SEVENTH FRAMEWORK PROGRAMME

Information & Communication Technologies
ICT

Cooperation Programme

Nippon-European Cyberdefense-Oriented Multilayer threat Analysis
[†]

Deliverable D3.1: Policy Enforcement Point Survey

Contractual Date of Delivery	November 30th 2013
Actual Date of Delivery	November 30th 2013
Deliverable Dissemination Level	Public
Editors	Youki Kadobayashi, Jouni Viinikka
Contributors	All *NECOMA* partners

The *NECOMA* consortium consists of:

Institut Mines-Telecom	Coordinator	France
ATOS SPAIN SA	Principal Contractor	Spain
FORTH-ICS	Principal Contractor	Greece
NASK	Principal Contractor	Poland
6CURE SAS	Principal Contractor	France
Nara Institute of Science and Technology	Coordinator	Japan
IIJ - Innovation Institute	Principal Contractor	Japan
National Institute of Informatics	Principal Contractor	Japan
Keio University	Principal Contractor	Japan
The University of Tokyo	Principal Contractor	Japan

[†] The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7-ICT-2013-EU-Japan) under grant agreement n° 608533, and the Strategic International Collaborative R&D Promotion Project of the Ministry of Internal Affairs and Communication, Japan.

Contents

1 Introduction

The main objective of workpackage 3 is to provide defense mechanisms against cyber attacks and malware. By defense mechanisms, we mean any element(s) of the protected network/system/application that can be *reconfigured* in response to an attack, coupled with the logic allowing its reconfiguration. Such a component is called a Policy Enforcement Point (PEP).

This report will survey existing components that could serve as Policy Enforcement Points for the needs of *NECOMA*. We are interested in understanding the capabilites - what type of countermeasures the equipment can provide - and their manageability - how we can reconfigure these elements.

We seek to reuse existing components, such as switches, routers, firewalls, intrusion prevention systems, proxies, user directories, virtual machine hypervisors as PEPs when available; and thus need to understand what possibilities they offer for countermeasure enforcement.

More specifically, we will study PEPs for *infrastructure* and *endpoint* layers. As defined in the Description of Work [55], infrastructure layer encompasses both networks and hosting centers that together form the service side of the Internet, and endpoint layer encompasses all the terminals used to access the aforementioned infrastructure.

For each domain, we shall provide a more precise definition, and for each component, provide a general description, identify its capabilities and the configuration options we have for managing that component.

One important parameter in the reconfiguration process is the threat analysis information which allows to understand against what the reconfiguration is supposed to react. In other words, we are looking for capabilities that can be linked with the types of threat analysis results coming from workpackage 2, as our objective in other tasks of this workpackage is to choose and deploy countermeasures based on threat analysis data.

Furthermore, we are looking for capabilities corresponding to the needs of the scenarios of workpackage 4 as these scenarios will serve as the test

setting for validating the work done in the project. One challenge in the choice of PEPs to include in this study comes from this deliverable being the first technical deliverable of the project - workpackage 1, workpackage 2 and workpackage 4 are still taking their form or haven't even started yet.

We have chosen to take a rather high-level approach on describing the capabilities and management possibilites. In each category of PEPs, the precise capabilities vary between different implementations and in many cases the number of different implementations is important, and for the management side the situation is maybe even worse. As the details of other technical work packages unwind, we can more easily identify which categories of PEPs are suitable for using the available threat data in the context of our scenarios.

We will discuss potential policy enforcement points for infrastructure layer in Chap. 2 and for endpoint layer in Chap. 3.

2 Policy Enforcement Points for Infrastructure

In this chapter, we will present potential policy enforcement points for infrastructure layer. As mentioned earlier, infrastructure layer encompasses both networks and hosting centers that together form the service side of the Internet.

At the network level, we will be looking at PEPs oriented for Denial of Service (DoS) attack mitigation and at the data-center level PEPs for defending against a little bit wider spectrum of attacks.

The network level PEPs consist of switches, routers, network firewalls, proxies and DDoS mitigation equipment; and SDN controllers and hardware. In addition to existing solutions, we will include in consideration related work in research, that could potentially be implemented as a new PEP.

2.1 Switches

We consider a switch to be a network device linking network segments or other network devices at OSI layer 2. If a switch is so called layer 3 switch with routing capabilities, we consider those capabilities apart in the Sect. 2.2.

2.1.1 Capabilities

Switches typically have capabilities falling into categories of access control and logging.

VLAN assignment Switches control the assignment of virtual LANs, based for example on the switch port or on the source MAC address. By changing the VLAN assignment of a port it is possible to quarantine a suspicious host to specific quarantine VLAN that can for example have limited access to other resources in the network.

ACLs Switches can filter, typically inbound, traffic on switch ports, based on either MAC address or IP address.

Bandwidth limiting A switch can limit the maximal bandwidth allowed for a port. There are potentially several ways to achieve this.

2.1.2 Management

For our purposes we consider high-end, manageable switches providing a real command line interface. So called smart switches or switches with web-based management provide more or less control over the switch through a web-based GUI, which is much more complicated to use in automated manner. Typically a managed switch can be configured interactively through a shell session, either by using a serial console, telnet or ssh session. Such an interface would typically allow issuing the required commands for setting up the countermeasure. Another option would be generating the whole configuration file, which could then be pushed to the switch for upload. This approach could also work for smart switches.

2.2 Routers

Routers, as well as layer 3 switches, provide variety of capabilities in addition to providing connectivity between networks. Such capabilities primarily depend on information contained in the packet headers, thus they are not appropriate instruments for enforcing policy at the application layer.

2.2.1 Capabilities

Mitigation Capabilities Routers provide a variety of mitigation capabilities, as outlined below.

- **ACL** Access Control List (ACL), or access list, provides mitigation capabilities by dropping attack packets, when the traffic feature of an attack is simplistic and when it can be expressed in a minimal set of ACLs.
- **Policy-based Routing** Similar to ACL, policy-based routing can divert traffic to different route such that attack traffic is segregated from the rest of traffic, thus minimizing damages to operational networks, e.g., by reducing congestion. Since policy-based routing is essentially a feature extension of ACL, the granularity of control is identical to ACL.
- **Black Hole Routing** Black hole routing is a traffic mitigation technique that each BGP border router on the destination/attacked

AS forwards the unwanted traffic to the Null device. In comparison with ACL, routers can save CPU resources by using the black hole routing. One prominent scheme is [79], which describes an operational technique that utilizes BGP routing to create BGP-triggered black hole routing and Sinkhole Tunnels. This scheme can distribute discard routes that are based on destination addresses of unwanted traffic. Another prominent scheme is [45], that couples unicast Reverse Path Forwarding (uRPF) [6] with the BGP black hole technique to distribute discard routes based on source addresses of unwanted traffic.

Sinkhole Routing Similar to the black hole routing, sinkhole routing gathers unwanted traffic to a certain point on the network instead of the Null interface of the border routers. According to Cisco Systems[14], sinkhole routing was originally used by ISPs to engulf attack traffic, in many cases drawing attacks away from a customer or other target. Recently, sinkholes have been used in enterprise environments to monitor attacks, detect scanning activity from infected machines, and generally monitor other malicious activities.

Shunt Routing Shunt routing is a bypass technique to route the suspected traffic through an inspection / clearing point on the network. The difference between sink hole routing and shunt routing is that shunt routing sends back the traffic from the inspection / clearing point to the normal path, and delivers the traffic to the original destination. Basically, black hole routing and sink hole routing can be applied only on the destination/attacked AS. On the other hand, shunt routing can be implemented at all possible entry points from which attacks can pass into the destination/attacked AS.

Shunt routing can be achieved by various tunneling techniques. One prominent such mechanism is CenterTrack [72], which introduces a Tracking Router (TR), a special type of router connected with the edge router physically or virtually with an IP tunnel in a network. All TRs should also be connected to a central TR via IP tunnels, resulting in a total overlay network. If an attack is detected, a victim node sends the relevant traffic feature information to a TR. The TR uses this information to analyze and block unwanted traffic. “Sinkhole tunnels” described in [79] is another shunt routing technique.

Multiprotocol Label Switching Multiprotocol Label Switching(MPLS) is a network architecture allowing packet routing based on labels[63]. When incoming on the border of an MPLS network, a label is assigned to the packet and withdrawn when outgoing. Labels are

assigned depending on Forwarding Equivalent Class(FEC), representative of packet attributes(ip, port, protocol, etc.). Each FEC may have specific criteria transmission(QoS, explicit route, etc.). Every core routers will take forwarding decision based only on the label, regardless other packet attributes.

An MPLS network can do Implicit Routing using Label Distribution Protocol (LDP) in addition to an Interior Gateway Protocol (IGP) like Open Shortest Path First (OSPF) or Intermediate System to Intermediate System (IS-IS). MPLS network also enable establishment of Explicit Routing, allowing Traffic Engineering (TE) an Quality of Service (QoS) through protocols such as Resource Reservation Protocol - Traffic Engineering (RSVP-TE) or Constraint-based Routing Label Distribution Protocol (CR-LDP). Implicit and explicit routing can be used along, by using explicit routing to do TE and QoS on particular part of the traffic and letting the implicit routing basically route the other part of the traffic.

For our purpose, this network architecture, thanks to explicit routing protocols, allow flow categorization for prioritization [28] and redirection purposes(black holing, traffic cleaning, etc.).

Capabilities are basically the same between RSVP-TE and CR-LDP, see 2.1 for a sum up of main capabilities for both of them.

Table 2.1: Main capabilities of RSVP-TE and CR-LDP, two signaling protocols for MPLS network

	RSVP-TE	CR-LDP
QoS	x	x
Traffic Parameters	x	x
Failure Notification	x	x
Failure Recovery	x	x
Loop Detection	x	x
Multi-Protocol Support	x	x
Management	x	x
Record Route objects	x	-
Path Preemption	x	x
Path Re-optimization	-	x

However, there is differences on particular points, that may become critical depending on the situation. For example, RSVP-TE use UDP or Raw IP to exchange all information where CR-LDP use UDP for discovery and TCP to exchange session information. Because of the use of UDP by RSVP-TE, the failure of a peer is detected only when neighbor routers get no answer to a refresh

message, while that in the case of CR-LDP, failure of a peer is detected almost instantly when the TCP session with neighbors is interrupted. On that point, CR-LDP seems to be more efficient with TCP session, but on the other side, if a TCP session is interrupted, every channel allocated to particular traffic(for QoS and TE purpose) between two routers are destroyed. That is why RSVP-TE is called ”Soft state” against CR-LDP which is ”Hard state”.

To conclude, both protocol have pros and cons. The choice between them has to be made depending on specific constraints of the network (high reactivity on route failure, low signaling information rate, etc.).

