03-03-2011

Deliverable DJ2.4.1: Specification of Advanced Methods for Incident and Security Threats' Detection and Mitigation in a Multi-Domain Environment

Deliverable DJ2.4.1

Contractual Date:	30-11-2010
Actual Date:	03-03-2011
Grant Agreement No.:	238875
Activity:	JRA2
Task Item:	Task 4
Nature of Deliverable:	R (Report)
Dissemination Level:	PU (Public)
Lead Partner:	GARR
Document Code:	GN3-10-352
Authors:	S. Venuti (GARR), A. Sevasti (GRNET), Michelle Danho (RENATER), Carlos Fuentes (REDIRIS), Vladimir Gazivoda (MREN), Jan Kohlrausch (DFN), Vojtech Krmicek (CESNET), Ladislav Lhotka (CESNET), Hank Nussbacher (IUCC), Alperen Sirin (ULAKBIM), Emre Yuce (ULAKBIM)

Abstract

This deliverable provides an overview of technologies and opportunities for advanced methods for incident and security threat detection, with solutions to be deployed both within the National Research and Education Networks and in the inter-domain environment, embedded in the multi-domain management processes.

Table of Contents

Executive Summary

This document provides an overview of new methods and technologies for incident detection and handling, highlighting opportunities for research conducted by JRA2 (Multi-Domain Network Service Research) Task 4 (Security). The primary goal of this research is to identify ways to improve how NRENs deal with security problems within their domains and within the multi-domain GN3 environment.

This deliverable and the work of JRA2 Task 4 in general are not intended to address any possible solution or method for detecting or mitigating threats, but focus on the most suitable solution in terms of feasibility for the GN3 environment. The main goal is to reach a harmonised environment of work and a common basis for interoperability. To achieve this goal, the deliverable first approaches security aspects by taking a close look at the threats that need to be faced in everyday work, including SPAM, malware, viruses, phishing and so on. Consideration is also given to how these threats evolve in order to study adequate countermeasures. For threats that are still unknown, the document discusses new methods that can be employed to detect them (using honeypots).

Against the backdrop of the threats that the GÉANT network, users and NRENs need to be protected from, the deliverable next discusses methods that can be employed to detect threats and prevent damage. Several incident detection methods need to be researched in this context, including tools for monitoring NetFlow packets and finding anomaly detection algorithms (Nfdump and NfSen, developed by SWITCH). Another branch of incident detection involves research on the automation of alerts, which are triggered if inappropriate behaviour is detected in the network.

Where incidents cannot be prevented, processes need to be in place to identify vulnerabilities that were exploited and introduce measures to prevent the incident from recurring. The document describes new research concerning best practises for investigating an incident through forensic analysis when the worst has already happened.

Once threats or attacks have been identified, it is important to mitigate the damage that could be done by informing the appropriate stakeholders within the network. In view of this, the document discusses information transfer within a multi-domain environment, including procedures and standards.

Finally, the deliverable moves the focus to the nuts and bolts that underlie security threat handling, discussing what network devices and methods may need to be put in place to find bad traffic, mitigate damages and automate procedures.

1 Introduction

The main goal of this document is to provide an overview of the technologies and opportunities for researching new incident detection methods and how they can be applied in the intra- and inter-domain environment of GN3.

The security activities within the National Research and Education Network (NREN) community have developed a valuable set of tools, procedures and countermeasures to handle evolving threats and mitigate their effects on the network, its component, services, and ultimately its end-users. However, most of the existing consolidated methods comprise reactive actions, and many of these actions still rely on the manual handling of sometimes complex procedures. There is a need to exploit better and more adaptable monitoring tools to enable early detection of problems, including better handling of complex traffic-data analysis, and to shorten the manual intervention time counteractions require.

The advanced security research will focus mainly on new, innovative and advanced methods for incident and security threat detection.

The document is structured as follows.

- Chapter 2 focuses on new threats that NRENs are experiencing, explaining the new behaviour of old, well-known threats and methods for finding new threats. The chapter also looks at the tools nfdump and NfSen which are currently used by many NRENs to analyse NetFlow.

- Chapter 3 focuses on methods for incident detection. This includes tools and best methods for early detection, i.e. algorithms for Anomaly Detection, early alerts and methods to investigate incidents while occurring or "post mortem" by forensic analysis.

- Chapter 4 investigates methods to mitigate the threats and the number of incidents, trying to find techniques to make the procedures as automated as possible. Problems that can be experienced in a multi-domain environment are being considered and an attempt is made to find methods to harmonise the type and the content of the information to be sent to the NREN involved in the incident. Particular attention is given to the X-ARF format.

- Chapter 5 explores new incident detection features that network devices offer. These devices include routers, security appliances (both firewalls or proxies and intrusion prevention/detection systems), and protocols (both in monitoring and mitigation functions).

2 Specifications of Information Sources

2.1 Existing Information Sources: Threat Trends

2.1.1 SPAM

Spam is unsolicited and bulk email sent without the verified permission of the receiver. Delivery failure messages, misdirected messages and messages from system administrators are examples of unsolicited emails which are not spam.

Spammers use lists of email addresses that have been harvested using different methods, for example, crawling web pages, using search engines, sending chain emails, attacking dictionaries by sending blank spam, analysing domain queries and using malwares or bots.

Different spam techniques and the content of spam have evolved over time as spam filters have become able to detect spam. Usually this is done through analysing spam content such as the subject or body part of the email. First examples of spam included plain text messages and identifying identical plain text sent in large quantities was not a big problem. Personalised emails (emails that include the receiver's name, e.g. "Dear John" etc.), spam obfuscation (e.g. changing "Viagra" to "v1agra"), American Standard Code for Information Interchange (ASCII) art and HyperText Markup Language (HTML) emails constituted the next steps in the evolution of spam content. Another method to bypass spam filters is using images in the message content. Spammers also use animated images or add noise to image spam. Also, blank spam is sent to identify true email addresses. If a blank mail is sent to a false address it is returned with a system or server or user error. If the blank mail is not returned, the email address is real. Blank spam may also include hidden HTML or JavaScript codes to download malware.

As spam content evolves, different countermeasures are taken to reduce spam activities. While there is no single way to stop spam, different approaches exist to reduce the amount of spam that is received. These include rejecting emails from domains/IPs that are likely to send spam (black/white listing, DNS-based Blackhole Lists (DNSBLs), DNS Sender Policy Framework (SPF) records, filtering the outgoing Simple Mail Transfer Protocol (SMTP) port, grey listing etc.), more advanced methods like analysing message content and deciding whether an email is spam.

Another measure to counter spam is deploying a honeypot. This is a computer system which imitates an open mail relay or an open proxy server. A spammer who probes the network for an open mail relay or a proxy server will try to send spam through the honeypot and thus reveal some information about their identity. Received spam may be discarded, used for DNSBLs, kept to be analysed or the spammer's IP may be revealed.

Other than taking administrative countermeasure against spam, end users should be informed about the spam issue. They should be warned not to reply to spam and not to open any spam attachments. Also, antivirus programs should be deployed and updated regularly. Announcements about recent phishing messages should be delivered to the users.

According to the “2010 Q3 Internet Threats Trend Report” [ITTRQ3], an average of 198 billion spam/phishing messages is sent every day. Pharmacy is reported to be the most popular topic in these messages. Another report, “State of Spam & Phishing” [SSPNov], prepared by Symantec includes recent statistics about spam activities in October.

2.1.2 Phishing

Phishing is a type of Internet fraud that mainly aims to obtain personal credentials for email accounts, bank accounts, epay systems or social media. Phishing messages usually include the address of a web page that is a copy of a legitimate web page. The users are urged to submit their personal information via a form in the copy page. After gaining access to users' credentials via the fake site, fraudsters may remove money from bank accounts or sell these credentials.

Phishing tricks include using similar Uniform Resource Locator (URL) addresses as the legitimate site. For a system hosted under "www.examplebank.com", phishing message may redirect user to an address like "www.login-examplebank.com". In addition, a link to a website in a phishing message can also open a different website on click. This can be achieved by sending HTML spam.

A DNS technique to hide phishing sites is Fast Flux. The basic idea behind this technique is to associate numerous IP addresses with a single fully qualified domain name, where the IP addresses are exchanged very frequently by changing DNS records.

Pharming is another phishing method that aims to gain access to user credentials. Fraudsters redirect users to a fake site by rearranging the DNS record for the official site rather than sending a link in an email. This may be done either on a user’s computer (e.g. by changing the hosts file) or on a DNS server (e.g. by exploiting a vulnerability).

Warning users against possible phishing attacks plays an important role in mitigating phishing attacks. To be aware of the latest phishing threats the “Anti Phishing Working Group” was established [APWG].

2.1.3 Malware/Viruses

Malware is the short name for malicious software including viruses, worms, trojans, rootkits, spyware and adware. They use different methods for spreading, infection and exploitation.

The very first examples of malwares can be considered as the Morris worm (1988) [WikiMorris] and the Melissa virus (1999) [WikiMelissa]. These examples should be analysed as their replication and propagation methods are still valid.

Today, malicious software uses various ways including social networks, instant messaging, Peer to Peer (P2P) connections, Internet Relay Chat (IRC), emails and Universal Serial Bus (USB) sticks for spreading. The purpose of a malware may be advertising (adware), analysing user behaviour (spyware), obtaining user credentials or making the user part of a botnet.

A new kind of threat that was detected in July 2010 is the Stuxnet worm. This malware is mainly designed to gain control of an industrial system by reprogramming logic controllers [WikiPLC]. According to the "W32stuxnet Dossier" by Symantec [NLE10], Stuxnet malware uses various Microsoft Windows vulnerabilities, updates itself through a P2P mechanism and attempts to bypass security products.

2.1.4 Botnet

"A botnet is a collection of software agents, or robots, that run autonomously and automatically. The term is most commonly associated with IRC bots and more recently malicious software, but it can also refer to a network of computers using distributed computing software." [WikiBot]

During 2009 botnets became wider and spread all over the world, posing the biggest current threat. Botnets are a big money affair; they can be rented to send spam or advertising or to create a distributed Denial of Service (DoS) to paralyse almost every service on the Internet.

In turf wars between different botnets Distriuted Denial of Service (DDoS) is used as the method of attack. This results in significant waste of network resources for service provider over the network, due to bandwidth saturation or a frozen CPU in the network-access router, and in a large number of infected or compromised hosts (estimated to be about 30 millions, according to InfoSecurity magazine [InfSecMag]).

Because of their large spread and the large amount of money that bot herders can earn, botnets are evolving in different ways:

- By infecting systems:

  Being invisible to the system allows botnets to pass antivirus software unnoticed and also to use root-kits installed on unusable sectors of a disk drive that are not accessible to the operating system.

  Also, a bot-virus often patches the system that it has compromised and exploits a vulnerability that allows it to be the only one to control the host computer. Botnets tend to be as specific as possible to

the task they are assigned to do: for example, there are spam-only botnets, like Conficker and DDoS-only botnets like the Mariposa botnet, while multipurpose botnets, like the Zeus botnet, are decreasing.

While botnets were previously used for spam, phishing and spyware, they are now also used to fetch any data from computers that can be used to send specific and targeted spam (not only email contacts or credit card numbers but also login credentials for social network platforms).

- Using the communication protocol:

  Botnets tend to pass from an architecture with a centralised server (single point of failure) to an architecture without server, using P2P protocol. This technique has the advantage to be almost invisible to the monitoring of network traffic. Botnets also tend to use an encrypted version of the protocol to bypass Intrusion Detection Systems (IDSs) (based on intercepting specific patterns in packets payload).

The best way to mitigate botnet spreading is to find methods to compose a reliable list of known Command and Control (C&C) servers (to be added to the ones available on internet, like ShadowServer) to throw away all the traffic to and from them (if possible).

2.1.5 DoS and DDoS

A DoS attack or DDoS attack is an attempt to make a computer resource unavailable to its intended users. Although the methods, motives and targets of a DoS attack may vary, it generally consists of the concerted efforts of a person or people to prevent an Internet site or service from functioning efficiently or at all, temporarily or indefinitely.

Many DoS attacks are used to extort money from website owners, and they are evolving quickly. The following DoS attacks exit:

- P2P DoS attacks use the absence of a central server to obfuscate the command communication channel, like the botnet.

- DoS attacks against wireless and mobile data networks cause congestion at the radio network controller (RNC) at a Global System for Mobile Communications (GSM) or General Packet Radio Service (GPRS) provider.

