Network-based Intrusion Prevention System Prototype with Multi-Detection
A Position Paper

Daniel Kavan, Klára Škodová, and Martin Klíma
Department of Applied Research, CertiCon, a. s, Václavská 12, 120 00 Prague 2, Czech Republic

Keywords: Intrusion Prevention System, Intrusion Detection System, Network Behavior Analysis, Signature-based Detection, CAMNEP, Snort, IDMEF, Iptables, Ipset, PCAP, Netflow, Traffic Generation.

Abstract: The ongoing need to protect key nodes of network infrastructure has been a pressing issue since the outburst of modern Internet threats. This paper presents ideas on building a novel network-based intrusion prevention system combining the advantages of different types of latest intrusion detection systems. Special attention is also given to means of traffic data acquisition as well as security policy decision and enforcement possibilities. With regard to recent trends in PaaS and SaaS, common deployment specific for private and public cloud platforms is considered.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

The protection of local networks is a key goal for the current network administrators. The cyber war is in progress and it becomes still more and more difficult to keep up with the modern threats. We are focusing on network security, namely on new generation intrusion prevention systems (IPS) that combine two major approaches of intrusion detection.

To prevent an intrusion, it has to be identified first, therefore the problem solution comprises of two main parts – *detection* and *prevention*, together forming an IDPS[1]. This paper describes a prototype of such a system.

1.1 Motivation

There are two major branches of intrusion detection systems (IDS). One builds on signature based detection, the second on behavior analysis. Each has strengths and weaknesses and our challenge is to combine them into one IPS. (Kazienko & Dorosz, 2003) and (Talawat, 2008) show that:

- *Signature-based detection* works reliably, but cannot respond to unknown threats.
- *Behavior analyzers and anomaly detectors* can reveal previously unseen problems, but they do not work as fast as signatures and for known threats they may guess what has already been proven (signatures exist).

Bare deployment of detectors of both types would provide a fair chance to capture problematic behavior from both categories. (Gómez, et al., 2009) demonstrates it is not uncommon to combine known detection approaches to achieve better results.

However, our ambitions go beyond such simple solution by incorporating different detection techniques, consuming the knowledge they provide about the traffic, and attempting to deduce further information based on the combination of data evaluation. Such combination can yield additional knowledge about actual threats and means to react.

1.2 Objectives

Our aim is to create an intrusion prevention system composed of complementary detection modules. Such system should function in regular (enterprise) networks as well as private/public cloud environments.

Our solution is based on sets of rules to be enforced within the network environment.

[1] Intrusion detection and prevention system. The term and an analysis of the principles used are covered by (Scarfone & Mell, 2007).

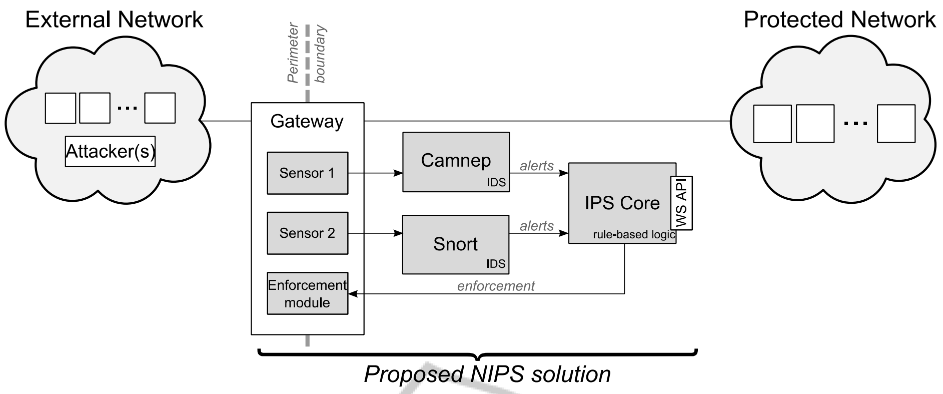

Figure 1: Protected segment schema.