Access Control Capabilities ACLs can restrict predetermined set of communications from entering or leaving particular network, based on IP address and TCP/UDP port number. They can be considered as rudimentary access control, whereas network firewalls, proxies and IDPS can provide more granular control up to application layer.

Encryption Capabilities Routers can provide encryption between configured set of peers, and they are typically implemented together with layered encapsulation of packets, often known as Virtual Private Network (VPN).

Logging Capabilities Routers can be configured to perform flow sampling and packet sampling, which is then diverted to external logging or analysis device. Samples can be used to detect volume anomalies through analysis software.

2.2.2 Management

Routers have distinct set of command-line interface which shares same characteristics with high-end switches. Typically, the command-line interface of both routers and high-end switches are designed for interactive use by network operators, such that the interface does not provide capabilities of fully functional programming languages, such as conditional statements and loop statements. Some routers provides a fully functional programming language within its own operating system, or an application programming interface for use by external software. Although a number of standards have been developed to provide uniform configuration interface, none of them have been successful to replace the command line interface.

2.3 Network Firewalls

A network firewall controls incoming and outgoing network traffic and determines whether the given packet should be allowed to continue on its way or not. The decisions are based on the firewall ruleset, typically based on layer 3 and layer 4 information contained in the packet headers (e.g. the source and destination IP addresses and ports and the protocol) ; and eventually on the session state of the connection.

2.3.1 Capabilities

Logging capabilities The firewalls can typically generate logs for more or less every packet passing through them - depending on the firewall the log might be a line of text or the actual packet. Because of performance and storage considerations the amount of logging is often limited in operational context. By adding the amount of logging as a countermeasure, for example for a given source and/or destination address, it would be possible to reinforce the data collection concerning suspicious communication endpoints.

Access control capabilities The firewalls can accept or deny network traffic based on the packet header information and eventually on the session state information. By modifying the ruleset it is possible to prohibit the communication between given endpoints e.g. in order to prohibit a suspicious external host from communicating with internal host or prevent data being sent out from a compromised host. This capability can apply only to the sessions being established after the countermeasure has been deployed or it might be able to possible to kill already established sessions, depending on the actual firewall.

Mitigation capabilities Firewalls can provide mitigation capabilities such as use of SYN cookies to avoid spoofed TCP traffic from reaching the target server. In addition, firewalls can provide some kind of rate limiting capabilities to, for example, limit the number of packets allowed from a given source IP address in a given time window.

2.3.2 Management

The configuration of the firewall varies largely from one model to another. Many software-based, open source solutions like Linux netfilter[1], OpenBSD pf[2], and FreeBSD ipfw[3] allow either loading the ruleset from a text file and

[1] http://www.netfilter.org

[2] http://www.openbsd.org/faq/pf/index.html

[3] http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/firewalls-ipfw.html

incremental modifications adding or removing rules through command line utilities, e.g. iptables for netfilter and pfctl for pf.

There exists also tools using a higher lever configuration language than provided by the firewall's native rule syntax, e.g. FireHOL[4] for netfilter.

One of the challenges in deploying countermeasures is to maintain the existing ruleset intact when adding or removing rules to enforce a countermeasure.

2.3.3 Prior Work on Network Firewalls

A lot of work has been done over the previous years in the area of (traditional) firewalls [13, 51, 52]. [88] and [46] describe different approaches to host-based enforcement of security policy. These mechanisms depend on the IP addresses for access control, although they could potentially be extended to support some credential-based policy mechanism. The Napoleon system [77] defines a layered group-based access control scheme. Policies are compiled to Access Control Lists (ACLs) appropriate for each application and pushed out to them at policy creation or update time. The STRONGMAN project [40, 42, 41] at the University of Pennsylvania is aiming at simplifying security policy management by providing an application-independent policy specification language that can be compiled to application-specific KeyNote credentials. These credentials can then be distributed to applications, hosts, and end users and used in an integrated policy framework. The Adage/Pledge system uses SSL and X.509-based authentication to provide applications with a library that allows centralized rights management. [5] presents an in-depth discussion of the advantages and disadvantages of credential-based access control. SnareWork [36] is a DCE-based system that can provide transparent security services (including access control) to end-applications, through use of wrapper modules that understand the application-specific protocols. Policies are compiled to ACLs and distributed to the various hosts in the secured network, although a pull-based method can also be used. Connections to protected ports are reported to a local security manager which decides whether to drop, allow, or forward them (using DCE RPC) to a remote host, based on the ACLs. Another approach [8] uses a network grouping language that is customized for each managed firewall at that firewall. The language used is independent of the firewalls and routers used. In [34], Ioannidis *et al.* introduced the concept of a "distributed firewall" in order to address the shortcomings of traditional firewalls which rely only on controlled entry points in restricted locations to enforce traffic filtering. This work presents the design and implementation of a distributed firewall able to cope with the new networking trends in-

[4] http://firehol.org/

cluding increased connectivity, high line speeds, extranets and large-sized networks in general.

2.4 Filtering Proxies

Filtering proxies are software components that inspect application-layer traffic, in order to prevent incidents from happening. They can also be used to keep track of possible incidents, or to provide traces of incidents for later analysis.

2.4.1 Capabilities

Detection Capabilities Filtering proxies implement inspection based on application-layer identifiers such as URL and domain names, together with network-layer addresses. URL-based or DNS-based filtering, as well as IP-based filtering can be commonly found in filtering proxies. It also conducts content inspection, such as virus scanning for malware mitigation, or dictionary-based scanning for data-loss prevention.

Mitigation Capabilities As stated above, filtering proxies are able to carry out inspection or to delegate this inspection for further content processing. Processing applied to either inbound or outbound traffic include blocking, stripping, privacy or load balancing.

Access Control Capabilities Some web proxies provide granular control so called application recognition, as web protocols have been used to deliver diverse web-based applications and cloud-based applications, and some enterprises define policies to limit certain types of such applications.

Authentication Capabilities Some filtering proxies provide user authentication in order to implement role-based access control.

Logging Capabilities Some filtering proxies also provide logging and reporting functions to provide better account of application-layer activities that may require later analysis.

Encryption Capabilities Some privacy-enhancing such as Privoxy[5] do not rely on encryption to ensure privacy, with more or less success. On the contrary, Tor[6] embeds original data into several layers of encryption. More generally, some anonymization proxies, known as Anonymous HTTPS proxy will rely on HTTPS to prevent their customers' activities from being monitored.

[5] http://www.privoxy.org
[6] http://www.torproject.org

2.4.2 Management

Filtering proxies are part of the network infrastructure and may be seen as part of the routing-capable devices. As a matter of fact, it will redirect or divert traffic from or to hosts on the network based on the analysis results carried out at the application layer.

Reverse proxies stand in front of or within application servers and are usually configured either via a configuration file present in the target application servers or via a web user interface when available. Detection capabilities rely often on regular expressions that are enabled or disabled through the configuration interface by selecting one or several attack classes. As for access control, either the administrator chooses to declare a blacklist of paths which are denied access to most users (in particular, external ones), or a whitelist of accessible paths which is built during a learning period. Authentication and encryption can be performed by filtering proxies, usually as external (through ICAP delegation[7]) or internal service respectively. In the latter case, the proxy acts as a certificate authority and holds the required certificates.

For HTTP(S) client proxies, the configuration usually features a whitelist or a blacklist of URLs or contents that users may or may not access. Proxies can be configured through configuration files in text format, or through a web interface.

2.5 Network-based IDPS

In general, an intrusion detection process is a method of monitoring events within a computer system or network and analyzing them for signs of possible violations or threats of violating computer security policies, acceptable use policies, or standard security practices. An intrusion detection and prevention system (IDPS) is a software component that automates the intrusion detection process and can also attempt to stop possible incidents. IDPS technologies offer many capabilities, and administrators can usually disable their prevention features, causing them to function as IDSs. A network-based IDPS monitors network traffic for particular network segments or devices and analyzes network, transport, and application protocols to identify suspicious activity.

2.5.1 Capabilities

Information Gathering Capabilities Network-based IDPSs offer information gathering capabilities such as collecting information on hosts and the network activity involving those hosts. An IDPS sensors might

[7] https://tools.ietf.org/html/rfc3507

be configured to create a list of hosts on the organizations network arranged by IP address or MAC address. Another capability related to information gathering is identifying the OSs and OS versions used by the organizations hosts. This can be achieved using various techniques such a as tracking which ports are used on each host or analyzing packet headers for certain characteristics, which could indicate a particular OS or OS family. Some IDPS sensors may also collect general information about network traffic related to the configuration of network devices and hosts, such as the number of hops between two devices. This information can be used to detect changes to the network configuration.

Logging Capabilities Usually Network-based IDPSs perform extensive logging of data related to detected events. Stored logs may be utilized afterwords to perform in-depth incidents investigation, confirmation of alerts validity and also for correlating events between different components within the network.

Examples of data fields logged by network-based IDPSs:

- Timestamp
- Connection or session ID
- Rating (e.g., priority, severity, impact, confidence)
- Network layer protocols
- Source and destination IP addresses
- Decoded payload data, such as application requests and responses
- Connection or session ID
- State-related information (e.g., authenticated username)
- Prevention action performed (if any).

Detection Capabilities Network-based IDPSs offer extensive and broad detection capabilities. Most products use a combination of signature-based detection, anomaly-based detection, and stateful protocol analysis techniques to perform in-depth analysis of the available data.

- Signature-based detection is a way of detecting events based on comparing current units of activity, such as a packet or a log entry, to a list of signatures using string comparison operations. Signature-based detection is very effective at detecting known threats but is not capable of detecting unknown threats.
- Anomaly-based detection is a process of comparing profiles of activities that are considered normal against observed events. The profiles are developed by monitoring the characteristics of typical activity over a period of time. If certain activities appear to

deviate significantly from the normal profiles, those activities are considered as malicious.

- Stateful Protocol Analysis relies on vendor-developed universal profiles that specify how particular protocols should and should not be used.

Prevention Capabilities Network-based IDPS sensors offer various prevention capabilities

- Passive - A passive sensor can attempt to end an existing TCP session by sending TCP reset packets to both endpoints.
- Inline - IDPS sensors offer firewall capabilities that can reject suspicious network activity as well as limiting bandwidth usage for particular protocols.
- Passive and Inline - IDPS sensors can instruct network security devices such as firewalls, routers, and switches to reconfigure themselves to block certain types of activity or route it elsewhere.

2.5.2 Management

IDPS products offer various management capabilities depending on the functionalities they provide. IDPS is capable of monitoring various network components and able to correlate data coming from them in order to effectively analyze events coming from the network. In addition, the IDPS should offer both, Passive and Inline, prevention capabilities. There are numerous IDPS solutions available that fulfill the specified requirements. In most cases, the commercial products do not offer much customized options such extending the system with custom-made plugins. Thus, our interest is aimed more at open-source solutions that could be easily extended and be able to interact in the future with components developed within the *NECOMA* project. Several such solutions were identified (e.g., ACARM-ng and Snort), offering full customization and compatibility with third party software that further supplement the IDPS system. Administration of those open-source IDPS is possible through scripting languages plugins, through a GUI or remotely with third party applications.

