Network Security Monitoring and Behavior Analysis

Best Practice Document

Produced by CESNET led working group on network monitoring (CBPD133)

Author: Pavel Čeleda

September 2011

Document No: GN3-NA3-T4-CBPD133
Version / date: 1.0.0 of 2011/09/07
Original language: English
Original title: “Network Security Monitoring and Behavior Analysis”
Original version / date: 1.0.0 of 2011/09/07
Contact: celeda@ics.muni.cz

CESNET bears responsibility for the content of this document. The work has been carried out by a CESNET led working group on network monitoring as part of a joint-venture project within the HE sector in Czech Republic.

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 23 8875, relating to the project 'Multi-Gigabit European Research and Education Network and Associated Services (GN3)'.

Table of Contents

Executive Summary

The purpose of this document is to provide an insight into Network Security Monitoring and Behavior Analysis for administrators of campus network and computer security incident response team members.

The document describes flow-based network security monitoring system and how to deploy them in a campus network. We show the process of NetFlow generation, collection and anomaly detection. Selected use cases demonstrate anomaly detection capabilities to reveal real world malware – the Chuck Norris botnet and Conficker worm.

Introduction

Ensuring security awareness in the campus 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 network 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.

Typically members of CERT (Computer Emergency Response Team) or CSIRT (Computer Security Incident Response Team) perform network security monitoring and behavior analysis. Based on these data they may better detect and prevent unauthorized usage of computers. Security monitoring provides valuable inputs for handling security incidents.

We present our best practices which may help reduce the requirements for network operator experiences and may help overtake long-term network surveillance. The system operator does not need to constantly observe network behavior, but can be focused on the security incident response and resolution. The system operator receives reports about security incidents and examines them – checks, if there are not false positives and in case of need he performs further necessary actions.

System Architecture

The proposed network security monitoring system has four main layers. The layers are typically distributed and each layer processes the output of the layer underneath, as shown in Figure 1. The system is based on GÉANT2 Security Toolset, which consists of the NetFlow analysis tool NfSen and the FlowMon appliance. Further we added anomaly detection tools and incident reporting.

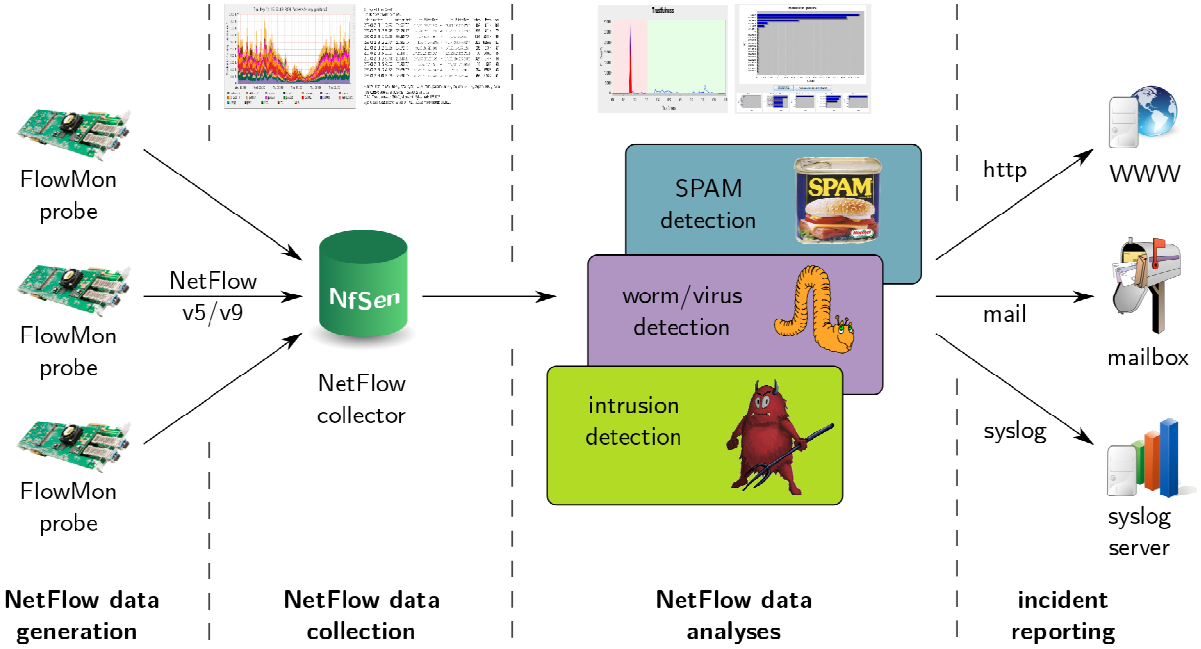


Figure 1: Flow-based network security monitoring system. Generated NetFlow data are stored into NetFlow collectors and used for anomaly detection and network forensics by university security team.

At the bottom of the whole architecture, there is one or more standard or hardware-accelerated FlowMon probes [LIB11] generating NetFlow data and exporting them to the collector. The open-source NFDUMP [HAAG1] and NfSen [HAAG2] tools handle NetFlow data. These tools also provide storage and retrieval of the traffic information for forensics analysis. NetFlow data are further processed by anomaly detection tools. Suspicious flows and network anomalies are either visualized in web user interface or reported to the network operator. Email reports and event notifications are sent to standard mailbox, syslog or to request tracking system.

Network Security Monitoring

To be able to provide a 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.

- 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 campus networks is an issue including high operational costs.
- Flow based statistics provide information from IP headers. They do not 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 does not allow advanced traffic analysis.

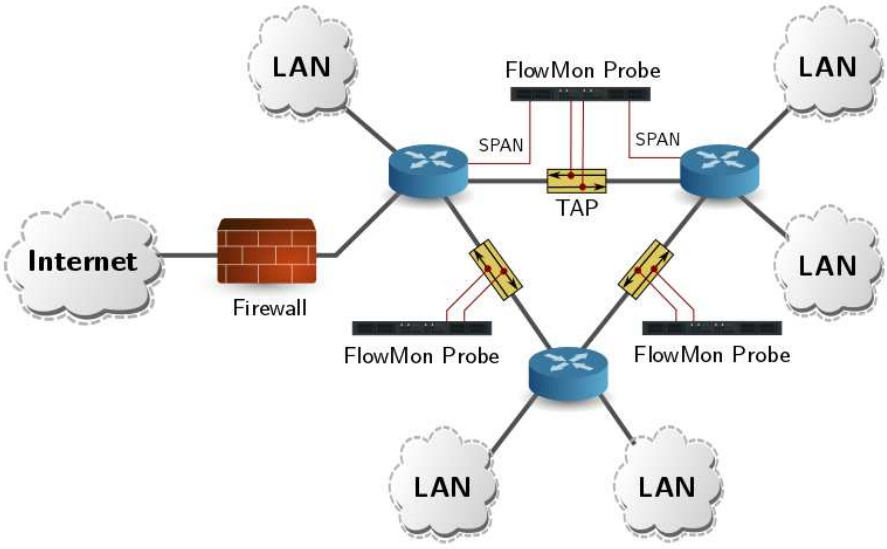


Figure 2: Traffic monitoring system deployment in a campus network.

We use flow data for their scalability and ability to provide a sufficient amount of information. Using flow statistics allows to work even with encrypted traffic. NetFlow initially available in CISCO routers is now used in various flow enabled appliances (routers, probes). Flow based monitoring allows us to permanently observe from small end-user network up to large NREN (National Research and Education Network) backbone links.

