Flow Based Security Awareness Framework for High-Speed Networks

Pavel Čeleda	Martin Rehák	Vojtěch Krmíček	Karel Bartoš
celeda@ics.muni.cz	mrehak@labe.felk.cvut.cz	vojtec@ics.muni.cz	bartosk@labe.felk.cvut.cz
Institute of Computer Science, Masaryk University, Brno Czech Republic	Department of Cybernetics, Czech Technical University in Prague, Czech Republic	Institute of Computer Science, Masaryk University, Brno Czech Republic	Department of Cybernetics, Czech Technical University in Prague, Czech Republic

Abstract

It is a difficult task for network administrators and security engineers to ensure network security awareness in the daily barrage of network scans, spaming hosts, zero-day attacks and malicious network users hidden in huge traffic volumes crossing the internet. Advanced surveillance techniques are necessary to provide near real-time awareness of threads, external/internal attacks and system misuse.

Our paper describes security awareness framework targeted for high-speed networks. It addresses the issues in detailed network observation and how to get information about who communicates with whom, when, how long, how often, using what protocol and service and also how much data was transferred. To preserve users' privacy while identifying anomalous behaviors we use the NetFlow statistics. Specialized standard and hardware-accelerated flow monitoring probes are used to generate unsampled flow data from observed networks.

We use several anomaly detection algorithms based on network behavioral analysis to classify legitimate and malicious traffic. Using network behavioral analysis in comparison with signature based methods allows us to recognize unknown or zero-day attacks. Advanced agent-based trust modeling techniques estimate the trustfulness of observed flows. The system operates in unsupervised manner and identifies attacks against hosts or networks from the network traffic observation. Using flow statistics allows the system to work even with encrypted traffic.

Incident reporting module aggregates malicious flows into the incidents. The intrusion detection message exchange format or plain text formated messages are used to describe an incident and provide human readable system output. Email event notification can be used to send periodical security reports to tools for security incident reports management.

Presented framework is developed as a research project and deployed on university and backbone networks. Our experiments performed on real network traffic suggest that the framework significantly improves the error rate of false positives while being computationally efficient, and is able to process the network speeds up to 1 Gbps in online mode.

Keywords: intrusion detection, network behavior analysis, anomaly detection, NetFlow, CAMNEP, FlowMon, Conficker.

1 Introduction

Ensuring security awareness in the high-speed networks is a human intensive task in these days. To perform such surveillance, we need a highly experienced network specialist, who has a deep insight to the netwok behavior and perfectly understands the network states and conditions. His usual work procedures consist of observing the traffic statistic graphs, looking for unusual peaks in volumes of transferred bytes or packets and consequently examining particular suspect incidents using tools like packet analyzers, flow collectors, firewall and system logs viewers etc. Such in-depth traffic analysis of particular packets and flows is a time consuming and requires excellent knowledge of the network behavior.

The presented framework introduces a new concept which dramatically reduces the requirements for network operator experiences and overtakes long-term network surveillance. The system operator doesn't need to constantly observe network behavior, but can be focused on the security incident response and resolution. The system can report either classified incidents or all untrustfull traffic. Consequently, the system operator receives reports about security incidents and examines them - checks, if their are not false positives and in case of need he performs further necessary actions.

The paper is organized as follows: After a short system overview, we present our FlowMon based traffic monitoring platform including generation and collection of NetFlow data. Then we describe CAMNEP framework and the reduction of false positives using trust modeling techniques. Finally, we conclude with real life example discussing detection of Conficker worm. We were not able due to the limited number of pages discuss all parts in deep detail. More detailed description of used methods and algorithms is available in our previous publications [10, 13].

2 System Architecture

The architecture of our system has four layers, which are typically distributed and where each layer processes the output of the layer underneath, as shown in Figure 1.

At the bottom of the whole architecture, there is one or more standard or hardware-accelerated FlowMon probes [13] generating NetFlow data and exporting them to the collector. The open-source NfSen [4] and NFDUMP [3] tools are used to handle NetFlow data. These tools also provide storage and retrieval of the traffic information for further forensics analysis. TASD (Traffic Acquisition Server Daemon) is an application managing communication between acquisition layer and agent layer via TASI protocol. TASD reads NetFlow data, performs preprocessing to extract aggregated statistics, estimates entropies and sends these data to the upper layer.

The preprocessed data is then used for cooperative threat detection, performed in the third layer called CAMNEP (Cooperative Adaptive Mechanism for Network Protection). Suspicious flows and network anomalies are either visualized in graphical user interface or passed to the top level layer, which is responsible for visualization and interaction with network operator. Email reports and event notifications are send to standard mailbox or to ticket request system e.g. OTRS (Open source Ticket Request System).