2.5.3 Prior Work on Network-based IDS and IPS Systems

In this section, we present former work on network-based intrusion detection and prevention systems emphasizing the ways used for detection as well as the respective performance of such approaches.

2.5.3.1 Pattern Matching

Pattern matching is the most critical operation that affects the performance of network intrusion detection systems. Pattern matching algorithms can be classified into single- and multi-pattern algorithms.

In single pattern matching algorithms, each pattern is searched in a given text individually. This means that if we have k patterns to be searched, the algorithm must be repeated k times. Knuth-Morris-Pratt [43] and Boyer-Moore [12] are some of the most widely used single pattern matching algorithms. Knuth-Morris-Pratt is able to skip characters when a mismatch occurs in the comparison phase using a partial-match table for each pattern. Each table is built by preprocessing every pattern separately. Boyer-Moore is the most widely used single-pattern algorithm. Its execution time can be sublinear if the suffix of the string to be searched for appears infrequently in the input stream, due to the skipping heuristics that it uses.

Multi-pattern string matching algorithms search for a set of patterns in a body of text simultaneously. This is achieved by preprocessing the set of patterns and building an automaton that will be used in the matching phase to scan the text. The automaton can be thought of as a state machine that is represented as a trie, a table or a combination of the two. Each character of the text will be searched only once. Multi-pattern matching scales much better than algorithms that search for each pattern individually. Multi-pattern string matching algorithms include Aho-Corasick [1], Wu-Manber [89] and Commentz-Walter [17].

Most Network Intrusion Detection Systems (NIDS) use finite automata and regular expressions [62, 57, 35] to match patterns. There have been many works focusing on improvements and optimizations of pattern matching algorithms in the current IDS systems. Coit *et al.* [16] improved the performance of Snort by combining the Aho-Corasick keyword trie with the skipping feature of the Boyer-Moore algorithm. Fisk and Vaghese enhance the Boyer-Moore-Horspool algorithm to simultaneously match a set of rules. The new algorithm, called Set-wise Boyer-Moore-Horspool [23], was shown to be faster than both Aho-Corasick and Boyer-Moore for sets with less than 100 patterns. Tuck *et al.* [78] optimized the Aho-Corasick algorithm by applying bitmap node and path compression.

Snort from version 2.6 and onwards uses only flavors of the Aho-Corasick for exact-match pattern detection. Specifically, it contains a variety of implementations that are differentiated by the type of the finite automaton they use (NFA or DFA), and the storage format they use to keep it in memory (full, sparse, banded, trie, *etc.*). It should be mentioned, however, that the best performance is achieved with the full version that uses a deterministic finite automaton (DFA) at the cost of high memory utilization [76].

2.5.3.2 Hardware Implementations

To speed-up the inspection process, many IDS implementations are based on specialized hardware. By using content addressable memory (CAM), which is suitable to perform parallel comparison for its contents against the input value, they are very well suited for use in intrusion detection systems [91, 92]. However they have a high cost per bit.

ASIC/FPGA Many reconfigurable architectures have been implemented for intrusion detection. Most approaches involve building an automaton for a string to be searched, generating a specialized hardware circuit using gates and flip-flops for the automaton, and then instantiating multiple such automata in the reconfigurable chip to search the streaming data in parallel. However, the circuit implemented on the FPGA to perform the string matching is designed based on the underlying hardware architecture to adjust to a given specific rule set. To adjust to a new rule set, one must program a new circuit (usually in a hardware description language), which is then compiled down through the use of CAD tools. Any changes in the rule set requires the recompilation, regeneration of the automaton, resynthesis, replacement and routing of the circuits which is a time consuming and difficult procedure.

Sidhu and Prasanna implemented a regular expression matching architecture for FPGAs [70]. Baker *et al.* also investigated efficient pattern matching as a signature based method [7]. In [19], the authors used hardware bloom filters to match multiple patterns against network packets at constant time. Attig *et al.* proposed a framework for packet header processing in combination with payload content scanning on FPGAs [4].

Several approaches attempt to reduce the amount of memory required to economically fit it in on-chip memory [7, 78, 20]. However, the on-chip hardware resource consumption grows linearly with the number of characters to be searched. In [73], the authors convert a string set into many tiny state machines, each of which searches for a portion of the strings and a portion of the bits of each string.

2.5.3.3 Assistance of Network Processors

Other approaches involve the cooperation with network processors in order to pipeline the processing stages assigned to each hardware resource [15], as well as the entire implementation of an IDS on a network processor [11, 18].

2.5.3.4 Cluster-based Approaches

In another line of work, cluster-based approaches have been proposed for keeping-up with the increasing link speeds [44, 87, 80, 65, 24, 66, 90]. Instead of having a single server to process all incoming traffic, a cluster

of servers is used instead. The major issue then is how to partition the incoming traffic to the back-end servers, while supporting stateful processing. Kruegel et al. [44] propose a stateful slicing mechanism that divides the overall network traffic into subsets of manageable size, which are then processed by different sensors. Foschini et al. [24] extend that work with a parallel matching algorithm that allows communication between the sensors through a dedicated control plane. SPANIDS [66] uses a specialized FPGA-based switch, that takes into account flow information and the load of each server when redirecting network packets. Xinidis et al. [90] present an active splitter architecture that provides early filtering to reduce the load of the back-end sensors. The cost however of these cluster-based solutions remains high, since it requires multiple processors, a distribution network, and a clustered management system.

2.5.3.5 Using GPUs

On the contrary, modern GPUs have low design cost while their increased programmability makes them more flexible than ASICs. Most graphic cards manufacturers provide high-level APIs that offer high programming capabilities and are further ensure forward compatibility for future releases, in contrast with most FPGA implementations that are based on the underlying hardware architecture and need to be reconfigured whenever a change occurs in the rule set. Furthermore, their low design cost, the highly parallel computation and the potential that are usually underutilized, especially in hosts used for intrusion detection purposes, makes them suitable for use as an extra low-cost coprocessor for time-consuming problems, like pattern matching. There have been many works trying to use GPU capabilities in order to improve the current state of IDS and IPS systems. PixelSnort [37] is a port of the Snort IDS that offloads packet matching to an NVIDIA 6800GT. The GPU programming was complicated, since the 6800GT did not support a general purpose programming model for GPUs (as the G80 used in our work). The system encodes Snort rules and packets to textures and performs the string searching using the KMP algorithm on the 16 fragment shaders in parallel. However, PixelSnort *does not* achieve *any* speed-up under normal-load conditions. Furthermore, PixelSnort did not have any multi-pattern matching algorithms ported to GPU. This is a serious limitation since multi-pattern matching algorithms are the default for Snort. Moreover, Marziale *et al.* [49] evaluated the effectiveness of offloading the processing of a file carving tool to the GPU. The system was implemented on the G80 architecture and the results show that GPU support can substantially increase the performance of digital forensics software that relies on binary string search. Vasiliadis *et al.* introduced Gnort [81, 82], which was the first attempt that sufficiently utilized the graphics processor for string searching and regular expression matching. Gnort [81] utilize a full state table representation,

which requires quite large memory amounts, but provide linear complexity independent of the number of patterns. Unfortunately, its single-threaded architecture restricts its scalability in the advent of multi-core CPUs. Many other approaches followed the above scheme [30, 71], without significant differences in the architecture and the performance benefits. In a more recent work, Vasiliadis *et al.* [83] introduced MIDeA, a multi-parallel intrusion detection architecture as a scalable solution for the processing and stateful analysis of network traffic. MIDeA parallelizes network traffic processing and analysis at three levels, using multi-queue NICs, multiple CPUs, and multiple GPUs in order to cope with multi-Gigabit networks. Kargus [38], is another recent highly-scalable NIDS that exploits GPUs to achieve high performance in incoming packets processing.

2.6 DDoS Mitigation Solutions

DDoS mitigation solutions are specialized in filtering out traffic related to denial of service attacks, distributed or not ((D)DoS), while trying to maintain better availability of the protected resource than with black hole routing or classic firewall functionality for example. These solutions also have a focus on the capability of handling very large traffic rates and volumes, a typical characteristic of many DDoS attacks event though low volume attacks exist, especially on the application layer.

As far as we know, there are currently only commercial, closed-source solutions available, such as 6cure Threat Protection[8], Arbor TMS[9], Corero's First Line of Defense[10]

2.6.1 Capabilities

Detection capabilities As these solutions are expressly aimed at filtering attack traffic they must first detect such traffic. Thus they provide capability of detecting different kinds of traffic anomalies e.g. in terms of rate and/or volumetry, in the versatility of traffic, and specific patterns for abusing protocol flaws.

Access control capabilities These solutions provide often also basic firewall style filtering capabilities, which allow enforcement of network level access control policies.

Mitigation capabilities First of all, these solutions provide ways to mitigate DDoS attacks by different means, varying from basic IP blacklist to

[8] http://www.6cure.com/contents/6cure-tp-productsheet_v1-0.pdf
[9] http://www.arbornetworks.com/products/peakflow/tms
[10] http://www.corero.com/resources/files/datasheets/solutions/Solution_Brief_First_Line_of_Defense_Overview.pdf

selective blocking based on behavioral analysis of the traffic source. In addition they may provide means for enforcing application level controls such as acceptable request types and/or parameter values.

2.6.2 Management

Being closed source, commercial solutions, their manageability is typically rather limited from the point of view of third party controllers such as those we aim to define and develop in project *NECOMA*. Some of the solutions can be controlled by creating and pushing a configuration file in a similar manner to loading a firewall rule set; some of the solutions require configuration through a Web interface.

2.7 SDN

In recent years Software-Defined Networking (SDN) has emerged as one of the most promising networking paradigms, featured with network programmability and dynamic network virtualization. In particular, the essential networking functionalities of hardware infrastructure are abstracted to the software plane, resulting in a decoupling between control plane and data plane. As such, the complexity of network administration and management could be significantly simplified, and a large variety of networking applications can be implemented and deployed through SDN controllers in flexible, scalable, and friendly ways. However, the defending landscape of SDN is meanwhile fundamentally reshaped because of the separation of control plane and data plane in network perimeters such as switches, routers and other networking appliances. It is thus significant and urgent to examine how the relevant security policies, either existing ones or novel ones, can be enforced at appropriate granularities, in order to better design and deploy cost-effective security mechanisms in this novel networking environments.

2.7.1 Capability

As one of the major advantages, the typical security mechanisms such as firewalls and IDSs could be easily implemented in SDN and have potential to enable centralized monitoring, prevention, detection, and reaction. More details about the potential capabilities are given as follows.

