Evaluation of Intrusion Detection Systems in IPv6 Networks

Max Schrötter[1], Thomas Scheffler[2] and Bettina Schnor[1]
[1]*Operating Systems and Distributed Systems, University of Potsdam, Potsdam, Germany*
[2]*School of Engineering - Energy and Information, Hochschule für Technik und Wirtschaft Berlin, Berlin, Germany*

Keywords: IDS, IDSv6, Benchmark, IPv6.

Abstract: This paper introduces a benchmark suite for the evaluation of intrusion detection systems in IPv6 environments. We use this benchmark to evaluate the prominent intrusion detection systems Snort, Zeek and Suricata. Further, an IPv6 Plugin Suite is presented and evaluated which enhances Snort by stateful attack detection. The results of our evaluation demonstrate the current abilities to detect IPv6 link-local attacks.

1 INTRODUCTION

Relatively early after the finalization of the IPv6 standard, it became apparent, that the automatic host configuration mechanism defined in the protocol did not work well with the proposed IPsec security mechanisms and that the Neighbor Discovery Protocol (NDP) (Narten et al., 2007) suffered from similar security problems as the Address Resolution Protocol (ARP) in IPv4 (Arkko et al., 2002).

The root cause of these issues lies in the early authentication problem (Nikander, 2002), which is common to all Layer 2 protocols without strong node authentication (most importantly Ethernet). Therefore, most mitigation techniques such as port-based Network Access Control (IEEE 802.1X), IP Security (IPSec) and SEcure Neighbor Discovery (SEND) propose cryptographic host authentication. However, this either requires some form of pre-configuration of credentials and keying material or resource intensive computations, which is not a practical solution within most networks. Another approach advocates the filtering of IPv6 messages on Layer 2 devices as specified by the IPv6 Router Advertisement Guard (Levy-Abegnoli et al., 2011) which requires hardware support in the switches as well as configuration effort.

The network research community is aware of the security issues of IPv6 and the Neighbor Discovery Protocol (NDP) (Hogg and Vyncke, 2009; Nikander et al., 2004) and the IETF has re-worked some of the standards in recent years to limit the attack surface of the protocols (Gont, 2013). However, countermeasures not always exist, since some of these problems are protocol inherent.

This leads to a third approach, that focuses on network monitoring and anomaly detection. Work started with (Beck et al., 2007), which introduced the host-based tool `ndpmon` that listens to all NDP messages in order to detect NDP anomalies and report them. Another suitable approach is based on the extension of Intrusion Detection Systems (IDS) with adequate protocol support for the detection of specific IPv6-attacks. The authors of (Schütte et al., 2012) developed an IPv6 Plugin Suite for the well known IDS Snort 2. This approach benefits from the already implemented packet capturing, decoding, configuration, logging mechanisms, and log analysis tools of the IDS framework. A new preprocessor module and several new IPv6-specific rule options make the definition of IPv6-specific attack signatures possible. In this paper, we present our findings on implementing a similar Plugin Suite that supports the new version of the IDS Snort 3.0 (see Section 4.1) and comparing its detection results with other popular intrusion detection systems (see Section 6). For the comparison, we propose an *IDSv6* benchmark suite consisting of IPv6 specific network attacks (see Section 3).

2 RELATED WORK

In 2015, Jon Mark Allen published a report which examined the IPv6 support by the intrusion detection systems Snort, Suricata, and Bro (Zeek) (Allen, 2015). One of his observation was the unsatisfying tool support at that time - such as not displaying the corresponding IPv6 address in case of Snort when viewing an alert from an IPv6 stack. Different to our

Schrötter, M., Scheffler, T. and Schnor, B.
Evaluation of Intrusion Detection Systems in IPv6 Networks.
DOI: 10.5220/0007840104080416
In *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications (ICETE 2019)*, pages 408-416
ISBN: 978-989-758-378-0

approach, Allen investigated web application attacks in IPv6 traffic, but no IPv6 specific attacks. All three IDS systems were tested with the same network captures of a web application vulnerability scan: one over IPv4, and one over IPv6. While the three IDS systems behaved different in their number of alerts, each IDS reported a comparable amount of alerts for the IPv4 and IPv6 traffic as one would expect.

