Global Information Assurance Certification Paper

Intrusion Prevention with L7-Filter

GSEC Gold Certification

Author: Rui Santos, rui.rgonzalez@gmail.com

Adviser: Jim Purcell

Accepted: August 12, 2008

Table of Contents

1. Introduction

Abstract

The purpose of this paper is to present the possibility of using L7-filter as an Intrusion Prevention tool. The goal of this paper is to explain how to use SNORT rules on L7-filter and to show this tool as an alternative to SNORT Inline. This paper will not conclude witch tool is better, it will only bring a new perspective on another use for L7-Filter, of the many that it already has. The motivation for doing this paper comes from doing a Master's Thesis Project which consisted of designing a web application tool for L7-Filter administration as a QoS tool.

Solution Description

L7-filter is a classifier for Linux's Netfilter that identifies packets based on application layer data. It can classify packets as Kazaa, HTTP, Jabber, Citrix, Bittorrent, FTP, Gnucleus, eDonkey2000, etc., regardless of the port. It complements existing classifiers that match on IP address, port numbers and so on. L7-filter has been used mainly as QoS tool.

SNORT is an open source network intrusion prevention and detection system that uses a rule-driven language, which combines the benefits of signature, protocol, and anomaly based inspection methods. With millions of downloads to date, Snort is the most widely deployed intrusion detection and prevention technology worldwide and has become the de facto standard for the industry. The Intrusion Prevention version of SNORT is called SNORT Inline.

As I've explained earlier the goal of this paper is to show how to use SNORT rules on L7-Filter. It's possible to use SNORT rules because they use signatures for attacks that are pattern based, just like L7-Filter patterns for protocols. What these applications do is to look for an expression in the payload of IP packets that matches the pattern of an attack, whether it is a virus, malware, network abuse or the use of unauthorized tools. You can check how to write patterns for L7-filter in (http://l7-filter.sourceforge.net/Pattern-HOWTO).

Other Solutions

There are some tools in the open source community that can translate SNORT rules into IPTables rules like FWSNORT. FWSNORT uses the string match capability of IPTables to look for expressions in the payload of IP Packets. The problem with IPTables string match is that it looks at the payload of an individual packet and not as a part of a communication between applications. This situation makes the rule vulnerable to IDS evasion techniques like packet fragmentation. To avoid this problem FWSNORT uses an IPTables tool called ConnectionTracking that allows tracking all the packets that belong to the same connection. But there are some issues with this solution, because FWSNORT works at the Network layer and not at the application layer. You can check the SANS paper (Intrusion Detection Evasion, Corbin del Carlo, 2003) to learn more about IDS evasion techniques.

SNORT Inline is another solution that I've mentioned earlier. SNORT Inline also uses IPTables but in a different way. It has an IPTables queue configured where the packets inspected by SNORT Inline are sent, after inspection SNORT Inline will decide if a connection is legit or if it's an attack and will be dropped. There have been some performance issues with SNORT Inline, but it seems to be the

best open source solution for an Intrusion Prevention System.

2. Why use L7 -filter

Heavily used systems may lack available resources to deploy an additional userland process for intrusion detection (such as Snort). L7-filter packet inspection takes place directly within the Linux kernel, and so this usually places a lightweight usage footprint on system resources as there is no need to copy data from kernel memory into a userland process (as is the case for a normal IPS). On systems where it is inappropriate to deploy a dedicated IDS/IPS because of resource constraints, this can be an important advantage of L7-filter.

L7-filter is inline to network traffic, it's an ideal candidate for taking action against certain attacks that are particularly malicious. For example, suppose that a new vulnerability is discovered within Linux server software (such as BIND) that is deployed in your infrastructure. If the Snort community develops a signature to detect attacks against this vulnerability, L7-filter can be configured to drop packets (via the IPTables DROP target) that appear to match the attack, and standard protocol responses can be issued by L7-filter via the REJECT target. If the server uptime is tied to a Service Level Agreement (SLA), then there may be a waiting period before it can be taken down and patched, and this assumes the availability of a patch to fix the vulnerability (which is not always the case). If the server software must remain globally available before an outage window can be scheduled to apply a patch, an inline prevention mechanism can provide valuable

protection against exploits for the vulnerability. In addition, because L7-filter policies are lightweight, they can usually be deployed alongside other prevention mechanisms such as Snort running in inline mode.

3. Translating SNORT Rules.

You can download freely SNORT rules from bleedingsnort site (http://www.bleedingsnort.com). SNORT rules are composed by several options. Each option on a SNORT rule has an IPTables equivalent.

SNORT rule header

This SNORT header instructs SNORT to match all TCP traffic from any from any source address to port 53 on any IP address within the 192.168.10.0/24 subnet looks like:

alert tcp any any -> 192.168.10.0/24 53

This header is equivalent to the following IPTables command:

[IPTablesfw]# IPTables -A FORWARD -p tcp -d 192.168.10.0/24 --dport 53 -j LOG

SNORT rule options

SNORT rules have several options, here are a few:

Content flags dsize uricontent

itype ip_proto offset icode

flow depth ttl replace

distance tos resp within ipopts

We are going to focus only on the options CONTENT and URICONTENT. If you wish to know more about the IPTables equivalent rules you can check the book ("Linux Firewalls", Michael Rash, 2007).