2.7.1.1 Enabling Security as an Application

The programmability and openness of SDN controller potentially enable security as on-demand service. Specifically, a wide range of customized security applications can be programmatically embedded into SDN via controller

layer on the basis of user, application, or flow. Unlike current security devices that are designed for implementing particular security functions, SDN users can create a security application by him/herself. For example, OpenFlow, a protocol facilitating the communication between SDN controller and data plane, defines a set of control APIs to manage openflow switches' behavior by flow table, which sets specific rules for the ongoing packets. In particular, OpenFlow supports a set of actions for handling a packet: drop, forward, enqueue and modify-field. Thanks to these APIs, a security application can be implemented. For instance, a firewall function can be easily implemented by simply specifying a drop action for a packet which has meets certain attributes like src/dst port number or src/dst ip address.

In fact, we have already seen some attempts about this. For example, a security applications oriented development framework, termed FRESCO, was proposed in [69], in which 16 modules were defined, and each of them has five interfaces: input, output, event, parameter and action. Then each interface can be further specified with different values, implementing the basic security functions such as IDS/IPS, firewall, traffic monitor. In addition, new modular libraries can be created using FRESCO scripts, which are then loaded and assembled as desirable security service, controlling network traffic through OpenFlow controllers and hardware.

2.7.1.2 Flexible Policy Enforcement

In traditional networks, the security policies are usually enforced by manual configuration at individual network components such as IDS/IPS, firewall, router ACLs, NATs and the like, which work as middleboxes and are installed in the physical path of the network. Thus, any modification or update of the policy may lead to a major change of network topology and configuration, incurring prohibitive operational cost and complexity. SDN has potential to alleviate such problems as the operators are empowered to gain a global and centralized control over high level abstractions, facilitating those security policies to be dynamically and optimally enforced in an automated manner. For example, a user can define a steering flow for a OpenFlow switch, which may forward packets to a middlebox and could be changed by a user-defined application. Thus, a new middlebox can be installed by simply adding a steering flow for the switch. Also, the programmable operation could reduce management complexity and cost.

To date, some research efforts have been observed on the related issues. For example, the authors of paper [29] has proposed a high-level policy language FSL for controller NOX, which can express basic network access controls, directionality in communication establishment (similar to NAT), network isolation (similar to VLANs), communication paths, and rate limits, supporting network-wide complex policy objectives. Also, a SDN-based policy enforcement layer named SIMPLE was reported in [59] for

middlebox-specific traffic steering, allowing network operators to specify a logical middlebox routing policy and automatically translates this into forwarding rules that take into account the physical topology, switch capacities, and middlebox resource constraints. A better trade-off between operational complexity, cost, and security benefits can therefore be achieved. Another middlebox-based approach, which is called FlowTags, was proposed in [22]. The approach requires the middleboxes to automatically add tags to the outgoing packets, which are then used for systematic police enforcement. In particular, if a middlebox changes a tag of a received packet, a SDN switch can easily recognize and steering the packets. However, to support this approach, some modifications of middlebox software might be necessary.

2.7.1.3 Adaptive Threat Monitoring

In SDN, specific flow rules could be dynamically established in order to re-direct or divert the flows of interest to a centralized or multiple PEPs for better pinpointing the anomalies. Such an operation could not be easily conducted in traditional networks. In paper [68], an approach termed CloudWatcher was proposed to enable security monitoring as a service.

In addition to the academic effort, we have also seen that some network and virtualization vendors, as well as standards groups, are working towards SDN-enabling security strategies. For example, DefenseFlow of Radware is claimed to be the industry's first SDN application that programs networks for DoS security, proactively defending against network flood attacks and providing network-wide attack mitigation services.

2.7.2 Management

It is clear that SDN controllers could seamlessly bridge the gap between network operators and underlying network infrastructure, significantly simplifying the management and operational complexity of security policy enforcement, potentially allowing network administrators to gain a centralized control over the entire network, and globally leveraging the landscape of the deployed security mechanisms to optimize their collective performance, ultimately achieving automated, adaptive, and flexible security control.

Another salient feature is that diverse management interfaces can be offered to network operator through SDN controllers, which are always impossible in traditional networks. For example, a user can remotely manage security devices through SDN controller by simply using a web UI or a dedicate program running on the operator's computer.

Moreover, the SDN applications can collaborate with each other like cloud management system (CMS), and such a collaboration is called *Orchestration*. One of the communication methods between the applications is

RESTful API, which can be used for security applications as well. For example, the firewall rules can be queried and updated using pieces of perl code via RESTful API. Although such a functionality is already available in virtual firewalls, SDN facilitates its integration and interoperability with other applications.

2.8 LISP

2.8.1 LISP Overview

LISP (Locator/ID Separation Protocol) [21] is a new routing method on the Internet. LISP enables separation of IP addresses into two new numbering spaces: Routing Locators (RLOCs) and Endpoint Identifiers (EIDs). RLOCs are IP addresses of network attachment points for routing and forwarding packets through the network. EIDs are nonrouteable IP addresses of end devices.

LISP defines a function for mapping between RLOCs and EIDs on Map Server[25]. When LISP router received a packet, the router send a request which RLOC has the packet's destination address to a map server. The packet is encapusreted with the address and forwarded to the router. At the router, the packet is decapsulated and transfered to the internal network by general IP routing.

2.8.2 Capability

LISP separate a IP address space into RLOC and EID. This separated architecture increases network mobility. If a network space is moved to the other, we just change a RLOC of the network.

The architecture could be use to lead a malicious traffic to another network by changing a map of RLOCs and EIDs.

2.8.3 Management

LISP router and map server have already available in a market. Also, some projects provide open source implementations of LISP router and map server. LISP router management is configured by same manner of other routers. An operator who is configure LISP network, they should register their RLOCs and EID information to LISP Map server.

2.9 HyperVisor

HyperVisor is software that can operate Virtual Machines (VMs) using host virtualization technology. By using abstraction of resources such as CPU

cores, memory regions, hard disk drives, and network interfaces that Operating Systems require to run, multiple VMs are possible to run on a physical host (bare metal).

From the standpoint of VM users, a VM is constructed using the virtual resouces provided by a HyperVisor and a user can install and operate their own OS on a VM. The benefits of deploying HyperVisor are that user can operate the number of VMs greater than the number of physical hosts, and operate each VM separately.

2.9.1 Capabilities

As typical implementations of HyperVisor which realize full virtuallization of VMs, there are VMware, Xen, and kvm. Also there are the implementations with partial virtualization such as OpenVZ and LXC (Linux Containers). They provide separated user-spaces to each VM, however the kernel on all VMs are shared.

Further, some of HyperVisor implementations include an implementation of virtual network switch. It is a network switch which is composed from software, and provides functions of network separation and filtering between HyperVisor and VM.

PEPs of HyperVisor are composed of resource access controls in HyperVisor, network access controls in HyperVisor, resource access controls of VM, network access controls of VM, and access controls of a virtual network switch.

2.9.2 Management

Resource access contorols in HyperVisor HyperVisor provides access controls of resources for VMs. More specifically, it provides the limit of CPU usage, limit of memory usage, selection and separation of networks, and exclusive control of external devices such as HDD and USB. An administorator can decide and provide the policies of the limitations. VMs use the resources that are completely separated by a HyperVisor, so there is no need to care about resource violations between VMs.

Network access controls in HyperVisor HyperVisor can separate and virtualize networks by using VLAN, VXLAN and MAC address control, and provide separated networks to the network interfaces of VMs. Regarding separation, HyperVisor can implement and deploy the policies such as providing an isolated network to each VM, providing multiple networks to each VM or providing a network to multiple VMs. Furthermore, some HyperVisor implementations have a function of packet

filtering. Such a HyperVisor can restrict access to itself to trusted networks and also can control the network access to VMs running on itself.

Resource access controls in VM VM can use the resources freely only within the limit of the virtual resources provided from HyperVisor. In other words, it is not possible to access the resources of a VM from other VMs. All policy management of the resources is confined within the VM.

Network access control in VM VM has access privileges only to networks provided by HyperVisor. Because the network is one of the virtual resources provided from HyperVisor, it is treated the same as other virtual resources. Thus, OS running on VM can perform packet filtering on a virtual host basis, if it has a packet filtering function.

Virtual Network Switch Virtual Network Switch is a Layer-2 network switch which is composed of software and is operated within HyperVisor. Open vSwitch is a typical implementation of Virtual Network Switch. It usually is located between the physical network interfaces of a physical host and the virtual network interfaces of VMs. Similar to a physical Layer-2 network switch, a virtual switch provides functions of network separation, for instance by using VXLAN, VLAN, MAC address filtering, and rule-based packet filtering. Thus it is possible to filter and classify packets per VM, providing more flexible networks to VMs.

3

Policy Enforcement Points for Endpoints

Endpoint policy enforcement, with the rapidly advancing technologies, has emerged to be a security issue as significant as securing the corporate network infrastructure itself. Not long ago most of the computing devices were located within the corporate premises under the direct control of the IT department. Now, however, the corporate networks are no longer limited to the corporate perimeters and beyond the managed devices that connect to the network through external providers and proxies, are expected to support unmanaged, non-standard devices such as smartphones, Macs and tablets. PCs have become more functional with the growth of variety of software and supported technologies. Adding to those devices such as tablets and smartphones expands the potential surface area for an attack.

The aforementioned devices are potential targets of many and varied attacks. Such attacks may include compromising PCs or mobile devices with malicious software that is used for data harvesting, monitoring keystrokes, monitoring screen activity or network traffic to intercept sensitive information. It may also include scanning the device's storage for specific types of files to stream them back to the C and C server. Malicious software can also be used to carry out an attack on networks. The infection can be spread to other devices connected to the same network exploiting the same vulnerabilities used to primarily perform the attack.

Devices or any kind of removable media, such as USB drives or external hard drives, can get stolen or lost.

In this chapter will explore enforcement solutions for endpoints that can help to tackle previously mentioned risks. The following list encompasses identified enforcement points for endpoints:

1. Antivirus solution
2. Host-based application firewalls
3. Disk encryption and file system-level encryption

4. Other required configurations, such as certain registry settings
5. Host-based IDPS
6. Network Access Control (NAC)
7. Browser

3.1 Antivirus software

Antivirus software is a computer program that detects, prevents, and takes action to disarm or remove malicious software programs, such as : computer viruses, malicious BHOs, hijackers, ransomware, keyloggers, backdoors, rootkits, trojan horses, worms, malicious LSPs, dialers, fraudtools, adware and spyware.

