Deployment of Snort IDS in SIP based VoIP environments

Jiří Markl, Jaroslav Dočkal

Jaroslav.Dockal@unob.cz

K-209 Univerzita obrany
Kounicova 65, 612 00 Brno
Czech Republic

Abstract

This paper describes deployment of Snort IDS for detection of attacks on VoIP infrastructure based on the SIP protocol. The paper provides a description of a proposed set of rules for the detection of various DoS attacks.

Keywords: VoIP, SIP, Snort, DoS, detection.

1 Introduction

Today's rapidly growing internet telephony networks aspire to replace legacy PSTN lines. There are a great deal of evident advantages of VoIP however this technology brings some new problems as well. One of the major problems is the need to achieve at least the same level of availability as in PSTN. Availability of VoIP service can be treated among others by malicious attacks. This paper describes techniques for detection of Denial of Service attacks on SIP based VoIP infrastructures. SIP protocol has been chosen because it seems to be the standard VoIP protocol of the future.

2 Snort IDS

Snort is a network-based intrusion detection system distributed under the GPL license. It performs real-time analysis of packets on networks and triggers alarms when specified conditions are met. Snort uses a flexible set of rules for the detection of malicious attacks on guarded networks. Snort also provides special modules called preprocessors used for more complex analysis. Snort can be used to monitor SIP networks for intrusion attempts with the use of rules for checking of VoIP specific traffic.

3 Snort configuration for SIP networks

This chapter contains information about the setup of preprocessors and about the deployment of some rules for the detection of malicious attacks against different components of SIP networks.

3.1 Portscan detections

In the worse case an attacker can get access to some of our servers (SIP proxy, STUN, RTP proxy) and take full control on its functionality. There are different ways how the attacker can get access to our system. Attackers often misuse known vulnerabilities in operating systems or applications. In most cases such attacks can begin by scanning opened ports on our server. Thus, it could be useful if we can get alerts about such activity. On the other hand this can result in a high number of alarms (because anybody can launch a portscan anytime) and thus the analyst can be flooded by an inordinate number of alerts. So we need to take in consideration all these aspects and make the decision if enabling portscan detection fits our needs. Snort IDS can be configured to make alerts when some portscans directed against particular servers are detected. Snort IDS provides several types of preprocessors intended for portscan detection.

Available portscan preprocessors:

- Portscan Detector – Basic Snort portscan detector.
- sfPortscan - Designed to be able to detect different types of scans which the Nmap utility can produce.
- Flow-Portscan - The goal is to catch one-to-many hosts and one-to-many ports scans.

Example of configuration of Portscan Detector:

```
preprocessor sfportscan: proto { all } \
scan_type { all } \
sense_level { low } \
ignore_scanners { 123.123.123.123 }
```

3.2 Detection of known exploits

Today we know of different security bugs in different software. These bugs can be used as weak points of our architecture and an attacker can launch an attack to exploit these bugs. For various reasons it is often usual that the signature for the detection of an attack exploiting a known bug is available earlier than the patch for the correction of the bug. Thus it can be useful to use available signatures for the detection of attacks against different parts of our SIP infrastructure.

3.3 Rules for the detection of common TCP/IP attacks

Common TCP/IP attacks can be launched against different points of the VoIP network. The main place where an attacker can direct his attack is the SIP proxy. Also contributing services like STUN, RTP proxy etc. are places where these attacks can be directed. The detection of these attacks can be done via set of Snort rules. Some proposed rules for detection of such attacks are depicted in this chapter.

3.3.1 TCP SYN flood attack

TCP SYN flood attacks are based on the exhaustion of the possible amount of open TCP connections on the SIP proxy. We can use the simple threshold rule for detection of unusually high counts of TCP SYN packets arriving to our SIP proxy. We need to set up the threshold high enough so that the number of false positives will be minimal (alerts mustn't be generated by normal traffic or by usual traffic peaks). We can utilize two different rules. The first rule is intended for the detection of TCP SYN attacks arriving from a single source (all packets have same source IP address), and the second rule is used for the detection of attacks where packets come from different sources (it can be an distributed attack or an attacker can spoof the of source IP addresses). The threshold for the second rule must be set much higher than the threshold in the rule for the single source IP address. The next difference between both rules is given by the "track" expression. In the first rule we use tracking by the source address (track by_src) in contrast with second rule where we are using tracking by the destination IP address (track by_dst).