Salih et al. present a tool which detects and classifies covered channels in IPv6 networks (Salih et al., 2015). Sources of covert channels are header fields which are not set properly. Salih et al. identify the following header fields as potential covert channels: Traffic Class, Flow Label, Payload Length, Next Header, Hop Limit, and Source Address. They propose a Machine Learning approach to detect covert channel implementing an enhanced feature selection algorithm supporting Naive Bayesian classifier. The results of the conducted experiments show a high detection performance of about 96 %. While this is a promising result, it is questionable whether these attacks should be detected by an intrusion detection system, since the misuse of IPv6 header fields may already be prohibited by a packet filter.

In order to prove that IPv6 attacks are real and exploitable, Marc Heuse implemented different link-local attacks in "The Hacker's Choice" (THC) toolkit (Heuse, nd). The toolkit implements several denial-of-service and fragmentation attacks as well as a covert channel attack where a destination option header is misused. The toolkit is easy to use also for non-security experts, constantly updated and runs on Linux. We use the THC toolkit as the base of our IDSv6 benchmark (see Section 3).

Fernando Gont developed the SI6 Networks' IPv6 toolkit[1]. Like the THC toolkit, the SI6 implements link-local attacks special to IPv6, but it is highly configurable and may be used for protocol analysis as well as attack purposes. The tools need to be parameterized correctly in order to be effective and usually require more protocol knowledge and configuration overhead than the THC toolkit. Contrary to the THC toolkit the SI6 implementation runs also on BSD-type UNIXes such as FreeBSD and MacOS.

3 THE IDSV6 BENCHMARK

One question that is going to be asked by researchers and practitioners alike is: "How effective is a particular IDS in detecting network attacks"? It is possible to use the THC toolkit and SI6 directly to measure the effectiveness of attack detection and defense mechanisms but this method has several drawbacks. Software versions change also for attack tools and may implement subtile changes that can lead to varying detection results. Necessary network setups depend on additional devices and their configuration, are time consuming to create and may be difficult to replicate exactly. Attributing changes in detection results to specific root causes becomes difficult, even if the same tools are used for different measurements.

The problem of comparatively benchmarking different Intrusion Detection Systems is not new. Having a common benchmark dataset eases comparison between different systems and makes trial runs easily reproducible. Between 1998 and 2000, Lincoln Laboratory of MIT released several datasets for the evaluation of IDS, the so called DARPA datasets[2]. The datasets were aimed to represent real-world traffic mixes interspaced with attack traffic. Right after release, these datasets drew immediate criticism regarding the modelling and generation of the attack and background traffic (McHugh, 2000), however, they are still used to the present day, simply because no other common benchmarks exist.

When we started developing the IPv6 Plugin Suite for Snort, we were looking for a quick way to verify our development efforts. In order to automatize attack detection testing and to minimize administrative overhead, we created the *IDSv6* benchmark suite, that now consists of several `pcap` files derived from real attack tools created in the network setup depicted by Figure 1. This collection of `pcap` files can be applied quickly and consistently in order to test detection rates and functionality of intrusion detection systems. The IDS present in the figure plays no active part during capturing of the attack traffic. It simply is a placeholder to indicate from what position within the network traffic patterns will be observed if the benchmark `pcap` files are used. IDS usually support direct input from capture files. If this is not possible, tools like `tcpreplay`[3] could be used to reproduce previously captured network events.

Table 1 lists all the different attacks that are included in the suite. With the exception of `dos_mld_chiron`, all attacks were created by the THC toolkit in Version v3.5-dev. We choose the THC toolkit because of its maturity and ease of use, which makes it likely to be used during real network attacks. The Chiron tool can be found on Github[4]. We used Version 1.0 of this tool.

The *IDSv6* benchmark is focused on attacks that

[1] https://www.si6networks.com/tools/ipv6toolkit/

[2] https://www.ll.mit.edu/r-d/datasets

[3] https://tcpreplay.appneta.com

[4] https://github.com/aatlasis/Chiron

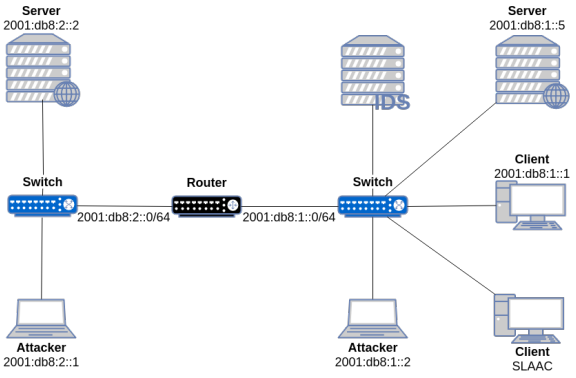

Figure 1: Testbed for the recording of the *IDSv6* benchmark `pcap` files.

are new to IPv6 and / or that exist because of missing early authentication. For example, it includes recordings of an attack created by the THC `parasite6` tool, which implements the IPv6 version of an ARP cache poisoning.