Antivirus programs running on endpoints use several methods to detect malicious software:

3.1.1 Capabilities

Signature Based Detection - Signature Based Detection is based on scanning the executable code of computer files and cross-referencing their content with the signatures of known viruses. Because new malicious software keeps emerging on an almost everyday basis, software vendors work constantly assessing new threats keeping their signature libraries up to date. With the new generation of malicious software such as ”polymorphic” and ”metamorphic” viruses some argue that signature based detection is currently obsolete. Despite that, is still remains as the core technique for detecting malicious software.

Heuristics - Heuristics analyses are applied to detect malicious software by using a rule-based approach. The analyser engine checks the files and processes running, against criteria that may indicate possible malware, assigning ”score points” when a certain rule is meet. If the score reaches particular score, the files are flagged as threats or potentially dangerous.

Once malicious software is detected, antivirus programs typically quarantine or encrypt detected dangerous files rendering them useless.

3.1.2 Management

Almost all antivirus solutions are closed-sourced and do not allow manipulation of the algorithms used to detect malware nor any reconfiguration of the antivirus application. They also offer very limited (if any) interaction with

third party components and systems. The existing open-source antivirus solutions has not proven to be efficient enough to be considered as a sufficient solution for an end-point PEP within the *NECOMA* project.

3.2 Host-based application firewalls

Host-based application firewalls are basically filters that operate between software components and network components of an operating system. Installing such a firewall implants it in key places on the application-network path, analysing the traffic between them against a specified rule set. If the analysed piece of traffic meets the criteria specified in the rule set, it is allowed to pass through the firewall, otherwise it is blocked. An application firewall can be split into two main components:

1. Packet level analysis component
2. Process level analysis component

The packet level analysis component takes care of analysing the packets themselves looking for malformed packed and detecting port scans. The component assesses if a packet should be allowed to pass or not depending on the networking criteria.

The process level analysis component checks if a particular process should be allowed to initiate a connection with a given host through a specific port or listen on a given range of ports.

3.2.1 New Directions

Currently the firewalls whether they run on the host or a network element are based on static rules that are being created either by a system administrator (network level) or a user (host level). Firewalls themselves are key elements where policies are enforced. The rules created on a host based firewall are mostly based on user experience and in some cases (if not configured correctly) could make users life hard (*e.g.* deny DNS traffic, users cannot use local printers, cannot access shared storage via Samba or NFS protocol etc). Many of the users would be keen on using a local host-based firewall on their device but the majority probably is not willing to deal with all the configuration parameters. Configuring those parameters correctly could lead to a firewall installation operational and effective.

Thus, it seems that may be space for a solution that allows automated blocking of IP addresses on most of the popular firewall implementations. Such an implementation could be part of two elements:

1. an output plugin of a PDP element (network element that is able to observe malicious traffic/ IPs/packets and create some primitive rules)

2. and an intelligent agent that runs on the firewall, or a host near the firewall that is able to create the appropriate rules and enforce the policy received.

3.3 File systems

File systems organize the information stored on a disk volume into files that are identified by names and indexed. There exist many different implementations that differ in structure and logic. File systems are usually characterized by the way they allocate space for the files to be written, the naming system they use, the granularity of the directory structure. This information is usually kept as metadata along with the size of the file, the access privileges or the creation and modification timestamps.

3.3.1 Capabilities

Access Control File systems are able to control who access the data stored on the target volume and in which manner. Typical privileges include `read`, `write` and `execute`. Access control is usually implemented through permission bits, access control lists (ACLs) or capabilities.

Accountability While preventing access from unauthorized users is desirable, we also need to monitor any suspicious access done by authorized users. File systems feature such ability to record events about user access to files, along with the update of the last access data in the metadata, especially in case the file has been altered.

Mitigation When a file system has been compromised through, for example, the execution of a malicious file, its propagation can be mitigated if the directory it resides is isolated. Such isolation capability may prevent access to external resources from within a certain directory. For example, the UNIX command `chroot` allows changing the file system root of the environment in which runs a process, possibly malicious.

Encryption Some file systems allow to encrypt part or whole of the disk volume. This is detailed in Section 3.4.

3.3.2 Management

While file access permissions are usually set by default at the time a file is created, their modification can be done through invoked commands or a graphical interface.

While metadata is a native feature, it may not be enough to ensure good security capabilities. Thus, the implementation of other capabilities may require to install some third-party programs to be installed. Typicall isolation

and encryption perfectly integrate with the file system but are not native capabilities.

Additionally, file system commands can be invoked in automated scripts.

3.4 Disk encryption and file system-level encryption

Encryption is a mechanism which protects data by ciphering the information with a cryptographic algorithm. Encrypted data cannot be read without deciphering it with the same algorithm and applying the same key that was introduced in the encryption process. There are two basic types of storage endpoint encryption:

1. Full disk encryption (FDE)
2. File system-level encryption

Full disk encryption is a technique which encrypts the whole disk when shutting the system down leaving only a boot volume unencrypted so that the system can start up again. This mechanism leaves all data unencrypted while the system is running what allows for normal functioning, but also because of that, is an attacker gains access to the device at run-time all files will be readable for him.

File system-level encryption is used to encrypt (using the same or similar process for encrypting) particular files or directories on the device storage drive. This allows using different keys for encrypting different parts of the hard drive. This mechanism decrypts the files only when they are required during system run-time. Thus, it might interfere with the normal system functioning but does not allow attackers to gain instant access to all files and directories.

3.5 Other required configurations, such as certain registry settings in Windows OS

One of the goals of operating systems is to be as functional and user-friendly as possible. However those extended functionalities and user friendliness come at a cost of lowering the security of the operating system and opening it to more potential threats. Those security flaws can be addressed by additional configuration applied to various system components such as the Windows Registry (in case of Windows OS). There are several methods to perform such changes, all of them having pros and cons. The three main identified methods are:

1. Manual configurations

2. Scripts

3. Custom ADM templates

Manual configuration is usually done with the use of Regedit application that is provided with the Windows OS (although there are other tools that can perform such changes). The problem with that solution is that it can be applied to one device at a time. If a system administrator has to perform a change to the registry in all of the operating systems running within a company, this process gets very inefficient.

A more sophisticated approach would be the use of scripts. Using scripts for modifying the registry is very common and also very powerful since scripts can change basically anything. Scripts can be created in various ways including the use of tools dedicated for that task. The drawback of using scripts is that those have to be triggered by an event such as the device starting up or restarting. If the computer is already started and the user has already logged on, the script won't automatically execute.

Custom Administrative Templates allow the registry changes to be refreshed automatically in the background. These templates use a simple coding syntax to establish the correct Registry path, value and value data that needs to be changed. Although the ADM templates provide a simple way to perform the registry changes, the problem is that they have to address the "allowed" area of the registry.

3.5.1 Capabilities

Access Control - Each key in the registry of Windows can have an assosiated security descriptor. The security descriptor contains an access control list (ACL) that describes which user groups or individual users are granted or denied access permissions. Windows Resource Protection works by setting discretionary access control lists (DACLs) and access control lists (ACLs) defined for protected resources.

3.5.2 Management

For *NECOMA* purposes we would consider managing permission (ACLs) on Windows systems from a centralised location preventing attackers from accesing critical resources hosed on the end-point. This kind of funtionality if offered by various comercial tools. However the software solutions provided by the vendors are closed-sourced and require a licence purchase.

3.6 Host-based IDPS

A host-based IDPS monitors the characteristics of a single host and the events occurring within that host for suspicious activity. Examples of the types of characteristics a host-based IDPS might monitor are wired and wireless network traffic (only for that host), system logs, running processes, file access and modification, and system and application configuration changes.

3.6.1 Capabilities

Logging Capabilities Host-based IDPSs typically perform extensive logging of data related to detected events. This data can be used to confirm the validity of alerts, to investigate incidents, and to correlate events between the host-based IDPS and other logging sources.

Detection Capabilities Most host-based IDPSs have the capability to detect several types of malicious activity. They often use a combination of signature-based detection techniques to identify known attacks, and anomaly-based detection techniques with policies or rulesets to identify previously unknown attacks. The types of events detected by host-based IDPSs vary considerably based primarily on the detection techniques that they use. Some host-based IDPS products offer several of these detection techniques, while others focus on a few or one.

Code Analysis Before code is run normally on a host, it can first be executed in a virtual environment or a sandbox to analyze its behavior and compare it to profiles or rules of known good and bad behavior.

Network Traffic Analysis This is often similar to what a network-based IDPS does.

Network Traffic Filtering Agents often include a host-based firewall that can restrict incoming and outgoing traffic for each application on the system.

Filesystem Monitoring This involves periodically generating message digests or other cryptographic checksums for critical files, comparing them to reference values, and identifying differences.

Log Analysis Some agents can monitor and analyze OS and application logs to identify malicious activity.

Network Configuration Monitoring Some agents can monitor hosts current network configuration and detect changes. Typically all network interfaces on the host are monitored, including wired, wireless, virtual private network (VPN), and modem.

Prevention Capabilities Host-based IDPS agents offer various intrusion prevention capabilities. Capabilities vary based on the detection techniques used by each product, the following items describe the capabilities by detection technique.

Code Analysis The code analysis techniques can prevent code from being executed.

Network Traffic Analysis Network traffic analysis techniques can stop incoming network traffic from being processed by the host and outgoing network traffic from exiting it.

Network Traffic Filtering Working as a host-based firewall, this can stop unauthorized access and acceptable use policy violations.

Filesystem Monitoring Filesystem Monitoring can prevent files from being accessed, modified, replaced, or deleted.

Other Capabilities Some host-based IDPSs offer non-IDPS capabilities such as antivirus software and Web or e-mail content filtering. Examples of such capabilites:

Process Status Monitoring Some products monitor the status of processes or services running on a host, and if they detect that one has stopped, they restart it automatically.

Network Traffic Sanitization Some agents, particularly those deployed on appliances, can sanitize (rewrite) the network traffic that they monitor deleting any unexpected content.

3.6.2 Management

Most host-based IDPSs have detection software known as agents installed on the hosts of interest. Each agent monitors activity on a single host and if IDPS capabilities are enabled, also performs prevention actions. Host-based IDPSs usually are confiured on the host itself by a console or a GUI provided by the software. Although more sophisticated IDPS agents transmit data to management servers, which may optionally use database servers for storage. Those IDPSs may also be configured remotely from a management and monitoring central server. Usually commercial products offer limited customization capabilites seldomly allowing third-party scripts to interact with the IDPS system. However, there are several open-sourced solutions available (e.g., ACARM-ng and Snort) that offer the aforementioned capabilities as well as extending their functionalities by custom plugin. Such open-sourced IDPS also offer compatibility with third-party software that extend the detection and prevention capabilities of the IDPS. Administration of those open-source IDPS is possible through scripting languages plugins, through a GUI or remotely with third party applications.