An important objective was to research ways to use the derived enforcement actions, considering the limits of particular enforcement-point platforms.

We want to combine the detection capabilities of IDS and based on this, to introduce the prevention, e.g. to enforce network traffic shaping.

We strive to demonstrate that the additional logic added to the used detection modules will yield benefits compared to simple combined deployment of detectors unaware of each other.

2 RELATED WORK

We build on existing ideas from fields of both signature-based and behavior detection. The advantages and drawbacks of both these approaches are outlined in Section 1.1. For our prototype purposes, these two groups are represented by Snort and CAMNEP, respectively.

(Roesch, 1999) published an initial form of his lightweight NIDS Snort. Since then the design was improved heavily while the key concept was kept – network traffic is checked against a database of known signatures in order to identify a threat.

(Rehák, et al., 2008) introduced the project CAMNEP as a representative of behavior network intrusion detection system (A-NIDS) strongly relying on agent technology based on A-Globe platform (Šišlák, et al., 2005).

According to the taxonomy by (García-Teodoro, et al., 2009), CAMNEP can be described as statistical and machine-learning-based detector featuring fuzzy logic techniques, clustering and outlier detection. (Lim & Jones, 2008) would identify CAMNEP as 'learned-model-based' as it automatically creates its model of normal network behavior.

Even though CAMNEP is based on previously described detection techniques, due to the employment of agent-based collective trust aggregation or history integration it is able to function with significantly lower rate of false positives than common anomaly detectors, (Rehák, et al., 2008).

3 PROPOSED SOLUTION

An intrusion prevention system consists of three basic modules: a sensor providing the input data acquisition, a data processor evaluating the input and deciding about actions to be performed, and an executor responsible for the results enforcement.

For the data evaluation, we choose a combination of signature-based detection tool Snort (that can work as a simple IPS as well) and CAMNEP, NBA[2] anomaly detector developed by Cognitive Security / Cisco Systems. Conceptually, other similar tools can be used; specific conditions when and how this can be done are described in section 5.1.

The initial IPS prototype relied on network behavior analysis provided by CAMNEP based on the traffic coming from a NetFlow[3] sensor[4]. Following engagement of Snort's output allowed building a versatile network-based intrusion prevention system.

Figure 1 depicts the resulting schema: Traffic coming from the external network enters the protected network via a gateway where the passing data can be collected for an analysis within the NIPS. Various aspects that can affect traffic data gathering by sensors are introduced in section 3.2. A behavioral and a signature-based detection system are engaged to compensate weaknesses of each other as discussed in section 3.1. Section 3.4 explains how alerts

[2] network behavior analysis

[3] metadata format described by (B. Claise, 2004)

[4] also known as „probe“

exported from both IDS are processed by the IPS core (configurable via a web service). The resulting actions are enforced by a specialized module at the gateway as described in section 3.5. Additionally, also other ways of intrusion prevention can be employed (see section 3.3).

Event data from detectors are preferably obtained as XML files in standard Intrusion Detection Message Exchange Format (IDMEF)[5] supported by the CAMNEP IDS. Alternative security alert retrieval methods can be used – e.g. IDS Snort alerts are obtained via shared MySQL database access.

3.1 Symbiosis of Signature and Behavior Detection

The NBA-only IDS core is not sufficient on its own to provide reliable information for actions to control the network traffic. Many known attacks do not display any statistical anomalies and therefore cannot be revealed by a behavior detector. Moreover, even a very specific output of an NBA analyzer points always to only a suspicious behavior and without information context, significant risk of false positive detection and thus an encroachment might occur. False alarms are considered even more harmful than false negatives (especially when used for automated responses).[6]

However, the necessary context information related to the detected event can be delivered to the IPS by the other detector, the signature IDS core. The advantage of the signature approach is its promptness allowing real-time interventions, high specificity and full sensitivity[7]. These qualities are redeemed by the dependency on the signatures developer and frequent regular database updates. In comparison, the behavior analysis is not only computationally expensive but it also needs a time to collect traffic flow sets and to compare the observed data with a previously built statistical model (further referred to as “model”). CAMNEP is capable to provide output in 5min intervals at the best, which can become critical for the intrusion prevention.