- DoS attacks against mobile phones cause battery drain by preventing the mobiles from going into sleep mode.

- DoS attacks that use compromised web servers as zombies, instead of botnets or DoSnets. Due to the larger bandwidth that is generally assigned to a web server, these attacks can cause as much damage as 50 bots with only one compromised server.

The only defence method to mitigate a DoS or DDoS attack is to find the actual origin and totally filter out or limit the bandwidth for the offending IP.

2.1.6 Prefix Hijacking

Prefix hijacking (also known as IP hijacking [IPhij]) involves the illegitimate takeover of certain IP addresses via false Border Gateway Protocol (BGP) announcements to the global routing table. The Internet is based on the BGP routing protocol, which allows any organisation with an Autonomous System Number (ASN) to announce their IP prefixes to the Internet. IP prefixes are an aggregate announcement of all IP addresses that belong to an organisation. Typical IP address prefixes might be a /24 (255 IP addresses) or a /16 (65025 IP addresses). There are many other prefix lengths.

Prefix announcements are via BGP to an upstream ASN – typically an Internet Service Provider (ISP) or NREN. These announcements are based primarily on trust but each peer is supposed to do sanity checking to determine whether the announcing ASN is allowed to announce a specific prefix. ISPs will check the RIPE whois database [RIPE] and build appropriate access list filters so that only valid IP prefixes can be delivered from a peer. However, not all ISPs do this sanity checking and therefore, either by mistake or on purpose, a wrong prefix can be announced. Typically, IP hijacking involves malicious attempts to announce prefixes to the global Internet and thereby divert incoming traffic to that IP range. This can then be leveraged into a “man in the middle" attack, where the captured data could be either be merely copied locally, or even manipulated and sent onward to the intended target.

A protocol rule of BGP is that a more specific prefix will take precedence over a more general prefix. Therefore, if an NREN announces a prefix of 100.1.0.0/22 (IP range of 100.1.0.0-100.1.3.255) a malicious announcement of 100.1.1.0/24 (IP range of 100.1.1.0-100.1.1.255) would usurp the previous announcement and all data destined to 100.1.1.0/24 would flow to the wrong destination on the Internet.

There are no known methods for protecting against this type of attack, only reactive email alerts can be sent. The NREN would then have to contact the ASN that is not doing proper filtering and ask them to correct the situation. This typically takes a number of hours.

2.1.7 Internet Alerts

Numerous free and online alerting systems can be used to determine whether an IP in a network is infected. These systems are based on honeypots and sensors spread across the Internet to report attacks they receive from various IPs. While these IPs may be spoofed, they often belong to the IP range of the targeted network, indicating that one or more of its systems have become compromised.

- Team Cymru

  The Team Cymru [CYMRU] site provides targeted reports specific to ASNs. These reports can, for example, include phishing sites, malware sites and botnet C&Cs. This is based on a data exchange agreement where Team Cymru provides the daily report data in exchange for data from the data exchange partner (NREN). This might be Darknet, BGP, flow, passive DNS or other data. Team Cymru also provide the TCConsole service [TCC], a web console for viewing graphical and statistical representations of malicious activities.

- Google Safe

  Google Safe [GoogleS] provides information about malware and phishing sites inside a network. The alerts provided by this site are based on ASNs and, in general, will only be sent to system contacts registered in RIPE Whois database for the ASN in question (see [GoogleB] for more information).

- Shadowserver

  The Shadowserver [Shade] organisation also provides daily email reports on bots, open proxies, spam relays, C&Cs etc. (see [ShadeRep] for more information).

- Cyclops and BGPmon

  Some sites can be used to help determine if an IP range is being hijacked, for example,. Cyclops [CYCLOPS] and BGPmon [BGPmon]. Each involves a process where the user registers their ASN and/or IP ranges and determines the level of reporting they would prefer. A user could, for example, receive an email alert if a more specific IP prefix is suddenly seen on the Internet that has a different ASN path than theirs. Since each system uses different BGP feeds and sensors, both should be used to ensure that potential IP hijacking alerts are not missed.

2.1.8 Advertising Honeypots

A honeypot is a decoy system placed in the network to detect and analyse attacker behaviours. A honeypot is useful for network security if it is able to get the attention of the attackers. To successfully analyse the manipulation of traffic using honeypots the following questions need to be considered:

- Where in the network should the honeypot be deployed?
- Which operating systems would be most useful for the honeypot to imitate to optimise the detection and analysis of attacks?
- Which methods do intruders use?

Intruder profiles depend on the position of the honeypot in the network:

- A honeypot deployed on a Local Area Network (LAN) will be targeted by internal intruders. An example of this is an employee searching the LAN for vulnerable machines.
- A honeypot deployed on a demilitarised zone (DMZ) can imitate the web server of the company. In this case the honeypot will be targeted by attackers outside and inside the company.

Honeypots may be classified based on whether they are deployed in a production or research network. Production honeypots are low interaction honeypots that are easier to deploy and less risky for the network. They provide less information about the attacks and the attackers than research honeypots do. Research

honeypots are used to research the threats that organisations face. They are complex to deploy and maintain, however, they capture detailed information about the threats.

Honeypots imitate valuable sources (such as databases, credentials, services etc.) to attract intruders. Selecting which sources a honeypot contains affects how much traffic it attracts. More popular services, such as SMTP or Hypertext Transfer Protocol (HTTP), receive more traffic than File Transfer Protocol (FTP). Deploying systems that have known vulnerabilities attracts more attackers. Honeypots also need to successfully imitate data sources to be attractive to attackers. The names chosen for hosts, databases and services should be interesting to attackers. For example, a honeypot imitating a mail server may have the host name “NREN_mailserver”.

The level of interaction (high, medium, low) determines the length of an intruder’s stay. Determining the level of interaction is a balancing act. If the level of interaction is high, the honeypot is more likely to be compromised.

It is also important to understand the path attackers follow when they find a vulnerable target. The main attack method used in an IPv4 network is scanning the address space for vulnerable targets. The honeypot should gain the interest of the attacker probing the network. This may be achieved either by placing the honeypot as a regular user computer/server or by redirecting all unused IP addresses to the honeypot (blackholing). The honeypot may be deployed by using the first few IP addresses of the block in case the intruder scans the subnet in order (e.g. 192.168.0.2 for 192.168.0.0/24 subnet).

In addition to probing, attackers use bots that crawl websites to find IP addresses that they then check for vulnerabilities. This type of attack may be attracted by placing hidden links to the honeypot which are not visible to legitimate users and cannot be clicked by them.

Another possible network configuration that aims to detect attacks and simultaneously protect a production server (e.g. web server) employs shadow honeypots [SHADOW-HONEY]. Shadow honeypots are tightly coupled with a production server, sharing state and content, and do not have to be directly accessible to a network attacker. Untrusted or suspicious traffic, destined towards the actual server, is diverted to the honeypot which evaluates it on a mirror of the contents and state of the production server. If the traffic is malicious, the honeypot takes appropriate measures to record as much information as possible about the attack, while the attack never reaches the production server. If the traffic is benign, the honeypot, having a mirror of the actual server content, is able to successfully serve a potential request or it may forward the traffic to the production server.

2.2 Deploying Honeypots

Honeypots have proven their importance in capturing malware including Internet worms and are very valuable in keeping track of global malicious activity on the Internet. In the context of the GÉANT project they are planned to complement network-based monitoring components to facilitate a precise analysis of global anomalies that are, for example, caused by Internet worms.

Two classes of honeypots have evolved from the different aims of honeypot deployment:

- Low-interaction honeypots

  Low-interaction honeypots are systems that emulate vulnerable services or backdoors. Their purpose is to collect as much statistical data as possible about known attacks and compromised hosts. This includes the capture of malware that is related to the attacks. These honeypots typically pose as finite state machines that emulates the behaviour of a vulnerable service. Each state corresponds to a specific step of the attack. Thus, for each attack a corresponding state machine has to be designed. However, the finite state machines require significantly less resources than the corresponding operational service or program. This efficiency allows a single physical machine to monitor a large network space.

  A drawback of low-interaction honeypots is the roughness of the simulation. A more accurate simulation would computationally be too expensive in comparison to the corresponding operational service. For that reason, most unknown attacks fail to interact with the simulated service before the vulnerability itself is triggered. To alleviate this, Dionaea implement significant parts of the Windows Server Message Block (SMB) protocol and Remote Procedure Call (RPC) services. However, this implementation is very expensive and can only cover fractions of the Windows functionality.

- High-interaction honeypots

  The limitations of low-interaction honeypots can be completely overcome by using a productive operating system as a honeypot (a high-interaction honeypot). This is highly advantageous for spotting new attacks and zero-day exploits, because all unknown network traffic and binary execution can be monitored However, since high-interaction honeypots deploy productive services they can be compromised like other insecure systems and pose a security threat to the Internet. In addition, attacker who compromised the honeypot must be prevented from being able to manipulate or create forged data. High-interaction honeypots also require much more resource than low-interaction honeypot which limits the monitored address space.

To combine the individual advantages each type of honeypot offers, sensor networks combine both types. Low-interaction honeypots are deployed to compile statistical data about attacks and to capture malware involved in these attacks. If new attacks occur that cannot be handled, they are processed by high-interaction honeypots. This allows the targeted vulnerability to be identified which is critical in the case of previously unknown vulnerabilities (these are commonly called zero-day vulnerabilities).

Well-known low-interaction honeypots are Nepenthes [NEPENTHES], Amun [AMUN], Honeyd [HONEYD], and Dionaea. While Honeyd focuses on the simulation of networks, Nepenthes, Amun and Dionaea aim at capturing malware. However, there is no further adaptation of Honeyd and Nepenthes to handle new attacks. Amun allows its users to enhance its detection capabilities by writing their own vulnerability modules. Among these, only Dionaea is actively supported by the HoneyNet project (GSOC Project #10) and SurfNet.

In contrast to other high-interaction honeypots, Argos has a generic attack detection engine that reliably stops the attack before the honeypot is compromised. Therefore, it can be more easily deployed than other high-interaction honeypots. No expensive monitoring of the honeypot is required to prevent abuse.

The following sections describe the different honeypots in detail and point out how they can be deployed for the GÉANT network.

During the GN3 project all the methods described below will be implemented and their advantages and disadvantages analysed so that recommendations can be made to NRENs, allowing them to decide which to implement.

2.2.1 Argos High-Interaction Honeypot

Argos is a high-interaction honeypot that deploys a native operating system which uses a generic method for attack detection. It can detect malicious activity on most common operating systems including Linux and Microsoft Windows. Although the attack detection is limited to the vulnerability class of buffer overflows and related vulnerabilities, this class is still the most threatening. The benefit of this focus is that the attack detection does not depend on a specific operating system or program, and that the method is very accurate and reliable. These features make Argos well suited to detect and analyse zero-day vulnerabilities.

Compared to Dionaea, Argos' attack detection is more accurate. While Dionaea treats all connections as malicious, Argos allows malicious activity and accidental received connections to be distinguished. However, it should be noted that Argos requires much more computation resources than Dionaea. For that reason, honeypot networks should deploy both types of honeypots to benefit from their individual advantages.

Argos uses the technique of dynamic taint analysis for attack detection. This method is based on the assumption that all data originating from the Internet is potentially malicious. The data is marked as tainted and the data flow of all tainted data is monitored. Malicious behaviour is detected through the inspection of the usage of tainted data. If such data manipulates the control flow in a specific way, evidence of a successful attack is found.

Technically, the specification of malicious behaviour in Argos consists of the direct execution of tainted data (e.g. in form of shell code) or if it is loaded in the instruction pointer of the Central Processing Unit (CPU). This happens, for example, if a function pointer or the return address of an activation record on the stack segment is overwritten by tainted data. It is important to note, that almost all exploits for buffer overflow or related vulnerabilities apply this technique to redirect the control flow of the process. Moreover, nearly all exploits for this class of vulnerabilities have to use tainted data in such an illegitimate way. For that reason, dynamic taint analysis is very accurate and reliable in detecting all successful attacks that target buffer overflows and related vulnerabilities. An additional advantage is that all data that is critical to the analysis of the attack is recorded. This data includes the exploit string and the shell code which is sent to the vulnerable program during the attack. Among other approaches, Microsoft's Vigilante, Argos, and Minos rely on dynamic taint analysis.

