirely flow-based algorithm that can detect security threats. The main reason for this paper is to address this issue by looking at a common area of concern: host intrusion by Secure SHell (SSH). SSH is a communications protocol used for remote shell access, which potentially allows full control over a machine's resources. For this reason, machines offering SSH access are frequent targets of intrusion attacks.

16th Twente Student Conference on IT January 27th, 2012, Enschede, The Netherlands.

Sperotto *et al.* have already analyzed flow data belonging to malicious SSH traffic and constructed a model which describes the flow characteristics during certain kinds of SSH intrusion attacks [6]. In theory, this model can also be used to detect these attacks. However, it has not yet been used in this way, and as such, it is still unknown whether this kind of flow-based detection is possible in practice. This culminates in the following research question:

Can SSH intrusion attacks be detected and analyzed in practice by solely using flow data?

In order to answer this question we have undertaken the development, implementation and testing of an algorithm for detecting specific kinds of SSH intrusion attacks, based on the theoretical model. The goal, in addition to determining the practical applicability of the algorithm, was to prepare a framework that can be used to test future implementations as well.

The remainder of this paper is structured as follows: Section 2 contains theoretical background on the concepts utilized in the paper. Section 3 discusses the created algorithm, including its theoretical basis and the choices made during development. The implementation of the algorithm is then described in Section 4, with particular attention to its architecture. This implementation was tested on real-world data from the University of Twente (UT) campus network. The results of these tests are laid out in Section 5. Interpretation of and discussion about these results is presented in Section 6. Finally, Section 7 contains the conclusions, as well as the answer to the main research question and opportunities for future work.

2. BACKGROUND

This section will introduce the key concepts used in the rest of the paper, as well as their relation to our research.

2.1 Network flows

Network flows describe network communication in terms of flow records, which contain traffic information aggregated by certain properties, collectively called the key [2]. Most commonly, these properties are the source and destination addresses and ports and the protocol used. A flow record describes the amount of traffic with these common properties that passes through a specific point in a network, as well as when this traffic began and ended.

Various vendors supply equipment capable of generating exporting flow records, often embedded in infrastructure hardware such as routers. These devices collect information about passing traffic, accrue this information into flow records, and then typically export these records to one or more devices, elsewhere in the network, that have been set up to collect and store flow records for later processing and analysis. The industry standard for this technology is

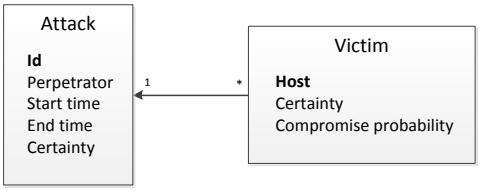

Figure 1. Attack metadata structure

Cisco's NetFlow [2], which is also the technology used in the research underlying this paper.

2.2 Intrusion attacks

An intrusion attack is defined as "'an attempt to gain unauthorized access to a host"' [5]. Many kinds of attacks fall under this umbrella term, but the goal is always to put the attacker in a position to use a target machine's resources for his own purposes. Intrusion attacks are one of the main classes of malicious activity over IP networks and one of the most prominent concerns for Intrusion Detection and Prevention Systems (IDPS) [5].

Intrusion attacks have a number of common properties. These include the host that has executed or is executing the attack (perpetrator), the hosts that are being or have been attacked (victims), the subset of victims to which the perpetrator has actually gained access (compromised victims) and the times at which the attack begins and ends (start and end times, respectively).

2.3 SSH

SSH is a protocol for accessing secure network services over an insecure network infrastructure [9]. The most common service is that of remote login, which enables a user to access a machine over a network as if physically typing on that machine's terminal. The nature of this service makes it highly valuable as a target for malicious activity. If one were able to get into a machine's SSH service, one could potentially gain access to all that machine's resources, including information stored on its hard drive, and also exploit the machine's position in its local network.

Because SSH merely exposes a machine's native shell, a user needs the information of a valid account on the target machine in order to use it.