A considerable property of the NBA is concealed in its strength – the ability of the model to learn over time can cause a creeping adaptation to an anomalous traffic that becomes a part of the norm.

[5] standardized by (Debar, et al., 2007) in RFC 4765

[6] this is the prevailing opinion supported not only by (Rehák, et al., 2008)

[7] This is true for all threats previously described in the signature database. Sensitivity is also known as *recall*.

In spite of all its drawbacks, the NBA proves to be irreplaceable in case of long-term enterprise network issues, important data leaks or suspicious events where no signature detectors can discover any problem. Employment of an NBA-based detector usually assumes a human-expert participation to investigate the conditions and circumstances. Such manual intervention could be replaced by a rule system based on the previous knowledge in order to provide the protective decision supporting information.

Symbiosis of both detection types (compared in Table 1) leads to an increase of overall capabilities of the newly developed IPS.

Table 1: Behavior and signature analysis comparison.

Property	*NBA*	*Signature-based*
determinism	anomaly detection	reliable attack detection
specificity	suspicious activity	precise threat identification
efficiency	time-expensive	real-time
autonomy	independent	updates necessary
adaptability	ability to learn new threats	only known signatures

3.2 Traffic Data Acquisition

All analyzers are directly dependent on the incoming data extracted from the network traffic using a sensor. The sensor must either be placed in the course of the data packets (an in-line sensor) or be sent a copy.

An example is a network tap or a spanning port[8] directing all data coming through the network device also towards the listening hardware or software sensors. However, this solution requires a special hardware that may not be available in the target network.

Another option for the information gathering is offered by the network-card promiscuous mode that enables “sniffing” of packets destined for other nodes in the local network. This way, both the inner traffic and the communication with external hosts can be evaluated. Nevertheless, many architectures do not allow promiscuous mode for security reasons.

[8] differences between these two are illustrated in (Network Instruments, 2013)

Alternative means of data acquisition can be provided by promiscuity emulation. The iptables[9] tool on the entry node of the observed network segment can be instructed to duplicate all the incoming traffic and forward it to a specified host via the non-terminating *TEE* target. This solution assumes administrative privileges in the gateway.

An important decision to be made about the position of the sensor is related to the network address translation (NAT). If NAT is performed on the entry point, placing the sensor in front of it can lead to a considerable loss of available data as the protected nodes are not individually observable and the internal communication is not visible at all. On the other hand, gathering data from inside of the protected segment causes missing the information about the traffic already blocked on the gateway. The previously mentioned iptables *TEE* target at the very beginning of the *forward* chain of the *mangle* table is the most appropriate solution (see Figure 2).

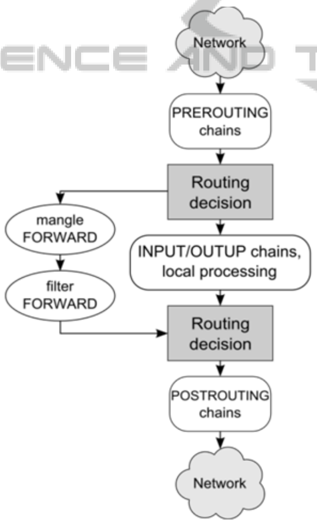

Figure 2: Simplified iptables traversal schema.

3.3 Dual Usage of Snort

Snort can be used either purely as an intrusion-detection product or also its prevention capabilities can be leveraged.