Different attacks affect different parts of the network. While most attacks are local to a specific subnet, some attacks were carried out across a network border. The `parasite6` attack, for example, affected only components of the `2001:db8:1::0/64` subnet. It manipulated the communication between Client (`:1`) and Server (`:5`). The Attacker (`:2`) send spoofed *Neighbour Discovery Advertisements* to the Client and thereby redirected traffic to himself or to a non-existing target on the network.

When we tried to upset the flow of Multicast traffic in the network, we found that the THC tool `dos_mld` was not able to disrupt multicast traffic in our testbed. We therefore chose to use the Chiron tool to generate a successful attack, that is called `dos_mld_chiron` in the benchmark. The attack vector of the tool is described in the talk MLD considered harmful (Rey et al., 2016). A multicast source in the `2001:db8:2::0/64` subnet sends continuous multicast messages to a client in subnet `2001:db8:1::0/64`. The attacker is placed in the same subnet as the client. The attacker stopped the flow of multicast messages by first becoming the local *MLD Querier* and sending a spoofed *MLD Report* removing listeners for this MLD group. Afterwards it sends a *Last Listener Query* and periodic *Generic Queries* to the unicast address of the router.

The `frag` attacks, which create variously fragmented packets, and the `denial6` attacks, which creates packets filled with too many extension headers and options, are conducted with the attacker placed in the `2001:db8:2::0/64` subnet and the client in subnet `2001:db8:1::0/64`.

The Benchmark inluding all `pcap` files is publicly available[5].

4 INTRUSION DETECTION SYSTEMS

This section gives a short overview of the evaluated intrusion detection systems. All three systems are open source, free to use and have done their homework and support the necessary decoding of IPv6 packets for the upper layer protocol analysis and attack detection. In section 6, we will further evaluate whether these systems can also detect IPv6-specific attacks.

4.1 Snort

Snort (Roesch, 1999) is a powerful and widely used IDS that provides basic IPv6 support. In prior work, it was recognized that Snort in version 2 lacks the ability to detect many attacks on IPv6-specific protocol mechanisms (Schütte et al., 2012) and had to be extended to handle those attacks.

Snort 3, also called Snort++, is the next generation of Snort. It fixes many of the performance and configuration shortcomings of Snort 2. Snort++ also gains a lot more flexibility by modularizing Snort preprocessors, and thereby gaining service specific inspectors. Despite the new codebase, we found, that it still lacks many of the features provided by the IPv6 Plugin Suite that had been developed for Snort 2 (Schütte et al., 2012).

We therefore made the decision to implement a corresponding IPv6 Plugin Suite for Snort++. This new Plugin Suite is inspired by the prior version and expands its functionality based on our *IDSv6* benchmark (Section 5 discusses the new Plugin Suite in more detail).

4.2 Zeek/Bro

Zeek, previously known as Bro[6], has been originally developed by Vern Paxson (Paxson, 1999) at UCB. The Zeek IDS emphasizes a clean separation between the decoding stage, implemented in C, and the analysis and alarm generation which is implemented in a domain-specific language called `bro script`.

Basic IPv6 support is enabled by default since Bro 2.1. Further, support for IPv6 fragmentation reassembly, tunnel decapsulation and IPv6 header chain analysis was added in subsequent versions.

[5]https://redmine.cs.uni-potsdam.de/projects/pcap
[6]https://www.zeek.org

Table 1: List of attacks in the *IDSv6* benchmark.