Argos is based on the Qemu machine emulator. In contrast to other virtual machines like VMware, Qemu emulates the complete hardware environment. Since Qemu is distributed as open source under the GNU General Public License (GPL), it can be arbitrarily modified and adapted for attack detection. The extensions implemented by Argos affect the CPU and the virtual network card (Network Interface Controller - NIC). The virtual NIC is extended to mark all received data as being tainted. The CPU is instrumented to perform the

dynamic taint analysis. Because of the generic attack detection, all operating systems, which are supported by Qemu, can be set up inside the Argos honeypot.

2.2.2 SSH Low-Interaction Honeypot

Currently, identity theft is a major threat on the Internet. There are different ways to take over the identity of a user. Attacks target weak passwords, vulnerable applications (especially web applications) and the authentication infrastructure of operating systems. A brute-force approach, in which a list of password is used to try to access the system, can be employed to guess passwords or to attack encrypted passwords. For the latter, the attacker has to gain access to a list of encrypted passwords which are used by all well-known operating systems for user authentication. This can be achieved, for example, by abusing a directory traversal vulnerability from which a large list of web applications suffered in the past. In addition, a large number of malware programs on Windows systems contain a keyboard sniffer that monitors the user input to applications (e.g. web applications). A promising target for stealing user passwords are Secure Shell (SSH) servers or clients. They are manipulated in such a way that passwords are recorded and sent to the attacker. Attackers also look for private SSH keys on compromised machines and try to abuse them to get access to connected systems. Usually, the stolen passwords are stored in a secret file on the local computer and are sent to a central server (typically known as a dropzone) that is under the control of the attacker.

A significant identity theft threat are stolen SSH passwords or keys that use the methods described above. This kind of incident is particularly important due to the widespread trust relationships between sites. The attacker can leverage these relationships to compromise a large number of related servers or sites. Especially Grid sites are at an increased risk as they are highly interconnected and require a large list of users to be shared by these sites that all connect from the Internet. A starting point is a stolen user account that allows the attacker to execute code with the privileges of the user. Typically, the attacker first tries to elevate his privileges to gain access to the root account. This makes it possible to modify the SSH server or client so that passwords of other users can be found out. In addition, the attacker is able to monitor SSH connections to other servers or sites. This can be leveraged to attack these sites in a viral manner.

There are different methods to detect scans for weak SSH passwords and compromises of SSH servers. SSH honeypots are a complement to network-based methods to detect malicious activity. While the latter can reveal high volume anomalies in SSH network traffic, honeypots are able to add details about these attacks, including the attacker's list of submitted passwords. A strategy to conceal scans is to significantly reduce the traffic volume and connections by distributing the load between multiple systems. Since the number of connections and the traffic load of each system are relatively low, such distributed scans are difficult to detect by network-based monitoring. However, such scans can be detected and analysed on a honeypot.

2.2.3 Nephentes and Dionaea Low-Interaction Honeypot

Nepenthes is a low-interaction honeypot designed to capture malware related to known attacks. It is motivated by the observation that most malware is distributed by well-known Internet worms and other automated attacks. Since the complete data exchange during these attacks is known, no real operating system is required to

communicate with the attacker. Instead a finite state machine identifies known protocol requests or other network traffic and selects a suitable constant answer which is sent to the attacker. The finite state machine is manually designed in a way that allows the attack to proceed. For example, most exploits for vulnerabilities in Windows have to start a protocol negotiation (e.g. to be able to call functions over Distributed Computing Environment / Remote Procedure Calls (DCE/RPC)) in order to be able to exploit the targeted vulnerability. The main advantage of the finite state machine is its efficiency. This allows a single machine to monitor a large network.

During most attacks, code is injected into the vulnerable process. The vulnerability is exploited in such a way that the control flow is diverted to execute this code. Historically, the code starts an interactive shell which allows the attacker to manually control the compromised system. For that reason, the code is typically denoted as shell code. The most important task of current shell code is to subsequently download and install other malware after the attack. The code can thus be leveraged to capture the malware. After the finite state machine processes the attack, the code is extracted and analysed.

Nepenthes architecture consists of different types of modules that emulate vulnerable services and backdoors (vulnerability modules) for parsing and analysing shell code (shell code parsing modules), for downloading files from HTTP and FTP servers, and for logging the results. The data flow between the modules is provided by a central dispatcher that invokes the appropriate module and supplies the data to it. For example, if shell code has been captured, the dispatcher for shell code passes this data successively to all shell code parsing modules until one is able to successfully parse the shell code. Additional modules for new attacks can be added to the Nepenthes framework.

For each new attack or vulnerability, an individual module has to be manually developed that implements the communication with the attacker or attack tool. In addition, a shell code parsing module is usually required for a further analysis of these attacks. This process is very error prone and time consuming and, therefore, the major drawback of Nepenthes.

Dionaea was designed to overcome these drawbacks. Although the basic idea of Dionaea is similar to Nepenthes, it has a more generic design. In contrast to Nepenthes, Dionaea focuses on protocol simulation and uses an advanced technology to interpret shell code. Instead of vulnerability modules, Dionaea implements the most relevant protocols including the Windows SMB and DCE/RPC functionality. Since this approach is more generic than vulnerability modules, even unknown attacks can be processed. For example, the protocol implementation covers the functionality required by the W32.Conficker worm to communicate with the honeypot until the shell code has been sent. Although no specific adaptation has been conducted for the Stuxnet malware, it is likely that Dionaea could capture this malware.

The shell code detection and interpretation of Dionaea is based on the Libemu library. Libemu uses a list of heuristics to pinpoint the starting point of assembly code (this code is usually denoted as shell code) which is executed during the attack. Having found this, the library executes the assembly code in a simulated Microsoft Windows environment and records important library calls, i.e. calls to Windows libraries that provide functions to communicate with the Internet (e.g. upload of Malware) and to execute uploaded programs. Thus, Libemu is able to identify and interpret shell code. This allows all malware to be captured that is intended to be

downloaded after the initial compromise. An advantage of this approach is that it is generic and can cope with highly obfuscated shell code.

2.3 NetFlow Analysis Tools

2.3.1 nfdump

nfdump [nfdump] is a set of tools to collect and process NetFlow data. It is fast and has a powerful pcap-like filter syntax (eg. tcpdump) that supports NetFlow versions 5, 7 and 9 as well as a limited set of sflow. It is also IPv6 compatible.

nfdump is an efficient and flexible tool for analysing NetFlow data quickly. It has a complete and versatile set of command line instructions. nfdump was tested and recommended during GÉANT2, and is already widely used among NRENs. It has been developed by SWITCH and is distributed under the Berkeley Software Distribution (BSD) License. JRA2 Task 4 will research it in depth and may develop its functionality further in collaboration with SWITCH:

- Currently, nfdump is unable to generate output to a database, it can only dump results as plain text or in the nfdump binary format. It would be useful to add functionality for dumping results to MySQL or another database management system.

- The nfdump filter syntax does not support filters like “src as = dst as”. Although there are not many cases where such filters would be needed, it would be desirable if nfdump's filter syntax supported such filters.

- nfdump does not have a tagging feature which some other flow-processing tools like flow-tools have. There are not many cases in which tagging would be needed, but it would be a desirable to have it.

2.3.2 NfSen

Nfsen is the graphical web-based front end for the nfdump netflow tools. It provides a user-friendly interface for accessing the nfdump functionality (viewing flows, traffic statistics, filtering traffic data) but also provides features for detecting incidents and investigating problems:

- Graphs can be generated that represent the time series of monitored elements or their aggregation.
- Automatic alerts can be triggered.
- New plugins can be created using the built-in API.

Nfsen can also assist in the detection of security threats:

- Graphical charts of traffic data can help in identifying anomalies in traffic (e.g. peaks, sudden increase or decrease of traffic).
- From an overview of traffic it is possible to drill down to the details of specific flows to find out what caused a potential anomaly.
- Different types of interesting traffic can be profiles and monitored.

3 Specifications of Cross-Domain Correlation Methods

3.1 Anomaly Detection Algorithms

From the viewpoint of practical network security administration, the most effective anomaly detection methods operate in or near real time. Such methods are able to provide an early warning and give administrators an opportunity to react quickly. In high-speed backbone networks such as GÉANT or most NREN backbones, large traffic volumes limit the selection of real-time anomaly detection methods to the following two fundamental approaches, or combination thereof:

- Non-specific detection based on easily computable metrics.
- Specific detection targeting of a certain class of anomalies only, such as DDoS attacks or botnets.

The first approach allows a broad spectrum of anomalies to be detected and, ideally, also indicates the type of the anomaly. The metrics are computed in real time, typically from either SNMP counters or IP traffic flow data (NetFlow).

The second approach then reduces the computational complexity by looking at only a small fraction of traffic data.

In both cases, other information sources may be taken into account, for instance IP routing data or information collected from honeypots.

3.1.1 Finding Patterns of Bad Traffic in NetFlow Data

Non-specific anomaly detection methods are based on the observation that large-scale attacks or network failures often have a measurable impact on certain metrics such as aggregated traffic volumes. If the “normal operational range” for such a metric can be determined, then the detection algorithm can flag any value that falls outside this range and take appropriate actions.

Standard statistical methods can be used to obtain a sound estimate of the “normal operational range”. A range that is too narrow would cause the detection method to generate false positives, while too broad a range would lead to false negatives. An additional complication lies in the fact that traffic characteristics are rarely static – apart from periodic fluctuations (annual and diurnal), most parameters also exhibit long-term trends. Therefore, most anomaly detection methods re-compute the normal range estimates in regular intervals. However, it is worth noting that regular adjustments of the operational range open the possibility for sophisticated attackers to misguide the detection method by repetitive injections of forged traffic with carefully tuned statistical properties [Rub08].

Brute-force attacks such as DDoS may be easily detected from volume metrics such as packet or byte counts or numbers of IP data flows. One of the anomaly detection methods that utilise volume metrics was developed as a NfSen plugin and tested by the JRA2 activity of the GÉANT2 project [MK06].

However, other types of attack have been demonstrated to have virtually no effect on volume metrics. More subtle metrics capable of detecting a larger set of anomaly types were proposed. These metrics usually indicate the distribution of a selected feature in a traffic sample. A feature in this sense can be any parameter of an IP data flow, typically the destination or source address, port etc. By far the most popular metric is entropy, either in its classical Shannon form [LCD04] or using alternative definitions [Tel09]. The GÉANT2 JRA2 activity tested NetReflex, a commercial tool based on the methods developed in [LCD04]. In the GN3 project, this tool is now used for detecting anomalies in the GÉANT backbone network [Rou10].

Some alternative methods for representing distributions of features draws inspiration from fractal geometry [Koh06]. In general, these methods analyse the scaling behaviour of the distribution of traffic features in a sequence of subdivisions of the feature space (e.g. by varying the prefix length of source and/or destination addresses). These methods have the following advantages:

- The sequences of IP prefixes of decreasing length may be easily computed by applying elementary bit shift operations to IP addresses.

- It is possible to obtain a single metric for joint distributions of multiple features, for instance source and destination addresses.

The fractal methods can either take into account just the simple occurrence of distinct values of traffic features or, in a more complex setting, various measures associated with traffic features, such as the aggregate byte count corresponding to the same pair of addresses.

To further investigate these anomaly detection methods, JRA2 will use NetFlow data from 24 routers connected to the GÉANT backbone as input. JRA2 will also try to store the GÉANT backbone data in a large repository of historical archive that can then be used for fractal geometry anomaly detection: the larger the timeslot of input data, the bigger its accuracy and efficiency might be. In the meanwhile (while collecting a sufficient amount of GÉANT backbone data) JRA2 are going to use the data as input for botnet detection as described below.

3.1.2 Detecting Botnets

Although the research community has presented various works focused on the detection of botnets, the methods discussed in these works have limitations that prevent them from being deployed in modern high-speed networks. These methods either target the detection of particular types of botnets (e.g. the detection of particular IRC patterns in botnet communications) or they provide very complex and resource-intensive algorithms (e.g. multidimensional clustering of the network traffic) that cannot be used for the real-time monitoring of high-speed networks.

Instead botnet detection should provide real-time time outputs to network administrators both in an intra-domain and multi-domain environment. It should be able to detect both existing, known botnet networks and new botnet types. In addition, it should provide an option to perform an implementation of proposed algorithms in the existing network infrastructure and connect the detection system outputs with the existing alert systems. The detection itself would use following resources, available in the JRA2 Task 4 environment:

- IP flow traffic data in NetFlow format. The proposed methods will process the NetFlow data as the main source of information about the current state of the monitored network. The aggregation of the network traffic in the form of NetFlow data enables network traffic analysis to be performed at high speeds in real time (see *Finding Patterns of Bad Traffic in NetFlow Data* on page 16), compared to the inspection of packets payload, which is very resource-intensive and not suitable for GÉANT.
- Threat detection results from the honeypots (see *Deploying Honeypots* on page 9).
- Information from both inter-domain and intra-domain environments.
- Internet alerts that provide lists of known sources of bad traffic (see *Internet Alerts* on page 7).