An active sensor, located directly in the flow of all data incoming and outgoing to/from the protected network, enables immediate control over the passing packets. The control actions include dropping, bandwidth limiting, and modification in form of normalization or sanitation. However, severe bottleneck issues are imminent using the in-line sensor at intensive traffic load. When the sensor is flooded with data and not able to accomplish their processing in real-time, high network latency or even loss of packets may occur or unchecked flows may pass through.

In our IPS prototype, a trade-off between such unintentional traffic limiting and a delayed intervention caused by the analyzer overhead and by internal processing was inevitable. We use an active sensor for the fast signature-based Snort in-line enforcement combined with passive collecting and/or thorough examination of data by both IDS modules (CAMNEP and Snort).

The optional protection feature of the Snort detector is available in several data acquisition modes (DAQ). The “NFQ”[10] was chosen as the most appropriate one for our purposes because it collaborates with the iptables allowing for user-space packets processing[11]. The active Snort sensor provides fast and reliable signature-based decision on blocking the malicious packet. In order to prevent unnecessary workload, this step can be prepended to the more complex evaluation by the passively listening analyzers controlled by the IPS-configuration rules. A way to achieve this setup is enqueuing the Snort’s NFQ into the *forward* chain of the *mangle* table while placing the enforcement of the IPS system in the *forward* chain of the *filter* table – see Figure 2.

3.4 Prevention Logic

The prototype was constructed as a rule system consisting of dynamic and static rules (see Figure 3). Dynamic rules sequentially process incoming events (alerts from IDS of various types, e.g. “port scan”, “DNS tunnel”, “attempted reconnaissance”) and respond with primary actions (traffic limiting and modification) and supplementary secondary actions (logging or user notification). Static rules setting is provided by whitelist and blacklist definitions and a default policy (packets “*deny*” or “*allow*”).

The 3-dimensional space of network connection endpoints (see Figure 4) is described by “tokens”, the substituting symbols for a range of IP addresses, set of L3/L4 protocols and a range of port numbers in case the protocol supports ports. As an example,

[9] a user-space application for Linux allowing kernel firewall management

[10] an inline module for Linux leveraging the QUEUE target in netfilter to move packets from the kernel-space to user-space for evaluation – see (Snort Team, 2013)

[11] (Leblond, 2013) refers to this act as “issuing a verdict”.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

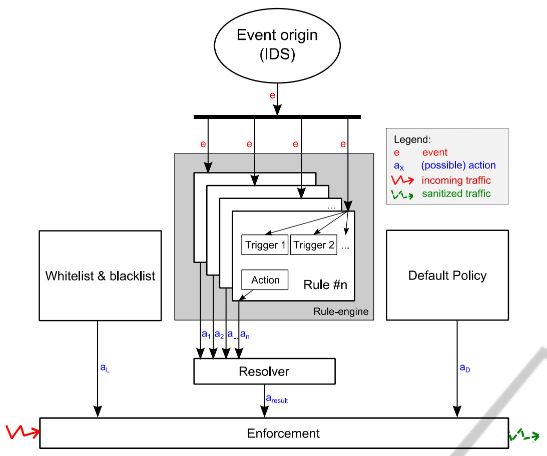

Figure 3: Event processing into actions.

the depicted token *t1* defines a set of hosts within the IP range *ip1–ip2* communicating via the UDP protocol on port *p1*; the token *t2* defines a node with the *ip3* address at the TCP protocol on the port *p2*. Such tokens are used to describe the hosts or host groups identified as carrying on the suspicious communication reported in an IDS alert. Another application of tokens is to determine the enforcement-action target as explained later.

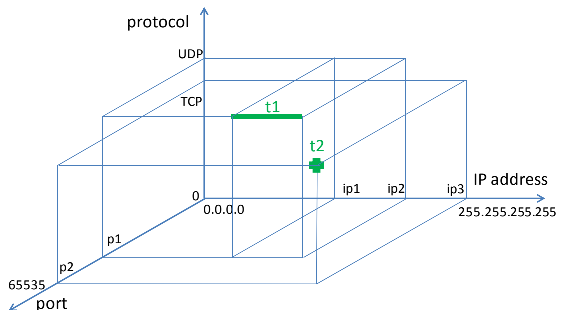

Figure 4: 3D network space described by tokens.