Attack	Description
covert_send6	Sends data covertly by putting it into a Destination Options Header.
denial 1-6	Various attacks to conduct a DoS attack against a host. It increases the decoding effort of the host by including a large amount of unknown options or extension header.
dos_mld_chiron	An attack that cancels an arbitrary MLD subscription by becoming the MLD querier and faking the multicast specific queries.
dos_new_ipv6	An attack that disrupts SLAAC by always replying with advertises and therefore prevents new hosts from obtaining an IP address.
fake_mldrouter6_advertise	Advertising a wrong MLD router via MRD to interfere with filtering of MLD snooping switches.
fake_mldrouter6_terminate	Terminates a MLD router via MRD to interfere with filtering of MLD snooping switches.
flood_*	Attacks that flood the network with various NDP and MLD messages.
frag 1-36	Various fragmentation attacks that send malformed fragments to circumvent firewalls or attack the victim's network implementation.
kill_router6	Removes the given router as gateway from all SLAAC configured systems in the network.
ndpexhaust26	An flooding attack that floods the network with ICMPv6 toobig messages.
parasite6_real	An neighbour cache poisoning attack to divert traffic to the attacker.
parasite6_fake	An neighbour cache poisoning attack to divert traffic to a not used MAC address.
redir6	A man in the middle attack that diverts the traffic with ICMPv6 redirect messages.
rsmurf6	An attack that sends a ping from a multicast address to the victim.
smurf6	An attack that sends a ping with the victims IP address to a multicast address.
toobig	An attack causing atomic fragments and therefore a DOS attack as described in RFC8021

4.3 Suricata

Suricata is the youngest of the three IDS projects developed by the Open Information Security Foundation[7], a non-profit organization founded in 2009. Its goal is to develop an IDS that is backward compatible with existing Snort rule sets, but not limited by Snort's development and architecture history. Suricata's architecture is targeted at parallel execution to benefit from modern multicore processor architectures.

Suricata promises IPv6 support only for packet decoding (IPv6, ICMPv6).

5 IPV6 PLUGIN SUITE

In this section we present the functionality of the IPv6 Plugin Suite for Snort 3. Due to the new architecture and code base of Snort 3, no existing code could be reused and the Plugin Suite needed to be developed from the ground up. As mentioned in Section 4.1 the Snort++ preprocessor stage is now split up in many Inspector sub stages. Also, all plugins are embedded in their own modules and *Inspector* and *Rule Options* are separate types of plugins. Therefore, we did not had to create one, but several plugins. The Plugin Suite is open source and can be found online[8].

5.1 NDP Inspector

In contrast to Snort 2, which ran all preprocessors on all captured packets, now only inspectors specific to that packet are executed. The types of packets for which an inspector can be written are predefined by Snort 3. There is an IP packet type but no sub-type for the different versions of the Internet Protocol. Therefore, our inspector is called on all IP packets.

The plugin needs some pre-configuration to be effective. For example it needs to know the address of the current IPv6 subnet, for which it is going to examine the correct execution of the *Neighbor Discovery Protocol*. The plugin then examines the source and destination addresses of all incoming IPv6 packets. If these addresses match the configured subnet, it tries to learn the corresponding IPv6 address together with its Layer 2 address and create a host-list for the subnet.

One problem, that any detection tool must avoid, is the state-explosion that is possible in IPv6 networks. Due to the large size of IPv6 subnets (2^{64} pos-

[7]https://oisf.net

[8]https://redmine.cs.uni-potsdam.de/projects/snort3-ipv6-plugin

sible hosts) it is comparatively easy to overwhelm detection tools by sending packets with randomly generated IPv6 *Host-IDs*.

Our original idea was to monitor the *Duplicate Address Detection* (DAD) process for each host and learn only hosts, for which DAD had been successfully completed. However, this strategy would require that each host on the subnet needs to perform DAD *after* our plugin starts and would exclude hosts where DAD was administratively disabled or broken. We therefore implemented a different strategy. Before a host address is learned and marked as active, the host must send at least two and receive one packet that can be observed by the plugin.

Once the plugin has a complete overview of the subnet, it becomes possible to detect attacks that exploit the authentication gap between Layer 2 and Layer 3.

One example for such an attack can be observed by looking at the mechanism of the `dos_new_ipv6` tool. The tool interferes with the stateless autoconfiguration process for new IPv6 addresses, by answering each DAD *Neighbor Solicitation* request with the message that the address is already present in the network. This attack effectively hinders an IPv6 host from correctly configuring an IPv6 address and entering the network. The plugin can detect this attack by checking protocol messages against the recorded network structure. If the DAD *Neighbor Solicitation* request is answered with a spoofed *Neighbor Advertisement* the plugin triggers an alert.