TCP SYN flood detection rule (from single source IP address):

```
alert tcp any any  -> $SIP_PROXY_IP any \
(msg: "TCP SYN packet flooding from single source"; \
threshold: type both, track by_src, count 200, seconds 20; \
flow:stateless; flags:S,12; sid:5000100; rev:1;)
```

TCP SYN flood detection rule (all TCP SYN packets with different source IP address):

```
alert tcp any any  -> $SIP_PROXY_IP any \
(msg: " TCP SYN packet flooding (simple or distributed)"; \
threshold: type both, track by_dst, count 10000, seconds 60; \
flow:stateless; flags:S,12; sid:5000150; rev:1;)
```

3.3.2 Smurf attack

In the Smurf attack scenario an attacker uses spoofed source IP addresses of ICMP echo request (ping) and an auxiliary broadcast network to amplify the attack stream. In this scenario ICMP echo responses (itype: 0) can arrive to our SIP proxy from different sources and thus we use a threshold rule with destination tracking (track by_dst).

```
alert icmp any any  -> $SIP_PROXY_IP any \
msg: "Smurf attack directed against SIP proxy"; \
itype: 0; \
threshold: type both, track by_dst, count 1000, seconds 60; \
sid:5000200; rev:1;)
```

3.3.3 Available rules for the detection of various TCP/IP attacks

Some rules available for the detection of common TCP/IP attacks are provided in Table 1. These rules are part of Snort's basic rule set. This table has been created on the basis of information obtained from Snort's web pages.

SID	Name	CVE ID
241	DDOS shaft synflood	2000-0138
268	DOS Jolt attack	1999-0345
269	DOS Land attack	1999-0016
270	DOS Teardrop attack	1999-0015
271	DOS UDP echo+chargen bomb	1999-0635, 1999-0103
275	DOS NAPTHA	2000-1039
529	NETBIOS DOS RFPoison	
1257	DOS Winnuke attack	1999-0153
1545	DOS Cisco attempt	
2486	DOS ISAKMP invalid identification payload attempt	2004-0184

Table 1: Available rules for detection of common TCP/IP attacks.

3.3.4 Available rules for the detection of intrusion on DNS servers

Some rules available for the detection of intrusions directed against DNS servers are provided in Table 2. These rules are also the part of Snort's basic rule set. They are intended for securing of DNS servers against known DNS exploits and attacks. This table has been created on the basis of information obtained from Snort's web pages.

SID	Name	CVE ID
252	DNS named iquery attempt	1999-0009
253	DNS SPOOF query response PTR with TTL of 1 min. and no authority	
254	DNS SPOOF query response with TTL of 1 min. and no authority	
255	DNS zone transfer TCP	1999-0532
256	DNS named authors attempt	
257	DNS named version attempt	
258	DNS EXPLOIT named 8.2->8.2.1	1999-0833
259	DNS EXPLOIT named overflow ADM	1999-0833
260	DNS EXPLOIT named overflow ADMROCKS	1999-0833
261	DNS EXPLOIT named overflow attempt	
262	DNS EXPLOIT x86 Linux overflow attempt	
264	DNS EXPLOIT x86 Linux overflow attempt	
265	DNS EXPLOIT x86 Linux overflow attempt ADMv2	
266	DNS EXPLOIT x86 FreeBSD overflow attempt	
267	DNS EXPLOIT sparc overflow attempt	
303	DNS EXPLOIT named tsig overflow attempt	2001-0010
314	DNS EXPLOIT named tsig overflow attempt	2001-0010
1261	EXPLOIT AIX pdnsd overflow	1999-0745
1435	DNS named authors attempt	
1616	DNS named version attempt	
1948	DNS zone transfer UDP	1999-0532
2921	DNS UDP inverse query	2001-0010
2922	DNS TCP inverse query	2001-0010
3153	DNS TCP inverse query overflow	1999-0009
3154	DNS UDP inverse query overflow	1999-0009

Table 2: Available rules for detection of intrusions on DNS.

3.3.5 Detection of attacks on STUN servers and RTP proxies

For the detection of attacks on STUN servers and RTP proxies, rules for the detection of common TCP/IP attacks can be used as has been proposed in chapter 3.3. Also, the detection of portscans as proposed in 3.1 and the detection of known exploits as proposed in 3.2 can be used. Further, we can use the detection of brute force flooding of STUN servers and RTP proxies.

3.4 Specific rules for the detection of attacks on SIP proxies

Here will be provided some rules for the detection of attacks specific for the SIP protocol. Most of the depicted rules are based on the detection of unusually high traffic of specific types. For such detection, Snort's threshold rules are used. The setting of threshold values must be done individually for every deployment and it must be based on knowledge of usual SIP traffic characteristics for a given network. The main drawback is that we can do the detection only in packets from unsecured sessions (no SSL or TLS can be used).

3.4.1 INVITE flooding

The INVITE flooding rule is designed for the detection of unusually high incoming traffic of INVITE messages from a particular IP address. This can be caused by an attacker's attempt to realize DoS or by bad configured or badly implemented UA.

The main drawbacks of this rule are that it cannot be used for the detection of distributed DoS or for detection of an attack where the source addresses of incoming packets are spoofed. The solution can be similar as in the case of TCP SYN flood detection (see 3.3.1), where we used second rule (with tracking by destination IP address) for the detection of floods where incoming packets have different source IP addresses.