The “rule-engine” uses a set of dynamic rules. A rule includes one or more triggers and a single enforcement action. The enforcement action, such as packet dropping, rejecting, or limiting, is applied to a set of tokens provided by the rule after its activation. So e.g., a “drop” action applied to the *t2* token from Figure 4 directs the gateway to drop all packets from or to a host with the *ip3* address sent via the TCP protocol on the host’s port *p2*.

A “trigger” can be thought of as a definition of the rule’s area of interest and if activated, it provides output tokens that are eventually used for the rule’s enforcement action as documented in Listing 1.

```
process_alert(alert_type, alert_tokens)
  for-each rule
    for-each trigger
      if trigger_regular
        if alert_type ∉ trigger_types
          trigger_ignore_alert // continue
        trigger_output
            := alert_tokens ∩ trigger_tokens
        if trigger_output empty
          trigger_ignore_alert // continue
        activate_trigger
      if trigger_rule
        if not dependee_rule_acivated
          trigger_ignore_alert // continue
        trigger_output
            := get_tokens_from_dependee_rule
        activate_trigger
    if not all_triggers_activated
      rule_ignore_alert // continue
    enforcement_action_input
          := ∩all_triggers trigger_output
    submit_enforcement_action
```

Listing 1: General incoming alert processing.

A regular trigger consists of a set of alert types and a set of tokens that the trigger should react to if a corresponding alert occurs. Such trigger is activated by an incoming alert if the alert type matches a type defined within the trigger and if the intersection of both the alert’s and the trigger’s tokens is not empty; this intersection constitutes the activated trigger’s output.

A special kind of trigger, a “ruletrigger”, represents a link to another rule, so (de)activation of the referenced rule implies (de)activation of such ruletrigger. The presence of ruletriggers allows creating trees of dependent rules.

In order for the rule enforcement action to be applied, all triggers within the rule must be activated. In case of activation of a trivial rule (with only one trigger), the enforcement action is applied on the network space defined by the trigger’s output. In case of a more complex rule (with multiple triggers), the logic works alike – the individual triggers’ outputs are further intersected with each other and only the result is applied via enforcement.

A rule with only some triggers activated is in a partially excited state, which is remembered for a defined time interval. If also the rest of triggers get activated within this period then the whole rule is activated; otherwise the activation of triggers expires.

As a demonstration of this mechanism, consider a rule with two triggers defined by the “port scan” and “reconnaissance” alert type, respectively, and a common IP range 198.51.100.0/24. The former trigger is activated by an incoming “port scan” alert identified in the traffic from the 198.51.100.101 and 198.51.100.102 addresses. A following “reconnaissance” alert warning against the 198.51.100.102 and 198.51.100.103 hosts activates the latter trigger.

This causes activation of the whole rule. The activated rule uses the token intersection of both the triggers' outputs, i.e. the 198.51.100.102 IP address, to enforce the "reject" action on the gateway of the protected network.

For cases when the rule's action is being already enforced, the system configuration allows an extension of the rule impact. In this way, other incoming relevant alerts can cause (a) an extension of the set of tokens, subject to the enforcement action, by a newly detected attacker address, (b) action amplification or action duration extension (e.g. packets being rejected by now become being dropped), or (c) a combination of both.

3.5 Enforcement

Network nodes protection can be reached by different means depending on the point of intervention in the data flow or the level of ISO/OSI model[12] to be affected. Network-based protection constitutes the only available solution for securing hosts of an undefined platform, which the protected nodes do not need to know of at all.