3.7 Network Access Control (NAC)

Network access control is a method of enforcing the security of a proprietary network by restricting the availability of network resources to endpoint devices that comply with a defined security policy. NAC uses a set of protocols to define and implement policies that are mandatory to comply with when devices attempt to access the network. NAC might also integrate an automatic remediation process (fixing non-compliant endpoints before allowing access) into the network systems, ensuring that the endpoint is operating securely before interoperability is allowed. The endpoint compliance check is usually performed by a preinstalled software agent prepared by the network administrators.

Such agents may perform the following policies check:

1. Checking if the antivirus software is up to date
2. Checking if all the software has up to date security patches applied
3. Checking if the host-based firewall is properly configured
4. Checking if crucial directories and files are properly secured
5. Checking if the registry settings are properly configured

But the software agents are not limited to that list and may be configured accordingly to the specific network needs. Typically software agents are centrally managed at management console, and they are collectively known as endpoint protection or vulnerability management products.

3.7.1 Prior Work on Network Access Control

Here we provide previous work related with Network Access Control (NAC) in different domains, such as file systems, distributed applications, databases *etc.*

In [50], Miltchev *et al.* provide a survey of decentralized access control in popular production and experimental distributed file systems in order to identify essential properties of such mechanisms. In [32], Ioannidis *et al.* introduced *Virtual Private Services* (VPS), which are distributed applications which require coordination amongst client, servers and networks to deliver a reliable, secure service to client. VPS are designed and built for a single ubiquitous security policy, which will be enforced everywhere (nodes, networks, *etc.*). This approach unifies the management of all access control under a single global policy in order to address the new security problems arose from the increasing number of applications coming from heterogeneous software components interconnected by a network. In [47], Levine *et al.* make an effort of applying an access control mechanism in the web where

traditional access control strategies would be inflexible and non-intuitive. They designed and implemented a prototype of their system called WebDAVA for a secure web browser application. Same techniques can be used in a distributed file system or in a resource allocation mechanism as part of an operating system. In this work [31], the problem of security policy consistency in decentralized heterogeneous systems is addressed. Two novel techniques are proposed for maintaining consistency in these environments and a demonstration is shown on how it is possible for multiple security elements to dynamically *share* and *exchange* state information, to consistently enforce security policies that span multiple access control nodes.

3.8 Browser

Information exchange over the Internet is essential component of modern web based applications and applications in general. An enormous amount of traffic exchange is performed through web browsers running on a variety of devices (traditional PCs, tablets, smart phones etc). Network firewalls and access lists can provide generic protection to a device, but in order for the user to experience the look and feel of the Web as we know it, a lot of applications should be able and allowed to execute code within the context of a browser and the device itself.

3.8.1 Capabilities

Detection Capabilities The modern web browsers have phishing detection, which shows an alert if the site is a suspected phishing site. As similar to antivirus software, there are two types of algorithms for distinguishing phishing sites.

URL Filtering URL filtering detects phishing sites by comparing the URL of a site where a user visits with its URL blacklist, which is composed of the URLs of phishing sites. Web bowser alerts when a user visit a site whose URL is listed on the blacklist. Unfortunately, registering all phishing URLs into the blacklist is not feasible due to the rapid increasing of phishing sites. It is, however, still core portion in phishing detection, as similar to the signature-based detection in the context of antivirus software.

Heuristics-based algorithm A heuristic checks if a site seems to be a phishing site. In the context of phishing, lifetime of domain name, popularity of website, legitimate-surrouding URL, similarity of visual image, and so on. Based on the detection result from each heuristic, the heuristic-based solution calculates the likelihood of a site being a phishing site and compares the likelihood

with the defined discrimination threshold. Different from URL filtering, a heuristic-based solution has a possibility to identify unreported phishing sites. Notice that heuristics only give a hint to detect phishing sites, but do not provide the accurately information. Thus, Heuristics-based solutions usually employ statistical techniques including machine learning and clustering for integrating the detection results of two or more heuristics to improve the detection accuracy.

Protection Capabilities Browser vulnerabilities often allow malicious codes to run in computers and/or to steal secret information stored in the browsers. The modern browsers equip protection capabilities for preventing their components from attacks

Cookie If the session information stored in HTTP Cookie was stolen, the session will be hijacked; HTTP requests involving the information allow the hijacker to purchase using the stolen session. Same-origin policy mechanism protects Cookie from malicious content.

JavaScript JavaScript is a dynamic scripting language that has been widely used. As part of web browsers, implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed. However, JavaScript has been abused by malware providers (*e.g.* code injection through cross site scripting, also known as XXS attacks, etc.) to exploit numerous vulnerabilities in web browsers' plugins and extensions. PEP for web browser can disable JavaScript in the untrusted websites.

Plugins Although not part of the browser per se, browser plugins and extensions extend the attack surface. A plugins blocker which works in the web browser disables plugins' execution in order to protect the plugins from malware.

Encryption Capabilities Browsers support encrypted communication in order to ensure confidentiality between client and server. This capability protects against forging the contents of the communication, as well as eavesdropping.

Credibility Assessment Capabilities Browsers present information, usually padlock icons, whenever a web server equips an SSL server certificate. The certificate is checked if it is valid; the certificate is issued by trustworthy third parties, is not expired, and the fully qualified domain name of the server matches the common name in the SSL certificate. Extended Validation SSL certificates also show information

prominently to users. If a site used EV SSL certificates, the background of the address bar in users' browser turned from white to green; it can be useful to notice that users are visiting legitimate enterprises.

3.8.2 Management

The configuation of web browsers enables/disables the detection capabilities. Internet Explorer equips Smart Screen Filter, anti-phishing protection, and it detects phishing website. Google Chrome and Mozilla Firefox employ Google's SafeBrowsing API[1], which has phishing and malware protection features. Apple Safari contains built-in anti-phishing feature which will show an alert if the site is a suspected phishing site.

There exists also tools for protecting browsers. For instance, NoScript extension[2] disables JavaScript and plugins in the untrusted web site.

3.8.3 Prior Work on Browser Security

In this section we present previous work related with browser security. We first present some studies on web exploitation and defenses. Then, we proceed with some more sophisticated kind of web attacks that threaten browser users, and finally we describe security risks related with the new trend of browser to constitute a separate operating system.

3.8.3.1 Web Exploitation and Defenses

There are many works like BEEP [39], Noncespaces [27], xHunter [2] and DSI [53] that focus on the detection and prevention of XXS attacks. Moreover, there are also frameworks [3] that can cope with both XXS and *return-to-JavaScript* attacks and are capable to prevent attacks that leverage the content-sniffing algorithms of web browsers [9]. Blueprint [75] is a server-only approach which guarantees that untrusted content is not executed. The application server pre-renders the page and serves each web document in a form in which all dynamic content is correctly escaped to avoid possible code injections. Enforcing separation between structure and content is another prevention scheme for code injections [61]. This proposed framework can deal with XSS attacks as well as SQL injections. As far as XSS is concerned, the basic idea is that each web document has a well defined structure in contrast to a stream of bytes, as it is served nowadays by web servers. In [84] the authors propose to use dynamic tainting analysis to prevent XSS attacks. Taint-tracking has been partially or fully used in other similar approaches [53, 67, 56, 54].

[1] https://developers.google.com/safe-browsing/
[2] http://noscript.net/

3.8.3.2 Sophisticated Web Attacks

Through the years, as the web technologies evolved, exploitation of web applications also became more sophisticated. Apart from XSS and CSRF, there are many ways to exploit a web application. By sending specifically crafted data, someone can exploit bugs in an AJAX based client [64] or by including web pages as a Cascade Style Sheet (CSS) someone can steal a user's secret, for example the subjects of her inbox [48]. Moreover, even devices such as routers that use web interfaces for configuration can be exploited by injecting specific client-side code [10]. Although many of these attacks are recently discovered, we believe that in the future there will be many web application exploits that will rely on such techniques.

3.8.3.3 Browser Operating Systems

Recently, there is an effort for applying operating systems' principles in the web browser aiming at the creation of a more secure browser architecture. The research community is trying to transform the web browser into a mini-operating system, which offers web application separation. For the latest proposals we refer the reader to [85, 86, 26, 60, 74]. All these systems promote a browser architecture that processes web applications like traditional operating systems process native applications. So, sandboxing techniques [58, 33] would be essential in order a browser operating system to be able to guarantee that two web applications cannot interfere with each other.

4 Analysis

In the preceding chapters, we have covered existing components that could serve as Policy Enforcement Points for the needs of NECOMA. Through the survey, we were able to understand capabilities and manageability of individual components.

This chapter provides some analysis to the collective capabilities of PEPs, their interoperability, as well as their placement across networks and systems. More specifically, we can observe the following trend when we examine capabilities and management possibilities across all of reviewed PEPs:

1. Skewed placement of policy enforcement points toward enterprises and end users: vendor products, such as firewalls and antivirus software, primarily focus on enterprises at infrastructure layer and end-users at endpoint layer. The rest of constituents in cyberspace are largely left behind, i.e., organizations other than enterprises, Internet infrastructure, cloud service providers, and infrastructure of end-users. Such tendency to gravitate product focus toward "enterprise" and "home PC" metaphor significantly affects open-source software, as well as the isolated nature of most PEPs.

2. Few approaches exist to contain ongoing, evolving and dynamic threats: we observed strong tendency to exclude adversaries from internal system through detect-and-protect cycle which is programmed into both hardware and software of PEPs. Most PEPs assume that networks of devices have been governed by single or hierarchical policy, which is true for enterprise and home setting, whereas the rest of the cyberspace may not have clear definition of internal system and its associated policy. As such, exclusion-oriented approach should be augmented with analysis, mitigation and containment.

3. Limited manageability and programmability: most PEPs are built to interact through Web-based user interace or unique configuration lan-

guage. Adding diversity of log formats, implementing countermeasures that works across multiple PEPs can be a significant challenge. While there has been significant amount of standardization activity in past years, much more light-weight approach to achieve interoperability across PEPs seems to be necessary. In some cases, the deployment of countermeasures can be carried over standardized protocols, e.g., in case of traffic redirection (blackhole or other) using BGP – such options would be interesting from the genericity point of view.

The limited manageability clearly limits the number of policy enforcement points available for efforts like *NECOMA*– if only available configuration means are a graphical user interface or an opaque configuration file format, such a component cannot be reconfigured by automated means.

This also underlines the importance of defense mechanisms that would use a certain level of abstraction in the reasoning in how to react to ongoing threats. There are several PEPs that share similar capabilities and that could be used to implement a similar countermeasure, alone or in collaboration. For instance, switches, routers, network firewall, network-based IPS, and host-based firewall could all be used to prevent IP-level communication from a given source to a given destination. Even in each category the details of a concrete configuration associated to a countermeasure are different, not to mention between the different categories of PEPs. The reasoning should take place at a more abstract level to determine what we want to achieve with the countermeasure, then we need to choose in some way where and with which PEP(s) we wish to actually enforce the countermeasure, and only then should we go down to the level of concrete device configurations.