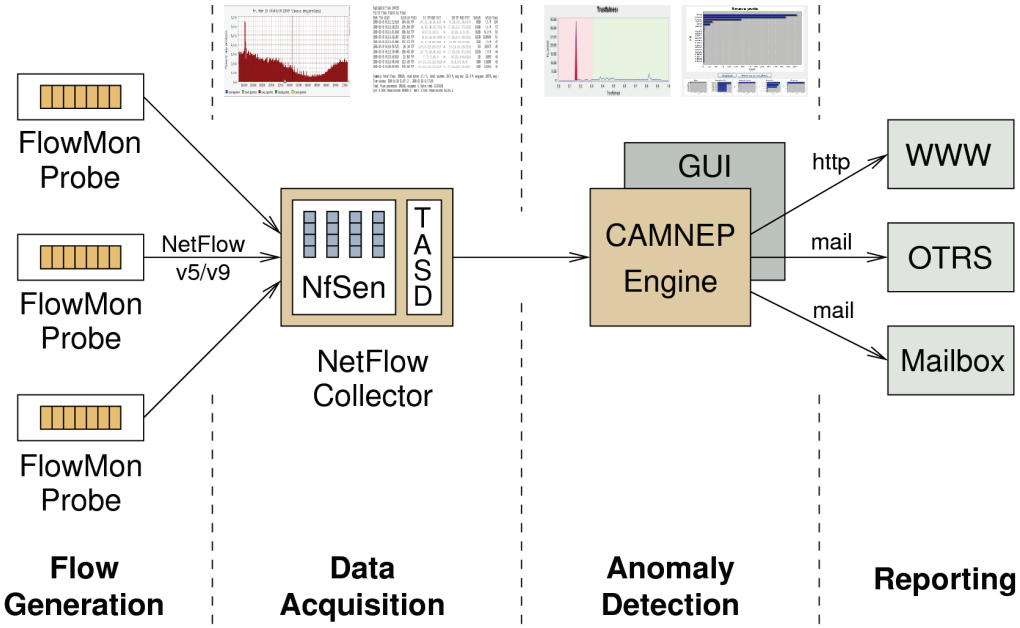

Figure 1: System block structure.

3 Flow Based Network Traffic Awareness

To be able to provide permanent network situational awareness we need to acquire detailed traffic statistics. Such statistics can be complete packet traces, flow statistics or volume statistics. To efficiently handle high-speed traffic the trade-off between computational feasibility and provided level of information must be chosen.

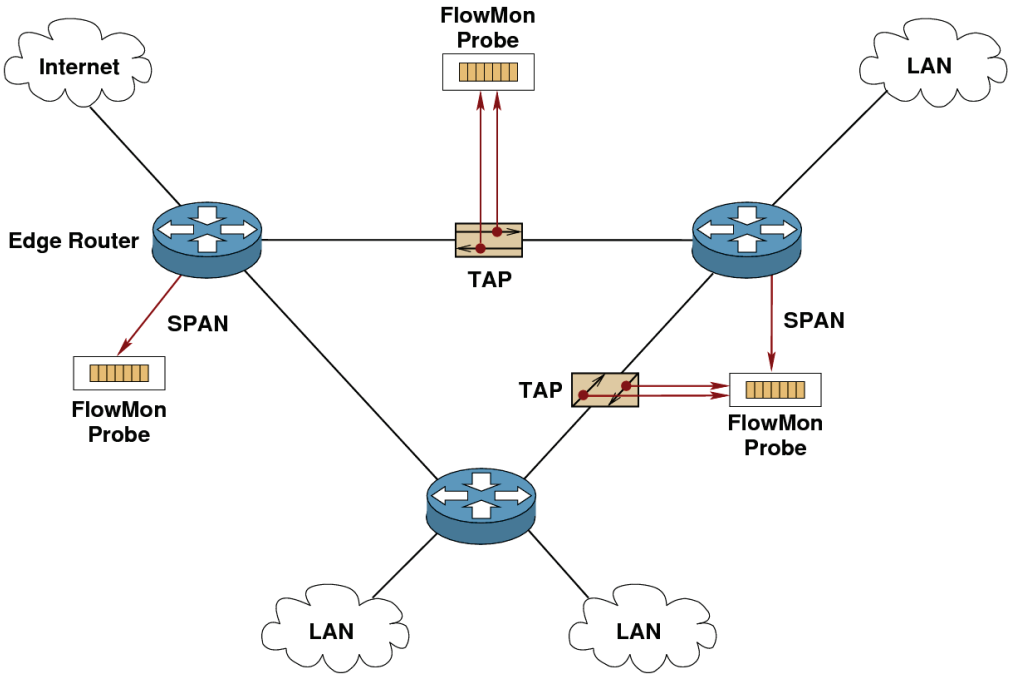

Figure 2: Traffic monitoring system deployment in operation network.

- Full packet traces traditionally used by traffic analyzers provide most detailed information. On the other hand the scalability and processing feasibility for permanent traffic observation and storing in high-speed networks is problematic including high operational costs.
- Flow based statistics provide information from IP headers. They don't include any payload information but we still know from IP point of view who communicates with whom, which time etc. Such approach can reduce up to 1000 times the amount of data necessary to process and store.
- Volume statistics are often easy to obtain in form of SNMP data. They provide less detailed network view in comparison with flow statistics or full packet traces and doesn't allow advanced traffic analysis.

In our work we have decided to use NetFlow data for their scalability and ability to provide sufficient amount of information. NetFlow initially available in CISCO routers is now used in various flow enabled devices (routers, probes). Flow based monitoring allows us to permanently observe from small end-user network up to large NREN (National Research and Education Network) backone links.

In general, flows are a set of packets which share a common property. The most important such properties are the flow's endpoints. The simplest type of flow is a 5-tuple, with all its packets having the same source and destination IP addresses, port numbers and protocol. Flows are unidirectional and all their packets travel in the same direction. A flow begins when its first packet is observed. A flow ends when no new traffic for existing flow is observed (inactive timeout) or connection terminates (e.g. TCP connection is closed). An active timeout is time period after which data about an ongoing flow are exported. Statistics on IP traffic flows provide information about who communicates with whom, when, how long, using what protocol and service and also how much data was transferred.