3. ALGORITHM

As part of our research, we developed an algorithm to test the practical applicability of the SSH intrusion model by Sperotto *et al.* [6]. This section outlines the various aspects of that algorithm.

The algorithm processes flow data and uses it to construct attack metadata in the form of the properties listed in section 2.2. In addition to these properties, the algorithm needs to represent the degree to which each detected event corresponds to an actual attack. As a consequence, the algorithm deals with several certainty levels.

First of all, for every event, there is a level of confidence that what is measured is actually an attack (*attack certainty*). Due to the single source assumption (see Section 3.1), this certainty is coupled to the certainty that the machine suspected of the attack is actually a perpetrator. Until it is sufficiently certain that an actual attack is occurring, that machine is labeled a suspect. Secondly, for every machine communicated with by a suspect or perpetrator, there is a level of certainty as to whether it is being targeted for an intrusion attempt (*victim certainty*). Finally, if a victim is attacked, the attack may be successful or not. The *compromise certainty* of a victim reflects the likelihood that the attack was successful and the perpetrator gained access to that victim. The algorithm keeps track of all certainty levels associated with an attack as percentages.

Table 1. Typical data flow metrics during SSH intrusion attacks

Phase	FPS	PPF	BPP
Scanning	100	1.5	100
Brute force	100	10 - 15	1000
Die-off	20	1	100

The data model of the reports generated by the algorithm is given in Figure 1.

3.1 Theoretical basis

The algorithm is based on the model of SSH brute force intrusion attacks described by Sperotto et al. [6]. This model includes a typical pattern for these kinds of attacks, consisting of three distinct phases.

1. **Scanning phase** - Initially, an attacker initiates many light-weight, short-lived connections to different target machines on the network, in order to find those that are running an SSH service.
2. **Brute force phase** - After collecting the potential victims, the attacker begins using brute force techniques to gain access to some of the targeted machines.
3. **Die-off phase** - Having potentially gained access to some machines, the attacker now runs whatever commmands he wishes on the compromised targets.

The model also describes typical flow metrics associated with these phases. Table 1 lists the average number of flows per second (FPS), the average number of packets per flow (PPF) and the average number of bytes per packet (BPP) measured in each phase. Flows-per-second and packets-per-flow are particularly useful, since they can be easily distinguished from regular network traffic in most situations. It should be noted that the model limits itself to attacks that fit the pattern listed above. Not all SSH intrusion attacks will conform to this. For example, attacks that focus on one predetermined victim will lack a scanning phase. Since the algorithm is based on the model, it is only able to identify attacks fitting it.

The model described in [6] only describes the pattern of flow data that is measured during an intrusion attack. In order to extract an algorithm for detecting and analyzing such attacks, a few further assumptions must be made. First of all, each attack is defined as coming from a single source (*i.e.* host). Allowing for every permutation of coordinated activity between two or more perpetrators would be unfeasible. Additionally, each phase of the attack must occur within reasonable time of the end of the last one. What exactly this reasonable time is will depend on the network or even the notoriety of the suspect, but for practical purposes a threshold must be chosen. Section 5 lists the threshold chosen for the university network tests as part of the implementation parameters. Finally, The algorithm considers only TCP flows going to or coming from the standard SSH port (port 22). Although SSH

services can in principle be set up and accessed via any port , SSH traffic cannot be reliably distinguished in flow data through by other means. Port 22 is reserved for SSH traffic and is by far the port most widely used, as well as targeted by attackers, so this limitation is not a problem in practice.

3.2 Operation

The algorithm structures its tasks around the clear distinction between attack phases. It detects attacks during the first phase and continues to update its analysis while the attack continues

Scanning: Establish suspects
The scanning phase of an attack is very recognizable, and notably distinct from regular traffic on a standard multipurpose network. A large number of small flows between a common machine and many different ones is a telltale sign that a scan is occurring. Once such activity is noted, the machine is registered as a suspect. Note that this can only be done by granting the assumption that the attack originates from a single source.