The following list of NetFlow-based anomaly detection methods are suitable for the real-time detection of botnet networks in high-speed networks.

- Detection of communication between known botnet C&C centres and bots.

  This detection method exploits the knowledge about known botnet C&C centres and provides a list of suspicious hosts (potential bots) in the monitored network in real time. It periodically retrieves the current lists of known C&C centres both from the deployed honeypots (see *Deploying Honeypots* on page 9) and from the Internet alert centres (see *Internet Alerts* on page 7), processes the current NetFlow traffic and reports new suspicious hosts in the observed network to the network operator.

- Detection of the centralised IRC bots.

  This algorithm analyses IRC traffic. It uses the NetFlow data to study its statistical properties and classifies data as standard IRC traffic or potential botnet IRC traffic. Each flow is classified using a defined set of criteria, including Transmission Control Protocol (TCP) flags, average packet sizes, bit-rate, number of packets in the flow, flow duration, port numbers etc.

- Detecting botnets by examining DNS traffic.

  This algorithm is based on the fact that each bot performs DNS queries in order to connect to the C&C server or to download its update from the server. The algorithm distinguishes between legitimate DNS queries from botnet queries, using metrics such as the amount of DNS queries outside the local network, the frequency of DDNS usage, similar DNS behaviour of host groups etc.

- Detecting P2P botnets.

  P2P botnets behave similarly to ordinary P2P networks and therefore they face the connection problems caused by Network Address Translation (NAT) devices, active firewalls and non-responsive hosts. In legitimate P2P networks these problems are solved by "super nodes". However, these are not used in the case of the P2P botnets as they introduce the botnet vulnerability. The ratio of the Internet Control Message Protocol (ICMP) codes with “destination unreachable” and TCP Reset flags represents such behaviour and can be used for the P2P botnet detection.

- Detection of the botnet structures using cluster correlation and spatial-temporal similarity.

  Several works presenting complex approaches to botnet detection were published [USENIX, RAID]. These were based on advanced statistical methods such as clustering the traffic and identifying similar malicious behaviour in the host communication. These approaches entail very intensive computational requirements and are not suitable for real-time monitoring.

  JRA2 will evaluate existing statistical approaches and try to design and implement a fast algorithm for anomalous traffic clustering and botnet identification that is suitable for real-time network monitoring. To maximise algorithm performance, strong flow filtering will be performed in the initial phase with the goal to reduce the amount of flow data for analysis.

It should be possible to implement any of these methods in the existing network infrastructure without special costs being incurred. The methods work independently of each other and provide output that is valuable to network administrators. An advanced correlation algorithm of the outputs could also lower the false positive rate. A modular implementation of the algorithms in the form of NfSen plugins is discussed in *NfSen Plugins* on page 19.

In the course of JRA2 Task 4, a feasibility study of the alternative methods will be conducted in order to recommend the method or methods that are most feasible within the project and suitable to be implemented inside a NREN, pointing out advantages and disadvantages.

3.1.3 NfSen Plugins

The NfSen plugins represent a suitable tool for the implementation of the proposed anomaly detection methods, as described in the previous section. Their advantages are a simple application interface, relatively easy deployment to the existing network infrastructure, user friendly output to the web interface and an option to use an advanced NfSen alerting (see *NfSen Alerts* on page 23).

The plugins are divided into backend and front-end plugins. The backend plugins are initialised the moment the NfSen collector is started. They provide several functions including regular data processing, alert conditions and alert actions. The front-end plugins visually display results of the backend processing through the NfSen web interface. Backend plugins are implemented using the Perl scripting language and front-end plugins are implemented as PHP: Hypertext Preprocessor (PHP) files. Both plugins may exchange relevant data over the standard UNIX socket. Details can be found in the NfSen documentation [NfSen].

Such plugin architecture allows specific anomaly detection methods to be implemented as single NfSen plugins that operate completely independently. This means that specific anomaly detection methods can be added or removed on the fly and be tested independently by the network administrator. It is also possible to implement tools that are necessary for collaboration by adding extensions for e.g. saving the generated alerts to the database, formatting the threat alerts to the user-defined formats (including Extensible Markup Language (XML) etc.).

3.2 Traffic Analysis from Honeypots

3.2.1 Statistical Analysis of Attack Data

The traffic analysis of honeypots depends on the type and its aim of deployment. Low-interaction honeypots are well suited for monitoring attacks on large unused networks. This data can be used to compile statistics about trends in current attacks on the Internet. Since nearly all systems attacking honeypots are compromised themselves, the attack data can be used to warn affected sites about the compromise.

Ideally, this data is automatically exported to Computer Security Incident Response Teams (CSIRTs) to be distributed to the appropriate contact persons. An appropriate format for the data exchange and reporting is X-ARF (see *Automatic Alerting and Opening Incidents* on page 27).

Honeypots like Dionaea are designed to capture malware that is related to attacks. Typically, this malware connects to a botnet C&C server and is remotely controlled over the Internet. Thus, a behavioural analysis of the malware reveals its control channels and helps to pinpoint bad hosts and C&C servers.

3.2.2 Analysis of Argos Data to Detect Unknown Attacks

The traffic analysis of high-interaction honeypots is more difficult as this type of honeypot aims to detect unknown attacks. Argos applies dynamic taint analysis to detect attacks. All data that originates from the network us regarded as tainted. The usage of tainted memory is monitored by Argos to detect malicious operations on this data which occur during attacks. Since this approach does not rely on any specific information, new attacks and zero-day exploits can be detected.

The resulting attack data, as included in an Argos alert (CSI file), contains the type of the alert, the state of the CPU and all tainted memory blocks at the moment the control flow was maliciously diverted. Although this data contains specific information about the attack, it does not directly identify the attack or the targeted vulnerability. Therefore, the challenge of the analysis is to understand the meaning of the alert data. Once this is accomplished, this knowledge can be used to identify the attack and the targeted vulnerability. Fortunately, the Argos CSI file provides some starting points for the identification process:

- The timestamp of the alert for correlation with other alerts.
- Often the Ethernet-frame containing the exploit data can be extracted. The IP address of the attacker and the source and destination port can be used to track down the attacker and to correlate this data with other alerts.
- The state of the CPU at the moment the alert was produced.

The timestamp and Ethernet-frame allow a correlation with Snort alerts that are produced at the same time. For all known worms and attacks, one or more corresponding Snort alerts can be expected, which allow attacks to be classified.

The state of the CPU can also be used to classify the Argos alert. It contains the values of all CPU registers at the exact moment the control flow was maliciously diverted. Most valuable is the program counter or instruction pointer because the value of this register allows attacks to be classified. On current 32-bit Intel architectures this register is called Extended Instruction Pointer (EIP). For example, buffer overflows on the stack segment overwrite the return address to divert the control flow. If the execution of the current function is completed, the return address is loaded from the stack into the instruction pointer. The attack is detected because the return address is overwritten by tainted data. Typically, the overwritten address points to the injected shell code. If so, the instruction pointer has the value of the overwritten return address where the shell code can be assumed to be. In addition to this value, Argos logs the preceding value of the instruction pointer (faulty EIP). In this example, this would be the address of the “RET” assembly command, which loaded the tainted value in the instruction pointer. In other words, this is the last legitimate instruction before the exploit would be able to take over the control. The experimental evaluation of Argos alerts showed that the values of the EIP and faulty EIP allow the targeted vulnerability to be classified. These values are unique for each successful attack that exploits the targeted vulnerability. However, it is likely, that the address space randomisation of some operating systems may lead to inconstant values for future attacks.

To support the analysis and interpretation of Argos data additional programs are required. Debuggers like OllyDBG or Ida Pro that run inside the Argos guest system help to analyse the state of the machine at the moment the control flow is diverted. As previously mentioned, Argos alert data includes the current value of the instruction pointer as well as the faulty value. The faulty EIP points to the last legitimate instruction before the exploit takes over control. Thus, if the debugger is attached to the targeted program, the last executed function can be determined. However, this may not be the function in which the memory corruption occurred. For instance, if the buffer overflow is caused by an insecure “strcpy()” call, the faulty EIP points to the “strcpy()” function and not to the vulnerable function. Technically, a stack trace is required to track down this function.

Another tool that supports the analysis of Argos data is a network traffic analyser like Wireshark. Wireshark captures full packets of network connections and displays their structure including the packet headers and payload. For that purpose, Wireshark contains a large number of dissectors for protocol-specific data. This, for example, allows the data that is contained in the packet payload of specific protocol fields to be assigned. By correlating it with the data of the Argos alert, the protocol field whose content contributed to the attack (e.g. caused the buffer overflow) can be identified.

The W32.Conficker worm, for example, exploited a buffer overflow in the "NetprPathCanonicalize" function (CVE-2008-4250). A call to this function and the passed arguments showed up in the Argos attack data, which was recorded before characteristic Snort signatures were published. Thus, the W32.Conficker worm could be identified before official signatures were available.

3.3 Finding a List of Bad Hosts or Botnet C&C Servers

Since nearly all hosts that attack honeypots are compromised themselves, the connections to honeypots can be monitored to produce a list of bad hosts. The vast majority of malware connects to a botnet control and C&C server and is controlled remotely over the Internet. A behavioural analysis of the malware can reveal its control channels and helps to pinpoint bad hosts and C&C servers. Thus, C&C servers can be identified by monitoring the Internet connections that originate from the malware executed on compromised hosts.

However, it is difficult to distinguish between a connection to a control server and an attack against other systems on the Internet. Therefore, complex measures have to be taken to prevent any abuse and a non-supervised operation of this kind of high-interaction honeypot is not advisable. However, as most honeypot software (such as Dionaea and Argos) prevent the underlying operating system of the honeypot from being compromised, no malware is installed and executed on the honeypot.

An alternative to executing malware on the honeypot is to execute it inside a sandbox. A sandbox is a system that is designed to run, monitor and record the actions of malware. At the time of writing no open-source sandboxes are available, however, the University of Mannheim and the Vienna University of Technology operate the following public sandboxes and allow software samples to be submitted to them:

- CWSandbox (University of Mannheim) runs a Windows operating system instrumented to monitor malware samples. Selected Windows libraries are modified to record library calls (Application Programming Interface (API) hooking).

- Anubis (Vienna University of Technology) is based on the Qemu virtual machine. Like CWSandbox the virtual machine is instrumented to monitor the activity of the malware.

Typically, all connections from and to the Internet are recorded, as are important system or library calls of the operating system running in the sandbox. This includes the monitoring of Windows registry and file system activities, and of the creation of processes. Any attempt of the malware to attack other systems is prevented wherever possible. For example, the running malware is stopped after a certain amount of time which reduces

the risk to systems on the Internet. The connections can be classified by deep packet inspection on the sandbox. Characteristic IRC patterns can identify connections to C&C servers or HTTP protocols which are typically used for botnet-controlled traffic. Most sandboxes export this information in XML format which allows the date to be automatically processed.

3.4 Finding Information about a DDoS – NetFlow Analysis

A DDoS leaves some characteristic patterns in netflow data. It has a high volume of traffic which typically produces a peak in the traffic volume. In addition, it is characteristic that a large number of systems in different networks connect simultaneously to a single host. Commonly, this is either a server (e.g. IRC or web server) or some type of network equipment like a router. However, a legitimate rush of traffic hitting a web or e-mail server can cause the same characteristic. The difference between a DDoS and such a traffic rush is often not obvious. For that reason, a future challenge is to develop algorithms that are able to reliably distinguish between these two events. JRA2 are going to investigate if it could be possible to distinguish the DDoS traffic from the normal intensive hitting of a service, using GÉANT backbone data.

3.5 NfSen Alerts

It is essential for a network operator to observe network behaviour and react to unusual behaviour. Automating this process is important to reduce the man power and time network monitoring requires. The NfSen alert interface [NfSenInt] is a good way to automate network monitoring since it provides the ability to trigger alerts when various conditions are met in flow data.

An NfSen alert consists of four major elements:

- Filter:

  The filter is applied to flow data before the alert conditions are checked. The syntax is same as for Nfdump/NfSen.