NetFlow Generators

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 [ZAD10]. 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.

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 by Liberouter project as part of JRA2 activity of the GÉANT2 project. The FlowMon appliances are now manufactured by INVEA-TECH company (university start-up company).

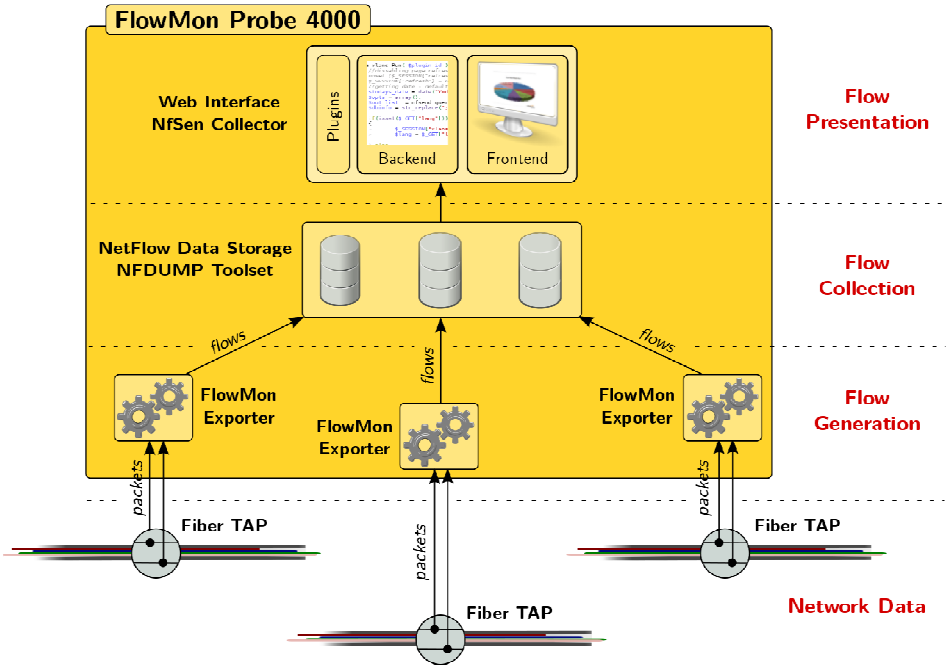


Figure 3: FlowMon appliance architecture.

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 on 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 a 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 [ZHA07].

NetFlow Collectors

The main task of collector is to store incoming packets into a database (SQL or flat file). The collector provides interface to graphical representation of network traffic, flow filtration, aggregation and statistics evaluation, using source and destination IP addresses, ports and protocol.

Figure 4 shows block diagram of NfSen (Netflow Sensor) collector. NfSen is a graphical web based front-end for the NFDUMP toolset. NfSen provides web user interface to display NetFlow as graphs or command-line interface may be used to process flow data. NfSen may be extended with user plugins to provide additional functionality.

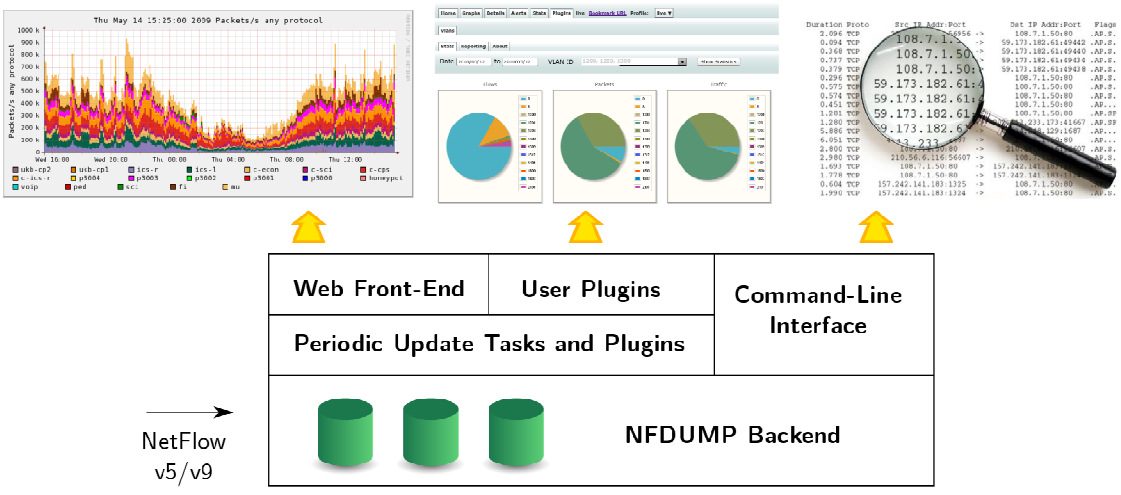


Figure 4: NfSen/NFDUMP collector block diagram.

NfSen profiles are specific views on the NetFlow data. A profile is defined by its name, type and one or more profile filters, which are any valid filters accepted by the NfSen tool. Network administrator may define own profiles and groups of profiles to graph flow data.

The collector interface provides also a tool for automatic alerting. The network administrator may define a set of alerts based on the conditions depending on the NetFlow data. The alert

itself may execute a specific action based on specific conditions e.g., an email is sent in case of outage in the network traffic.

The NfSen collector supports API (Application Programming Interface) for integration of extension plugins. This interface consists of two parts – front-end part accessible through the web interface and performing direct interaction with the user and the back-end part processing requested data and preparing them for the plugin interface. The front-end part is implemented as the set of PHP pages, the back-end part is implemented in the Perl.

Anomaly Detection and Behavior Analysis

Network behavior analysis (NBA) is an intrusion-detection technique that uses the patterns in network-traffic structures and properties to identify possible attacks and technical problems with minimal impact on user data privacy. The analysis is not based on content of the transferred information but on network traffic statistics in the NetFlow format. NBA is only one of the principal approaches to intrusion detection. Intrusion detection systems are classified by their location (on a host, a wired network, or a wireless network), detection methodology (signature matching, anomaly detection, or stateful protocol analysis), or capability (simple detection or active attack prevention).

Anomaly detection 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 [ANUK04] or parameter distribution characteristics [ANUK05]. 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 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.

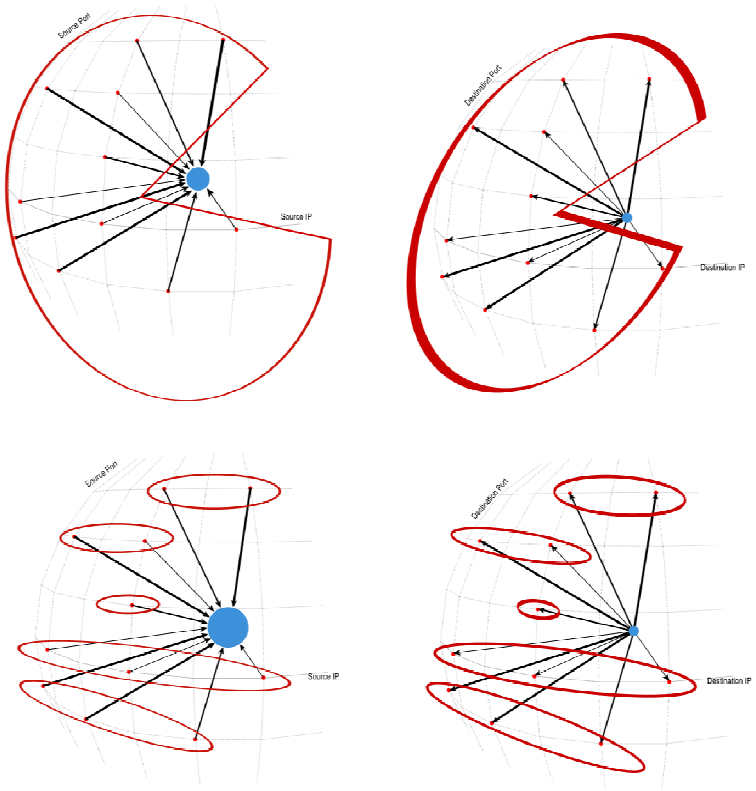


Figure 5: Anomaly detection methods – flow counts from/to important IP/port combinations.