To acquire NetFlow statistics routers or dedicated probes can be used. Currently not all routers support flow generation. Enabling flow generation can consume up to 30 - 40 % of the router performance with possible impacts on the network behavior. On the other hand dedicated flow probes observe the traffic in passive manner and the network functionality is not affected.

In our system we use FlowMon probes [13]. The FlowMon probe is preferred one due to implemented features which contain support for NetFlow v5/v9 and IPFIX standard, packet/flow sampling, active/inactive timeouts, flow filtering, data anonymization etc. The probe firmware and software can be modified to add support for other advanced features. Hardware-accelerated probes support line-rate traffic processing without packet loss. Standard probes are based on commodity hardware with lower performance. The FlowMon probe was developed in Liberouter project and is now maintained by INVEA-TECH company.

To provide input for probes TAP (Test Access Port) devices or SPAN (Switched Port Analyzer) ports can be used. TAP devices are non-obtrusive and are not detectable on the network. They send a copy (1:1) of all network packets to a probe. In case of failure the TAP has built-in fail-over mode. The observed line will not be interrupted and will stay operational independent and any potential probe failure. Such approach enables us to deploy monitoring devices in environments with high reliability requirements.

SPAN (port mirroring) functionality must be enabled on router/switch side to forward network traffic to monitoring device. It's not necessary to introduce additional hardware in network infrastructure but we need to reconfigure the router/switch and take in count some SPAN port limits. Detailed comparison between using TAP devices or SPAN ports is described in [14].

4 Agent-Based Anomaly Detection Layer

The anomaly detection layer uses several *network behavior analysis* [12] (NBA) algorithms embedded within autonomous, dynamically created and managed agents. Each *detection agent* includes one anomaly detection method and a trust model, a knowledge structure that aggregates a long-term experience with specific traffic types distinguished by agent.

Anomaly detection (AD) paradigm is appropriate for NetFlow data processing due to its nature and due to the relative effective low-dimensionality of network traffic characteristics, either in the volume [5] or parameter distribution characteristics [6]. The anomaly detection methods use the history of traffic observations to build the model of selected relevant characteristics of network behavior, predict these characteristics for the future traffic and identify the source of discrepancy between the predicted and actually observed values as a possible attack. The main advantage of this approach is its ability to detect the attacks that are difficult to detect using by the classic IDS systems based on the identification of known malicious patterns in the content of the packets, such as zero-day exploits, self-modifying malware, attacks in the ciphered traffic or resource misuse or misconfiguration.

The use of the collaborative agent framework helps us to address the biggest problem of the anomaly-based intrusion detection systems, their error rate, which consists of two related error types. *False Positives* (FP) are the legitimate flows classified as anomalous, while the *False Negatives* (FN) are the malicious flows classified as normal. Most standalone anomaly detection/NBA methods suffer from a very high rate of false positives, which makes them unpractical for deployment. The multi-stage collaborative process of the CAMNEP system removes a large part of false positives, while not increasing the rate of false negatives, and deploys a range of self-optimization and self-monitoring techniques to dynamically measure and optimize the system performance against a wide range of known attack techniques.

The collaborative algorithm, which has been described in [11] is based on the assumed independence of false positives returned by the individual anomaly detection algorithms. In the first stage of the algorithm, each of the agents executes its anomaly detection algorithm on the flow set observed during the last observation period, typically a 5 minute interval. The AD algorithms update their internal state and return the anomaly value for each of the flows in the observed batch. These anomaly values are exchanged between the agents, so that they all can build the identical input data for the second stage of processing, when they update their trust models.

The trust models of each agent cluster the traffic according to the characteristics used by the anomaly detection model of the agent. This results in a different composition of clusters between the agents, as any two flows that may be similar according to the characteristics used by one agent can be very different to other agent's models. Once the agents build the characteristic profiles of behavior, they use the anomaly values provided by the anomaly detection methods of all agents to progressively determine the appropriate level of trustfulness of each cluster. This value, accumulated over time, is then used as an individual agent's assessment of each flow.

In the last stage of the algorithm, the dedicated aggregation agents aggregate the trustfulness values provided by the detection agents using either predetermined aggregation function, or a dynamically constructed and selected aggregation function based on the self-monitoring meta-process results. The final assignment of the final anomaly/trustfulness values positions the flows on the [0,1] interval and allows the user to visualize the status of the network during the evaluated time period.

The flows evaluated as untrusted or suspicious (i.e. being under dynamically determined thresholds on the on the trustfulness interval) are analyzed to extract meaningful events that are then classified to several generic and user defined categories of malicious behavior before being reported to the user.

The deployment of this relatively complex algorithm is straightforward, as it is able to autonomously adapt to the network it is deployed on and to change the algorithm properties to match the dynamic

nature of network traffic. On the lower level, the algorithm is based on the assumption of independence [1] between the results provided by the different agents. This reduces (in several stages described above) the number of false positives roughly by the factor of 20 and more with respect to the average standalone AD method, making the AD mechanism fit for operational deployment.