Brute Force: Confirm attacker, identify victims
If a number of large flows are detected from a host that was recently marked as a suspect, the likelihood that that host is actually perpetrating an intrusion attack increases such that the host is now considered a perpetrator. In addition, the flows can now be used to identify the machines that are being targeted for brute force attacks.

Die-off: Identify compromised victims
If the brute force phase of a confirmed attack is followed by light flows between the perpetrator and one or more of its victims, it is almost certain that those victims have been compromised.

3.3 Parameters

It is important to keep in mind that the algorithm makes use of a number of parameters, such as thresholds. The best values of these parameters can vary with the conditions of the network, or even time. The values chosen for the tests in Section 5 have been derived from empirical data from the UT campus network gathered by Sperotto *et al.* [6].

During the scan phase, the distinguishing attributes of malicious traffic coming from a source are many light flows to many different target addresses. The *Suspect Flow Count Threshold* is the minimum number of flows generated by a single source. The *Suspect PPF Maximum* is the maximum average Packets-per-flow in these flows. The *Suspect FPS Threshold* is the minimum number of flows per second. Flows coming from a single source that fit these parameters will mark that source as a suspect for a certain length of time, the *Suspect Retention Time*.

Once a recognized attack proceeds into the brute force phase, the traffic patterns change. The algorithm keeps track of all suspects' activity and measures whether and when this activity matches brute force patterns in order to confirm the suspect as a perpetrator and identify its victims. The *Brute Force Flow Threshold* is the minimum number of flows occuring from the suspect to any individual target before that target is considered a victim. The *Brute Force PPF Maximum* is the maximum number of packets (on average) in each of those flows.

After the brute force phase, the algorithm watches for the residual activity predicted by the model that indicates a compromise of one or more of the victims. The *Compromise Flow Maximum* is the maximum number of flows communicated from an attacker to a brute force-attacked victim. The *Compromise PPF Threshold* is the minimum average number of packets in each of those flows. If there is any activity between the attacker and a registered victim that meets these criteria, the algorithm marks that victim as compromised.

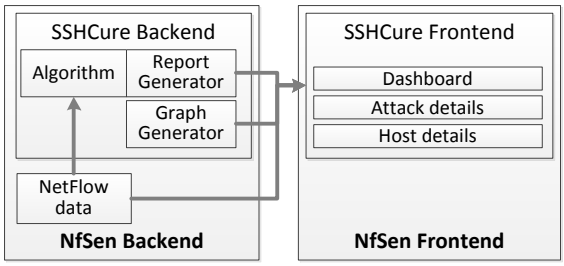

Figure 2. SSHCure architecture

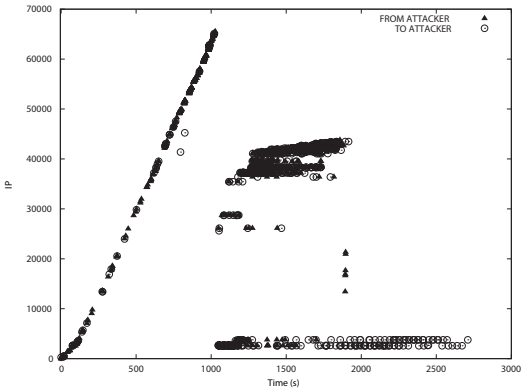

Figure 3. Sample Activity/Time graph, from [6]

.

4. IMPLEMENTATION

This section describes the actual implementation of a piece of software that utilizes the algorithm described in Section 3 to analyze NetFlow data and generate information about detected SSH intrusion attacks. NfSen [3] was chosen as a platform for the implementation of the algorithm. In addition to being the industry standard solution for NetFlow both for its ready infrastructure for developing plugins and the large existing body of knowledge about the package. NfSen includes a flow data collector and processor, and a Web interface for viewing and analyzing this data. NfSen runs a program called a *capture daemon* that captures netflow data sent by specialized collection hardware (or in some cases, software). This data is processed and analyzed by NfSen in 5-minute chunks, and this is also the way in which plugins such as SSHCure access the data.

The algorithm was implemented and developed as a plugin for NfSen named SSHCure (pronounced as *she-cure*) [4]. SSHCure is designed to both generate reports of SSH intrusions and allow navigation and analysis based on these reports.

4.1 Architecture

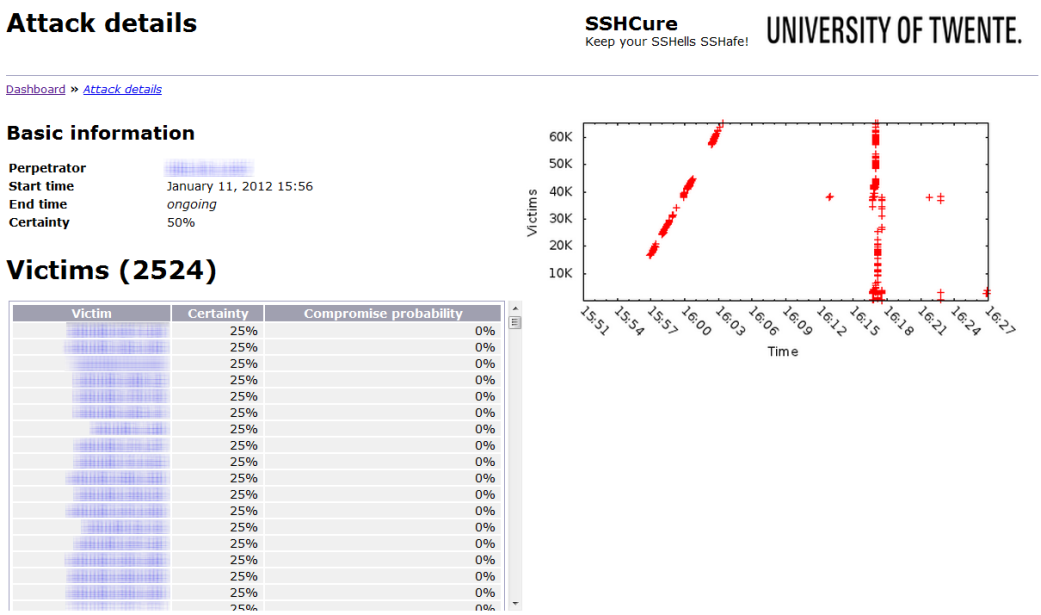

Figure 4. SSHCure frontend attack details page

SSHCure is composed of several distinct functional units. The design of the application is such that it is easy to modify one unit without needing to change any of the others. For example, the algorithm could be upgraded to deliver more accurate confidence estimates or to detect other types of attacks (Section 7 elaborates on this). As per requirements for NfSen, SSHCure's frontend is written in PHP-generated HTML, JavaScript and CSS, while the backend is written in Perl. The architecture is shown in Figure 2. This figure shows that the Graph- and Report Generators, implemented in the SSHCure backend, use flow data provided via NfSen and the algorithm. The frontend then uses these reports and graphs to allow browsing and inspection of attacks by users.

The core of SSHCure is the use of the algorithm to generate attack reports. These reports include all the information described in Section 3. The *Report Generator* works by analyzing incoming netflow data in intervals of 5 minutes, keeping track of suspects and ongoing attacks between cycles. The reports are written to an SQLite database [8] whose structure is modeled after the report format given in Figure **??**. While an ongoing attack is being analyzed, the report of that attack can change. As the generator receives new information, confidence levels may increase, new victims may be identified or victims may be determined to have been compromised. Once an attack is finished, the report is assigned an end time and will not change any further.

The ability to do further investigation based on the information the algorithm produces is key to the interpretation of test results for answering our primary research question. SSHCure includes the *Graph Generator*, which generates two types of graphs associated with attack reports. The most important graph type is the activity/time graph, which shows the victim IP addresses and times of each flow associated with an attack. This type of graph provides visual confirmation as well as insight into the attack. Figure 3 shows an activity/time-graph for a typical attack captured by SSHCure. The three phases of the attack are clearly visible in in the graph: The rising line in the left portion shows connections between the perpetrator and many target machines during the scanning phase. The next part shows the long, heavy flows to a few targets associated with the brute force phase. The last part has regular flows to a subsection of the brute force victims, indicating that they have been compromised during the die-off phase.

The SSHCure interface uses both the generated reports and graphs to allow users to navigate, inspect, analyze and investigate attack occurrences. It is accessed through a web interface that is incorporated into the main NfSen web page. The main page, or *Dashboard*, shows an overview of detected attacks that can be navigated through time. It also shows the most prolific perpetrators and victims for the currently selected time. By clicking on a specific attack the user can open the attack details page for that attack. The *attack details* page (Figure 4) shows the activity/time graph for that attack, as well as the metrics and information about the associated perpetrator and victims (including confidence levels). The page can also load the flow data relevant to the attack, enabling the user to do further investigation. The *host details* page shows attack-related data for a host, which can be either a victim or a perpetrator (or, in very rare cases, both). It shows the events in which the host was involved and statistical information about these events such as the number of times it was attacked or compromised.

5. TESTS

To test the viability of the algorithm, it was used to analyze NetFlow data collected from the UT campus network. Aim of the tests was to show that the algorithm is able to detect SSH intrusion attacks in practice. An additional but non-essential goal was to determine the algorithm's success rate (the chance that an attack is reported by the algorithm, i.e. a true positive) and error rate (the chance that a reported attack is not actually an attack, *i.e.* a false negative).

The data set used to feed the algorithm spans one week,

Table 2. SSHCure input parameters for UT campus network

Parameter	Value
Suspect Flow Count Threshold	1000 flows
Suspect PPF Maximum	3 packets / flow
Suspect FPS Threshold	3 flows / second
Suspect Retention Time	10 minutes
Brute Force Flow Threshold	100 flows
Brute Force PPF Threshold	15 packets / flow
Compromise Flow Maximum	300 flows
Compromise PPF Threshold	15 packets / flow

Table 3. Reports generated from UT campus network

Date	Crt	Vics	Cmps	TP
Jan 11 20:04	25%	999	0	Yes
Jan 11 21:40	40%	1	0	No
Jan 11 22:08	25%	1000	0	Yes
Jan 11 23:02	25%	1004	0	Yes
Jan 12 01:07	25%	2000	0	Yes
Jan 12 07:40	25%	1593	0	Yes
Jan 12 08:42	25%	1260	0	Yes
Jan 12 10:04	25%	2673	0	Yes
Jan 12 10:22	25%	1000	0	Yes
Jan 12 12:50	40%	1	0	No
Jan 12 13:11	25%	869	0	Yes
Jan 12 13:16	25%	19	0	No
Jan 12 13:23	25%	1000	0	Yes
Jan 12 15:40	25%	1000	0	Yes
Jan 12 20:55	25%	1394	0	Yes
Jan 13 02:38	25%	999	0	Yes
Jan 13 02:44	25%	999	0	Yes
Jan 13 02:38	25%	999	0	Yes
Jan 13 14:16	25%	1288	0	Yes
Jan 13 17:17	25%	1000	0	Yes
Jan 14 10:16	25%	999	0	Yes
Jan 14 14:39	25%	24	0	No
Jan 14 21:06	25%	1461	0	Yes
Jan 14 23:19	25%	856	0	Yes*
Jan 14 23:24	50%	128	0	Yes*
Jan 14 23:29	25%	397	0	Yes*
Jan 14 23:34	25%	311	0	Yes*
Jan 14 23:59	25%	265	0	Yes*
Jan 15 06:05	25%	1612	0	Yes
Jan 15 09:27	25%	1001	0	Yes
Jan 15 10:20	25%	797	0	Yes
Jan 15 14:50	40%	2	0	No
Jan 15 14:54	25%	1004	0	Yes
Jan 15 15:10	25%	998	0	Yes
Jan 15 18:29	25%	1291	0	Yes
Jan 15 19:50	25%	999	0	Yes
Jan 16 10:42	25%	1000	0	Yes
Jan 16 10:55	25%	1762	0	Yes
Jan 16 11:20	25%	1479	0	Yes