The biggest issue of the anomaly-based intrusion detection systems is their error rate, which consists of two related error types. False positives are the legitimate flows classified as anomalous, while the False negatives 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 NBA systems use various approaches like a multistage collaborative process of detection, trust modeling, autonomous adaptation, etc. to make the anomaly detection operationally deployable.

Anomaly Detection Case Studies

In this part we describe use cases demonstrating the anomaly detection capabilities from the user (e.g., network administrator or incident analyst) perspective. We focus on the detection of the real world malware – the Chuck Norris Botnet [CEL10] and the Conficker worm [POR09]. The detection was performed on real network data acquired from campus network. Using network security monitoring system, we were able to reveal and describe previously unknown Chuck Norris Botnet.

Chuck Norris Botnet Detection

To protect our campus network we use network monitoring system as shown in Figure 1. The network-based approach allows us to see all activities against and from our network. Typically we observe various network scan attempts, password brute force attacks and exploits coming from outside. Such activities are often regarded as a normal part of nowadays Internet traffic.

At the beginning of December 2009 our attention was attracted by an increased amount of Telnet scans (TCP port 23). The use of the Telnet protocol should be discontinued for security related shortcomings and replaced by Secure Shell (SSH) protocol. Any Telnet activity, especially on the public Internet, is suspicious. Figure 6 depicts trends in Telnet scan activities observed in the campus network from October 2009 to July 2011.

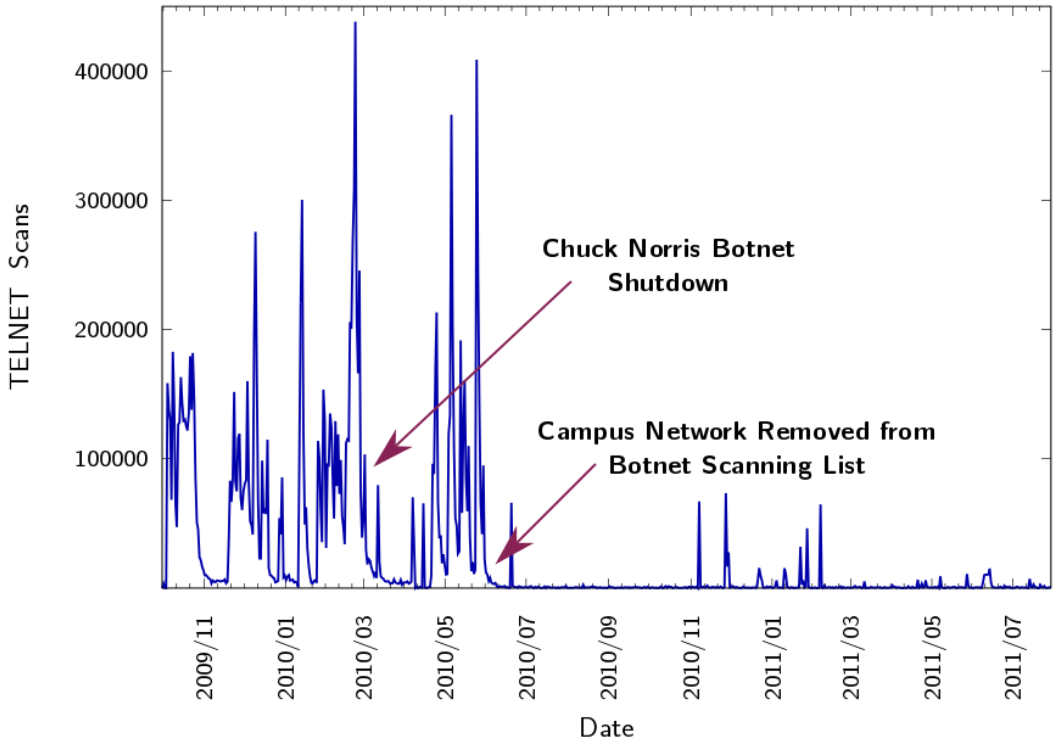


Figure 6: Number of Telnet scans on TCP port 23 in the campus network.

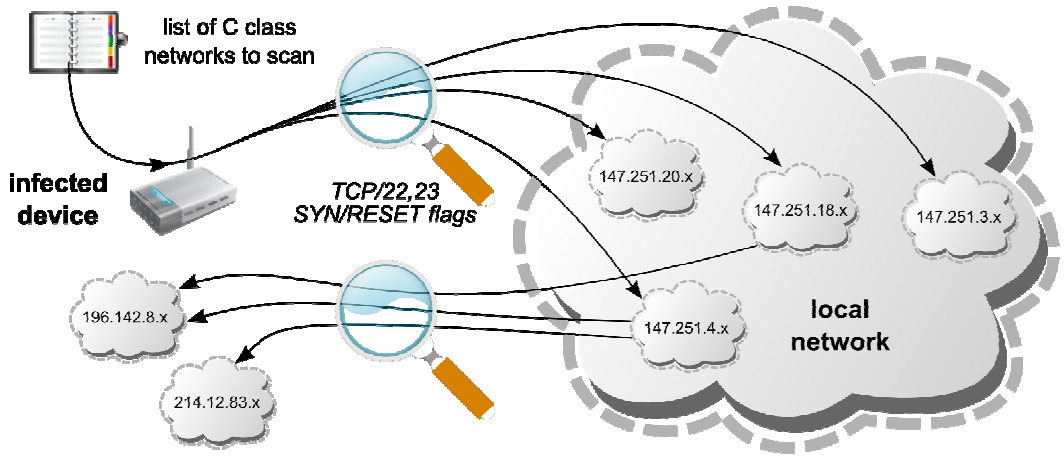
By checking the attack sources we have identified world wide located networks of DSL modems and home routers. Further investigation revealed IP addresses of C&C centers including botnet distribution sites. The botnet got the Chuck Norris moniker from a comment in its source code *[R]anger Killato : in nome di Chuck Norris !*

In this section, we show how to design own anomaly detection plugin for NfSen collector to detect the Chuck Norris Botnet. Any network connected device leaves communication traces e.g., connection establishment, data downloads and uploads, etc. Using NetFlow we may save these communication traces. Further analysis reveals details about a device behavior and may be used to detect illicit network activities. We identified a set of generic NetFlow signatures, representing particular botnet lifecycle phases. Any occurrence of such signatures in the observed network could indicate a presence of possible botnet infection.