5 CAMNEP's Contribution To Security Awareness

The CAMNEP system provides a wide spectrum of tools that facilitate the network surveillance. It can be run in two basic modes - a server side mode with no graphical interface, providing only reporting capabilities and a GUI mode with full graphical interface providing sophisticated tools for interactive network analysis based on the system outputs.

The **server side mode** has several possibilities of incident reporting. The network administrator can define the format of the report (short text, full text, XML, flows list), the receivers of the report (email addresses or disk folders), intervals of generating these reports, a level of details of reports etc. The system also provides basic web interface, where the incident reports can be accessed remotely by web browser. In this mode the network operator just regularly checks the security incident mailbox or ticket system connected to the CAMNEP and consequently performs further analysis of reported security incidents using third party tools or the CAMNEP run in the GUI mode.

The **GUI mode** provides full graphical interface with advanced tools for detailed incident analysis on the flow level. These tools include the trustfulness-based histogram of the current traffic, ability to select and analyze a subset of traffic, in order to display various traffic characteristics such as significant source and destination IP addresses, ports, protocols, entropies, aggregations etc. The network operator can also perform DNS name resolution, ping and whois requests and other, user defined actions directly from the graphical interface. The interface also allows an experienced user to check particular anomaly detection methods behavior, set the weights of anomaly detection methods in the trust model, display the graphical representation of the trust models etc.

The integration of several anomaly detection methods into the CAMNEP system allows to detect a wide scale of anomalous network traffic and network attacks. It includes preliminary intruders activities during reconnaissance phase like *fingerprinting*, *vertical scans* of the particular machines and *horizontal scans* of the whole networks. In the case of more serious network threats the system detects *ssh brute force attacks* and other brute force attacks on passwords, can reveal *botnet nodes*, *worm/malware spreading, Denial of Service* and *Distributed Denial of Service attacks* and other various classes of network attacks specified by CAMNEP operator. It is particularly relevant as an extrusion detection tool that allows the administrators to detect the misuse of their network assets against the third parties, typically as a part of a botnet.

The network administrator can configure the reporting from the system on a graduated level of severity, based on the incident degree of trust/anomaly and the attack class it is attributed to. False positives can be ignored, irrelevant attacks logged, more sever incidents can generate tickets in the event management system and the most important events can be pushed directly to the administrator via email. Due to its nature, the system is able to detect new attack variants, that may not fall into any of the predefined classes. Such incidents fall into the default incident category and the administrators can investigate them, typically when the affected hosts start to misbehave in the future.

6 Real World Example - Conficker Worm Detection

In this part we will describe the use case demonstrating the framework capabilities from the user (e.g. network administrator or incident analyst) perspective at real world example. In the previous papers we have presented the possibilities of the CAMNEP system to detect horizontal and vertical scans [7, 8], here we will focus on the description of the *Conficker worm* [9] and how it was detected by our system.

This experiment was performed on real network data acquired from academic network links where the system is deployed to observe both incoming and outgoing traffic.

Conficker, also known as *Downup*, *Downadup* and *Kido*, is a computer worm that surfaced in October 2008 and targets the Microsoft Windows operating system. It exploits a known vulnerability in Microsoft Windows local networking, using a specially crafted remote procedure call (RPC) over port 445/TCP, which can cause the execution of an arbitrary code segment without authentication. It was reported that Conficker had infected almost 9 million PCs [2] at the time of this writing (March 2009). Its authors also proactively modify the worm code and release new variants, in order to protect the botnet from the orchestrated international response against its command and control infrastructure.

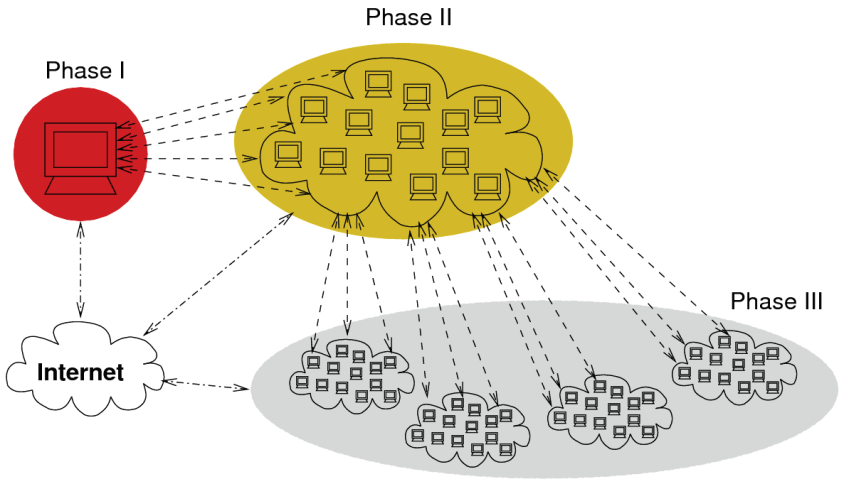

Figure 3: Conficker worm spreading activity in monitored network.

Figure 3 illustrates the Conficker worm propagation inside the university network. An initial victim was infected during *phase I*. The main phase - *phase II* - started at 9:55:42 with massive scanning activity against computers both in local network and internet with the goal to discover and infect other vulnerable hosts (destination port 445 - targeting Microsoft security issue described above). One hour later, a lot of university computers are infected and again try to scan and propagate (*phase III*) to further computers both in university network and internet.