4. Lack of PEPs for new terminals such as smartphones. Even though the current smartphones are more and more like small computers, their operating environment is much more controlled than for general purpose computers: the only PEP encountered in this survey that could apply for smartphones would be the web browser.

Of course, most of PEPs are still under active development; capabilities of individual components have been enriched and their demarcation as a product have been consistently refactored. Regardless of these recent developments, we observe that the above trend remains largely intact.

The NECOMA project covers the major part of the spectrum of technologies that are used to prevent, detect and mitigate modern cyberthreats, through the participation of experts in the field of network infrastructure, cloud computing, trustworthy computing, cybersecurity information, web

security, incident response and forensics, in addition to the proven capabilities of the FP7 WOMBAT project such as malicious code acquisition and analysis. It should be noted that modern threats are manifesting through unexpected combination of different technologies; the NECOMA project targets the development and experimentation of vigilance and mitigation capabilities across such diverse set of technologies.

5 Conclusion

This report delivered a survey of existing security mechanisms (PEPs) that can be reconfigured in response to an attack. More specifically, we are looking into mitigating the effect of an attack or the effectiveness of malware. The mitigation can take place at various locations and layers, from the servers to the client machines, passing through the core network; and from the link layer to the application layer and even users.

The survey covered both infrastructure and endpoint PEPs, where individual capabilities and their manageability were reviewed. Throughout our survey, we noted: somewhat skewed placement of policy enforcement points toward enterprises and end users; few approaches were made to contain ongoing threats; and some PEPs provide limited manageability and programmability.

This survey confirms our initial observation and motivation to start NECOMA Project:

1. Development of actionable threat knowledge methodologies: there is a clear need to measure the state of the Internet threat, and traditional detect-and-protect cycle within individual PEPs needs to be complemented with reconfiguration-oriented and mitigation-oriented approaches, as well as higher-level comprehension of the threat landscape that induce appropriate action of human operators and managers.

2. Development of advanced cyberdefense mechanisms: PEPs could be harnessed by reconfiguring them in response to an attack. Modern threats are compositional and they can be instantiated through variety of means, thus we need to identify key control points in networked systems and combine them as necessary.

3. A complete pipeline from information to reaction: most PEPs require individual configuration based on countermeasure information, which

in turn requires translation from threat knowledge. At least for specific threat scenario, we should be able to construct a pipeline from information to reaction such that experts and devices in the different stages of pipeline can work together.

Based on this overview, and with respect to further work in this workpackage, it seems like existing PEPs could cover the needs of Task 3.3 *Resilience mechanisms for infrastructure*. For Task 3.4 *Resilience mechanisms for endpoints*, the development of new PEPs was already part of initial plans. With respect to the WP4 *Case studies*, the question whether we need to develop new PEPs or use existing ones remains more open, but for example it is clear that we lack in diversity in identified PEPs for smartphone user protection.

Bibliography

[1] A. V. Aho and M. J. Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, June 1975.

[2] E. Athanasopoulos, A. Krithinakis, and E. P. Markatos. Hunting Cross-Site Scripting Attacks in the Network. In *Proceedings of the 4th Workshop on Web 2.0 Security & Privacy (W2SP)*, Oakland, CA, May 2010.

[3] E. Athanasopoulos, V. Pappas, A. Krithinakis, S. Ligouras, and E. P. Markatos. xJS: Practical XSS Prevention for Web Application Development. In *Proceedings of the 1st USENIX WebApps Conference*, Boston, US, June 2010.

[4] M. Attig and J. Lockwood. A framework for rule processing in reconfigurable network systems. In *Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '05)*, pages 225–234, Washington, DC, USA, 2005. IEEE Computer Society.

[5] T. Aura. Distributed access rights management with delegation certificates. In *Secure Internet Programming*, pages 211–235.

[6] F. Baker and P. Savola. Ingress Filtering for Multihomed Networks. RFC 3704 (Best Current Practice), mar. 2004.

[7] Z. K. Baker and V. K. Prasanna. Time and area efficient pattern matching on FPGAs. In *Proceedings of the 2004 ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays (FPGA '04)*, pages 223–232, New York, NY, USA, 2004. ACM.

[8] Y. Bartal, A. Mayer, K. Nissim, and A. Wool. Firmato: A novel firewall management toolkit. *ACM Trans. Comput. Syst.*, 22(4):381–420, Nov. 2004.

[9] A. Barth, J. Caballero, and D. Song. Secure Content Sniffing for Web Browsers or How to Stop Papers from Reviewing Themselves. In *Proceedings of the 30th IEEE Symposium on Security & Privacy*, Oakland, CA, May 2009.

[10] H. Bojinov, E. Bursztein, and D. Boneh. XCS: Cross Channel Scripting and Its Impact on Web Applications. In *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*, pages 420–431, New York, NY, USA, 2009. ACM.

[11] H. Bos and K. Huang. Towards software-based signature detection for intrusion prevention on the network card. In *Proceedings of 8th International Symposium on Recent Advances in Intrusion Detection (RAID)*, Seattle, WA, September 2005.

[12] R. S. Boyer and J. S. Moore. A fast string searching algorithm. *Communications of the Association for Computing Machinery*, 20(10):762–772, October 1977.

[13] W. R. Cheswick and S. M. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1994.

[14] Cisco Systems. Remotely Triggered Black Hole Filtering in IP Version 6 for Cisco IOS, Cisco IOS XE, and Cisco IOS XR Software. http://www.cisco.com/web/about/security/intelligence/ipv6_rtbh.html.

[15] C. Clark, W. Lee, D. Schimmel, D. Contis, M. Kone, and A. Thomas. A hardware platform for network intrusion detection and prevention. In *Proceedings of the 3rd Workshop on Network Processors and Applications (NP3)*, 2004.

[16] C. Coit, S. Staniford, and J. McAlerney. Towards faster string matching for intrusion detection or exceeding the speed of Snort. In *Proceedings of DARPA Information Survivability Conference & Exposition II (DISCEX '01)*, June 2001.

[17] B. Commentz-Walter. A string matching algorithm fast on the average. In *Proceedings of the 6th International Colloquium on Automata, Languages and Programming*, pages 118–131.

[18] W. de Bruijn, A. Slowinska, K. van Reeuwijk, T. Hruby, L. Xu, and H. Bos. SafeCard: a Gigabit IPS on the network card. In *Proceedings of 9th International Symposium on Recent Advances in Intrusion Detection (RAID)*, Hamburg, Germany, September 2006.

[19] S. Dharmapurikar, P. Krishnamurthy, T. S. Sproull, and J. W. Lockwood. Deep packet inspection using parallel bloom filters. *IEEE Micro*, 24(1):52–61, 2004.

[20] S. Dharmapurikar and J. Lockwood. Fast and scalable pattern matching for content filtering. In *Proceedings of the 2005 ACM symposium on Architecture for networking and communications systems (ANCS '05)*, pages 183–192, New York, NY, USA, 2005. ACM.

[21] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. The Locator/ID Separation Protocol (LISP). RFC 6830 (Experimental), Jan. 2013.

[22] S. K. Fayazbakhsh, V. Sekar, M. Yu, and J. C. Mogul. Flowtags: Enforcing network-wide policies in the presence of dynamic middlebox actions. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, pages 19–24, New York, NY, USA, 2013. ACM.

[23] M. Fisk and G. Varghese. Applying fast string matching to intrusion detection. Technical Report In preparation, successor to UCSD TR CS2001-0670, University of California, San Diego, 2002.

[24] L. Foschini, A. V. Thapliyal, L. Cavallaro, C. Kruegel, and G. Vigna. A Parallel Architecture for Stateful, High-Speed Intrusion Detection. In *Proceedings of the 4th International Conference on Information Systems Security (ICISS)*, 2008.

[25] V. Fuller and D. Farinacci. Locator/ID Separation Protocol (LISP) Map-Server Interface. RFC 6833 (Experimental), Jan. 2013.

[26] C. Grier, S. Tang, and S. King. Secure Web Browsing with the OP Web Browser. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 402–416. IEEE, 2008.

[27] M. V. Gundy and H. Chen. Noncespaces: Using Randomization to Enforce Information Flow Tracking and Thwart Cross-Site Scripting Attacks. In *Proceedings of the 16th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 8-11, 2009.

[28] N. Hachem, H. Debar, and J. Garcia-Alfaro. HADEGA: A novel mpls-based mitigation solution to handle network attacks. In *Performance Computing and Communications Conference (IPCCC), 2012 IEEE 31st International*.

[29] T. Hinrichs, J. Mitchell, N. Gude, S. Shenker, and M. Casado. Expressing and enforcing flow-based network security policies. Technical report, 2008.

[30] N.-F. Huang, H.-W. Hung, S.-H. Lai, Y.-M. Chu, and W.-Y. Tsai. A GPU-Based Multiple-Pattern Matching Algorithm for Network Intrusion Detection Systems. In *Proceedings of the 22nd International Conference on Advanced Information Networking and Applications - Workshops (AINAW)*, 2008.

[31] S. Ioannidis. *Security Policy Consistency and Distributed Evaluation in Heterogeneous Environments*. University of Pennsylvania, 2005.

[32] S. Ioannidis, S. M. Bellovin, J. Ioannidis, A. D. Keromytis, K. G. Anagnostakis, and J. M. Smith. Virtual private services: Coordinated policy enforcement for distributed applications. *I. J. Network Security*, 4(1):69–80, 2007.

[33] S. Ioannidis, S. M. Bellovin, and S. J. M. Sub-operating systems: A new approach to application security. In *Proceedings of the 10th workshop on ACM SIGOPS European workshop*, 2002.

[34] S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith. Implementing a distributed firewall. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*, CCS '00, pages 190–199, New York, NY, USA, 2000. ACM.

[35] C. IOS. IPS deployment guide. http://www.cisco.com.

[36] C. J. and S. S. A transparent security framework for tcp/ip and legacy applications. Technical report, Intellisoft Corp., 1996.

[37] N. Jacob and C. Brodley. Offloading IDS computation to the GPU. In *Proceedings of the 22nd Annual Computer Security Applications Conference on Annual Computer Security Applications Conference (ACSAC '06)*, pages 371–380, Washington, DC, USA, 2006. IEEE Computer Society.

[38] M. A. Jamshed, J. Lee, S. Moon, I. Yun, D. Kim, S. Lee, Y. Yi, and K. Park. Kargus: A highly-scalable software-based intrusion detection system. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 317–328, New York, NY, USA, 2012. ACM.

[39] T. Jim, N. Swamy, and M. Hicks. Defeating Script Injection Attacks with Browser-Enforced Embedded Policies. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 601–610, New York, NY, USA, 2007. ACM.