Scanning and Infiltration of Vulnerable Devices

In the case of scanning, we observe a large amount of flows from outside network against devices in the monitored network. The attackers perform a horizontal scan, i.e. they initiate a new connection against every possible IP address in the scanned network. These connections are against one of the service ports, i.e. SSH port (22), Telnet port (23) or HTTP port (80). This behavior results in a large number of unsuccessful connections, because not every scanned IP address is alive and it doesn’t accept a new connection at the scanned port. Such connections do not finish TCP three way handshake and therefore they contain

only SYN (S) or RESET (R) TCP flags, but not ACKNOWLEDGE (A) flag. When we filter out such connections (TCP connections with TCP flags S or SR against the port 22, 23 or 80), we will get a list of devices outside our network, which could be (but not necessarily) a part of botnet.

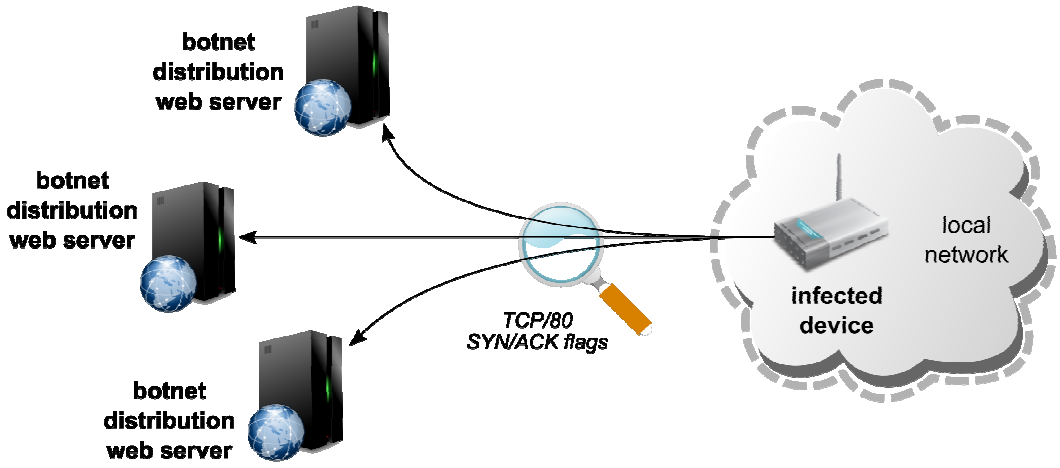



```
(dst net local_network) and (dst port 22 or dst port 23) and (proto TCP)
 and ((flags S and not flags ARPUF) or (flags SR and not flags APUF))
```

Figure 7: NFDUMP filter for botnet scanning detection.

Initialization and Update

In the case of bot code download, we observe an HTTP connection from a local network to the web server with update files. To perform bot update download, a regular HTTP connection against HTTP port (80) has to be established, so we filter out only connections containing TCP flags S, A and not flag R. By the analysis of the bot code we get a list of used IP addresses or domain names, so we are able to filter out only connections against these domain names/IP addresses. The main limitations of this approach is no possibility to distinguish a regular request for WWW pages and the connection performing bot code download from the same web server. Such a situation is frequent in the case of virtual web hosting, where a single IP address is used for hosting multiple web domains (including the domains with bot code).

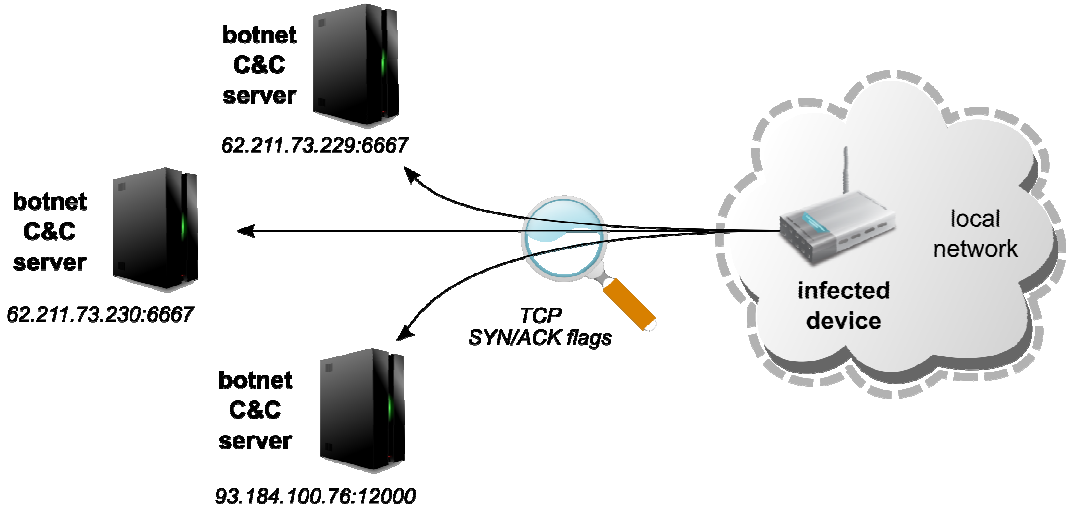



```
(src net local_network) and (dst hosts web_servers) and (dst port 80)
 and (proto TCP) and (flags SA and not flag R)
```

Figure 8: NFDUMP filter for botnet initialization and update detection.

Communication with C&C Center

Because the botnet member needs to communicate with C&C server regularly, we may see a set of TCP connections to the IRC server. These connections use a set of predefined IRC ports (including default IRC port 6667, the IRC port numbers can vary depending on the particular botnet). By the analysis of the botnet code we get the IRC port number and the domain names or IP addresses of the IRC C&C servers.




```
(src net local_network) and (((dst ip 62.211.73.229 or dst ip
62.211.73.230) and port 6667) or (dst IP 93.184.100.76 and port 12000))
and (proto TCP) and (flags SA and not flag R)
```

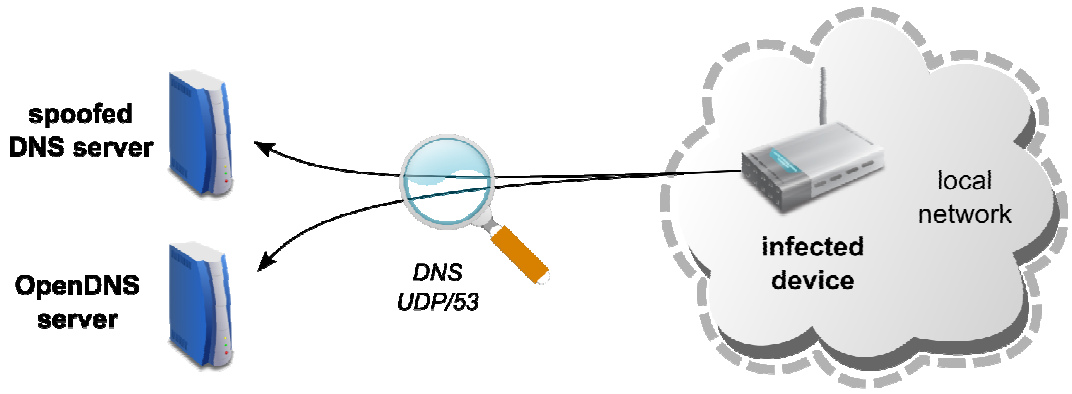
Figure 9: NFDUMP filter for botnet C&C communication detection.

Attacking

The detection of bot attacks is difficult, because we can see only the flow activity generated by the infected device. We are not able to inspect commands and scripts executed at the device itself. Even so we may detect several types of botnet attacks analyzing NetFlow data from monitored network.

DNS spoofing attack is based on the misconfiguration/change of the DNS server IP addresses in the infected device. The user is then redirected to the spoofed botnet DNS server and becomes a target of the phising attack. In such case, we can detect connections to the DNS servers (using UDP protocol/port 53) located outside local network, not ordinary used in the local network. These connections would indicate possible infection of the inspected device.

Detection of the most typical harmful activity, (D)DoS attacks, generated by the infected devices, is not trivial. We need to distinguish all regular connection going outside the inspected device from the attack traffic. For the (D)DoS attacks are used e.g., UDP floods, TCP SYN floods or TCP PSH ACK floods. Such attack represents a significant change in the device behavior. Therefore we can detect it e.g., by using behavior profiles of each device. Another way is a detection of the spoofed source IP addresses in the monitored traffic (i.e. addresses not originated in the local network).




```
(src net local_network) and ((dst ip OpenDNS server) or
(dst ip botnet DNS server)) and (proto UDP) and (dst port 53)
```

Figure 10: NFDUMP filter for botnet DNS spoofing attack detection.

NfSen Detection Plugin

We developed tool called cndet (NfSen plugin) to perform real-time botnet detection [CYB10]. The plugin identifies botnet from flow data using several previously described behavior patterns. It detects infected devices in the local network as well as monitors botnet activity outside the local network (by observing incoming connections).

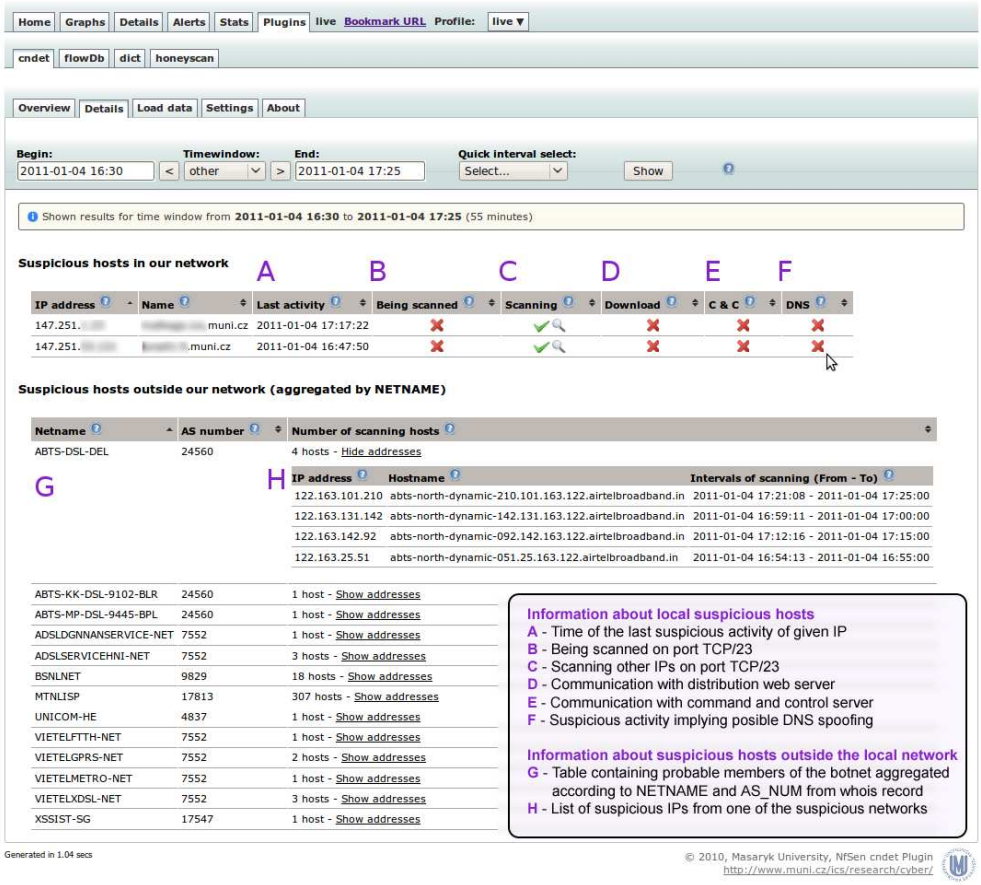


Figure 11: NfSen detection plugin.

The presented approach for revealing botnets may be used for the detection of any botnet variant with similar behavior. Some of the behavior patterns cannot be generally defined and must be adapted to the specific type of botnet (or only for particular revision of the botnet). Detection methods applicable to any type of botnet, include:

- The botnet scanning (the botnets are using a fixed set of ports).
- The (D)DoS attacks (well-known behavior changes in the NetFlow data during the (D)DoS attack).
- The DNS spoofing attacks (DNS connections going to the nonstandard DNS servers).
- The communication with IRC C&C servers (in the case of network with a strict policy restricting IRC connections).

The signatures for revealing botnet download/initialization and the communication with the IRC C&C server (in the case of academic/open network) have to be customized to the particular botnet. The IP addresses and ports of web servers and IRC C&C servers may vary depending on the botnet variant.

Conficker Worm Detection

In the second use case we compare anomaly detection using command-line tool NFDUMP and a dedicated detection system. We show how to detect Conficker worm in the campus network.

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 [FSEC09].

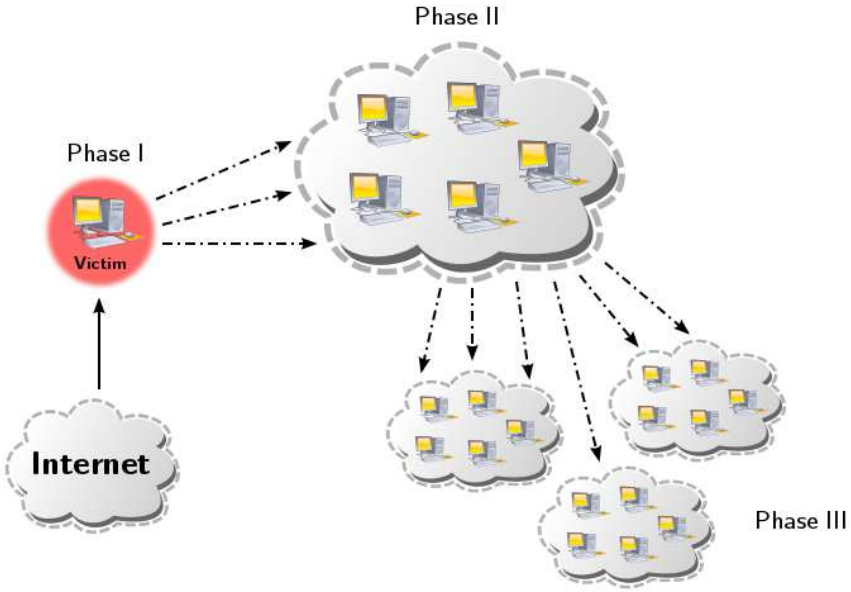


Figure 12: Conficker worm spreading activity in monitored campus network.