Access to L3/L4 (our target level) can be configured to a great extent, thus it is possible by various administration tools. Probably the most well-known of these tools is iptables which has also been employed within this IPS.

Network protection settings must be modified atomically in order to avoid exposing the internal network to the outside world while not missing any packets. Iptables itself is capable of such behavior, but a reasonable and common practice is using the ipset[13] tool in cooperation with iptables to simplify IP addresses and/or ports definitions. For such combination, a specific approach has to be applied to preserve the atomicity of the key operation. This approach is shown in Figure 5:

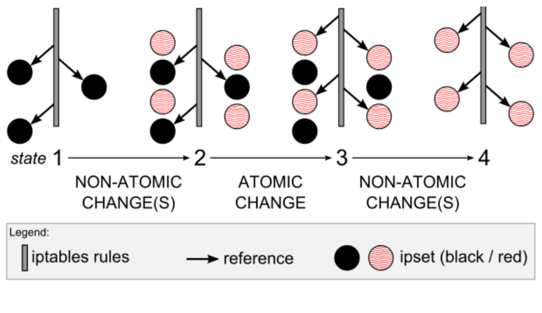

Figure 5: Atomic iptables and ipsets workaround.

The newly needed ipsets (thought of as "red" in an alternating red-black sequence) have to be created first (non-atomically one by one as they are not used yet – state transition 1-2). Then the iptables chains referencing the old ipsets (called "black") have to be flushed and simultaneously the new chains with references to the new red ipsets imported (which happens atomically via the "iptables-restore" command; 2-3). The old black ipsets can be non-atomically deleted after that as they are no longer referenced and used (3-4). In the next enforcement turn, new black sets can be created and applied analogically.

4 EXPERIMENTS

The prototype behavior has been evaluated in several environments emulating different network setups from cloud to a local deployment in a small to middle-sized enterprise network. Experiments were performed in a dedicated network segment representing a set of running services in cloud or in a corporate network (possibly a demilitarized zone). Experimental intrusion attempts were conducted from the outside of NAT to simulate an external attacker.

All traffic for this model was generated artificially. The traffic consisted of generated attacks and artificial background. Legitimate UDP communication between individual hosts was simulated with the RUDE and CRUDE[14] tools. A source for massive traffic to camouflage malicious actions was the Linux netcat utility. Netcat CLI was used to emulate a web server, to run as a port scanner or to be used as an HTTP client on both UDP and TCP protocols.[15]

The IDPS-testing framework Pytbull[16] provided prepared experiments covering a range of parameterized attacks (its modules comprise fragmented packets, evasion techniques, shell codes, denial of service and other types of exploits). We also used the Python API to extend the tests and adjust them to our needs.

Hydra[17] was used for multi-threaded cracking of various services authentication. A DoS attack effect was reached by flooding the target using LOIC[18], the

[12] defined in (International Organization for Standardization, 1996)

[13] user-space tool for organizing (mostly) IP addresses for efficient lookup, integrated with iptables

[14] http://rude.sourceforge.net, "Collector for RUDE", resp.

[15] http://www.howtoforge.com/useful-uses-of-netcat

[16] http://pytbull.sourceforge.net

[17] http://thc.org/thc-hydra

[18] http://sourceforge.net/projects/loic

ping tool, hping[19] or by target resource exhaustion with Slowloris[20]). Network reconnaissance attempts including horizontal and vertical scanning were accomplished by the Nmap security scanner[21].

The tcpdump[22] and tcpreplay[23] provided a way of capturing and emitting network traffic in the form of PCAP files. Contagio Malware Dump[24] served as a public source of current intrusion traffic recordings for our experiments.

Finally, two penetration-testing Linux distributions, BackTrack and Kali, were engaged for advanced simulation of threats.

A different setup was designed for real-world traffic observations – e.g. a network tap placed on our company's outer gateway to the Internet.