- Alert Condition:

  This is probably the most important element. The interface for creating alerts allows conditions to be created that are based on flow/packet/byte, sum/rate/average or any statistic value. Up to six conditions can be logically linked with OR or AND. However, while this interface may not be sufficiently powerful to create sophisticated sets of conditions, the plugin interface of NfSen can be used to craft special conditions according to requirements. Alert conditions can be written in backend plugins which are written in Perl. A key subroutine used for this purpose is “alert_condition”. If an “alert_condition” subroutine is detected when a backend plugin is being loaded, then a condition for that plugin becomes available in the alert creation dialogue. Any sophisticated anomaly detection or forecasting algorithm can be implemented in this subroutine to craft alert conditions.

- Trigger Rate:

  This element takes three inputs: one specifies if an alert will be triggered once or every time the condition is met, one specifies how many times the condition should be met for the trigger to run, one specifies for how many cycles the alert will be blocked after the trigger. If the alert is configured to be triggered once, then it should be manually activated again after it has been triggered.

- Alert Action:

  This specifies what action is taken when a trigger is set off. The alert creating interface has three options for an alert action: do nothing, send an email or execute an action specified in one of loaded backend plugins. A key subroutine called “alert_action” is used to specify the alert actions. Any action can be executed as a result of a trigger being set off, for example, logging, mailing or changing firewall configurations etc.

Once alerts have been created according to requirements, the NfSen alerts tab displays a list of the created alerts and their states (see the NfSen documentation for a list and definitions of the possible states [NfSen]). This list can also be used to edit or remove alerts.

In practice, the key subroutine “run” can be used to implement filter, alert condition and alert action functionalities since it runs every cycle (5 minutes by default) but the trigger rate and alert status features of NfSen cannot be used with it. The Dynamic DDoS Detector [DDD] is an example of a plugin in which all the functionality is implemented in the “run” subroutine.

NfSen alerts can be used to detect botnet-related activity, peaks, DDoS or any other network behaviour, and any required reaction to these behaviours can be automated with the help of NfSen plugins. DDoS or peak detection can be done with simple NfSen alert conditions without plugin support. For botnet detection, the methods mentioned in *Finding Patterns of Bad Traffic in NetFlow Data* on page 16 can be implemented in the “alert_condition” subroutine that is to be used by NfSen alerts.

3.6 Investigating Incidents – “Post Mortem” Detection

Computer security incidents can be detected either on the network layer or on the compromised host. In both cases the following steps have to be taken:

- Evidence for a security incident:

  Is the recorded event really a security incident or does it only look like one? For example, a legitimate traffic rush might cause an anomaly on the network. Therefore, the first step is to find evidence of a compromise or malicious event.

- Extent of the incident:

  The extent of the incident must be investigated. The incident often affects more than one host. As compromised machines are usually abused to attack other systems on Internet, it is crucial to determine the full extent of an incident. This can be done in the following ways:

- Analysis of netflows:

  If the incident was detected on the network layer, the netflows from and to the affected host or network reveal other involved systems. For example, the netflows to an attacked host can identify hosts that are involved in a DDoS attack. The same technique can be applied to explore the extent of a botnet following the netflows to or from the C&C server.

- Forensic analysis of a compromised host:

  This aims to identify the vulnerability which has been abused in order to compromise the system. The vulnerability needs to be known to recover from the incident without taking the risk of a re-infection.

  The forensic analysis also aims to explore the activity of the attacker on the compromised system. This may reveal malware or recorded logfiles the attacker left behind, which can be used to identify other systems affected by the incident.

3.6.1 Forensic Analysis of a Compromised System

As previously stated, the forensic analysis of a compromised host can identify abused vulnerability or vulnerabilities, and investigates the activities of the attacker on the compromised host.

Investigating data gathered on the network or host layer can identify the vulnerability. Abnormal connections in the netflow data point to vulnerable applications. They could be produced by an exploit sent to such an application (e.g. a web server).

A different approach is to use a vulnerability scanner like Nessus or an AV-product to identify vulnerabilities or malware. These tools can be deployed from the network or executed on the affected machine. Moreover, the logfiles on the system may have recorded events that are related to the attack and may give hints to other compromised systems:

- What were the attacker's privileges on the compromised system? If the attacker reached root privileges, everything on the system could be modified. In addition, a root-kit may be running which hides processes, files and other resources on the compromised system.

- Are backdoors installed? Typically, a bot is running which offers a backdoor. Moreover, backdoors are often combined with a root-kit that hides them.

- Which programs have been installed on the compromised machine?

- Are other systems attacked?

Most software tools intended to attack other systems leave log files on the compromised machine. Typically, these files contain information about targeted systems and the result of the attack. Other important log files are produced by keyboard sniffers or manipulated software. In such case, passwords recorded in these files may be compromised .

A web or IRC server can potentially be abused as a C&C server or a dropzone that serves as a centralised storage collecting log entries (e.g. containing stolen passwords) from distributed malware. In either case the server is at the centre of a distributed network of compromised hosts. This network might be completely or at least partly revealed by its network traffic or log files. Typically, IRC or web server record connection attempts from clients. Therefore, these log files are valuable for exposing a botnet.

JRA2 are going to use the data from Argos (see *Analysis of Argos Data to Detect Unknown Attacks* on page 20) to study how these operations can be supported.

4 Specifications of Mitigation Tools

4.1 Automatic Alerting and Opening Incidents

Currently, malware and compromises are usually spread automatically. Due to the number of compromised hosts and the speed at which malware spreads, manual methods for incident or alert reporting fail. Instead, methods are required that automate this process and that include the following components:

- Automatic incident detection:

  The first step in the chain of automatic alerting are components to record attack or incident data. In GÉANT these are honeypots and components to record and analyse netflow data.

- Data analysis:

  To efficiently react to alerts, additional information about their scope and extend are required. In many cases, the attack data allows to reliably identify the vulnerability or the malware the attacking host is infected with. For example, if Dionaea captures the binary of the W32.Conficker worm, the source itself is likely to be compromised by this worm. This can also be applied to SSH password guessing attacks and the majority of known malware.

- Contact information:

  It is crucial to know to which contact the alert should be sent. This information is stored in a database, which assigns networks, autonomous systems or IP addresses to contacts. Thus, the database allows identifying the appropriate contacts for a list of systems or networks which produced the alert.

- Format for incident or alert data exchange:

  An appropriate format for the exchange of alert or incident data is ARF or the extended X-ARF (see *X-ARF Format* on page 29). This format contains machine- as well as human-readable parts. Because of its lightweight, though structured, form it is increasingly used by CERTs for data exchange.

- Infrastructure for the automatic distribution of incident data:

  An infrastructure is required that interconnects all components. This includes the authentication of the communicating sites as well as the security of the alert data on the transport.

Although automated solutions for distributing incident reports exit on the layer of autonomous systems, they will need to be extended to a larger constituency or ideally to the appropriate administrative contact.

4.1.1 Issues in a Multi-Domain Environment

One of the biggest risks of a multi-domain incident in a GN3 environment is caused by many machines running the same software, same operating system and same level of patch, because once an attacker gains access to one machine, all other machines can be accessed as well. A common attack vector could be used to compromise all hosts that run GN3 services because all the service hosts run the same software. An attacker could also disrupt one of the nodes of a service to corrupt the data of a GN3 service. The availability of the entire GN3 service could also be affected by malicious activities.

Their collaborative nature is a feature of all GN3 services. They have been developed based on the assumption that information is shared across domains (when needed). For example, the collaboration between the nodes of the perfSONAR monitoring service allows a user to see the network delay between two points in different domains. A disruption in one of the nodes of the service infrastructure in a foreign domain would, therefore, have a negative impact on the users' perception of the service.

The solution proposed for dealing with security threats to multi-domain services requires a good level of coordination between the NRENs involved in the services. NREN CERTs, Service Development Teams and Service Owners need to work together. In particular, the relationship between NREN CERTs and Service Owners is critical, because it allows the scope of the incident to be mitigated, and those supporting the infrastructure to be warned about the incident, so appropriated actions can be taken to protect the service.

One of the most frequent problems encountered in incident handling is the low level of procedure and process automation. Most of the steps a security incident handler takes when faced with a security threat are manual. Some of these steps could be automated to speed up the process and free up resources, so they can be used more efficiently. For example, steps related to gathering information (such as contact information of an IP in the constituency or of a GN3 service) and steps for processing and triaging the received report, especially if has a structured format, as the tracking system can extract all the information from the report without manual intervention. Once the report is uploaded to the tracking system, the system can be programmed to execute different actions. For example, identify who needs to be notified of the incident, select the proper text template for contacting this person, or even analyse the netflow data for traffic information that can confirm the reported malicious activity.

Automating security incident handling procedures would offer significant advantages to the multi-domain GN3 environment, where many actors are involved in solving security incidents and where quick action can be the key to resolving threats. Automation would aid in preventing the exploitation of new vulnerabilities in GN3 services. It could also increase the number of reports that CERTs are able to process and handle, significantly increasing the volume of gathered information. This, in turn, would result in better and more accurate responses to security problems, as incident response teams would be able to analyse and evaluate more information. To find methods to communicate inside the multi-domain environment JRA2 will collaborate with SA2 (Multi-Domain Network Services) Task 4 (Security) and use procedures researched by JRA2 Task 1.

4.2 Harmonisation of the Information to Send

To contain and prevent security incidents, CSIRTs need to have a common understanding of them and agreed operational procedures for exchanging and handling information. For the GN3 project, the coordination of handling multi-domain incidents between the CSIRTs and Service Development team requires shared information to be in a structured and harmonised format that can be understood by all involved. A harmonised format will allow the root of the problem to be understood properly, and improve both the quality of the shared information and the quality of the response that is given.

Messages being exchanged between the teams are human readable, but defined as a wire format. It is the responsibility of each team to translate the messages to a local database format that is suitable for the incident tracking system used by each team, before any initial action is taken over the incident. The use of non-formatted messages could cause problems within the multi-domain incident handling process like:

- Misunderstandings between involved parties.
- Bad triaging of incidents.
- The skipping or losing of important information about the incident.

Therefore, the exchange of security information between any two entities must be structured and described in a consistent manner that is understood by all of the involved entities and parties. This format should :

- Be human readable.
- Be automatically parsable.
- Provide a clear definition of the format and the content, avoiding confusion about when the information is sent, received or studied.
- Be based on a standard format like XML.
- Easily integrate with ticket systems that are currently used in the GN3 environment.

Building an harmonised format for exchanging security information between all the parties involved in multi-domain security incidents will improve the incident handling process in areas such as timing, triaging of reports, prioritisation of information, etc.

4.2.1 X-ARF Format

A lightweight but structured format for the exchange of computer incident information is X-ARF. In contrast to other formats such as Incident Object Description and Exchange Format (IODEF), it is less complex which eases its deployment. Although IODEF is more powerful and covers more information, its complexity prevents widespread deployment. Also, only a few and limited software products for the creation and handling of IODEF documents are available. Thus, the aim of X-ARF is to introduce a lightweight format which focuses on the most important information and can be easily deployed and extended. It is designed to transfer data about port

scans, malware and other incidents. Currently, the format is accepted by a growing list of CSIRTs and broad acceptance can be expected.

X-ARF is an email format which is grouped into three parts:

1. The first part is human readable and can contain arbitrary text.
2. The second part contains structured data which uses the YAML markup language for structuring. A JSON scheme exists to ensure that the content is well structured.
3. The third part is intended for unspecified data. This can be log data related to the incident, or it may be used for malware files attached to the X-ARF report.

Currently, X-ARF alerts for SSH scans and for reporting malware can be produced by existing modules. However, plans exist to use X-ARF in the future also for the data export of other honeypots including Dionaea. Thus, the challenge will be to promote the format in order to make it attractive for a larger community of CSIRTS and security teams.