7 Traditional NetFlow Analysis Using NFDUMP Tool

In the following text, we will show how the worm progressed in the campus network and propagated beyond from the infected machines. To protect user privacy all IP addresses and domain names are changed (anonymized). The infected host *172.16.96.48 - victim.faculty.muni.cz* inside the university network started to communicate at 9:41:12, 11.2.2009 :

Flow start	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Packets	Bytes	Flows
09:41:12.024	0.307	UDP	172.16.96.48:49417	->	224.0.0.252:5355	.....	2	102	1
09:41:12.537	0.109	UDP	172.16.96.48:60435	->	224.0.0.252:5355	.....	2	102	1
09:41:14.446	30.150	ICMP	172.16.92.1:0	->	172.16.96.48:3.10	.....	25	3028	1
09:41:14.446	30.148	UDP	172.16.96.48:137	->	172.16.96.255:137	.....	25	2238	1
09:41:21.692	3.012	UDP	172.16.96.48:60436	->	172.16.92.1:53	.....	2	162	1
09:41:21.692	3.012	UDP	172.16.92.1:53	->	172.16.96.48:60436	.....	2	383	1
09:41:21.763	31.851	UDP	172.16.96.48:5353	->	224.0.0.251:5353	.....	9	867	1
09:41:24.182	20.344	UDP	172.16.96.48:60438	->	239.255.255.250:3702	.....	6	6114	1
09:41:24.470	0.049	UDP	172.16.96.48:138	->	172.16.96.255:138	.....	3	662	1
09:41:26.069	31.846	UDP	172.16.96.48:60443	->	239.255.255.250:1900	.....	14	2254	1
09:41:39.635	0.103	UDP	172.16.96.48:55938	->	224.0.0.252:5355	.....	2	104	1
09:41:40.404	0.000	UDP	172.16.96.48:60395	->	172.16.92.1:53	.....	1	50	1
09:41:40.405	0.000	UDP	172.16.92.1:53	->	172.16.96.48:60395	.....	1	125	1
09:41:40.407	0.101	UDP	172.16.96.48:52932	->	224.0.0.252:5355	.....	2	100	1
09:41:42.134	0.108	UDP	172.16.96.48:51504	->	224.0.0.252:5355	.....	2	104	1

```
09:41:42.160   0.099  UDP  172.16.96.48:52493 ->      224.0.0.252:5355 .....    2     102    1
09:41:42.461   0.112  UDP  172.16.96.48:55260 ->      224.0.0.252:5355 .....    2     102    1
09:41:43.243   0.000  UDP  172.16.96.48:64291 ->        172.16.92.1:53 .....    1      62    1
09:41:43.244   0.000  UDP  172.16.96.48:50664 ->        172.16.92.1:53 .....    1      62    1
09:41:43.244   0.000  UDP   172.16.92.1:53    ->    172.16.96.48:64291 .....    1     256    1
09:41:43.246   0.000  UDP   172.16.92.1:53    ->    172.16.96.48:50664 .....    1     127    1
09:41:43.246   0.384  TCP  172.16.96.48:49158 ->     207.46.131.206:80  A.RS    4     172    1
09:41:43.437   0.192  TCP  207.46.131.206:80  ->    172.16.96.48:49158  AP.SF   3     510    1
09:41:43.631   0.000  UDP  172.16.96.48:63820 ->        172.16.92.1:53 .....    1      62    1
09:41:43.673   0.000  UDP   172.16.92.1:53    ->    172.16.96.48:63820 .....    1     256    1
09:41:44.374   0.105  UDP  172.16.96.48:51599 ->      224.0.0.252:5355 .....    2     104    1
09:41:45.170  14.645  UDP  172.16.96.48:137   ->     172.16.96.255:137 .....   11     858    1
09:41:45.876   0.109  UDP  172.16.96.48:61423 ->      224.0.0.252:5355 .....    2     102    1
09:41:45.881   0.000  UDP  172.16.96.48:54743 ->      224.0.0.252:5355 .....    1      51    1
09:41:52.792   0.109  UDP  172.16.96.48:52975 ->      224.0.0.252:5355 .....    2     104    1
09:41:54.719   0.000  UDP  172.16.96.48:62459 ->        172.16.92.1:53 .....    1      62    1
```