4.1 Testing Results

Using Pytbull test-sets, that are close to real attack schemas, we were able to induce all possible combinations of our analyzers detection, i.e. situations where only signatures indicate a threat, where only behavior analysis signals an alert, and where both or none of them succeed. This result demonstrates the added value of the detection methods combination.

The Hydra experiments confirmed that a massive clutter with legitimate connection requests on a service could not be caught by signature detector while its quantitative anomality and time distribution allow safe identification of an attack. The DoS and Nmap related tests confirmed shortcomings of a signature-based approach as well.

Data obtained from the Contagio Malware Dump database split up into an undetected part, almost 52 % detected by Snort, and roughly 9 % which both types of IDS warned about. Thus for the malware communication typically displaying statistical insignificance of traffic features, the necessity of the signature-based detection was demonstrated.

5 CONCLUSION

This paper presents ideas and technological methods for building a network-based intrusion prevention system with an ample potential.

[19] http://hping.org

[20] http://ha.ckers.org/slowloris

[21] http://nmap.org

[22] http://www.tcpdump.org

[23] http://tcpreplay.synfin.net

[24] Security database; http://contagiodump.blogspot.com

Approaches to employ multiple detection engines have been outlined and prototyped, with emphasis on signature-based and anomaly-based (NBA) detection composition, specifically using Snort and CAMNEP, respectively. Projected benefits of such arrangement have been explained in the manner where the resulting system should be able to protect key nodes or subnets against threats that neither of the detection subsystems would reliably react to on their own.

Configuration logic of the prototype provides means for modeling of arbitrary dependency trees including alerts combined from the different detection subsystems, their evaluation and enforcement actions for entry-point firewall application.

At the moment, enforcement can be performed on Linux gateways via iptables and ipset which safe ways of operating have been found for.

5.1 Arbitrary Detection Integration

During the design and implementation phases of the project, we have always had detection extensibility in mind. We chose IDMEF (as discussed in section 3) to be the cornerstone exchange format as it is designed exactly for this purpose – to exchange intrusion detection information – and serves this function better than its counterparts (e.g. in comparison with CEF[25]).

New detection modules can be used to enhance capabilities of our prototype and depending on their output details (whether they implement IDMEF and how), changes to the prototype, needed to be done in order to integrate them, may even be little to none.

Employment of an IDS using different export structure and/or format should be possible as well, but the complexity of such integration process can vary dramatically. A successful proof-of-concept of such approach is the Snort integration we have conducted as described in section 3. In that case, the prototype is able to read intrusion event data from a shared database.

When the detection composition schema is changed (because of a new detection unit), the IPS configuration must be brought up-to-date. Configuration of the complete system needs to respect new types of events the joint IDSs produce. The XML-based rules configuration is extendable, hence this is possible. However, the logic of the events should be revised and tested in order for the combination of

[25] Common Event Format (ArcSight, Inc., 2009)

chosen detection subsystems to interact sensibly and offer practical benefits.

5.2 Future Work

There are matters that need further work and research. The prototype is at the point of being further tested. Experiments outlined in section 4 yielded promising results regarding the combined detection capabilities of the prototype. Real-world employment, result evaluation and tuning are a next logical step along our efforts. We have to prove to what extent the solution resists against new, still unknown attacks.

Current rulesets used for our testing deployments need improving, as they may be too naive for real-world malware and threats. Based on detection and environment knowledge, extended sets of rules must be developed for particular cases tailored to given production environments to fully utilize the projected advantages of these and other detection combinations. This is also a key requirement for demonstrating potential capabilities of the prototype. General rulesets may also be created, but the full extent of the beneficial impact is yet to be determined.

Areas regarding deployment and enforcement possibilities have been considered and a noticeable amount of ground research has been done, however there is also still much to work on:

Cloud platforms are a prime candidate for our solution. However, inquiries done in this field show that along with them striving to deliver all-around secure environment, they also severely limit the execution possibilities for employment of different intrusion detection methods compared to classic LAN cases (promiscuous mode, iptables kernel support, custom routing and network management as analyzed in section 3.2).