Figure 12 illustrates the Conficker worm propagation inside the campus 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 the 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 campus computers are infected and again try to scan and propagate (*phase III*) to further computers both in the campus network and Internet.

Traditional NetFlow Analysis Using NFDUMP Tool

We 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 inside the campus network started to communicate at 9:41:12, 11.2.2009:

Flow start	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Packets	Bytes	Flows
09:41:12.024	UDP	172.16.96.48:49417	->	224.0.0.252:5355	.....	2	102	1
09:41:12.537	UDP	172.16.96.48:60435	->	224.0.0.252:5355	.....	2	102	1
09:41:14.446	ICMP	172.16.92.1:0	->	172.16.96.48:3.10	.....	25	3028	1
09:41:14.446	UDP	172.16.96.48:137	->	172.16.96.255:137	.....	25	2238	1
09:41:21.692	UDP	172.16.96.48:60436	->	172.16.92.1:53	.....	2	162	1
09:41:21.692	UDP	172.16.92.1:53	->	172.16.96.48:60436	.....	2	383	1
09:41:21.763	UDP	172.16.96.48:5353	->	224.0.0.251:5353	.....	9	867	1
09:41:24.182	UDP	172.16.96.48:60438	->	239.255.255.250:3702	.....	6	6114	1
09:41:24.470	UDP	172.16.96.48:138	->	172.16.96.255:138	.....	3	662	1
09:41:26.069	UDP	172.16.96.48:60443	->	239.255.255.250:1900	.....	14	2254	1
09:41:39.635	UDP	172.16.96.48:55938	->	224.0.0.252:5355	.....	2	104	1
09:41:40.404	UDP	172.16.96.48:60395	->	172.16.92.1:53	.....	1	50	1
09:41:40.405	UDP	172.16.92.1:53	->	172.16.96.48:60395	.....	1	125	1
09:41:40.407	UDP	172.16.96.48:52932	->	224.0.0.252:5355	.....	2	100	1
09:41:42.134	UDP	172.16.96.48:51504	->	224.0.0.252:5355	.....	2	104	1
09:41:42.160	UDP	172.16.96.48:52493	->	224.0.0.252:5355	.....	2	102	1
09:41:42.461	UDP	172.16.96.48:55260	->	224.0.0.252:5355	.....	2	102	1
09:41:43.243	UDP	172.16.96.48:64291	->	172.16.92.1:53	.....	1	62	1
09:41:43.244	UDP	172.16.96.48:50664	->	172.16.92.1:53	.....	1	62	1
09:41:43.244	UDP	172.16.92.1:53	->	172.16.96.48:64291	.....	1	256	1
09:41:43.246	UDP	172.16.92.1:53	->	172.16.96.48:50664	.....	1	127	1
09:41:43.246	**TCP**	**172.16.96.48:49158**	**->**	**207.46.131.206:80**	**A.RS.**	**4**	**172**	**1**
09:41:43.437	**TCP**	**207.46.131.206:80**	**->**	**172.16.96.48:49158**	**AP.SF**	**3**	**510**	**1**
09:41:43.631	UDP	172.16.96.48:63820	->	172.16.92.1:53	.....	1	62	1
09:41:43.673	UDP	172.16.92.1:53	->	172.16.96.48:63820	.....	1	256	1
09:41:44.374	UDP	172.16.96.48:51599	->	224.0.0.252:5355	.....	2	104	1
09:41:45.170	UDP	172.16.96.48:137	->	172.16.96.255:137	.....	11	858	1
09:41:45.876	UDP	172.16.96.48:61423	->	224.0.0.252:5355	.....	2	102	1
09:41:45.881	UDP	172.16.96.48:54743	->	224.0.0.252:5355	.....	1	51	1
09:41:52.792	UDP	172.16.96.48:52975	->	224.0.0.252:5355	.....	2	104	1
09:41:54.719	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	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Packets	Bytes	Flows
09:55:42.963	TCP	172.16.96.48:49225	->	100.9.240.76:445	...S.	1	48	1
09:55:42.963	TCP	172.16.96.48:49226	->	209.13.138.30:445	...S.	1	48	1
09:55:42.963	TCP	172.16.96.48:49224	->	71.70.105.4:445	...S.	1	48	1
09:55:42.964	TCP	172.16.96.48:49230	->	150.18.37.52:445	...S.	1	48	1
09:55:42.965	TCP	172.16.96.48:49238	->	189.97.157.63:445	...S.	1	48	1
09:55:42.965	TCP	172.16.96.48:49235	->	46.77.154.99:445	...S.	1	48	1
09:55:42.965	TCP	172.16.96.48:49237	->	187.96.185.74:445	...S.	1	48	1
09:55:42.965	TCP	172.16.96.48:49234	->	223.62.32.43:445	...S.	1	48	1
09:55:42.966	TCP	172.16.96.48:49236	->	176.77.174.109:445	...S.	1	48	1

```
09:55:42.966   TCP  172.16.96.48:49239 ->  121.110.84.84:445  ...S.   1       48     1
09:55:42.966   TCP  172.16.96.48:49243 ->  153.34.211.79:445  ...S.   1       48     1
09:55:42.967   TCP  172.16.96.48:49244 ->    59.34.59.14:445  ...S.   1       48     1
09:55:42.967   TCP  172.16.96.48:49245 ->  172.115.82.70:445  ...S.   1       48     1
09:55:42.967   TCP  172.16.96.48:49246 ->   196.117.5.44:445  ...S.   1       48     1
09:55:42.968   TCP  172.16.96.48:49258 ->    78.33.209.5:445  ...S.   1       48     1
09:55:42.968   TCP  172.16.96.48:49248 ->      28.36.5.3:445  ...S.   1       48     1
09:55:42.968   TCP  172.16.96.48:49259 ->     91.39.4.28:445  ...S.   1       48     1
09:55:42.968   TCP  172.16.96.48:49254 -> 112.96.125.115:445  ...S.   1       48     1
09:55:42.969   TCP  172.16.96.48:49262 ->    197.63.38.5:445  ...S.   1       48     1
09:55:42.969   TCP  172.16.96.48:49268 ->   36.85.125.20:445  ...S.   1       48     1
09:55:42.969   TCP  172.16.96.48:49261 ->  170.88.178.77:445  ...S.   1       48     1
09:55:42.969   TCP  172.16.96.48:49260 ->  175.42.90.106:445  ...S.   1       48     1
09:55:42.969   TCP  172.16.96.48:49263 ->    15.70.58.96:445  ...S.   1       48     1
```