Each host is identified by MAC and IP-Address in the data structure and the plugin is able to distinguish different virtual machines from the same virtualisation host. The data structure used to manage active hosts is a combination of a hash map and a double linked list and has a user configurable size. The hash map allows for quick access to entries by a key and the linked list allows for a fast *least recently used* (LRU) replacement strategy, should the number of entries outgrow its configured size. A fast replacement of entries is needed to avoid that our plugin itself becomes the target of a DoS flooding attack.

The plugin also records the amount and type of NDP and MLD messages present in the network in order to detect flooding attacks. An alert can be triggered by the amount of packages seen within a time window. Time and traffic parameters for flooding detection are configurable and their exact values depend on the size and kind of network under observation.

There are also some attacks such as `kill_router6` that can be detected without having to analyze the underlying network structure. Some of the necessary checks, that needed to be implemented for Snort 2 within a plugin, are now already covert by Snort 3 decoder inspectors. This includes the check for the right order and amount of IPv6 extension headers. The mentioned `kill_router6` attack, however, is not yet covered and needs to be implemented within the Plugin Suite. It detects *Router Advertisement* messages that are send with a lifetime of zero and can be misused to signal to hosts that a particular router is going down and it should be removed from routing tables.

In total, our new NDP inspector supports 17 alerts described in Table 2. Not all of them are used to detect attacks, some also exist for debugging reasons (5,8,9).

The plugin does not create alerts directly but generates so called detection events. These detection events must be configured to trigger alerts via Snort rules. We suggest to use the priorities shown in Table 2. The priority is an integer value stating the severity level of an event. It starts with the value 1, which represents the highest priority. Priorities can also be set by assigning a *classtype* to a rule. The *classtype* priorities vary from 1 to 4, where 4 is the lowest priority.

The following example shows the use of such an alert within a newly defined Snort rule:

```
alert (msg:"Possible Neighbour Cache
    Poisoning";
   sid: 15;
   gid:1001;
   rev:1;
   classtype:mitm;)
```

And the following *class definition*:

```
{ name = 'mitm', priority = 1,
  text = 'man in the middle attack' }
```

Snort identifies events uniquely by assigning several ID values, such as *Generator ID* (GID), *Signature ID* (SID), and *Revision ID* (REV). The GID indicates which Snort module generated the event. Normal detection rules are evaluated by the *text rule inspector*, which uses the default value of 1 (`gid:1`). Our IPv6 Plugin Suite uses the GID value 1001 (`gid:1001`).

Each module itself may signal many different events and each of these events needs to be identified clearly. This is done using SID values. SID values may be re-used between modules and may sometimes have internal guidelines regarding the value range that may be applicable for user-definable IDs. The assigned SID values for our IPv6 Inspector Plugin are given in Table 2 and currently range from 1 to 17.

REV values are mainly relevant for user-defined text rules, which may be rewritten over time and here the REV value will be used to indicate that a rule had been altered from a previous state.

Table 2: Snort 3 IPv6 Inspector Plugin Alerts.

SID	PRIO	MESSAGE
1	1	Spoofed Redirect Message
2	2	Neighbor Discovery flood detected
3	1	Router Kill detected
4	1	Router Advertisement from non-Router
5	4	New Router Alert
6	2	Change Router Flag Alert
7	2	Change Router Prefix Alert
8	4	New DAD
9	4	New Host Alert
10	3	DAD Conflict Alert
11	1	Spoofed DAD Conflict Alert
12	3	Unsolicited Advertise Alert
13	2	MLD flood detected
15	1	Possible Neighbour Cache Poisoning
16	1	MLD Query from non-Router MAC
17	1	MLD Router Advertisement from non-Router

5.2 IPv6 Rule Options

The other function provided by the IPv6 Plugin Suite is the extended support for multiple options that expand the Snort rule language. These options enable the user to write rules that can be IPv6 aware and specific. Snort rules typically target Layer 4 traffic and are agnostic to different versions of the Internet Protocol. While this is the desired behaviour to detect the majority of network attacks, it hinders the detection of attacks that target specific Network Layer functions.