A sample of a reported SSH scan is:

Part I (free text):

```
Dear DFN-CERT,

this is an automated report for ip address 10.11.229.178 in format "X-ARF"
generated on 2010-10-22, 10:26:46.044883
IP address 192.168.229.178 produced 1892 log lines, sample log lines attached.
```

Part II (structured YAML key-value format)

```
Category:            abuse
Source-Type:         ipv4
Report-Type:         login-attack
Service:       tsshp 0.7
Report-ID:           12877360064637@localhost
Reported-From: tsshp@localhost
Source:        10.11.229.178
Schema-URL:    http://www.x-arf.org/schema/abuse_login-attack_0.1.1.json
Attachment:          text/plain
Date:                2010-10-22
User-Agent:          tsshp 0.7
Port:                22
```

Part III (arbitrary content; recorded SSH connections):

```
2010-10-19 20:33:34,888 WARNING login attempt from: '10.11.229.178:42658', to:
'192.168.39.234:22221',
user: 'root', password: 'XXXXXXXXX', failure #1
2010-10-19 20:33:35,471 WARNING login attempt from: '10.11.229.178:42658', to:
'192.168.39.234:22221',
user: 'root', password: 'XXXXXXXXX', failure #2
--- MARK ---
2010-10-20 21:19:52,473 WARNING blocked attempt from: '10.11.229.178:55558',
to: '192.168.37.233:22221'
2010-10-20 21:19:53,080 WARNING blocked attempt from: '10.11.229.178:55575',
to: '192.168.37.233:22221'
```

5 Network Devices

Security features inside network devices primarily depend on the device's purpose and its hardware capabilities (see Appendix A for an overview of the state-of-art for security features in network devices). More sophisticate security features require more device resources, which can be demanding. Also, due to the changing nature of communication technologies and associated new communication protocols and services, the traffic pattern is changing significantly. There is a significant rise in video conferencing and other media traffic which is sensitive traffic with a low resilience to packet loss or packet delay. This traffic also requires a lot of bandwidth and has a high impact on device performance.

Due to better hardware performance, network devices that used to only offer a dedicated set of services (for example, routing, switching, firewall services etc.) now also offer extended functionalities. These devices can serve as aggregate points and offer anomaly detection and prevention or functionality such as media gateways etc. in addition to switching, routing and some firewall functionality.

Alongside the rise of distinct media traffic, there is also an increasing need for converged communication that converges distinct media such as voice traffic, Internet and others into a common interface on a single device. This device must guarantee a minimum flow performance for such sensitive traffic. Security features and QoS support are now must-have features on border routers/devices for institutions that want to use the Internet as the main backbone for a wide variety of communication technologies.

In this context, especially for large enterprise and backbone networks like the GÉANT network, security features on routing devices tend to overload devices, due to a large volume of traffic.

5.1 Specification for BGP Policies Inside Network Devices

The Border Gateway Protocol (BGP) backs the core routing decisions on the Internet. It maintains a table of IP networks or 'prefixes' which designate network reachability among autonomous systems (AS). BGP is described as a path vector protocol. It does not use traditional Interior Gateway Protocol (IGP) metrics, but makes routing decisions based on path, network policies and/or rulesets. For this reason, it is more appropriately termed a reachability protocol rather than a routing protocol. [WikiBGP]

As BGP is prone to vulnerability and bad traffic injection at the border level of the network, it could be a severe issue in a multi-domain environment if these vulnerabilities were exploited.

During the project JRA2 will research security issues in BGP and how BGP configuration could be used to mitigate malicious traffic by preparing a testing environment.

JRA2 will also investigate currently implemented security methods in BGP configurations in GÉANT and possible security breach cases until now. The Activity will propose testing of various encryption algorithms and their impact on BGP sessions and examine implementing BGP Route Origin Validation via a method known as Resource Public Key Infrastructure (RPKI). In GÉANT, MD5 authentication is currently in place but the leading edge on BGP is the Origin authentication the Activity intends to investigate.

5.2 Specification for Detection and Mitigation

This part of the project will study how network devices could be used to detect and mitigate the bad traffic, in collaboration with other subtasks' needs. With the help of network devices and some existing or new features, JRA2 will explore in collaboration with other GN3 Tasks how methods and solutions for detection of bad traffic be can be researched and implemented. Giving consideration to the needs of GN3 and the specification of currently implemented and researched methods, JRA2 will research smart techniques for instructing devices to operate on traffic in order to upgrade the existing solutions. The Activity will also research how network devices can support anomaly detection techniques in collaboration with currently implemented (software) solutions.

After the detection of attacks, JRA2 will investigate techniques to instruct devices to react to traffic using methods such as blackholing of bad traffic to ward off DDoS attacks.

6 Conclusions

This document has demonstrated the wide variety of threats that need to be faced, how they may evolve and what risks they pose to the GÉANT community. As most of the threats are still unknown (and likely to remain unknown for the foreseeable future), it is difficult to mitigate them. JRA2 Task 4 found that the best approach to the problem is to study different methods to identify threats as early as possible, with the scope to notify the NREN that the problem is originating from, or to recommend the best configuration for network devices in order to mitigate threats, for example, by sink-holing bad traffic.

The first step to become aware of threats is to try to collect them. The best method for this is to deploy both low-interaction and high-interaction honeypots. This could enable NRENs to create lists of "bad hosts", machines that have been compromised by viruses to connect to the honeypots. NRENS could use these lists to take appropriate countermeasures against the bad hosts or as input for NetFlow analysis tools to find more compromised hosts by analysing the flows that involve all machines that are connecting to the bad hosts.

Another way to detect security incidents is to analyse network traffic by applying both anomaly detection algorithms and using appropriate tools. JRA2 Task 4 found that there is not sufficient time to study the validity of an anomaly detection algorithm (at least 5 years of network-traffic data would be needed), although the Task is planning to develop a new algorithm, which could be tested in future GÉANT projects. However, the Task has found ways of using the GÉANT backbone traffic data to develop new plugins for the nfdump and NfSen tools that were already used during the GN2 project for detecting botnets and other malicious activities.

JRA2 Task 4 will also use network-device-like appliances to help detect botnet and malicious traffic, and to mitigate threats by filtering them with ACLs or sink-holing.

Due to the multi-domain nature of the GÉANT environment, the Tasks expects a large number of machines inside the community to become compromised. Therefore, it is important to automate the notification of the appropriate CSIRT teams as much as possible. To better automate the operations involved, is also important to establish a standard format for sending the needed information. JRA2 Task 4 found the X-ARF format to be a valid candidate and will develop all tools and methods to output X-ARF format.

Finally, as a good level of collaboration is important inside a multi-domain environment, JRA2 Task 4 will try to study a common program for basic operations or common procedures for incident handling, based on the knowledge-base tool developed in SA2 Task 4.

Appendix A State of the Art for Security Features in Network Devices

A.1 Vulnerabilities of Network Devices

General threats to network devices include unauthorised access, session hijacking, rerouting, masquerading, DoS, eavesdropping, information theft etc. In addition to threats to a router from the network, dial up access to a router exposes it to further threats.

Attack techniques include:

- Password guessing.
- Routing protocol attacks.
- SNMP attacks.
- IP fragmentation attacks – to bypass filtering.
- Redirect (address) attacks.
- Circular redirect – for denial of service.

Session replay attacks use a sequence of packets or application commands that can be recorded, possibly manipulated and then replayed to cause an unauthorised action or give unauthorised access.

Rerouting attacks can include manipulating router updates to cause traffic to flow to unauthorised destinations. These kinds of attacks are sometimes called “route injection” attacks.

Masquerade attacks occur when an attacker manipulates IP packets to falsify IP addresses. Masquerades can be used to gain unauthorised access or to inject bogus data into a network.

Session hijacking may occur if an attacker can insert falsified IP packets after session establishment via IP spoofing, sequence number prediction and alteration, or other methods.

Resource starvation attacks usually involve flooding the router with traffic or requests designed to consume all of a limited resource. Target resources may be bandwidth, memory or even computation.

Careful device configuration can help prevent a (compromised) site from being used as part of a DDoS attack, by blocking spoofed source addresses. DDoS attacks use a number of compromised sites to flood a target site with sufficient traffic or service requests to render it useless to legitimate users.[attyp]

A.2 Routers

Updating routing tables and forwarding data packets between portions of a network are two of the primary tasks of a router, as well as additionally filtering traffic. [rousec]

The reason for committing to the expense and effort of deploying a dedicated, purpose-built router rather than a general purpose machine with a “standard” operating system (OS), partly concerns performance. Consolidating network routing and related functions on a dedicated devices restricts access to and limits the exposure of those critical functions. A specialised router operating system can be smaller, more comprehensible, and more thoroughly tested than a general purpose OS.

A.2.1 Implemented Security Features in Routers and Router Classes

Security in the router is implemented across the four major planes of router architecture, data, control, management, and service, with much of the security conversation around routers involving the services plane. (Sometimes control, management, and services are simply grouped into the "control plane".)

To secure a router, the possible threats to each plane must be considered. Threats to the management, control and service planes mostly concern unauthorised access to the router or interference with router operation. Threats to the data plane usually concern violations of network security for the networks that the router supports. Three aspects of securing routers can be observed: physical security, network security and management security.

There are a number of ways to provide physical security for a router. The room that contains router should be free of electrostatic or magnetic interference, have controls for temperature and humidity, installed UPS and available spare components. The router should be placed in a locked room accessible only to authorised personnel. Since the operating system is the crucial component of the router, it should be determined what features the network needs, and the router feature list should be used to select the version of the operating system. The latest stable release of the operating system that meets the feature requirements should be used. Many default services are unnecessary and should be disabled as they may be used by an attacker for information gathering or exploitation. The most common method of securing networks with routers is by applying packet filters. [guisec]

To create adequate packet filter rules and a secure network, the following best practice rules should always be followed:

- Permit only required protocols and services.

- Reject risky protocols and services.

With adequate configuration of packet filters, routers can also be able to mitigate some common security threats such as DOS, spoofing, TCP syn attacks, land attacks, smurf attacks etc.

Management security involves controlling access to a router, updating policy, logging and operational security management. For each of these there are best practice guidelines that should be followed.

Access to a router must be highly secured, using dedicated secured connections. There may be also be a need to have more than one level of administrator or more than one administrative role. Regularly updating routers will fix known vulnerabilities, improve performance and support new features (perhaps some that allow more advanced security policies). Using the information in a log, the administrator can tell whether a router is working properly or has been compromised. In some cases, it can show what types of probes or attacks are being attempted against the router or the protected network. Maintaining the security of a router over its operational lifetime requires regular assessment, testing, and correction. Another important aspect of lifetime security is preparing for problems. Keeping up-to-date backups of router configurations and installed operating systems releases is essential for quick and reliable recovery from security compromises or simple hardware failures. In the case of a security compromise, it is highly desirable to preserve the evidence, so that it can be used in a forensic investigation or even prosecution.

A.2.2 Next Generation Routers and Security Features

New technologies give developers more freedom in developing router models. Most of the SME routers support firewall and IPDS features as well as other security-related features. However, where routers are to be used as backbone or border routers, bandwidth requirements are high and implementing various security solutions can degrade the routers' performance. The trend of router developing tends to improve the hardware performance of border routers , and their main security features are based on packet filtering and good practice security recommendation for all operational planes of the router. By creating good a security policy that addresses physical integrity, core static configuration, dynamic configuration, and network traffic, the router can be used as a device that can address most of the security threats.

As traffic-processing engines and operating system architectures improve, the border router will be able to play a larger security role due to its ability to segregate control, management and services plane functions from the data plane. Security appliances will continue to play a role in identifying new threats, while the responsibility for stopping more mature network-level attacks is likely to migrate to the border router.

In recent research, WAN managers preferred to continue the separation of security from the border router, but they also saw the value in integrating additional security features in the border router. This calls for the need to integrate security in each of the planes in the border router. The data plane can be secured using techniques such as packet filtering, unicast RFP, flexible packet matching, QoS, and IP source tracking. The control plane can be secured using receive packet filtering, control plane policing, MD5 authentication, and ICMP techniques. The management plane can be secured using techniques such as CPU and memory thresholding, password

and SNMP security, and remote terminal access control, while securing the services plane will depend on the IP services deployed.

One of the main trends that involve importing extensive security role into network device is converged communication aggregation. The border router has two significant responsibilities in enabling converged communications: interfacing with service providers and enabling high-availability and high-performance voice and video communications across the WAN. New types of traffic introduce new complexity to the architecture and role of the routers. At the service provider interface, the border router sits between the enterprise and a growing number of communications services ranging from SIP trunking, over PSTN gateways, to inter-company dialling services. In support of voice and interactive video traffic, the border router must be able to provide consistent service levels, regardless of the network load. Additionally, rapid recovery and traffic control mechanisms must be in place to ensure voice and video session stability at all times.

The role of the border router will continue to gain importance and become more sophisticated as critical business applications migrate from the LAN to the WAN. The requirements for the next generation of border routers must take this into consideration and provide the capability to be adaptable to new types of traffic and traffic patterns in order to ensure consistent and reliable performance across the WAN links as new business services are added.

