Development of a Snort IPv6 Plugin
Detection of Attacks on the Neighbor Discovery Protocol

Martin Schütte[1], Thomas Scheffler[1] and Bettina Schnor[2]
[1]*Department of Electrical Engineering, Beuth University of Applied Sciences, Berlin, Germany*
[2]*Department of Computer Science, Potsdam University, Potsdam, Germany*
{mschuette, scheffler}@beuth-hochschule.de, schnor@cs.uni-potsdam.de

Keywords: IPv6, Neighbor Discovery, Intrusion Detection System.

Abstract: This paper describes the implementation and use of a preprocessor module for the open source Intrusion Detection System Snort. Our implementation utilizes preprocessor APIs for the extension of Snort and provides several new IPv6-specific rule options that make the definition of IPv6-specific attack signatures possible. The preprocessor detects attacks against the IPv6 Neighbor Discovery Protocol and can identify suspicious activity in local IPv6 networks. This includes misconfigured network elements, as well as malicious activities from attackers on the network. To our knowledge this is the first such implementation in an Open Source IDS.

1 INTRODUCTION

The IPv6 protocol was standardized in 1998 and is now implemented in virtually all operating systems and network devices. Since then, network security has developed considerably but there was little incentive to strengthen IPv6 against attacks, because deployment in "real networks" (i.e. outside of academic or industrial test labs) has remained marginal. This combination leads to the present situation where widespread IPv6 deployment is imminent (due to IPv4 address depletion), but protocol design and implementation in devices leave ample opportunity for local network based man-in-the-middle and denial-of-service attacks.

Intrusion Detection Systems (IDS) are established tools for the detection of attacks against computers and networks. In the past, the spread of the IPv4-based Internet has lead to a co-evolution of network security research, engineering, and IDS development. For the IPv6-based Internet, this co-evolution is to be expected as well, so there is evident demand for new advances in detection capabilities.

Snort (Roesch, 1999) is a powerful and widely used IDS, that already supports the IPv6 protocol. Based on practical experiments we analyzed capabilities and limitations for the detection of critical events in IPv6 networks (Eraña and Scheffler, 2010). Here we found that Snort provides basic IPv6 support, but lacks the ability to detect many attacks on IPv6-specific protocol mechanisms.

We developed a plugin for the Snort IDS that provides the ability to detect IPv6-specific attacks and makes Snort easily deployable in common network setups. It combines previously implemented and new detection functionality and can be integrated into every Snort IDS installation. The main goal lies in the tracking of the Neighbor Discovery Protocol (NDP) used for IPv6 autoconfiguration. Additionally it makes specific IPv6 protocol fields and extension headers available to the Snort detection engine. A network administrator is now able to write specific Snort rules that enable the detection of link-local denial-of-service attacks like those implemented in "The Hacker's Choice" toolkit (Heuse, nd).

2 RELATED WORK

Our work builds upon related work describing known security issues in IPv6 and NDP (Hogg and Vyncke, 2009; Nikander et al., 2004). Existing work on IDSs and IPv6 support only describes the necessary decoding of IPv6 packets for upper layer processing, but not the detection of IPv6-specific attacks using general purpose, open source IDS products.

The root cause of most problems 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 like port-based Net-

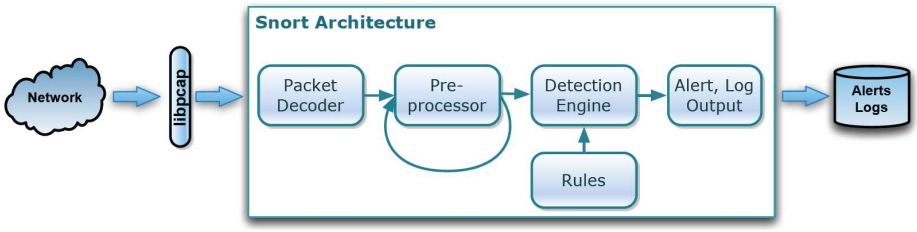

Figure 1: Schematic data flow in the Snort IDS.

work Access Control (IEEE 802.1X), IP Security (IPSec) and SEcure Neighbor Discovery (SEND) propose a cryptographic host authentication. Another approach are filter criteria for NDP messages on layer 2 devices as specified by the IPv6 Router Advertisement Guard (Levy-Abegnoli et al., 2011).

One existing software tool that detects NDP anomalies is "ndpmon" (Beck et al., 2007), a host-based tool that listens to all Neighbor Discovery messages to track active nodes on its subnet. Our solution builds upon this approach but extends an existing IDS as a framework that already implements packet capturing, decoding, configuration, logging mechanisms, and log analysis tools.