A Snort rule is based on the following scheme:

```
action proto source dir dest ( body )
```

A rule must contain an *action* that the rule should invoke and a *protocol* for which this rule is created. The *source*, *direction* and *destination* are optional. The body contains a *Signature ID*, *Generator ID* and *Revision ID* to identify the rule. It also contains a message that will be logged on match. There are more options in the body that further specify the rule, but most of the available options are signature options to differentiate detected packets.

The following example shows a rule, that uses the Snort rule options *flow* and *http_uri*:

```
alert http (msg:"Access to dubious URI!";
    flow:established, to_server;
    http_uri:"attack";
    sid:1000001;)
```

The rule examines HTTP messages contained within an established TCP session which are directed to the server. Should the URI of the HTTP command contain the string `attack`, it triggers an alert.

The IPv6 Plugin Suite adds the following options to target IPv6 specific rules within Snort 3 (Table 3):

Table 3: IPv6 Plugin Suite Options.

`ipv: n`	IP version, `n` is allowed to be 4 of 6
`ip6_flow: n`	matches the IPv6 *Flow Label*, `n` has a value between 0 and 2^{20}
`ip6_exthdr: n`	checks the whole IPv6 header-chain for a *Next Header* value equal to `n`
`ip6_rh: n`	checks the *Routing Type* of a IPv6 Routing Header, `n` is an 8 bit value
`ip6_optionId: n`	checks the presence of an option with ID `n` in a Hop-by-Hop or Destination header.

This allows Snort to filter on IPv6-specific attributes, as the following rule shows:

```
alert ip (msg:"Fragmented IPv6 packet with
        router alert!";
    ip6_exthdr: 44;
    ip6_optionId: 5;
    priority: 2;
    gid:1;
    sid:1000002;)
```

The IPv6 Standard (Deering and Hinden, 2017) defines a *Next Header* value of 44 for the Fragment header. This rule will fire if Snort detects a *Fragment* header in the header chain (`ip6_exthdr: 44`) and when there is a *Hop-by-Hop* or *Destination* header which carries the option 5 for a *router alert* (Faucheur, 2011) (`ip6_optionId: 5`). This rule has been given the second highest priority (`priority: 2`).

6 IDSV6 BENCHMARK RESULTS

In this section we present the results from our experiments running the *IDSv6* benchmark against the intrusion detection systems Snort, Zeek and Suricata in order to test their ability to detect protocol-level attacks targeted at IPv6 and compare the results against our IPv6 Plugin Suite.

The systems under test were Snort 3.0.0 (beta) with and without the IPv6 Plugin Suite, Zeek 2.5 and Suricata 3.2.1. For Snort the community rules, the snort subscriber rules[9] and the built-in rules were in-

[9]https://www.snort.org/downloads

stalled and activated. For Suricata all available rule sets from suricata-update[10] were installed. Also all Zeek modules were activated.

The outputs from intrusion detection systems are often very verbose. It happens frequently that not only one, but several warnings are generated for an attack. Sometimes these warnings fit to the attack, but sometimes they are misleading, redundant and of no benefit for the system administrator. This makes it difficult for the *IDSv6* benchmark to automatically produce a meaningful score value. Instead, the output was analyzed manually to determine which attacks are correctly detected. The complement set, i.e. the set of attacks which are not recognized, shows where the systems have to be improved.

The *IDSv6* benchmark consists of 62 attacks, this includes 36 attacks that use various forms of fragmentation to evade attack detection by firewalls and to trigger undesired behaviour in host systems. We will briefly discuss our findings for each individual IDS. Table 4 summarizes the results of our experiments.

6.1 Zeek

Zeek is capable to detect fragmentation overlap attacks (`fragmentation6 1-4,6-8,17,18,28,29`) and gives correct warnings ("fragment_inconsistency" or "fragment_overlap"). It also detects most other fragmentation anomalies as in fragmentation test 13 where Zeek warns "excessively_large_fragment" when an ICMPv6 request (Ping) with a payload size of 65486 Bytes is sent. In summary, Zeek detects 22 of the 36 fragmentation attacks.

Zeek still fails completely to detect abnormalities like the flooding attacks and all other attacks based on IP spoofing.

6.2 Suricata

Suricata detects 22 of the 33 fragmentation attacks, but the detection sets of Zeek and Suricata are not the same, Suricata detects correctly one-shot fragments (`fragmentation6 9-12,25-29`) with the alert "IPv6 useless Fragment extension header". In contrast to Zeek, Suricata does not detect all overlapping and large fragments. It also fails to detect the presence of two fragmentation headers in a packet.

Apart from fragmentation attacks, Suricata successfully detects 5 of the 7 `denial6` attacks. For not correctly detected attacks, Suricata gives a lot of false positive warnings. For the `parasite6` attack, Suricata warns "HTTP Host header invalid" for a correct HTTP request created by curl. In almost all attacks it also falsely detects "zero length padN option" which is a valid padding option for 2 bytes. In total Suricata detects 29 attacks.

[10]https://github.com/OISF/suricata-update

Table 4: IDSv6 Benchmark Results.

Attack	Snort3-beta	Snort3-ipv6-suite	Zeek	Suricata
covert_send6	✓	✓	✗	✓
denial6-1, 2, 3, 4, 7	✓	✓	✗	✓
denial6-5	✗	✗	✗	✗
denial6-6	✓	✓	✗	✗
dos_mld_chiron	✗	✓	✗	✗
dos_new_ipv6	✗	✓	✗	✗
fake_mldrouter6_advertise	✗	✓	✗	✗
fake_mldrouter6_terminate	✗	✗	✗	✗
flood_advertise6, mld26, mld6, router26, rs6, solicitate6	✗	✓	✗	✗
frag-1, 2, 3, 4, 15, 16, 17, 18, 26, 28, 29, 31, 32, 33	✓	✓	✓	✓
frag-5, 19, 20, 21, 34	✗	✗	✗	✗
frag-6, 7	✗	✗	✓	✓
frag-8, 13, 14, 30, 35, 36	✗	✗	✓	✗
frag-9, 10, 11, 12, 27	✓	✓	✗	✓
frag-22, 23, 24	✓	✓	✗	✗
frag-25	✓	✓	✗	✓
kill_router6	✗	✓	✗	✗
ndpexhaust26	✗	✓	✗	✗
parasite6_fake, real	✗	✓	✗	✗
redir6	✗	✓	✗	✗
rsmurf6	✓	✓	✗	✓
smurf6	✓	✓	✗	✗
toobig	✓	✓	✗	✗
Total = 62	33	47	22	29

6.3 Snort

Snort without the IPv6 Plugin Suite detects 6 of the 7 `denial6` attacks. It detects the `denial6-6` attack that Suricata missed and warns correctly "short fragment, possible DOS attempt" and "fragmentation overlap". However, it fails to detect the fragmentation over-

lap attacks (`fragmentation6 6-8`). On the other hand, Snort is the only IDS that detects some incomplete fragments (`fragmentation6 22,23,24`). In total, Snort detects 23 of the 36 fragmentation attacks. Snort also creates correct warnings for `covert_send6` and `rsmurf6` as Suricata did, but with more precise descriptions. Additionally, it detects `smurf` and `toobig`.

6.4 Snort with IPv6 Pugin Suite

When we include the IPv6 Plugin Suite in Snort, we are now able to detect 47 of 62 attacks of the *IDSv6* benchmark. Snort gains the ability to detect NDP and MLD flooding attacks. Because of the implemented inspector, it also detects NDP related attacks. For example, a `fake_mldrouter_advertise` is detected by comparison with the configured router list. However, Snort still fails to detect the `fake_mldrouter_terminate` attack, because we currently have no way do distinguish legitimate *Multicast Router Termination* messages from spoofed ones.

The plugin has no negative influence on the native detection ability for `denial6` and `fragmentation6` attacks. All previously detected attacks are still recognized by Snort.

6.5 Usability

All three tested IDS were able to directly examine recorded `pcap` files and supported multiple output modules including a log output. We found that the created default log files varied greatly in readability. Zeek/Bro produces human-readable tabulated output, while Suricata and Snort fail to write condensed human readable log files with their respective log modules, even though they have JSON output modules. The log files contain one entry for each packet that a rule matches and therefore creates an alert.

Especially in the case of Suricata, critical warnings can easily be missed by system administrators, because they may be hidden within a flood of irrelevant warnings. This problem is increased by the fact that almost all relevant attacks were categorized as a low priority warning. Suricata provided some warnings with higher priority like "BAD-TRAFFIC IP Proto 103 PIM", but which were (with the exception of one) either false positives or irrelevant for the attack.