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

```
Flow start   Proto   Src IP Addr:Port        Dst IP Addr:Port Flags Packets  Bytes Flows
10:48:10.983   TCP  172.16.96.31:50076 -> 145.107.246.69:445  AP.S.    30     1259     1
10:48:25.106   UDP  172.16.96.49:63593 ->  38.81.201.101:445  .....     6     1408     1
10:48:25.894   TCP  172.16.96.47:51875 ->  169.41.101.97:445  AP.S.    29     1298     1
10:48:26.001   TCP  172.16.96.49:63778 ->   43.28.146.45:445  AP.S.    18      906     1
10:48:26.948   TCP  172.16.96.50:52225 ->  104.24.33.123:445  AP.S.    10      537     1
10:48:27.466   TCP  172.16.96.35:55484 ->   109.18.23.97:445  AP.SF   102   146397     1
10:48:28.443   TCP  172.16.96.37:53098 -> 102.124.181.67:445  AP.S.    15      804     1
10:48:28.473   TCP  172.16.96.38:60340 ->   222.50.79.96:445  AP.S.    23     4549     1
10:48:28.797   TCP  172.16.96.37:53174 ->  212.82.132.58:445  AP.S.    19      861     1
10:48:29.267   TCP  172.16.96.34:64769 ->   34.56.183.93:445  AP.S.    17     1696     1
10:48:29.409   TCP  172.16.96.34:64756 -> 89.109.215.111:445  AP.S.    17     3037     1
10:48:29.492   TCP  172.16.96.44:57145 ->    32.113.4.81:445  AP.S.    15     2562     1
10:48:29.749   TCP  172.16.96.43:52707 ->   138.8.147.38:445  AP.S.    16     1725     1
10:48:30.159   TCP  172.16.96.49:63902 ->  203.101.75.18:445  AP.S.    22     2316     1
10:48:31.116   TCP  172.16.96.31:50766 ->  194.125.49.68:445  ...S.     2       96     1
10:48:31.117   TCP  172.16.96.31:50768 -> 193.114.216.37:445  ...S.     2       96     1
10:48:31.117   TCP  172.16.96.31:50769 ->   37.107.5.111:445  ...S.     2       96     1
10:48:31.117   TCP  172.16.96.31:50770 ->  126.96.239.95:445  ...S.     2       96     1
10:48:31.118   TCP  172.16.96.31:50776 ->   43.87.170.91:445  ...S.     2       96     1
10:48:31.119   TCP  172.16.96.31:50778 ->  103.13.70.122:445  ...S.     2       96     1
10:48:31.127   TCP  172.16.96.31:50784 ->  200.68.202.35:445  ...S.     2       96     1
10:48:31.129   TCP  172.16.96.31:50791 ->   56.39.208.87:445  ...S.     2       96     1
10:48:31.131   TCP  172.16.96.31:50797 -> 59.104.110.104:445  ...S.     2       96     1
```

Worm Detection And Analysis With NBA Tool

CAMNEP is a network behavioral analysis tool which use several anomaly detection algorithms to classify legitimate and malicious traffic. The goal of the CAMNEP project was to design and implement network intrusion detection system for high-speed networks. System observes network traffic using FlowMon probes, detects network anomalies using agent technologies and visualizes malicious traffic. More detailed description is available at CAMNEP homepage [CAMNEP]. The CAMNEP tool is now developed and maintained by Cognitive Security company (university start-up company).

The CAMNEP sorts the flows by trustfulness, placing the potentially malicious events close to the left edge of the histogram that we can see in Figure 13. 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 14, 15, and 16. 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 17.

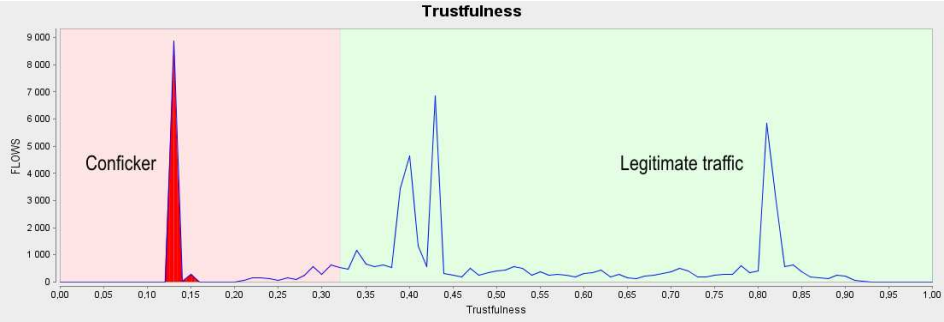


Figure 13: Aggregated trustfulness of flows during the Conficker worm spreading activity. We can see that the Conficker traffic (leftmost peak, red) is separated from the rest of the traffic.

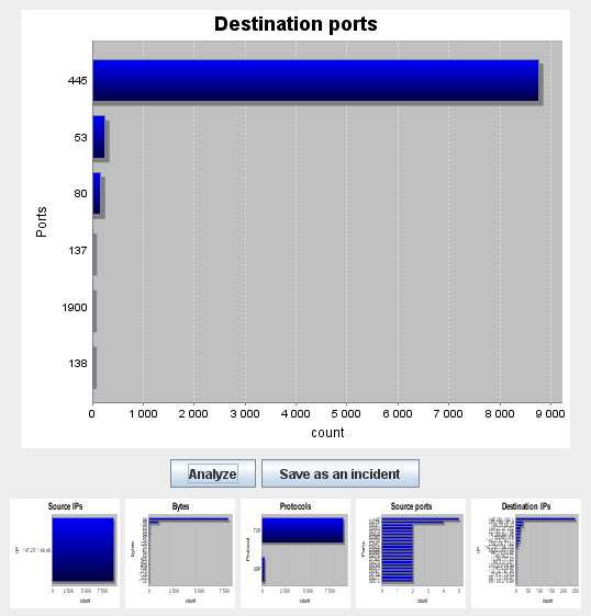


Figure 14: Incident analysis window representing Conficker worm traffic distribution by destination ports – the majority of traffic is the typical Conficker worm destination port 445 used for file sharing on Microsoft Windows machines.

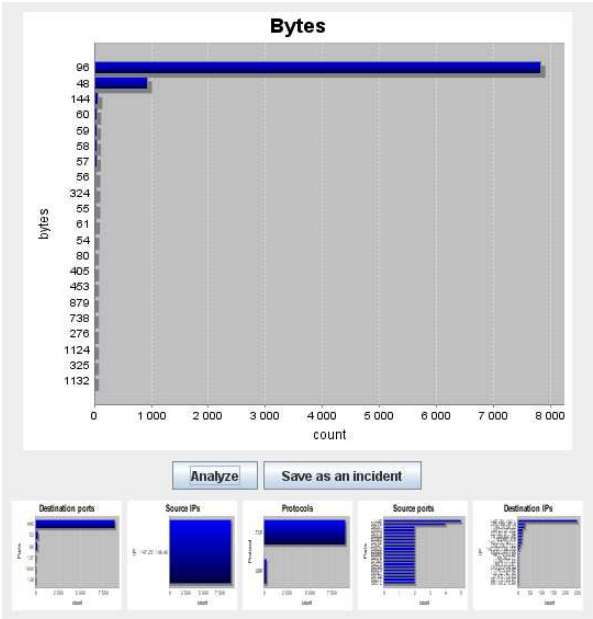


Figure 15: Incident analysis window representing Conficker worm traffic distribution by a number of bytes in flows – the majority of traffic is 96 bytes large.