As for enforcement, our current implementation leverages iptables and ipset for Linux. There are other similar tools available on different platforms (access-lists on Cisco network devices, PF[26] for BSD, and RRAS[27] in Windows Server), their engagement is potentially possible (we have briefly experimented with a few in order to confirm this), but whether such usage would be entirely practical is yet to be confirmed.

Considering the near-future course of our IPS prototype development, some of the main subjects of the work to be done are large data measurements in both artificial and real enterprise networks with actual imminent threats present.

[26] stateful packet filter comparable to iptables

[27] Routing and Remote Access Service – http://technet.microsoft.com/en-us/library/dd469714.aspx

ACKNOWLEDGEMENTS

This research has been supported by Czech Ministry of the Interior within the project named “Applied research of a method of an efficient management of network security of data centers, servers and cloud services verified by practical measurements”, grant number VG20122014086, PID VG2VS/242.

REFERENCES

ArcSight, Inc., 2009. *Common Event Format.* [Online] Available at: http://mita-tac.wikispaces.com/file/view/CEF+White+Paper+071709.pdf [Accessed December 2013].

B. Claise, E., 2004. *Cisco Systems NetFlow Services Export Version 9.* [Online] Available at: http://tools.ietf.org/html/rfc3954

Debar, H., Curry, D. & Feinstein, B., 2007. *The Intrusion Detection Message Exchange Format.* [Online] Available at: http://tools.ietf.org/html/rfc4765

Fielding, R. T., 2000. Representational State Transfer. In: *Architectural Styles and the Design of Network-based Software Architectures.* Irvine: University of California, pp. 76-97.

García-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G. & Vázquez, E., 2009. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security,* 28(1-2), pp. 18-28.

Gómez, J. et al., 2009. Design of a snort-based hybrid intrusion detection system. In: *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living.* Berlin: Springer, pp. 515-522.

International Organization for Standardization, 1996. *ISO/IEC standard 7498-1:1994.* [Online].

Kazienko, P. & Dorosz, P., 2003. *Intrusion Detection Systems (IDS) Part I.* [Online] Available at: http://www.systemcomputing.org/ssm10/intrusion_detection_systems_architecture.htm [Accessed February 2014].

Leblond, E., 2013. *Using NFQUEUE and libnetfilter_queue.* [Online] Available at: https://home.regit.org/netfilter-en/using-nfqueue-and-libnetfilter_queue/ [Accessed November 2013].

Lim, S. Y. & Jones, A., 2008. *Network anomaly detection system: The state of art of network behaviour analysis..* s.l., s.n., pp. 459-465.

Network Instruments, 2013. *TAP vs SPAN.* [Online] Available at:

http://www.networkinstruments.com/includes/popups/taps/tap-vs-span.php [Accessed February 2014].

Rehák, M. et al., 2008. CAMNEP: An intrusion detection system for high-speed networks. *In Progress in Informatics, number 5*, pp. 65-74.

Roesch, M., 1999. *Snort: Lightweight Intrusion Detection for Networks.* Seattle, WA, s.n., pp. 229-238.

Scarfone, K. & Mell, P., 2007. Guide to intrusion detection and prevention systems (idps). *NIST special publication,* Volume 800, p. 94.

Snort Team, 2013. *SNORT Users Manual 2.9.6.* [Online] Available at: http://s3.amazonaws.com/snort-org/www/assets/166/snort_manual.pdf

Šišlák, D. et al., 2005. A-globe: Agent development platform with inaccessibility and mobility support. *Software Agent-Based Applications, Platforms and Development Kits,* pp. 21-46.

Talawat, T., 2008. *Five major types of IDS.* [Online] Available at: http://advanced-network-security.blogspot.com/2008/04/three-major-types-of-ids.html [Accessed November 2013].

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS