

fwsnort-1.0.5: iptables intrusion detection

By Russ McRee – ISSA member, Puget Sound (Seattle), WA, USA chapter



Prerequisites

Linux
iptables
IPTables::Parse module to parse iptables policies (included)

Similar Projects

IPCop¹

By sheer happenstance, I recently revisited Michael Rash's cIPHERDYNE.org, thus ringing the proverbial toolsmith bell. All of Michael's offerings are well worthy of toolsmith consideration, but fwsnort presents an opportunity to promote a combination of many things I am fond of, including Snort, iptables, emergingthreats.net, and a bit of Ubuntu hardening for good measure.

fwsnort creates an application layer IDS/IPS by injecting iptables with Snort functionality. Allow me to be clear here: Michael has written so much on fwsnort, Linux firewalls, iptables, psad, and others, that it doesn't make sense for me to rehash what has already been so expertly penned. Thus, as with any toolsmith effort, I engaged the project author directly and Michael was immediately engaging. As the fwsnort creator, developer, and author of *Linux Firewalls: Attack Detection and Response with iptables, psad, and fwsnort*, I thought it best to let Michael speak for himself about project plans and highlights:

A challenge for intrusion detection systems is to handle network layer packet fragmentation in the same manner as a targeted IP stack does when under an attack which involves fragmentation. Artificially fragmenting an attack is easy with a tool such as fragroute, but not all IP stacks handle fragmentation in the same way. If an IDS cannot apply the same algorithm to incoming packets that the target uses to reassemble packets, then the IDS can either miss attacks (cause false negatives) or set off bogus alarm bells when there really isn't any attack (issue false positives). In the case of fwsnort, by using the iptables connection tracking modules, signature inspection takes place after the defragmentation process is complete, and for attacks directed at the local system

there is no uncertainty about which algorithm to use – inspection happens against reassembled data that is as the host (Linux in this case) itself sees. The frag3 preprocessor in the Snort IDS can apply different defragmentation algorithms to different hosts, but this requires manual configuration. There are many features in Snort that fwsnort cannot emulate, but fragmented attacks directed against the local system are handled quite well.

fwsnort can handle multiple application layer content matches, and this combined with the power of other matching criteria that can be expressed by iptables (such as matching on connection state) allows fwsnort to achieve a 77% translation rate (7191 out of 9353 rules) for the Snort-2.3.3 and Emerging Threats rules together. The rules distributed in the older Snort-2.3.3 are the last rules distributed under the GPL with Snort. The latest VRT rules from Sourcefire are not translated as well by fwsnort because of the increasing usage of features such as PCRE's, and there is no PCRE engine that runs directly in the kernel where iptables can use it.

iptables is a really well tested piece of code that runs in all major Linux distributions, and because iptables is a firewall, this code runs inline to the packet data path by definition. Further, by running inline, iptables is in an ideal position to drop malicious packets on the floor before forwarding them on to their intended target. This puts fwsnort into the category of a basic IPS instead of just offering detection features.²

Michael's plans for future fwsnort releases also include supporting the u32 match, now that it has been re-integrated with iptables and with the Linux kernel. It is possible to add support in fwsnort for more complex application layer matches that are similar to the tests performed by the Snort byte_test keyword. The u32 match allows bytes to be specified within payload data, have a mask applied, then compared to a numeric range, creating a powerful tool for effective attack detection. Integrating this outstanding feature with fwsnort is a top priority for Michael so expect it in the next release.

¹ <http://www.ipcop.org>.

² Notes from Michael Rash. Extra reading: http://www.ukuug.org/newsletter/17.1/#ids_s_mike_.

fwsnort currently relies on psad for syslog and email reporting, but additional support is planned that will take the form of a web interface. The existing email reports provided by combining psad with fwsnort can be quite informative however.

fwsnort utilization

Harden Ubuntu

As my test servers all run Ubuntu, I felt it relevant here to point you to a great Ubuntu hardening resource, Brian Provost's *The Big Ol' Ubuntu Security Resource*.³ Recommendations include modifying default settings including shared memory, SSH root login, and limiting super user access, enabling automatic security updates, securing the home directory, a list of essential security installs, bootup security, and an additional list of secondary security enhancements. It's great for those of you new to Ubuntu, and an excellent checklist for you old hands who may have missed something.

Emerging threats

Matt Jonkman's *Emerging Threats*, formerly *Bleeding Snort/Threats*, is a goldmine of cutting edge Snort rules with particular attention to bot analysis. Check out emergingthreats.net, contribute rules, and watch for near real time trends in rule coverage.

Matt let me know that, by the time this article is live, the production release of sidreporter will be available. Sidreporter allows anonymous IDS/IPS hits, and with data flowing through that channel, emergingthreats.net will have a ton of new rulesets to generate.

fwsnort at work

Again, Michael's written so much on the tools he's developed, I'd simply advise you to read what's available at cipherdyne.org, but most relevant to this article, and getting under way with fwsnort, is the freely available Chapter 10 of his *Linux Firewalls book*.⁴

Down to business. To work with fwsnort in my test environment, I configured iptables on my test server as follows:

```
iptables -L
iptables -A INPUT -m state --state
ESTABLISHED,RELATED -j ACCEPT
```

```
root@holisticinfosec01:/etc/fwsnort/snort_rules# fwsnort
-----
Snort Rules File      Success  Fail    Ipt_apply Total
[+] attack-responses.rules  16      1      15      17
[+] backdoor.rules         65     11     20     76
[+] bad-traffic.rules      9       3       0     12
[+] chat.rules             29      1      10     30
[+] ddos.rules             18     14       3     32
[+] deleted.rules         257    14     118    271
[+] dos.rules              9       7       1     16
[+] emerging-web.rules     73     83     69     156
[+] exploit.rules          36     46       7     82
[+] misc.rules             42     18     10     60
[+] multimedia.rules       4       6       3     10
[+] other-ids.rules        3       0       3       3
[+] p2p.rules              18      0       8     18
[+] policy.rules           20      1       3     21
[+] rpc.rules              37     91     11    128
[+] rservices.rules        13      0       2     13
[+] scan.rules             14      4       7     18
[+] sql.rules              42      4       2     46
[+] web-attacks.rules      46      0      46     46
[+] web-cgi.rules          348     2     344    350
[+] web-client.rules       9      16       9     25
[+] web-misc.rules         300    28    285    328
[+] web-php.rules          115    11     115    126
-----
[+] Generated iptables rules for 1523 out of 1884 signatures: 80.84%
[+] Found 1091 applicable snort rules to your current iptables
    policy.

[+] Logfile: /var/log/fwsnort.log
[+] iptables script: /etc/fwsnort/fwsnort.sh
```

Figure 1 – fwsnort execution results

```
iptables -A INPUT -p tcp --dport ssh -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -j DROP
iptables -I INPUT 1 -i lo -j ACCEPT
iptables -I INPUT 5 -m limit --limit 5/min -j LOG --log-
prefix "iptables denied: " --log-level 7
sh -c "iptables-save > /etc/iptables.rules"
iptables-save5
```

Installing fwsnort is as easy as downloading the appropriate version,⁶ unpacking it via `tar zxvf fwsnort-1.0.5.tar.gz`, `cd` to `fwsnort-1.0.5`, and run `./install.pl` as root. You'll be prompted to download the emerging-all ruleset. Do so if you wish to take advantage of all this ruleset offers. I passed this up based on my rule preferences, in favor of grabbing a specific Emerging Threats rule set described below. As a believer in a well tuned Snort implementation, I utilize only installation-relevant rules and eliminate the families that aren't necessary. This results in a lower cost to the system, better performance, and fewer packet drops. If you're running an Apache web server, you don't likely want to monitor web-iis.rules. First, I grabbed the `emerging-web.rules`.

```
cd /etc/fwsnort/snort_rules
wget http://www.emergingthreats.net/rules/
emerging-web.rules
```

3 <http://www.itsecurity.com/features/ubuntu-secure-install-resource>.

4 http://www.nostarch.com/download/firewalls_ch10.pdf.

5 <https://help.ubuntu.com/community/IptablesHowTo>.

6 <http://www.cipherdyne.org/fwsnort/download>.

I then cleaned up the other rule sets not relevant to my installation (yours will vary):

```
rm -f dns.rules experimental.rules finger.rules
ftp.rules icmp-info.rules icmp.rules imap.rules
info.rules local.rules mysql.rules netbios.rules
nntp.rules oracle.rules pop2.rules pop3.rules
porn.rules shellcode.rules snmp.rules smtp.rules
telnet.rules tftp.rules virus.rules web-coldfusion.rules
web-frontpage.rules web-iis.rules x11.rules
```

Running fwsnort will produce the following results displaying what rules successfully applied to the iptables configuration, those that failed, those that were applied, and totals (see Figure 1).

Run the resulting fwsnort.sh script: /etc/fwsnort/fwsnort.sh, achieving the following results.

```
root@holisticinfosec01: /etc/fwsnort/snort_rules# /etc/fwsnort/
fwsnort.sh
```

```
[+] Adding attack-responses rules. Rules added: 15
[+] Adding backdoor rules. Rules added: 20
[+] Adding chat rules. Rules added: 18
[+] Adding ddos rules. Rules added: 3
[+] Adding deleted rules. Rules added: 118
[+] Adding dos rules. Rules added: 1
[+] Adding emerging-web rules. Rules added: 69
[+] Adding exploit rules. Rules added: 7
[+] Adding misc rules. Rules added: 26
[+] Adding multimedia rules. Rules added: 3
[+] Adding other-ids rules. Rules added: 3
[+] Adding p2p rules. Rules added: 8
[+] Adding policy rules. Rules added: 3
[+] Adding rpc rules. Rules added: 11
[+] Adding rservices rules. Rules added: 2
[+] Adding scan rules. Rules added: 7
[+] Adding sql rules. Rules added: 2
[+] Adding web-attacks rules. Rules added: 46
[+] Adding web-cgi rules. Rules added: 344
[+] Adding web-client rules. Rules added: 9
[+] Adding web-misc rules. Rules added: 285
[+] Adding web-php rules. Rules added: 115
[+] Finished.
```

To test my installation I threw a format strings and directory traversal attack at the server and was treated to immediate results. I've color matched the Snort SID in the log entry to the Snort rule that generated it.

```
Sep 4 16:58:54 holisticinfosec01 kernel: [29516.356098]
[772] SID1122 ESTAB IN=eth0 OUT= MAC=00:0c:6e:3c:
b4:71:00:13:e8:e8:81:37:08:00 SRC=192.168.248.101
DST=192.168.248.104 LEN=686 TOS=0x00 PREC=0x00
TTL=128 ID=24429 DF PROTO=TCP SPT=7963 DPT=80
WINDOW=16425 RES=0x00 ACK PSH URGP=0
```

```
Sep 4 16:59:45 holisticinfosec01 kernel: [29567.666562] [234]
SID2003099 ESTAB IN=eth0 OUT= MAC=00:0c:6e:3c:
b4:71:00:13:e8:e8:81:37:08:00 SRC=192.168.248.101
```

```
DST=192.168.248.104 LEN=40 TOS=0x00 PREC=0x00
TTL=128 ID=24638 DF PROTO=TCP SPT=7987 DPT=80
WINDOW=16126 RES=0x00 ACK URGP=0
```

The associated Snort rules that fwsnort used to generate the log entries above are:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS
$HTTP_PORTS (msg:"WEB-MISC /etc/passwd"; flow:
to_server,established; content: "/etc/passwd"; nocase; classtype:
attempted-recon; sid:1122; rev:5;)
```

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS
$HTTP_PORTS (msg:"ET WEB-MISC Poison Null Byte";
flow:established,to_server; uricontent:"|00|"; depth:400;
reference:cve,2006-4542; reference:cve,2006-4458; refer-
ence:cve,2006-3602; reference:url,www.security-assessment.
com/Whitepapers/0x00_vs_ASP_File_Uploads.pdf; classtype:
web-application-activity; sid:2003099; rev:3;)
```

To test further I then added a TYPOLight installation to my server and hit it with SQLi, XSS, and CSRF attacks. This effort resulted in log entries matching content from my attacks.

```
Sep 4 18:03:59 holisticinfosec01 kernel: [33417.152209]
[438] SID882 ESTAB IN=eth0 OUT= MAC=00:0c:6e:3c:
b4:71:00:13:e8:e8:81:37:08:00 SRC=192.168.248.101
DST=192.168.248.104 LEN=624 TOS=0x00 PREC=0x00
TTL=128 ID=10412 DF PROTO=TCP SPT=10173 DPT=80
WINDOW=16425 RES=0x00 ACK PSH URGP=0
```

```
Sep 4 18:03:18 holisticinfosec01 kernel: [33375.938525]
[323] SID1334 ESTAB IN=eth0 OUT= MAC=00:0c:6e:3c:
b4:71:00:13:e8:e8:81:37:08:00 SRC=192.168.248.101
DST=192.168.248.104 LEN=1062 TOS=0x00 PREC=0x00
TTL=128 ID=10175 DF PROTO=TCP SPT=10146 DPT=80
WINDOW=16227 RES=0x00 ACK PSH URGP=0
```

The rules associated with these log entries are:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS
$HTTP_PORTS (msg:"WEB-CGI calendar access"; flow:
to_server,established; uricontent: "/calendar";
nocase; classtype:attempted-recon; sid:882;
rev:5;)
```

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS
$HTTP_PORTS (msg:"WEB-ATTACKS echo command
attempt"; flow:to_server,established; content: "/
bin/echo"; nocase; classtype:web-application-
attack; sid:1334; rev:5;)
```

The example above indicate fwsnort's obvious detective capabilities, but once you satisfied with the results of your tuning and optimization, you can easily turn fwsnort into an intrusion prevention system as well. Remember, by default fwsnort only logs malicious packets, but utilizing the --ipt-drop and --ipt-reject command-line options will change the game entirely. The best example of this configuration in use can be seen in a recent blog post from Michael discussing the detection and rejection of remote file include (RFI) attack attempts against his web server. Read the post for yourself, but here's the IPS-related details that sends that bad guys packing (color coded for easy logic flow).

- Bad guy looks for `s_loadenv.inc.php`
- The newly created `s_loadenv.rules` include this rule:

```
alert tcp $EXTERNAL_NET any ->
$HTTP_SERVERS $HTTP_PORTS (msg:"s_load-
env.inc.php DOCUMENT_ROOT attempt"; flow:
established,to_server; uricontent:"/s_load-
env.inc.php?"; uricontent:"DOCUMENT_ROOT=";
uricontent:"http://" classtype:web-applica-
tion-attack; reference:url,www.cipherdyne.org/
blog/2008/09/01.html; sid:20080001; rev:1;)
```
- Issuing fwsnort with the above ruleset enabled and an `--ipt-drop` set for ye olde black hole does the trick:

```
fwsnort --include-type s_loadenv --ipt-
drop > /dev/null
```

Good stuff, yes. I see this attack in my production web server logs all the time too...hmm, what should I do? ;-)

Benefits and drawbacks

You should have a reasonable understanding of iptables and Snort, but if comfortable here, there are no drawbacks, only the benefit of watching the bad guy try and fail if you config-

ure accordingly. I've added fwsnort to all my Ubuntu servers, both test and production, and am benefiting from the enhancement to my system's security posture.

In conclusion

You'll enjoy experimenting with fwsnort, psad too (likely topic for a future toolsmith), and learn a great deal from Michael's writings and development efforts. Inspect, drop, repeat. Inspect, drop, repeat. Go forth and snort mightily! Cheers...until next month.

Acknowledgments

Michael Rash, for his significant contributions to this article, and for his contributions to the greater community via cipherdyne.org.

About the Author

Russ McRee, GCIH, GCFA, CISSP, is a security analyst working in the Seattle area. As an advocate of a holistic approach to information security, Russ' website is holisticinfosec.org. Contact him at russ@holisticinfosec.org.