14 minutes later, at 9:55:42, it starts massive scanning activity against computers both in local network and internet with the goal to discover and infect other vulnerable hosts (destination port 445).

```
Flow start  Duration Proto Src IP Addr:Port        Dst IP Addr:Port Flags Packets Bytes Flows

09:55:42.963  0.000  TCP  172.16.96.48:49225 ->   100.9.240.76:445  ...S.   1       48     1
09:55:42.963  0.000  TCP  172.16.96.48:49226 ->  209.13.138.30:445  ...S.   1       48     1
09:55:42.963  0.000  TCP  172.16.96.48:49224 ->    71.70.105.4:445  ...S.   1       48     1
09:55:42.964  0.000  TCP  172.16.96.48:49230 ->   150.18.37.52:445  ...S.   1       48     1
09:55:42.965  0.000  TCP  172.16.96.48:49238 ->  189.97.157.63:445  ...S.   1       48     1
09:55:42.965  0.000  TCP  172.16.96.48:49235 ->   46.77.154.99:445  ...S.   1       48     1
09:55:42.965  0.000  TCP  172.16.96.48:49237 ->  187.96.185.74:445  ...S.   1       48     1
09:55:42.965  0.000  TCP  172.16.96.48:49234 ->   223.62.32.43:445  ...S.   1       48     1
09:55:42.966  0.000  TCP  172.16.96.48:49236 -> 176.77.174.109:445  ...S.   1       48     1
09:55:42.966  0.000  TCP  172.16.96.48:49239 ->  121.110.84.84:445  ...S.   1       48     1
09:55:42.966  0.000  TCP  172.16.96.48:49243 ->  153.34.211.79:445  ...S.   1       48     1
09:55:42.967  0.000  TCP  172.16.96.48:49244 ->    59.34.59.14:445  ...S.   1       48     1
09:55:42.967  0.000  TCP  172.16.96.48:49245 ->  172.115.82.70:445  ...S.   1       48     1
09:55:42.967  0.000  TCP  172.16.96.48:49246 ->   196.117.5.44:445  ...S.   1       48     1
09:55:42.968  0.000  TCP  172.16.96.48:49258 ->    78.33.209.5:445  ...S.   1       48     1
09:55:42.968  0.000  TCP  172.16.96.48:49248 ->      28.36.5.3:445  ...S.   1       48     1
09:55:42.968  0.000  TCP  172.16.96.48:49259 ->     91.39.4.28:445  ...S.   1       48     1
09:55:42.968  0.000  TCP  172.16.96.48:49254 -> 112.96.125.115:445  ...S.   1       48     1
09:55:42.969  0.000  TCP  172.16.96.48:49262 ->    197.63.38.5:445  ...S.   1       48     1
09:55:42.969  0.000  TCP  172.16.96.48:49268 ->   36.85.125.20:445  ...S.   1       48     1
09:55:42.969  0.000  TCP  172.16.96.48:49261 ->  170.88.178.77:445  ...S.   1       48     1
09:55:42.969  0.000  TCP  172.16.96.48:49260 ->  175.42.90.106:445  ...S.   1       48     1
09:55:42.969  0.000  TCP  172.16.96.48:49263 ->    15.70.58.96:445  ...S.   1       48     1
```

One hour later, a lot of university computers are infected and again try to scan and propagate to further computers both in university network and internet:

```
Flow start  Duration Proto  Src IP Addr:Port       Dst IP Addr:Port Flags Packets  Bytes Flows

10:48:10.983  29.934  TCP  172.16.96.31:50076 -> 145.107.246.69:445  AP.S.    30     1259    1
10:48:25.106  30.826  UDP  172.16.96.49:63593 ->  38.81.201.101:445  .....     6     1408    1
10:48:25.894  30.189  TCP  172.16.96.47:51875 ->  169.41.101.97:445  AP.S.    29     1298    1
10:48:26.001  32.111  TCP  172.16.96.49:63778 ->   43.28.146.45:445  AP.S.    18      906    1
10:48:26.948  10.745  TCP  172.16.96.50:52225 ->  104.24.33.123:445  AP.S.    10      537    1
10:48:27.466  24.770  TCP  172.16.96.35:55484 ->   109.18.23.97:445  AP.SF   102   146397    1
10:48:28.443  28.866  TCP  172.16.96.37:53098 -> 102.124.181.67:445  AP.S.    15      804    1
10:48:28.473  10.572  TCP  172.16.96.38:60340 ->   222.50.79.96:445  AP.S.    23     4549    1
10:48:28.797  30.748  TCP  172.16.96.37:53174 ->  212.82.132.58:445  AP.S.    19      861    1
10:48:29.267  32.783  TCP  172.16.96.34:64769 ->   34.56.183.93:445  AP.S.    17     1696    1
10:48:29.409   7.773  TCP  172.16.96.34:64756 -> 89.109.215.111:445  AP.S.    17     3037    1
10:48:29.492  34.993  TCP  172.16.96.44:57145 ->    32.113.4.81:445  AP.S.    15     2562    1
10:48:29.749  26.004  TCP  172.16.96.43:52707 ->   138.8.147.38:445  AP.S.    16     1725    1
10:48:30.159  12.609  TCP  172.16.96.49:63902 ->  203.101.75.18:445  AP.S.    22     2316    1
10:48:31.116   3.004  TCP  172.16.96.31:50766 ->  194.125.49.68:445  ...S.     2       96    1
10:48:31.117   3.003  TCP  172.16.96.31:50768 -> 193.114.216.37:445  ...S.     2       96    1
10:48:31.117   3.003  TCP  172.16.96.31:50769 ->   37.107.5.111:445  ...S.     2       96    1
10:48:31.117   3.003  TCP  172.16.96.31:50770 ->  126.96.239.95:445  ...S.     2       96    1
10:48:31.118   3.002  TCP  172.16.96.31:50776 ->   43.87.170.91:445  ...S.     2       96    1
10:48:31.119   3.001  TCP  172.16.96.31:50778 ->  103.13.70.122:445  ...S.     2       96    1
10:48:31.127   2.993  TCP  172.16.96.31:50784 ->  200.68.202.35:445  ...S.     2       96    1
10:48:31.129   2.991  TCP  172.16.96.31:50791 ->   56.39.208.87:445  ...S.     2       96    1
10:48:31.131   2.990  TCP  172.16.96.31:50797 -> 59.104.110.104:445  ...S.     2       96    1
```

8 Worm Detection And Analysis With CAMNEP

The detection layer presented in Section 4 sorts the flows by trustfulness, placing the potentially malicious events close to the left edge of the histogram that we can see in Figure 4. The red peak (highlighted by the aposteriori GUI-level filtering) can be easily discovered, and analyzed in the traffic analysis tool, as displayed at Figures 5, 6, and 7. These figures illustrate the several relevant characteristics of event flows during the initial attack phase. Internal representation of the event in the trust model of one of the agents can be seen in Figure 8.