3 SNORT ARCHITECTURE

Snort was created in 1998 by Martin Roesch. It is probably the most widespread open source IDS and provides multiple interfaces for third-party rules and plugins. One of its most powerful features is its rule language, which allows everyone to define and share their own signatures in a relatively simple plain-text format. Figure 1 shows the main data processing stages of the IDS. The main architectural elements that influence the detection capabilities of the IDS are the Detection Engine and the Preprocessors which are discussed next.

3.1 Detection Engine

The core of Snort's detection capabilities lies in the evaluation of configured attack signatures. A simple rule notation allows developers and users to easily configure and customize the IDS policy for their own networks. A rule header describes very basic attributes (action, protocols, direction and endpoints) and is followed by a rule body which contains a number of options to select certain packets or sequences of packets.

Basic IPv6 support was added to Snort in version 2.8 in 2007. Following versions added checks to the decoder (e.g. resulting in decoder alerts for packets

```
alert icmp any any -> any any (itype: 3;    \
  msg: "dest unreachable or time exceeded?";\
  sid: 1000000; rev: 1;)
```

Figure 2: Ambiguous Snort signature without IP protocol distinction.

with a multicast source address) and the ability to normalize IPv6 packets. In regard to detection signatures the developers decided to pass all IPv6 values the same way as IPv4 values to the detection engine, thus a rule option for the IPv4 time-to-live (e.g. `ttl: 100`) will also match an IPv6 hop limit. This has the advantage that all existing signatures continue to work well on IPv6, but has the disadvantage that signatures cannot distinguish between IPv4 and IPv6 packets. For example, the rule in Figure 2 is ambiguous and matches different ICMP messages in IPv4 and IPv6.

Rules are designed to be stateless, and have very limited capabilities to keep state, for example by setting own flags on TCP connections. Therefore, the rule management does not have to consider dependencies among different rules. However, it cannot be used to detect patterns in connectionless protocols, such as ICMP and IPv6 autoconfiguration.

3.2 Snort Preprocessors

Snort preprocessors generally have three functions: They implement their own checks to trigger alerts, they normalize data to simplify detection rules and they provide new rule options for the detection layer. For example, the *http inspect* preprocessor implements preprocessor alerts, rule options, and additionally normalizes HTTP-specific data such as URIs.

The preprocessor stage can include multiple modules; providing functions such as defragmentationand portscan detection. Preprocessors may perform additional payload decoding for a number of application layer protocols such as HTTP, SMTP and DNS.

Snort provides a dynamic plugin API; so it is possible to develop third-party preprocessors and deploy them as dynamically linked libraries. These plugins get access to the packet data and some of Snort's subsystems, like event logging queues. These capabil-

ities make the preprocessor stage most suitable for the inspection of additional connectionless protocols; therefore we chose to implement our IPv6 autoconfiguration checks as a preprocessor plugin using the dynamic preprocessor API.

4 IPV6 PLUGIN FUNCTIONALITY

Our preprocessor plugin provides two services: First it adds new IPv6-specific rule options to Snort's signature language, second it maintains its own network view by tracking all neighbor discovery messages in the IPv6 subnet.

4.1 Rule Options

A new set of rule options makes IPv6 protocol values available for Snort detection signatures; these values include the basic header fields, extension headers, and options contained in destination or hop-by-hop headers. Examples are the version number of the Internet Protocol, the flow label value, inclusion of routing headers, or included router alert options. Most rule options are implemented with a set of operators to enable not only exact matching (like with `ip6_flow: 0`), but also negation and comparisons (like `ip6_extnum: >3`). A list of new rule options is given in Table 1. Figure 3 shows an improved version of the ruleset in Figure 2. By using the plugin's `ipv:` rule option for IP protocol selection, we can now correctly distinguish between the different error semantics of the ICMP and ICMPv6 types.

Table 1: IPv6 rule options provided by our preprocessor.

`ipv`	IP version
`ip6_tclass`	Traffic Class
`ip6_flow`	Flow Label
`ip6_exthdr`	Extension Header
`ip6_extnum`	Num. of Ext Hdrs.
`ip6_option`	Dst-/HbH-Option
`ip6_optval`	Dst-/HbH-Option Value
`ip6_rh`	Routing Header
`icmp6_nd`	is NDP pkt
`icmp6_nd_option`	NDP Option

We already used the new rule options to implement several IPv6-specific signatures to check for unusual header values and packets. We expect these rule options to become important building blocks for the detection of future IPv6-specific attacks.

```
alert icmp any any -> any any (              \
  ipv: 4; itype: 3;                          \
  msg: "ICMPv4 dest unreachable";            \
  sid: 1000002; rev: 1;)

alert icmp any any -> any any (              \
  ipv: 6; itype: 3;                          \
  msg: "ICMPv6 time exceeded";               \
  sid: 1000003; rev: 1;)
```

Figure 3: New Snort signatures, with rule options for IP protocol distinction.