A.3 Security Appliances

A security appliance device can be a proprietary device from a number of world-known developers (e.g. Cisco, Juniper, Symantec, Checkpoint, Trendmicro, etc.) or from less popular developers (e.g. Cyberoam, Sonicwall, Contentwatch, Astaro, e-Safe etc.). These security appliances have proprietary device architectures and software solutions for addressing security threats. A security devce can also be a commercial or opensource software security appliance implemented on thin clients or ordinary personal computers with additional network interfaces. In the case of thin clients, these network appliances lack many of the features of a fully equipped PC, and are often referred to as "closed box systems" as they provide a complete solution consisting of limited hardware and software that is needed to perform a single or a specialised set of functions. This type of hardware device allows for quick installation, ease-of-use and low maintenance, and is typically managed through a web browser. Appliances are increasingly used in network security to replace more traditional software-based security solutions.

All of the above network security appliances comprise the following security tools to address known security threats:

- Firewall.
- Intrusion detection/prevention system (IDPS).
- Antivirus scanning.
- Content filtering (including spam filtering and spyware blocking).

For the purpose of this document, firewall and IDPS issues are explored further below.

Popular with business and enterprise, Unified Threat Management (UTM) is a category of security appliances that integrates a range of security features in a single appliance. UTM appliances combine firewall, gateway anti-virus, content filtering and IDS and intrusion prevention capabilities into a single platform.

A.3.1 Firewall

Firewalls are devices or programs that control the flow of network traffic between networks or hosts that employ different security postures. Several types of firewall technologies are available. One way of comparing their capabilities is to look at the Transmission Control Protocol/Internet Protocol (TCP/IP) layers that each is able to examine.

TCP/IP communications are composed of four layers that work together to transfer data between hosts. Basic firewalls operate on one or a few layers — typically the lower layers — while more advanced firewalls examine all layers. Firewalls that examine multiple layers can perform more granular and thorough examinations. Firewalls that understand the application layer can potentially accommodate advanced applications and protocols, and provide services that are user-oriented. For example, a firewall that only handles lower layers cannot usually identify specific users, but a firewall with application layer capabilities can enforce user authentication and log events to specific users. [guifire] [wikifire]

The information in the following sections has been drawn from [guifire].

A.3.1.1 *Packet Filtering*

The most basic feature of a firewall is the packet filter. Older firewalls that were only packet filters were essentially routing devices that provided access control functionality for host addresses and communication sessions. Unlike more advanced filters, packet filters are not concerned about the content of packets. Their access control functionality is governed by a set of directives referred to as a ruleset. Packet filtering capabilities are built into most operating systems and devices capable of routing. The most common example of a pure packet filtering device is a network router that employs access control lists.

Stateless packet filters are generally vulnerable to attacks and exploits that take advantage of problems within the TCP/IP specification and protocol stack. Spoofing attacks, such as using incorrect addresses in the packet headers, are generally employed by intruders to bypass the security controls implemented in a firewall platform.

A.3.1.2 *Stateful Inspection*

Stateful inspection improves on the functions of packet filters by tracking the state of connections and blocking packets that deviate from the expected state. As with packet filtering, stateful inspection intercepts packets at the network layer and inspects them to see if they are permitted by an existing firewall rule, but unlike packet filtering, stateful inspection keeps track of each connection in a state table.

A.3.1.3 *Application Firewalls*

A newer trend in stateful inspection is the addition of a stateful protocol analysis capability, referred to by some vendors as deep packet inspection. Stateful protocol analysis improves upon standard stateful inspection by adding basic intrusion detection technology — an inspection engine that analyses protocols at the application layer to compare vendor-developed profiles of benign protocol activity against observed events to identify deviations.

Another feature found in some application firewalls involves enforcing application state machines, which are essentially checks on the traffic's compliance to the standard for the protocol in question. This compliance checking, sometimes called "RFC compliance" because most protocols are defined in RFCs issued by the Internet Engineering Task Force (IETF), can be a mixed blessing. Many products implement protocols in ways that almost, but not completely, match the specification, so it is usually necessary to let such implementations communicate across the firewall. Compliance checking is only useful when it detects and blocks communication that can be harmful to protected systems.

Firewalls with both stateful inspection and stateful protocol analysis capabilities are not fully fledged intrusion detection and prevention systems (IDPS), which usually offer much more extensive attack detection and prevention capabilities.

A.3.1.4 *Application Proxy Gateways*

An application-proxy gateway is a feature of advanced firewalls that combines lower-layer access control with upper-layer functionality. These firewalls contain a proxy agent that acts as an intermediary between two hosts that wish to communicate with each other and never allows a direct connection between them. Each successful connection attempt actually results in the creation of two separate connections — one between the client and the proxy server and another between the proxy server and the true destination. The proxy is meant to be transparent to the two hosts — from their perspectives there is a direct connection. Because external hosts only communicate with the proxy agent, internal IP addresses are not visible to the outside world. The proxy agent interfaces directly with the firewall ruleset to determine whether a given instance of network traffic should be allowed to transit the firewall.

A.3.1.5 *Dedicated Proxy Servers*

Dedicated proxy servers differ from application-proxy gateways in that, while dedicated proxy servers retain proxy control of traffic, they usually have much more limited firewalling capabilities. Many dedicated proxy servers are application-specific, and some actually perform analysis and validation of common application protocols such as HTTP. Because these servers have limited firewalling capabilities, such as simply blocking traffic based on its source or destination, they are typically deployed behind traditional firewall platforms. Dedicated proxy servers are generally used to decrease firewall workload and conduct specialised filtering and logging that might be difficult to perform on the firewall itself.

A.3.1.6 *Limitation of Firewall Inspection*

Firewalls can only work effectively on traffic that they can inspect. Regardless of the firewall technology chosen, a firewall that cannot understand the traffic flowing through it will not handle that traffic properly — for example, allowing traffic that should be blocked. Many network protocols use cryptography to hide the contents of the traffic which firewalls cannot read. Another limitation faced by some firewalls is understanding traffic that is tunnelled, even if it is not encrypted. The content may still be unencrypted, but if the firewall does not understand the particular tunnelling mechanism used, the traffic cannot be interpreted.

In all these cases, the firewall's rules will determine what to do with traffic it does not (or, in the case of encrypted traffic, cannot) understand. An organisation should have policies about how to handle traffic in such cases, such as either permitting or blocking encrypted traffic that is not authorised to be encrypted.

A.3.2 Intrusion Prevention/Detection System

Intrusion detection is the process of monitoring the events occurring in a computer system or network and analysing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices. Incidents have many causes, such as malware (e.g., worms, spyware), attackers gaining unauthorised access to systems from the Internet, and authorised users of systems who misuse their privileges or attempt to gain additional privileges for which they are not authorised. Although many incidents are malicious in nature, many others are not; for example, a person might mistype the address of a computer and accidentally attempt to connect to a different system without authorisation.

An intrusion detection system (IDS) is software that automates the intrusion detection process. An intrusion prevention system (IPS) is software that has all the capabilities of an intrusion detection system and can also attempt to stop possible incidents.

IDS and IPS technologies offer many of the same capabilities, and administrators can usually disable prevention features in IPS products, causing them to function as IDSs. Accordingly, for brevity the term intrusion detection and prevention systems (IDPS) is used to refer to both IDS and IPS technologies. [guiipds] [wikiips]

The information in the following sections has been drawn from [guiipds].

A.3.2.1 *Uses of IDPS Technologies*

IDPSs are primarily focused on identifying possible incidents. For example, an IDPS could detect when an attacker has successfully compromised a system by exploiting a vulnerability in the system. The IDPS could also log information that could be used by the incident handlers. Many IDPSs can also be configured to recognise violations of security policies. Also, some IDPSs can monitor file transfers and identify any that might

be suspicious, such as copying a large database to a user's laptop. Many IDPSs can also identify reconnaissance activity, which may indicate that an attack is imminent.

IPS technologies are differentiated from IDS technologies by one characteristic: IPS technologies can respond to a detected threat by attempting to prevent it from succeeding. They use several response techniques, which can be divided into the following groups:

- The IPS stops the attack itself.
- The IPS changes the security environment.

  The IPS could change the configuration of other security controls to disrupt an attack. Some IPSs can even cause patches to be applied to a host if the IPS detects that the host has vulnerabilities.
- The IPS changes the attack's content.

  Some IPS technologies can remove or replace malicious portions of an attack to make it benign. This might cause certain attacks to be discarded as part of the normalisation process.

Another common attribute of IDPS technologies is that they cannot provide completely accurate detection. When an IDPS incorrectly identifies benign activity as being malicious, a false positive has occurred. When an IDPS fails to identify malicious activity, a false negative has occurred. It is not possible to eliminate all false positives and negatives; in most cases, reducing the occurrences of one increases the occurrences of the other.

Most IDPS technologies also offer features that compensate for the use of common evasion techniques. Evasion is modifying the format or timing of malicious activity so that its appearance changes but its effect is the same. Most IDPS technologies can overcome common evasion techniques by duplicating special processing performed by the targets.

A.3.2.2 *Types of Events Detected*

The types of events most commonly detected by network-based IDPS sensors include the following:

- Application layer reconnaissance and attacks

  (E.g. banner grabbing, buffer overflows, format string attacks, password guessing, malware transmission). Most network-based IDPSs analyse several dozen application protocols. Commonly analysed ones include Dynamic Host Configuration Protocol (DHCP), DNS, Finger, FTP, HTTP, Internet Message Access Protocol (IMAP), Internet Relay Chat (IRC), Network File System (NFS), Post Office Protocol (POP), rlogin/rsh, Remote Procedure Call (RPC), Session Initiation Protocol (SIP), Server Message Block (SMB), SMTP, SNMP, Telnet, and Trivial File Transfer Protocol (TFTP), as well as database protocols, instant messaging applications, and peer-to-peer file sharing software.
- Transport layer reconnaissance and attacks

  (E.g. port scanning, unusual packet fragmentation, SYN floods). The most frequently analysed transport layer protocols are TCP and UDP.

- Network layer reconnaissance and attacks

  (E.g. spoofed IP addresses, illegal IP header values). The most frequently analysed network layer protocols are IPv4, ICMP, and IGMP. Many products are also adding support for IPv6 analysis. The level of IPv6 analysis that network-based IDPSs can perform varies considerably among products. Some products provide no IPv6 support or can simply alert administrators that IPv6 activity is present. Other products can do basic processing of IPv6 and tunneled IPv6 traffic, such as recording source and destination IP addresses, and extracting payloads (e.g., HTTP, SMTP) for in-depth analysis. Some products can do a full analysis of the IPv6 protocol, such as confirming the validity of IPv6 options, to identify anomalous use of the protocol. Organisations with a current or future need to monitor IPv6 activity should carefully evaluate the IPv6 analysis capabilities of network-based IDPS products.23

- Unexpected application services

  (E.g. tunneled protocols, backdoors, hosts running unauthorised application services). These are usually detected through stateful protocol analysis methods, which can determine if the activity in a connection is consistent with the expected application protocol, or through anomaly detection methods, which can identify changes in network flows and open ports on hosts.

- Policy violations

  (E.g. use of inappropriate websites, use of forbidden application protocols). Some types of security policy violations can be detected by IDPSs that allow administrators to specify the characteristics of activity that should not be permitted, such as TCP or UDP port numbers, IP addresses, website names, and other pieces of data that can be identified by examining network traffic.

A.3.2.3 *Common Detection Technologies*

IDPS technologies use many methodologies to detect incidents: signature-based, anomaly-based and stateful protocol analysis. Most IDPS technologies use multiple detection methodologies, either separately or integrated, to provide more broad and accurate detection.

Signature-Based Detection

A signature is a pattern that corresponds to a known threat. Signature-based detection is the process of comparing signatures against observed events to identify possible incidents.

Signature-based detection is very effective in detecting known threats but largely ineffective in detecting previously unknown threats, threats disguised by the use of evasion techniques, and many variants of known threats.

Signature-based detection is the simplest detection method because it just compares the current unit of activity, such as a packet or a log entry, to a list of signatures using string comparison operations. Signature-based detection technologies have little understanding of many network or application protocols and cannot track and understand the state of complex communications. They also lack the ability to remember previous requests

when processing the current request. This limitation prevents signature-based detection methods from detecting attacks that comprise multiple events if none of the events contains a clear indication of an attack.