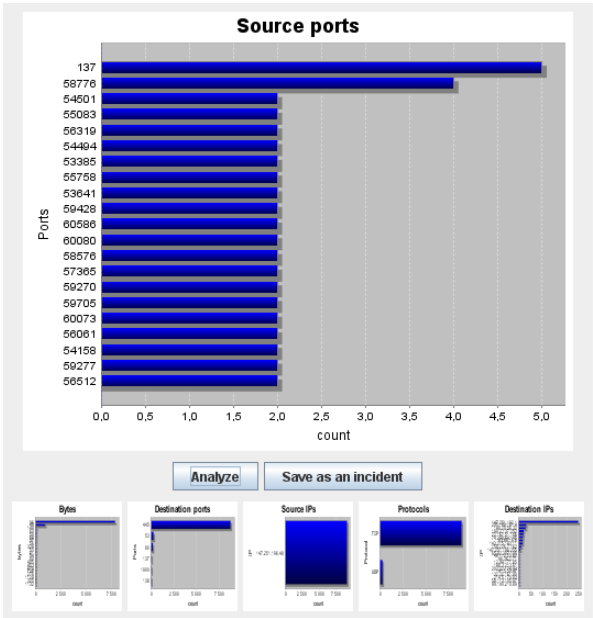


Figure 16: Analyzer window representing Conficker worm traffic distribution by source ports shows great variability (only 21 are shown).

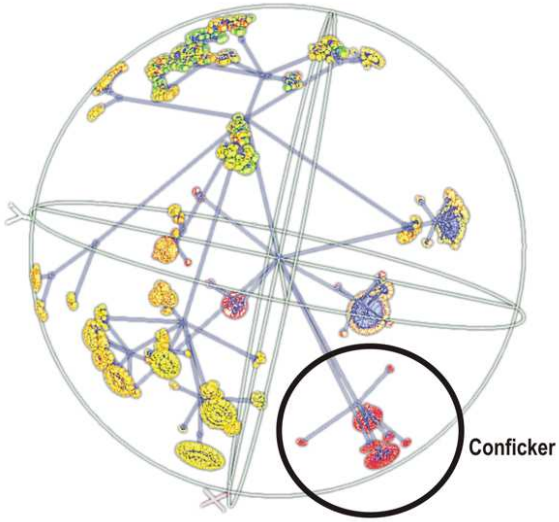


Figure 17: 3D representation of the trust model showing the whole traffic. The Conficker worm traffic (marked and red colored) is clearly separated from the legitimate traffic situated on the top of the sphere.

Conclusion

Continuous Internet traffic increase requires the deployment of high-speed network links to match the bandwidth demands. Besides all beneficial effects, this new high-bandwidth infrastructure presents novel challenges in the domain of security, as the manual oversight of such high traffic volumes is nearly impossible.

Figure 18 shows an optimal network behavioral analysis and incident handling framework. Various constrains (technical, budget, legal, etc.) may appear during deployment of such a framework. In previous sections we described system and tools we use to fulfill our daily operational needs. On the other hand, they are other solutions which may better fit to your needs. We mention some findings we learned in the past to help you to decide.

Accurate traffic measurement is a crucial part of network anomaly detection. However, it is even more important for on-line network behavior analysis systems, that must work at line rate, and can not afford the classification errors related to sampling. Sampling can negatively impact the detection. This may lead to later discovery of initial stages of infection, and make the reaction less efficient.

Many existing systems that have a real-time traffic input have scalability issues that restrict their deployment to lower speed links. Fine-tuned tools and operating systems are essential to overcome these limitations. Consider hardware acceleration for 10 Gb/s, 40 Gb/s and 100 Gb/s network monitoring.

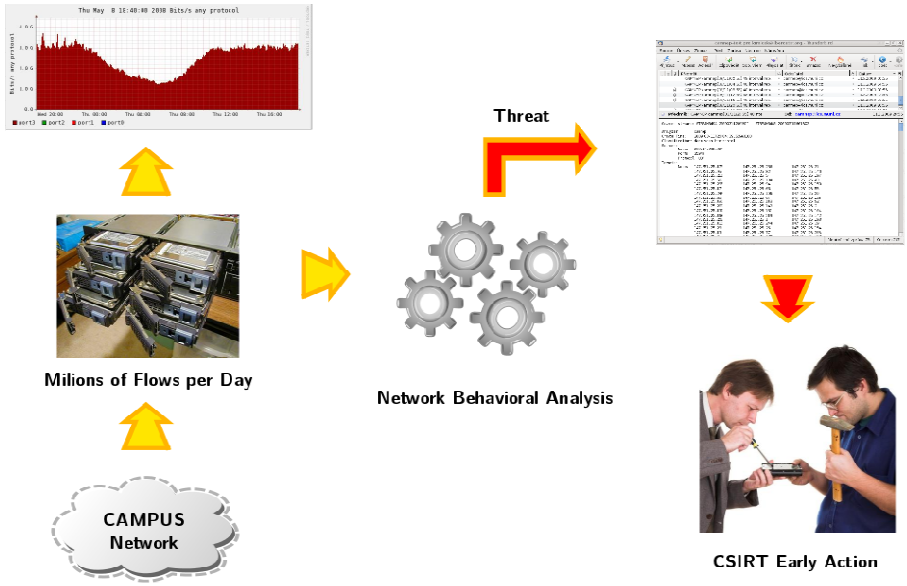


Figure 18: Network behavioral analysis and incident handling framework.

Detecting attacks in flow data sets is typically based on the anomaly detection paradigm. Detection methods build a traffic model and compare the model's predicted results with the actual traffic observed. The differences are assumed to result from an attack or technical problems, and they are reported to system administrators for further analysis. NBA systems may suffer from a high rate of false positives (legitimate flows labeled as malicious) and consequently a high volume of false alarms.

References

[ANUK04] 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.

[ANUK05] 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.

[CAMNEP] CAMNEP Project. http://www.muni.cz/ics/camnep, http://agents.felk.cvut.cz/projects/camnep/

[CEL10] Čeleda et al. Chuck Norris Botnet Analysis. http://www.muni.cz/ics/research/projects/4622/web/chuck_norris_botnet

[CYB10] CYBER Project. Chuck Norris Botnet - NfSen Detection Plugin. http://www.muni.cz/ics/cyber

[FSEC09] F-Secure. Preemptive Blocklist and More Downadup Numbers, 2009. http://www.f-secure.com/weblog/archives/00001582.html.

[HAAG1] Peter Haag. NFDUMP - NetFlow processing tools. http://nfdump.sourceforge.net/, 2011.

[HAAG2] Peter Haag. NfSen - NetFlow Sensor. http://nfsen.sourceforge.net/, 2011.

[LIB11] Liberouter Project. FlowMon Probe. http://www.liberouter.org/flowmon/index.php, 2011.

[POR09] Phillip Porras, Hassen Saidi, and Vinod Yegneswaran. An analysis of conficker's logic and rendezvous points. Technical report, 2009. http://mtc.sri.com/Conficker.

[ZAD10] Martin Žádník. Network Monitoring Based on IP Data Flows. Best Practice Documents, 2010. http://www.terena.org/activities/campus-bp/pdf/gn3-na3-t4-cbpd131.pdf.

[ZHA07] 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.

Glossary

API	Application Programming Interface
CAMNEP	Cooperative Adaptive Mechanism for Network Protection
C&C	Command & Control
CERT	Computer Emergency Response Team
CSIRT	Computer Security Incident Response Team
DDoS	Distributed Denial of Service Attack
DNS	Domain Name System
DSL	Digital Subscriber Line
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
IPFIX	Internet Protocol Flow Information Export
IRC	Internet Relay Chat
NBA	Network Behavior Analysis
NfSen	Netflow Sensor
NREN	National Research and Education Network
RPC	Remote Procedure Call
SNMP	Simple Network Management Protocol
SPAN	Switched Port Analyzer
SQL	Structured Query Language
SSH	Secure Shell
TAP	Test Access Port
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
WWW	World Wide Web