Table 4. Aggregated test results

Total number of attacks	42
Captured attacks	31 (42*)
False positives	5
False negatives	11 (0*)
Success rate	74% (100%*)
Error rate	84%

from Monday, January 9 until Monday, January 16, 2012. Conditions on the campus network are such that average SSH intrusion attack frequency is at least 1 per day [6]. Therefore, the data set was likely to contain many attack instances. Preliminary runs of the algorithm were used to establish the parameter values best suited to the conditions on the campus network. The parameters used in the final tests are listed in Table 2. Refer back to Section 3.3 for the precise role of each parameter.

The tests were conducted as follows: The algorithm was run on the data set, resulting in a list of reports. The flow data associated with each report was then analyzed manually by looking for the patterns laid out in the model by Sperotto *et al.* [6]. In this way it was determined whether each report concerned a true positive by the standards of the model. To determine the success rate, the total number of attacks occurring during the span of the data set was measured.

The reports generated from the data set are listed in Table 3. Each report includes the date and time of the attack, the attack certainty given by the algorithm (Crt), the number of victims (Vics), the number of compromises (Cmps), and whether the attack was found to be a true positive (TP). Statistical results are listed in Table 4.

There are some notes to the report data. The attacks captured between 23:19 and 23:59 on January 14 are actually part of the same attack. These have been counted as such in the statistics. On several other occasions, attacks originating from the same IP address have been erroneously combined into one attack. Section 6 will elaborate on these facts. They have also been accounted for in the statistics. Statistics as they would be if the attacks had been logged seperated have been included in Table 4 followed by asterisks.

6. DISCUSSION

This section contains our interpretation of the test results and relevant discussion points to be drawn from these.

6.1 Algorithm

The algorithm used in this research, while robust and grounded in thorough research, has some limitations. Another algorithm could be built using techniques to translate the developed Hidden Markov Model directly. Such techniques are already being applied [1]. Doing this would result in an algorithm that more directly fits the model, and consequently may be able to detect attacks earlier or more reliably than our current algorithm. Note that any algorithm based on the model must neccesarily be extended with the ability to identify the hosts involved in an attance, since the model merely describes traffic patters.

In the course of implementing the algorithm, a few considerations presented themselves, forcing us to make some pragmatic decisions.

Sampling

In theory, every network connection is captured and aggregated in the NetFlow data fed into the algorithm. However, not all collector equipment will actually do this. To prevent hardware capacity limits being reached, some collectors will apply 'packet sampling': selecting some of the collected packets to be included in the flow data and discarding others. Typical sampling rates are 1:10, 1:100 or even 1:1000. This has serious consequences for the algorithm, especially during the scanning phase. Scanning flows average 1.5 packets per flow (see Table 1), which means that a single packet omitted from the flow data will

result in the flow not being recorded. As a consequence, depending on the sampling rate, the algorithm may not be able to detect the characteristic pattern of the scanning phase, and entire attacks may go unnoticed. This is a limitation of the input to the algorithm, not the algorithm itself. However, given more time and testing opportunities, it should be possible to modify the algorithm to deal with these situations. Extrapolations could be made to detect the scanning pattern from partial records.

In our testing environment, sampling was not used. Therefore this consideration has no impact on the test results.

Resource constraints
Due to the design of the NfSen plugin mechanism, plugins are allocated limited amounts of memory, and a limited execution time. Consequently, the algorithm can not guarantee detection of all victims for each attack. Priority is given to hosts that are targeted for the brute force phase over those that are merely scanned. Although victim identification is not a primary goal for our implementation, such a feature is desirable in actual network monitoring applications and its inclusion should be investigated further.

6.2 Test results

Investigation of the flow data associated with the test results clearly shows that the algorithm detects actual malicious activity that conforms to the characteristics of SSH intrusion attacks outlined in the model. Additionally, there are some remarks to be made about the results. First of all, the captured attacks average to 1123 scanned victims per attack, but almost no compromises are detected. Since we can be certain that all flow records are fed into the algorithm, there are two possible causes for this finding: Either there were no successful intrusions during the testing period at all, which is very possible, or there is a flaw in either the testing parameters or the reasoning behind the algorithm on this point.

Another remark is that the algorithm has some flaws at the implementation level. For example, at least one attack was detected more than once (the attacks from January 14, 23:19 until 23:59 in Table 3), while several other attacks with the same perpetrating host were incorrectly identified as part of the same event. We find that this is due to a problem with the suspect retention mechanism within the algorithm implementation, which can be corrected. The effect on the statistics is an artificial decrease of the success rate, since some attacks were detected but not logged as separate events). There is no effect on the error rate, since all incidents concern true positives.

Lastly, we are able to say something about the way the characteristics of SSH intrusion attacks have evolved in the past years. Sperotto *et al.* predicted that, due to both increasing network speeds and evolving tools, the time scale of attacks would be compressed [6] [7]. In fact, we find that this is true for some of our captured attacks, while others show gaps between phases. We cannot say yet whether this is due to human intervention by the attacking party, a consequence of infrastructure and end-point configuration, or another cause or combination of causes. The algorithm captures these attacks regardless of these variations, so further study is easily possible. Another prediction by Sperotto *et al.* was that attack frequency would increase. We also find this to be fulfilled, increasing from a measured frequency of about 1 attack per day in late 2008 to about 6 per day in early 2012. Seasonal influences may account for some of this difference.

7. CONCLUSIONS

Based on the test results we can confidently state that the algorithm is able to identify SSH intrusion attacks with a high degree of certainty and a low degree of error. The main research question,

Can SSH intrusion attacks be detected and analyzed in practice by solely using flow data?

is therefore answered affirmatively. Furthermore, the algorithm demonstrates that more specific conclusions, such as the identity of an attacker, the identities of its victims and even which hosts have likely been compromised, can still be drawn using nothing but flow data. This fact further reduces the need for deep packet inspection in security systems, allowing for more scalable IDPS solutions.

Now that the practical applicability of SSH intrusion detection has been demonstrated, research should be directed towards expanding the scope of this detection. It remains an open question whether other types of intrusion attacks, for example using FTP or RDP, could be detected using NetFlow data. Other open questions are the detectability through NetFlow data of attacks using other methods than brute force.

Since the main reason for applying flow data analysis to network security is scalability, it would be worthwhile to investigate the ways in which flow data analysis could be combined with other detection methods to create security monitoring systems that are scalable while maintaining the ability to detect many types of attacks reliably.

Acknowledgements

Special thanks to Luuk Hendriks for his extraordinary contributions to the design and development of SSHCure, and to Rick Hofstede for his invaluable advice and feedback.

8. REFERENCES

[1] Y.-S. Chen and Y.-M. Chen. Combining incremental hidden Markov model and Adaboost algorithm for anomaly intrusion detection. pages 3–9, 2009.

[2] B. Claise. Cisco Systems NetFlow Services Export Version 3 (RFC3954), October 2004.

[3] P. Haag. NfSen - Netflow Sensor. http://nfsen.sourceforge.net, September 2011.

[4] L. Hellemons and L. Hendriks. SSHCure - SSH intrusion detector. http://sshcure.sourceforge.net, January 2012.

[5] K. Scarfone and P. Mell. Guide to Intrusion Detection and Prevention Systems (IDPS). http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf, February 2007.

[6] A. Sperotto, R. Sadre, P.-T. De Boer, and A. Pras. Hidden Markov Model Modeling of SSH Brute-force Attacks. In *Proceedings of the 20th IEEE/IFIP International Workshop on Distributed Systems: Operation and Management (DSOM 09)*, volume 5841, pages 164–176, 2009.

[7] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller. An Overview of IP Flow-based Intrusion Detection. volume 12, pages 343–356, 2010.

[8] SQLite Consortium. SQLite. http://www.sqlite.org, January 2012.

[9] T. Ylonen. The Secure Shell (SSH) Authentication Protocol, January 2006.