[40] A. D. Keromytis. Strongman: A scalable solution to trust management in networks. 2001.

[41] A. D. Keromytis, K. Anagnostakis, S. Ioannidis, M. B. Greenwald, and J. M. Smith. Managing access control in large scale heterogeneous networks. In *In Proceedubgs of the NATO NC3A Symposium on Interoperable Networks for Secure Communications (INSC*, 2003.

[42] A. D. Keromytis, S. Ioannidis, M. B. Greenwald, and J. M. Smith. The strongman architecture. In *In Proceedings, DARPA Information Survivability Conference and Exhibition*, pages 178–188. IEEE Press, 2003.

[43] D. E. Knuth, J. Morris, and V. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):127–146, 1977.

[44] C. Kruegel, F. Valeur, G. Vigna, and R. Kemmerer. Stateful intrusion detection for high-speed networks. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 285– 294, May 2002.

[45] W. Kumari and D. McPherson. Remote Triggered Black Hole Filtering with Unicast Reverse Path Forwarding (uRPF). RFC 5635 (Informational), Aug. 2009.

[46] W. LeFebvre. Restricting network access to system daemons under sunos. In *Proceedings of the Third USENIX UNIX Security Symposium*, pages 93–103. USENIX Assoc, 1992.

[47] A. Levine, V. Prevelakis, J. Ioannidis, S. Ioannidis, and A. D. Keromytis. Webdava: An administrator-free approach to web file-sharing. In *Proceedings of the Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, WETICE '03, pages 59–. IEEE Computer Society, 2003.

[48] H. Lin-Shung, W. Zack, E. Chris, and J. Collin. Protecting Browsers from Cross-Origin CSS Attacks. In *CCS 10: Proceedings of the 17th ACM Conference on Computer and Communications Security*, New York, NY, USA, 2010. ACM.

[49] L. Marziale, Golden G. Richard III, and V. Roussev. Massive threading: Using GPUs to increase the performance of digital forensics tools. In *Proceedings of the 7th Annual Digital Forensics Research Workshop (DFRWS 2007)*, September 2007.

[50] S. Miltchev, J. M. Smith, V. Prevelakis, A. Keromytis, and S. Ioannidis. Decentralized access control in distributed file systems. *ACM Comput. Surv.*, 40(3):10:1–10:30, Aug. 2008.

[51] J. Mogul, R. Rashid, and M. Accetta. The packer filter: An efficient mechanism for user-level network code. In *Proceedings of the Eleventh ACM Symposium on Operating Systems Principles*, SOSP '87, pages 39–51, New York, NY, USA, 1987. ACM.

[52] J. C. Mogul. Simple and flexible datagram access controls fot unix-based gateways. In *Proceedings of Summer 1080 USENIX Technical Conference*, 1989.

[53] Y. Nadji, P. Saxena, and D. Song. Document Structure Integrity: A Robust Basis for Cross-site Scripting Defense. In *Proceedings of the 16th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 8-11, 2009.

[54] S. Nanda, L. Lam, and T. Chiueh. Dynamic Multi-Process Information Flow Tracking for Web Application Security. In *Proceedings of the 8th ACM/IFIP/USENIX international conference on Middleware*. ACM New York, NY, USA, 2007.

[55] *NECOMA* consortium. Project *NECOMA* - Description of Work. Call FP7-ICT-2013-EU-Japan proposal n° 608533, Apr. 2013.

[56] A. Nguyen-tuong, S. Guarnieri, D. Greene, J. Shirley, and D. Evans. Automatically Hardening Web Applications Using Precise Tainting. In *Proceedings of the 20th IFIP International Information Security Conference*, pages 372–382, 2005.

[57] V. Paxson. Bro: A system for detecting network intruders in real-time. In *Proceedings of the 7th conference on USENIX Security Symposium (SSYM '98)*, pages 3–3, Berkeley, CA, USA, 1998. USENIX Association.

[58] V. Prevelakis and D. Spinellis. Sandboxing applications. In *In Proceedings of the USENIX Technical Annual Conference, Freenix Track*, pages 119–126, 2001.

[59] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu. Simple-fying middlebox policy enforcement using sdn. In *SIGCOMM*, pages 27–38, 2013.

[60] C. Reis and S. Gribble. Isolating web programs in modern browser architectures. In *Proceedings of the 4th ACM European Conference on Computer Systems (EuroSys)*, pages 219–232. ACM, 2009.

[61] W. Robertson and G. Vigna. Static Enforcement of Web Application Integrity Through Strong Typing. In *Proceedings of the 18th USENIX Security Symposium*, Montreal, Quebec, August 2009.

[62] M. Roesch. Snort: Lightweight intrusion detection for networks. In *Proceedings of the 1999 USENIX LISA Systems Administration Conference*, November 1999.

[63] E. Rosen, A. Viswanathan, and R. Callon. RFC 3031: Multiprotocol label switching architecture, Jan. 2001.

[64] P. Saxena, S. Hanna, P. Poosankam, and D. Song. FLAX: Systematic Discovery of Client-side Validation Vulnerabilities in Rich Web Applications. In *Proceedings of the 17th Annual Network and Distributed System Security Symposium (NDSS)*.

[65] L. Schaelicke, K. Wheeler, and C. Freeland. SPANIDS: a scalable network intrusion detection loadbalancer. In *CF '05: Proceedings of the 2nd conference on Computing frontiers*, pages 315–322, New York, NY, USA, 2005. ACM.

[66] L. Schaelicke, K. Wheeler, and C. Freeland. SPANIDS: A Scalable Network Intrusion Detection Loadbalancer. In *Proceedings of the 2nd Conference on Computing Frontiers (CF)*, 2005.

[67] R. Sekar. An Efficient Black-box Technique for Defeating Web Application Attacks. In *Proceedings of the 16th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 8-11, 2009.

[68] S. Shin and G. Gu. Cloudwatcher: Network security monitoring using openflow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?). In *ICNP*, pages 1–6, 2012.

[69] S. Shin, P. A. Porras, V. Yegneswaran, M. W. Fong, G. Gu, and M. Tyson. Fresco: Modular composable security services for software-defined networks. In *NDSS*. The Internet Society, 2013.

[70] R. Sidhu and V. Prasanna. Fast regular expression matching using FPGAs. In *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM01)*, 2001.

[71] R. Smith, N. Goyal, J. Ormont, K. Sankaralingam, and C. Estan. Evaluating GPUs for Network Packet Signature Matching. In *Proceedings of the International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2009.

[72] R. Stone. CenterTrack: An IP Overlay Network for Tracking DoS Floods. In *USENIX Security Symposium*, pages 199–212, Aug 2000.

[73] L. Tan, B. Brotherton, and T. Sherwood. Bit-split string-matching engines for intrusion detection and prevention. *ACM Transactions on Architecture and Code Optimization*, 3(1):3–34, 2006.

[74] S. Tang, H. Mai, and S. King. Trust and Protection in the Illinois Browser Operating System. In *Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation (OSDI)*. USENIX, 2010.

[75] M. Ter Louw and V. Venkatakrishnan. Blueprint: Precise Browser-neutral Prevention of Cross-site Scripting Attacks. In *Proceedings of the 30th IEEE Symposium on Security & Privacy*, Oakland, CA, May 2009.

[76] The Snort Project. Snort users manual 2.8.0. http://www.snort.org/docs/snort_manual/2.8.0/snort_manual.pdf.

[77] D. J. Thomsen, D. O'Brien, and J. Bogle. Role-based access control framework for network enterprises. In *Proceedings of the 14th Annual Computer Security Applications Conference (ACSAC)*, pages 50–58. IEEE Computer Society, 1998.

[78] N. Tuck, T. Sherwood, B. Calder, and G. Varghese. Deterministic memory-efficient string matching algorithms for intrusion detection. In *Proceedings of the IEEE Infocom Conference*, pages 333–340, 2004.

[79] D. Turk. Configuring BGP to Block Denial-of-Service Attacks. RFC 3882 (Informational), Sep. 2004.

[80] M. Vallentin, R. Sommer, J. Lee, C. Leres, V. Paxson, and B. Tierney. The NIDS cluster: Scalable, stateful network intrusion detection on commodity hardware. In *Proceedings of the 10th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 107–126, 2007.

[81] G. Vasiliadis, S. Antonatos, M. Polychronakis, E. P. Markatos, and S. Ioannidis. Gnort: High Performance Network Intrusion Detection Using Graphics Processors. In *Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2008.

[82] G. Vasiliadis, M. Polychronakis, S. Antonatos, E. P. Markatos, and S. Ioannidis. Regular Expression Matching on Graphics Hardware for Intrusion Detection. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2009.

[83] G. Vasiliadis, M. Polychronakis, and S. Ioannidis. Midea: A multi-parallel intrusion detection architecture. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 297–308, New York, NY, USA, 2011. ACM.

[84] P. Vogt, F. Nentwich, N. Jovanovic, E. Kirda, C. Kruegel, and G. Vigna. Cross-Site Scripting Prevention with Dynamic Data Tainting and Static Analysis. In *Proceeding of the 14th Annual Network and Distributed System Security Symposium (NDSS)*, 2007.

[85] H. J. Wang, X. Fan, J. Howell, and C. Jackson. Protection and Communication Abstractions for Web Browsers in MashupOS. In *SOSP*, pages 1–16, 2007.

[86] H. J. Wang, C. Grier, A. Moshchuk, S. T. King, P. Choudhury, and H. Venter. The Multi-Principal OS Construction of the Gazelle Web Browser. In *Proceedings of the 18th USENIX Security Symposium*, Montreal, Canada, August 2009.

[87] K. Watanabe, N. Tsuruoka, and R. Himeno. Performance of network intrusion detection cluster system. In *Proceedings of The 5th International Symposium on High Performance Computing (ISHPC-V)*, 2003.

[88] V. Wietse. Tcp wrapper: Network monitoring, access control, and booby traps. In *Proceedings of the Third USENIX UNIX Security Symposium*, pages 85–92. USENIX Assoc, 1992.

[89] S. Wu and U. Manber. A fast algorithm for multi-pattern searching. Technical Report TR-94-17, 1994.

[90] K. Xinidis, I. Charitakis, S. Antonatos, K. G. Anagnostakis, and E. P. Markatos. An Active Splitter Architecture for Intrusion Detection and Prevention. *IEEE Transactions on Dependable and Secure Computing*, 3:31–44, January 2006.

[91] F. Yu, R. H. Katz, and T. V. Lakshman. Gigabit Rate Packet Pattern-Matching Using TCAM. In *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP '04)*, pages 174–183, Washington, DC, USA, October 2004. IEEE Computer Society.

[92] S. Yusuf and W. Luk. Bitwise optimised CAM for network intrusion detection systems. In *Proceedings of International Conference on Field Programmable Logic and Applications*, pages 444–449, 2005.