4.2 Preprocessor

The preprocessor hooks into Snort's packet processing and receives all IPv6 packets. This allows it to inspect every packet and keep state information for the monitoring of the connectionless IPv6 autoconfiguration.

The basic functions of the preprocessor are stateless checks of all IPv6 packets with extension headers for consistency and standard conformance. For example a packet should never include an empty extension header (carrying only padding), or padding options containing data. Packets like this show unusual node behavior and the preprocessor will raise an alert.

The main purpose of our plugin is to build and maintain its own network view by observing all ICMPv6 Neighbor Discovery messages. This allows it to keep track of all routers and hosts with their tentative and acquired IP addresses. Every change in this network view triggers a Snort event; this happens every time a new host or router appears on the net, a router changes its advertisement, or a duplicate address detection fails.

The network model of the preprocessor is simple and regards every observed pair of MAC and IP addresses as a host; those hosts that send router advertisement messages are considered routers and their announced network attributes (prefix, lifetime, flags) are stored to detect later changes in these values.

One of the most common IPv6 related security issues is the resource exhaustion attack, for example against a node's neighbor cache or routing table (Wheeler, nd). An attacker can easily generate lots of neighbor or router announcements (and "flood" the network), while the target has to keep state and allocate memory for every new binding. Because this attack vector applies equally to all network monitoring tools, our plugin is designed to handle a flooding situation. All seen nodes are kept in three collections, each with a maximum number of entries: one for routers, one for active hosts, and one for new hosts with tentative addresses. The last one does not only increase the level of detail but also strengthens

Table 2: IPv6 preprocessor alerts.

SID	Message
1	RA from new router
2	RA from non-router MAC address
3	RA prefix changed
4	RA flags changed
5	RA for non-local net prefix
6	RA with lifetime 0
7	new DAD started
8	new host in network
9	new host with non-allowed MAC addr.
10	DAD with collision
11	DAD with spoofed collision
12	mismatch in MAC/NDP src ll addr.
13	extension header has only padding
14	option lengths $\neq$ ext length
15	padding option data $\neq$ zero
16	consecutive padding options

the preprocessor against denial of service attacks. In case of a simple flooding attack, where the attacker sends bogus neighbor announcements to our network, the number of tentative address entries will exceed the limit and further addresses are simply ignored. A more sophisticated attack may keep state to fill the active host list as well, thus causing a limited denial of service, but even then the plugin will only use a fixed amount of memory and is not susceptible to memory exhaustion attacks.

The preprocessor is configurable and can be supplied with additional information about the monitored network; among these are the used IP address prefix, router MAC addresses, and host MAC addresses. If these options are configured, then all IPv6 packets are checked against them and an alarm is raised if e. g. an unexpected address prefix is announced. A list of all implemented alerts is given in Table 2.

5 CONCLUSIONS

The IPv6 protocol has several weaknesses in its neighbor discovery and autoconfiguration services. Most of these problems arise from the unsolved early authentication problem and the implicit assumption that all link-local nodes are trustworthy. Thus, an attacker with physical network access and control over a connected node is usually able to assume a man-in-the-middle position and also to perform various denial-of-service attacks against particular hosts or the router.

A new IPv6 plugin was developed to extend the Snort IDS with integrated IPv6-specific detection routines. It includes neighbor discovery tracking to alert when new hosts and routers appear on-link. It also provides additional rule options that expose IPv6 specific header fields to the Snort detection module. The rule options facilitate the writing of new detection signatures using the flexibility of Snort's rule language, for example to detect attacks from the THC toolkit. The integration into the Snort infrastructure facilitates an easy deployment and integration into existing IDS setups. A number of test cases verify the plugin's functionality and demonstrate that known flooding and neighbor discovery attacks can be detected.

The plugin may become part of future Snort releases; its current development repository is available at https://github.com/mschuett/spp_ipv6.

ACKNOWLEDGEMENTS

This work was carried out as part of the *IDSv6* research project (http://ipv6-ids.de), funded by the German Federal Ministry of Education and Research (BMBF).

REFERENCES

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.

Eraña, E. I. and Scheffler, T. (2010). IPv6 Intrusion Detection mit Snort. In *Forschungsbericht der Beuth Hochschule für Technik Berlin*. Beuth Verlag GmbH Berlin-Wien-Zürich.

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.

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.

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

Wheeler, J. S. (nd). IPv6 NDP Table Exhaustion Attack.