Anomaly-Based Detection

Anomaly-based detection is the process of comparing definitions of what activity is considered normal against observed events to identify significant deviations. An IDPS using anomaly-based detection has profiles that represent the normal behaviour of such things as users, hosts, network connections, or applications. An initial profile is generated over a period of time (typically days, sometimes weeks) sometimes called a training period. The major benefit of anomaly-based detection methods is that they can be very effective at detecting previously unknown threats.

Anomaly-based IDPS products often produce many false positives because of benign activity that deviates significantly from profiles, especially in more diverse or dynamic environments. Another noteworthy problem with the use of anomaly-based detection techniques is that it is often difficult for analysts to determine why a particular alert was generated and to validate that an alert is accurate and not a false positive, because of the complexity of events and number of events that may have caused the alert to be generated.

Stateful Protocol Analysis

Stateful protocol analysis is the process of comparing predetermined profiles of generally accepted definitions of benign protocol activity for each protocol state against observed events to identify deviations. Unlike anomaly-based detection, which uses host or network-specific profiles, stateful protocol analysis relies on vendor-developed universal profiles that specify how particular protocols should and should not be used. The “stateful” in stateful protocol analysis means that the IDPS is capable of understanding and tracking the state of network, transport, and application protocols that have a notion of state. Stateful protocol analysis methods use protocol models, which are typically based primarily on protocol standards from software vendors and standards bodies (e.g. IETF RFC). The protocol models also typically take into account variances in each protocol’s implementation.

The primary drawback of stateful protocol analysis methods is that they are very resource-intensive because of the complexity of the analysis and the overhead involved in performing state tracking for many simultaneous sessions. Another serious problem is that stateful protocol analysis methods cannot detect attacks that do not violate the characteristics of generally acceptable protocol behaviour, such as performing many benign actions in a short period of time to cause a denial of service. Yet another problem is that the protocol model used by an IDPS might conflict with the way the protocol is implemented in particular versions of specific applications and operating systems, or how different client and server implementations of the protocol interact.

A.1.1 Positioning Security Appliances in the GÉANT Network

As the GÉANT network comprises NRENs, with all the differences and specific local networks, no specific recommendation can be made at the time of writing. A wide variety of issues must be taken into account before solutions can be proposed. An adequate anomaly detection on the backbone, if security appliances are

adopted by NRENs, must point to any devices that are breached, not configured adequately or have other, similar problems.

By implementing security appliances on the connecting point between NRENs and the backbone network, inter-domain problems regarding malicious traffic can be effectively resolved. However, running such a system from a global point may entail a problem for inter-domain services. As no system is 100% foolproof, a significant number of false positive on important types of traffic can be expected. The speed of the reaction to such cases is of great importance, and it may be difficult to achieve adequate results.

Other methods of intrusion detection and/or prevention may involve observing anomaly detection on the backbone and reacting upon positive detection. In those cases, the management of the intrusion is shifted to the global GÉANT network, and the mitigation of the detected anomaly is implemented at the source of the problem (i.e. the NREN the malicious traffic originated from). In this scenario, both the detection of malicious traffic and its impact on multi-domain services has to be taken into account.

A.2 Protocols and Features for Network Monitoring and Anomaly Detection

Apart from features inside network devices that perform threat recognition and mitigation, there are protocols that extract collected traffic data from devices for further external processing. Beside the Netflow and Sflow protocols, the port mirroring switching feature also copies traffic to a dedicated interface. With proper analysers, traffic collected using these protocols and methods can be further processed and analysed.

Currently, NetFlow is widely used in the GÉANT network and acquired data is analysed using various tools, mainly NfSen.

A.2.1 NetFlow and sFlow

Like NetFlow, sFlow is a push technology that sends reports to a collector. However, while NetFlow is a software based technology, sFlow uses a dedicated chip that is built into the hardware. This approach removes the load from the CPU and memory of the router or switch.

Alcatel, Allied Telesis, Extreme Networks, Foundry Networks, HP, Hitachi, Juniper Networks, NEC and a few others have devices with integrated sFlow chips. sFlow is not nearly as widely deployed as NetFlow so fewer collectors are available.

sFlow is a sample-only technology where every X packet is sampled, its length noted and the majority of the packet is discarded before it is sent to the collector. Because the technology is sample based, an accurate representation of 100 percent of the traffic per interface is nearly impossible.

If NetFlow is properly configured and the hardware is not overloaded, this technology can be nearly 100 percent accurate at representing who is communicating through the device.

A.2.2 Port Mirroring

Port Mirroring enables administrators to monitor network traffic and switch performance. It works by forwarding a copy of all inbound and outbound packets from one port of a switch to another port (designated by an administrator).

To use Port Mirroring an administrator must specify from which inbound and outbound packets are copied and to which port the packets will be sent. A copy of every inbound and outbound packet destined for the first port will be sent to the second port as well. When configuring port mirroring at least one source and one destination port need to be specified. These are the port number from which traffic is copied and to which traffic is copied.

A protocol analyser is used on the port that receives the copied data. This allows taffic to be captured and analysed without affecting the normal operation of the switch.

Some switches are only capable of supporting one-to-one port mirroring. Here, a copy of each incoming and outgoing packet from a single port is forwarded to another port on the switch. This means there can only be a single source port. In a many-to-one relationship, multiple source ports can be configured to forward a copy of all inbound and outbound traffic to one destination port.

Port mirroring functionality is mainly for switches and thus limited for LAN analyses.

References

[AMUN]	http://amunhoney.sourceforge.net/
[APWG]	http://www.antiphishing.org
[ATTYP]	Dijiang Huang, Qing Cao, Amit Sinha, Marc J. Schniederjans, Cory Beard, Lein Harn, and Deep Medhi “New Architecture for Intra-Domain Network”, COMMUNICATIONS OF THE ACM, November 2006.
[BGPmon]	http://bgpmon.net
[CYCLOPS]	http://cyclops.cs.ucla.edu
[CYMRU]	https://www.team-cymru.org
[DDD]	http://blog.synacknetworks.com/2010/01/10/using-nfesn-to-identify-dosddos-attacks/
[GUIFIRE]	Karen Scarfone, Paul Hoffman “Guidelines on Firewalls and Firewall Policy”, National Institute of Standards and Technology, September 2009.
[GUIIPDS]	Karen Scarfone, Peter Mell “Guide to Intrusion Detection and Prevention Systems (IDPS)”, National Institute of Standards and Technology, February 2007.
[GUISEC]	Timothy Grance, Marc Stevens, Marissa Myers “Guide to Selecting Information Technology Security Products”, National Institute of Standards and Technology, October 2003.
[GoogleB]	http://googleonlinesecurity.blogspot.com/
[GoogleS]	http://safebrowsingalerts.googlelabs.com/
[HONEYD]	Niels Provos, “A Virtual Honeypot Framework”, In Proceedings of the 13th USENIX Security Symposium
[InfoSecMag]	http://www.infosecurity-magazine.com/view/13620/bredolab-downed-botnet-linked-with-spamitcom
[IPhij]	http://en.wikipedia.org/wiki/ip_hijacking (accessed 03/12/2010)
[ITTRQ3]	"2010 Q3 Internet Threats Trend Report", Commtouch http://www.commtouch.com/download/1850
[Koh06]	Kohler, E. et al. Observed structure of addresses in IP traffic. IEEE/ACM Transactions on Networking 14(6): 2006, p. 1207–1218.
[LCD04]	Lakhina, A.; Crovella, M.; Diot, C. Mining anomalies using traffic feature distributions. Proc. ACM SIGCOMM’05, 2005, p. 217–228.
[MK06]	Mohacsi, J.; Kiss, G. Anomaly Detection for NfSen/nfdump Netflow Engine with Holt-Winters Algorithm. Meeting of the GN3/JRA2 activity, Espoo, Finland, September 2006. http://bakacsin.ki.iif.hu/~kissg/project/nfsen-hw/JRA2-meeting-at-Espoo_slides.pdf
[NEPENTHES]	Paul Baecher, Markus Koetter, Maximillian Dornseif, Felix Freiling, “The nepenthes platform: An efficient approach to collect malware”, In Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID 06).

[nfdump] http://nfdump.sourceforge.net/

[NfSenInt] http://nfsen.sourceforge.net/#mozTocId859236

[NfSen] http://nfsen.sourceforge.net

[NLE10] Nicolas Falliere, Liam O Murchu, and Eric Chien, W32.Stuxnet Dossier Version 1.3 (November 2010), Symantec http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf

[RAID] Giroire, F., Chandrashekar, J., Taft, N., Schooler, E., Papagiannaki, D., "Exploiting Temporal Persistence to Detect Covert Botnet Channels", RAID '09: Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection, Springer-Verlag, 2009.

[RIPE] https://www.db.ripe.net/whois

[Rou10] Routly, W. et al. Anomaly detection in backbone networks: building a security service upon an innovative tool. In: TERENA Networking Conference, Vilnius, 2010. http://tnc2010.terena.org/schedule/presentations/show.php?pres_id=65

[ROUSEC] Multiple authors "Router Security Guidance Activity of the System and Network Attack Center (SNAC)" National Security Agency, December 15 2005

[Rub08] Rubinstein, B. I. P. et al. Compromising PCA-based Anomaly Detectors for Network-Wide Traffic. Technical Report UCB/EECS-2008-73, Berkeley: UCB, 2008.

[SFRFC] Peter Phaal, Marc Lavine, "sFlow Version 5", sFlow.org, July 2004

[SHADOW-HONEY] K. G. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. Markatos, A. D. Keromytis, "Detecting targeted attacks using shadow honeypots", In Proceedings of the 14th USENIX Security Symposium

[Shade] http://shadowserver.org

[Shaderep] http://www.shadowserver.org/wiki/pmwiki.php/Involve/GetReportsOnYourNetwork

[SSPNov] "State of Spam & Phishing", Report #47, November 2010, Symantec, http://www.symantec.com/content/en/us/enterprise/other_resources/b-state_of_spam_and_phishing_report_11-2010.en-us.pdf

[TCC] https://www.team-cymru.org/Services/TCConsole

[Tel09] Tellenbach, B. et al. Beyond Shannon: characterizing Internet traffic with generalized entropy metrics. In S. B. Moon et al. (Eds.): PAM 2009, 2009, p. 239–248.

[USENIX] Gu, G., Perdisci, R., Zhang, J., Lee, W., "BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection", 17th USENIX Security Symposium, 2008.

[WikiBGP] http://en.wikipedia.org/wiki/Bgp (accessed 03/12/2010)

[WikiBotnet] http://en.wikipedia.org/wiki/Botnet (accessed 03/12/2010)

[WIKIFIRE] http://en.wikipedia.org/wiki/Firewall_%28computing%29 (accessed 03/12/2010)

[WIKIIPS] http://en.wikipedia.org/wiki/Intrusion_prevention_system (accessed 03/12/2010)

[WikiMelissa] http://en.wikipedia.org/wiki/Melissa_virus (accessed 03/12/2010)

[WikiMorris] http://en.wikipedia.org/wiki/Morris_worm (accessed 03/12/2010)

[WikiNetFlow] http://en.wikipedia.org/wiki/Netflow (accessed 03/12/2010)

[WikiPLC] http://en.wikipedia.org/wiki/Programmable_logic_controller (accessed 03/12/2010)

Glossary

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASN	Autonomous System Number
BGP	Border Gateway Protocol
BSD	Berkeley Software Distribution
C&C	Command and Control
CPU	Central Processing Unit
CSI file	The Argos system saves memory dumps in CSI files.
CSIRTs	Computer Security Incident Response Teams
DCE/RPC	Distributed Computing Environment / Remote Procedure Calls
DNS	Domain Name System
DNSBLs	DNS-based Blackhole Lists
DoS	Denial of Service
DDOS	Distributed Denial of Service
FTP	File Transfer Protocol
GPL	GNU General Public License
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IRC	Internet Relay Chat
ISP	Internet Service Provider
LAN	Local Area Network
NAT	Network Address Translation
NIC	Network Interface Controller
NREN	National Research and Education Network
P2P	Peer to Peer
PHP	PHP: Hypertext Preprocessor
RPC	Remote Procedure Call
RRD	Round-Robin Database
SMB	Server Message Block
SMTP	Simple Mail Transfer Protocol
SPF	Sender Policy Framework

SSH	Secure Shell
TCP	Transmission Control Protocol
URL	Uniform Resource Locator
USB	Universal Serial Bus
X-ARTF	Extension of the Abuse Reporting Format (ARF)
XML	Extensible Markup Language