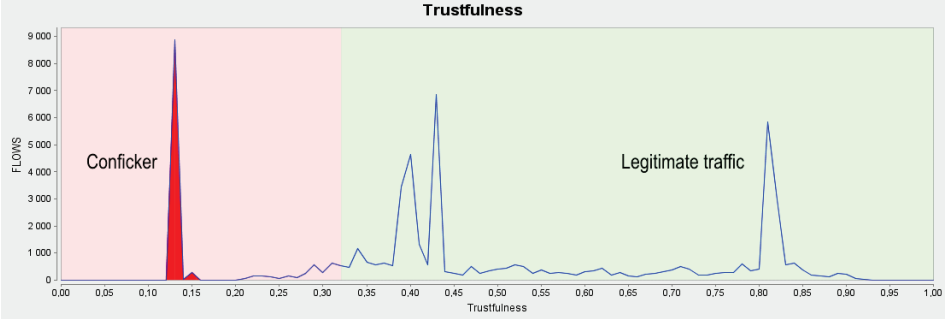

Figure 4: Aggregated trustfulness of flows during the Conficker worm spreading activity. We can see that the Conficker traffic (leftmost peak, red) is separated from the rest of the traffic.

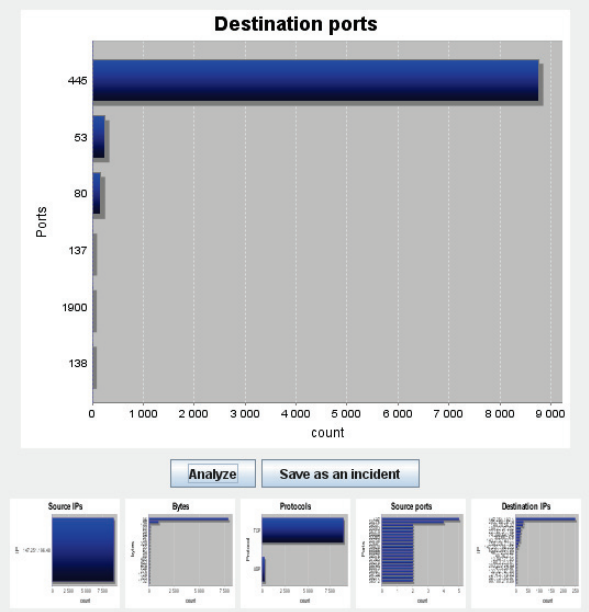

Figure 5: Incident analysis window representing Conficker worm traffic distribution by destination ports - the majority of traffic is the typical Conficker worm destination port 445 used for file sharing on Microsoft Windows machines.

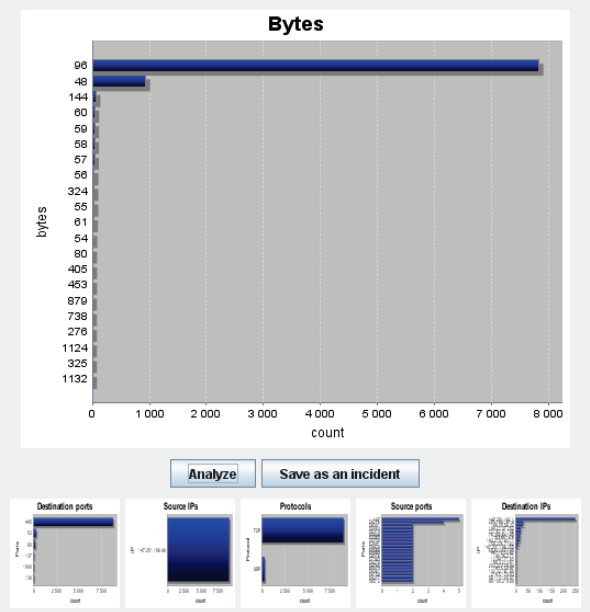

Figure 6: Incident analysis window representing Conficker worm traffic distribution by a number of bytes in flows - the majority of traffic is 96 bytes large.

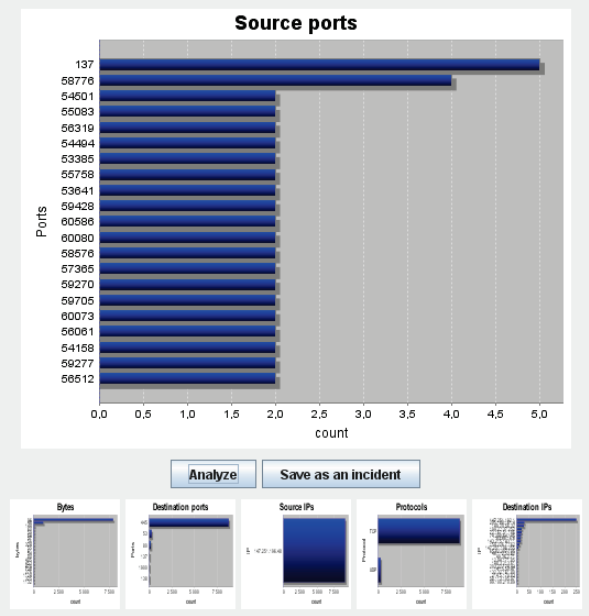

Figure 7: Analyzer window representing Conficker worm traffic distribution by source ports shows great variability (only 21 are shown).