7 CONCLUSION

When we compare the detection results for attacks against the IPv6 *Neighbor Discovery* process with results from (Schütte et al., 2012), we find that the current generation of IDS made little progress regarding the detection of such attacks. The results over all three systems, Snort, Suricata and Zeek, show that the intrusion detection systems are only able to detect basic IPv6 abnormalities such as fragmentation and header errors. Although similar detection functionality does already exist for IPv4 (Snort, for example, already has the functionality to detect ARP poisoning) the IPv6 version was never implemented.

The development of our IPv6 Plugin Suite greatly enhances the detection capabilities for genuine IPv6 attacks. When our IPv6 Plugin Suite is used, Snort exhibits an improved detection rate that results in the correct identification of 76% of the attacks covered by the *IDSv6* benchmark suite, whereas it previously only detected about 53%.

Another area that is in need of improvement is the usability of IDS. We think that the correct interpretation of events should not be difficult and although in most cases alerts from IDS will be fed into monitoring systems, it is still beneficial for administrators to be able to directly read and understand the alerts generated by the system.

Zeek currently shows the most readable output. Even the new Snort 3, which is still in development, does not create very human readable reports and therefore makes it necessary to use third party log analytics and visualization tools such as ELK[11].

REFERENCES

Allen, J. M. (2015). A Performance Comparison of Intrusion Detection Systems with Regard to IPv6. Technical report, SANS Institute InfoSec Reading Room.

Arkko, J., Aura, T., Kempf, J., Mäntylä, V.-M., Nikander, P., and Roe, M. (2002). Securing IPv6 neighbor and router discovery. In *WiSE '02: Proceedings of the 1st ACM workshop on Wireless security*, pages 77–86, New York, NY, USA. ACM.

Beck, F., Cholez, T., Festor, O., and Chrisment, I. (2007). Monitoring the Neighbor Discovery Protocol. In *The Second International Workshop on IPv6 Today - Technology and Deployment - IPv6TD 2007*, Guadeloupe.

Deering, S. and Hinden, R. (2017). Internet Protocol, Version 6 (IPv6) Specification. RFC 8200, Internet Engineering Task Force.

Faucheur, F. L. (2011). IP Router Alert Considerations and Usage. RFC 6398, Internet Engineering Task Force.

[11] https://www.elastic.co/elk-stack

Gont, F. (2013). Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery. RFC 6980, Internet Engineering Task Force.

Heuse, M. (nd). THC IPv6 attack tool kit.

Hogg, S. and Vyncke, E. (2009). *IPv6 Security*. Cisco Press, Indianapolis, IN 46240 USA.

Levy-Abegnoli, E., de Velde, G. V., Popoviciu, C., and Mohacsi, J. (2011). IPv6 Router Advertisement Guard. RFC 6105, Internet Engineering Task Force.

McHugh, J. (2000). Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security (TISSEC)*, 3(4):262–294.

Narten, T., Nordmark, E., Simpson, W., and Soliman, H. (2007). Neighbor Discovery for IP version 6 (IPv6). RFC 4861, Internet Engineering Task Force.

Nikander, P. (2002). Denial-of-Service, Address Ownership, and Early Authentication in the IPv6 World. In Christianson, B., Malcolm, J., Crispo, B., and Roe, M., editors, *Security Protocols*, volume 2467 of *Lecture Notes in Computer Science*, pages 12–21. Springer, Berlin/Heidelberg.

Nikander, P., Kempf, J., and Nordmark, E. (2004). IPv6 Neighbor Discovery (ND) Trust Models and Threats. RFC 3756, Internet Engineering Task Force.

Paxson, V. (1999). Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31(23-24):2435–2463.

Rey, E., Atlasis, A., and Salazar, J. (2016). Mld considered harmful.

Roesch, M. (1999). Snort: Lightweight Intrusion Detection for Networks. In *Proceedings of the 13th USENIX conference on System administration*, pages 229–238.

Salih, A., Ma, X., and Peytchev, E. (2015). Detection and Classification of Covert Channels in IPv6 Using Enhanced Machine Learning. In *Proc. of The International Conference on Computer Technology and Information Systems*.

Schütte, M., Scheffler, T., and Schnor, B. (2012). Development of a Snort IPv6 Plugin - Detection of Attacks on the Neighbor Discovery Protocol. In *9th International Conference on Security and Cryptography (SECRYPT)*, Rom, Italy.