Content

With the content option of the SNORT we have the pattern expression that is going to be matched against the IP Packet Payload. Let's see this SNORT rule:

alert udp any any -> any 53 (msg: "DNS /bin/sh attempt"; content: "/bin/sh";

sid: 100001)

The pattern is "/bin/sh". We are going to create a pattern file on the protocols directory called as the sid number of the rule "100001.pat". The content of this file will be the name of the pattern and the expression:

This is the pattern for the SNORT rule alert udp any any -> any 53 (msg: "DNS /bin/sh attempt"; content: "/bin/sh";

```
#sid:10001

100001

/bin/sh
```

Pattern File

This is the IPTables equivalent rule.

```
[IPTablesfw]# IPTables -A FORWARD -p udp --dport 53 -m layer7 -- l7proto
100001  -j LOG --log-prefix "SID100001
```

Uricontent

In L7-filter there is no way to decode a URL encoded to its normalized content. uricontent keyword in the Snort rule language searches for the NORMALIZED request URI field. This means that if you are writing rules that include things that are normalized, such as %2f or directory traversals, these rules will not alert. The reason is that the things you are looking for are normalized out of the URI buffer. The followig example was taken from the ("Snort Users Manual 2.8.2", The Snort Project, May 7 2008):

For example, the URI:

/scripts/..%c0%af../winnt/system32/cmd.exe?/c+ver

Will get normalized into:

/winnt/system32/cmd.exe?/c+ver

Another example, the URI:

/cgi-bin/aaaaaaaaaaaaaaaaaaaaaaaaa/..%252fp%68f?

Will get normalized into:

/cgi-bin/phf?

So L7-filter doesn't really fully support the URICONTENT option. Anyway, it

can be useful to translate the URICONTENT to a L7-filter pattern in case a URL isn't encoded. You can create a pattern file just like we've done with the content option.

4. A Script to Translate SNORT Rules.

I've made a simple script in PERL that converts a snort rule into a L7-filter pattern file. This is how it looks:

```
#!/usr/bin/perl


$DEFAULT_ACTION = "log,pass";

die("Usage: snort2l7.pl <snort rule >\n") unless(@ARGV);

                  $uricontent = "";
                  $content = "";
                  $msg = "";
                  $classtype = "";
                  $reference = "";
                  $sid = "";
#This for is necessary to concatenate al the arguments in a string

                  foreach $rule (@ARGV) {
                  $rules= $rules . " " .  $rule;

                  }
#We are going to filter some SNORT options in the string that has the rule
                  foreach $rule (split(/;\s+/, $rules)){

                        if ($rule =~ /uricontent:\s*(.*)/) {
                              $uricontent = $1;
                        } elsif ($rule =~ /content:\s*(.*)/) {
                              $content = $1;
                        } elsif ($rule =~ /msg:\s*(.*)/) {
                              $msg = $1;
                        } elsif ($rule =~ /sid:\s*(\d+)/) {
                                    $sid = $1;
                  }
                  }

                  print "# (sid $sid) $msg";
                  print "\n";
                  print $sid;
                  print "\n";
                  if (!($uricontent eq "")) {
                        if (!($content eq "")) {
                              print $uricontent;
                              print $content;
                        } else {
                              print $uricontent;

                        }
                  } else {
                        print $content;
                  }

                        print "\n\n";
```

Script that translates Snort Rules

In order to produce a L7-filter pattern file with this script you have to execute it in the following way:

```
[root@localhost ~]# /usr/bin/perl snort2l7.pl "alert tcp $EXTERNAL_NET
any -> $HTTP_SERVERS $HTTP_PORTS msg:"WEB-ATTACKS /bin/ps command
attempt"; flow:to_server,established; uricontent:"/bin/ps"; nocase;
classtype:web-application-attack; sid:1328; rev:6;" >> 1328.pat
```

Command Executed

The output will be the 1328.pat L7-filter pattern file. This is how it looks:

```
# (sid 1328) WEB-ATTACKS /bin/ps command attempt
1328
/bin/ps
```

Pattern File

5. Future work.

There are so many applications available in the open source community that uses in some way IPTables. A person could think that everything has been done already. But it seems to me that the next challenge that the open source community has to face is how these wonderful tools can be integrated. In what concerns L7-filter the challenge is the same. I've presented in this paper a perl script that

converts the SNORT content and uricontent field into a L7-filter Pattern file, the next challenge is to produce a framework of scripts that given an input SNORT rule it produces an IPTables equivalent rule.

Personally I would like to see an application that would integrate nessus and L7-filter, so it could prevent attacks to servers who aren't or can't be patched. In my opinion this is one of the biggest issues in patch management.

6. Conclusion.

IPTables with L7-filter has proven to be a very flexible tool. It has been used as an Application Layer Firewall and as QoS tool. Could it be use as an Intrusion Prevention System? I believe it could be used in some situations as an IPS, when there exists constraints in resources or simply by practicality. L7-filter used alone as an IPS can be vulnerable to some IDS evasion techniques and some of the SNORT rules can't be translated to L7-filter as I've mentioned earlier. That's why it should be used with the awareness of its limitations and the risks should be quantified.

A good approach for mitigating these risks would be using L7-filter in front of a SNORT sensor. In this configuration we can take advantage of the L7-filter performance, doing the filtering of unauthorized tools, virus and malware traffic at level one and leave the remaining traffic for the SNORT deep inspection at level two. Using L7-filter as the only IPS in the network isn't a good idea, unless the alternative is not having an IPS. Nonetheless I think the future of this tool isn't being used as an IPS, but to be integrated with other applications in order

to expand their capabilities.

7. References.

Designing and Implementing Linux Firewalls and QoS using netfilter, iproute2, NAT, and L7-filter. Lucian Gheorghe. Packet Publishing (2006).

LINUX FIREWALLS. Michael Rash. No Starch Press. (2007).

Intrusion Detection Evasion, Corbin del Carlo, SANS (2003).

Snort Users Manual 2.8.2, The Snort Project, (May 7 2008).