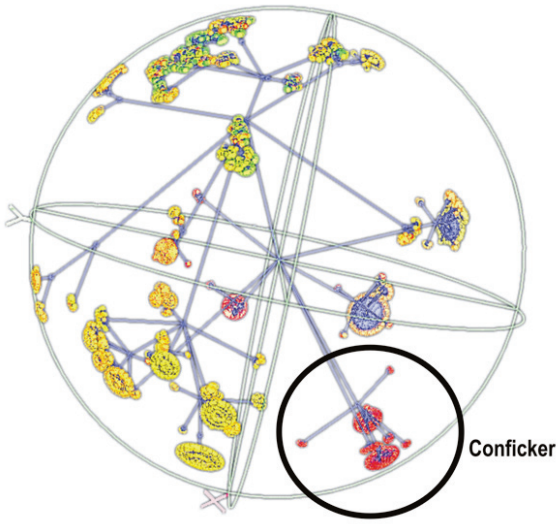

Figure 8: 3D representation of the trust model showing the whole traffic. The Conficker worm traffic (marked and red colored) is clearly separated from the legitimate traffic situated on the top of the sphere.

9 Conclusion

In our work, we extend possibilities of security tools especially NetFlow collectors [4] used by CERT (Computer Emergency Response Teams) to detect network security incidents. We target the problem of high knowledge demands on network security engineers and limits of human operator to efficiently supervise all network traffic in near real-time.

Presented flow based network intrusion detection system is able to identify significant malicious traffic events often hidden in a normal traffic overview. In our pilot deployment we showed that instead of analyzing thousands lines of flow data, or observing only the aggregated values for the whole network, the operator can efficiently investigate only reported events. Real world example shows Conficker worm detection and describes the analysis using raw NetFlow data compared to straightforward trustfullness histogram of CAMNEP.

Early in 2009 the CAMNEP tool was deployed for operational use of Masaryk University CERT. Together we are working on best practices how to deploy and use CAMNEP by network security engineers. We are also working on improved long-time stability and reduction of the false positive and false negative rates.

Acknowledgement

This material is based upon work supported by the ITC-A of the US Army under Contract No. W911NF-08-1-0250. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the ITC-A of the US Army.

This work was supported by the Czech Ministry of Defence under Contract No. SMO02008PR980-OVMASUN200801 and also supported by Czech Ministry of Education grants 6840770038 (CTU) and 6383917201 (CESNET).

References

[1] Paul Barford, Somesh Jha, and Vinod Yegneswaran. Fusion and filtering in distributed intrusion detection systems. In *In Proceedings of the 42nd Annual Allerton Conference on Communication, Control and Computing*, 2004.

[2] F-Secure. Preemptive Blocklist and More Downadup Numbers, 2009. http://www.f-secure.com/weblog/archives/00001582.html.

[3] Peter Haag. NFDUMP - NetFlow processing tools. http://nfdump.sourceforge.net/, 2007.

[4] Peter Haag. NfSen - NetFlow Sensor. http://nfsen.sourceforge.net/, 2007.

[5] Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosis Network-Wide Traffic Anomalies. In *ACM SIGCOMM '04*, pages 219 - 230, New York, NY, USA, 2004. ACM Press.

[6] Anukool Lakhina, Mark Crovella, and Christophe Diot. Mining Anomalies using Traffic Feature Distributions. In *ACM SIGCOMM, Philadelphia, PA, August 2005*, pages 217 - 228, New York, NY, USA, 2005. ACM Press.

[7] Rehak Martin, Pechoucek Michal, Grill Martin, Bartos Karel, Celeda Pavel, and Krmicek Vojtech. Collaborative approach to network behavior analysis. In *Global E-Security*, pages 153 - 160. Springer, 2008.

[8] Rehak Martin, Pechoucek Michal, Celeda Pavel, Krmicek Vojtech, Bartos Karel, and Grill Martin. Multi-Agent Approach to Network Intrusion Detection (Demo Paper). In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, pages 1695 - 1696. InescId, 2008.

[9] Phillip Porras, Hassen Saidi, and Vinod Yegneswaran. An analysis of conficker's logic and rendezvous points. Technical report, 2009. http://mtc.sri.com/Conficker.

[10] 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.

[11] Martin Rehak, Michal Pechoucek, Martin Grill, and Karel Bartos. Trust-based classifier combination for network anomaly detection. In *Cooperative Information Agents XII*, LNAI/LNCS. Springer-Verlag, 2008.

[12] Karen Scarfone and Peter Mell. Guide to intrusion detection and prevention systems (idps). Technical Report 800-94, NIST, US Dept. of Commerce, 2007.

[13] Pavel Celeda, Milan Kovacik, Tomas Konir, Vojtech Krmicek, Petr Springl, and Martin Zadnik. FlowMon Probe. Technical Report 31/2006, CESNET, z.s.p.o., 2006. http://www.cesnet.cz/doc/techzpravy/2006/flowmon-probe/.

[14] 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.