For partial reduction of false positives, white-listing of known SIP proxies is used. From these devices a great number of INVITE messages can arrive and thus it can be a potential place where false positives can occur without the use of white-listing. A higher amount of INVITE messages can also arrive from IP addresses of NAT devices used in particular networks. In the case of a known large network using NAT with a higher amount of SIP traffic to our proxy, we can also use white-listing to decrease the number of false positives. The depicted white-listing solution can have a drawback in the case when an attacker spoofed the source addresses of packets in such a way that they seem to come from some IP address on the white list (from an IP address of a known SIP proxy or network using NAT), because in this scenario no alert will be launched. White-listing is provided by Snort's suppress rules.

Example of a rule for alerting of INVITE flood attacks:

```
alert ip any any -> $SIP_PROXY_IP $SIP_PROXY_PORTS \
(msg:"INVITE message flooding"; content:"INVITE"; depth:6; \
threshold: type both, track by_src, count 200, seconds 60; \
sid:1000100; rev:1;)
```

```
#Suppresion of alerting for known proxy 147.32.121.12
suppress gen_id 1, sig_id 1000100, track by_src, ip 147.32.121.12
```

3.4.2 REGISTER flooding

The rule for the detection of REGISTER flooding is quite similar to the INVITE flooding detection rule. The only difference is in using REGISTER messages for flooding instead of INVITE. White-listing can also be used to decrease false positives especially in the case when packets are arriving from behind a NAT. The number of REGISTER messages from other SIP proxies will probably be very low. Only in some cases, for example if we are using server based replication of user registrations to achieve increased reliability, it is useful to use white-listing.

Example of a rule for alerting of REGISTER flood attacks:

```
alert ip any any -> $SIP_PROXY_IP $SIP_PROXY_PORTS \
(msg:"REGISTER message flooding"; content:"REGISTER"; depth:8; \
threshold: type both, track by_src, count 200, seconds 60; \
sid:1000200; rev:1;)
```

```
#Suppresion of alerting for known network with NAT 147.32.121.12
suppress gen_id 1, sig_id 1000200, track by_src, ip 147.32.121.12
```

3.4.3 Other TCP/UDP flooding

An attacker can also flood our proxy with packets containing senseless content. The increased amount of traffic from single source IP address is also usually detected here. In this case we use higher threshold values than in INVITE and REGISTER flood signatures.

Example of a rule for alerting of common TCP/UDP flood attacks:

```
alert ip any any -> $SIP_PROXY_IP $SIP_PROXY_PORTS \
(msg:"TCP/IP message flooding directed to SIP proxy"; \
threshold: type both , track by_src, count 400, seconds 60; \
sid:1000300; rev:1;)
```

3.4.4 Unresolvable DNS

In the case an attacker tries to exhaust our proxy's resources with an attack using unresolvable DNS names we can guard the number of "No such name responses" from the DNS server and if the number is higher than a given threshold we can generate an alert about some unusual or suspicious activity.

Example of a rule for alerting of Unresolvable DNS attacks:

```
alert udp $DNS_SERVERS 53 -> $SIP_PROXY_IP any \
msg:"DNS No such name treshold"; \
content:"|83|"; offset:3; depth:1; \
threshold: type both , track by_src, count 2000, seconds 60; \
sid:1000400; rev:1;)
```

3.4.5 Unauthorized responses on INVITE/REGISTER messages

A high number of outgoing 401 Unauthorized messages can be a symptom of a brute force authentication attack on the SIP proxy. These responses are generated when a user tries to send REGISTER messages with the wrong credentials to a SIP proxy that requires the authorization of registration requests. Therefore, we can implement the threshold rule for content based lookup for 401 Unauthorized responses. A similar rule can also be employed to detect unauthorized INVITE messages; in this case we need to detect responses where the 407 Proxy Authentication Required string occurs.

Threshold rule to detect 401 Unauthorized responses:

```
alert ip any any -> $SIP_PROXY_IP $SIP_PROXY_PORTS \
(msg:"INVITE message flooding"; \
content:"SIP/2.0 401 Unauthorized"; depth:24; \
threshold: type both, track by_src, count 100, seconds 60; \
sid:1000600; rev:1;)
```

3.4.6 SQL injection detection rules

Here, some non-threshold and content-detection based rules for the detection of SQL injection attacks will be provided. We use content detection rules for detection of SQL statements like DROP, DELETE, SELECT, INSERT, UPDATE, UNION etc. Another possible detection is the detection of the injection of expressions like, for example 1'or'1'='1, which are always evaluated as true.

Detection of always true expressions injection:

```
alert ip any any -> $SIP_PROXY_IP $SIP_PROXY_PORTS \
(msg:"SQL Injection – Injection of always true expression"; \
pcre:"/\w*\'or/ix"; \
sid: 1000500; rev:1;)
```

Detection of SQL statements:

```
#DROP statement injection:
alert ip any any -> $SIP_PROXY_IP $SIP_PROXY_PORTS \
(msg:"SQL Injection – Injection of DROP statement"; \
pcre:"/\'drop/ix"; \
sid: 1000510; rev:1;)
```

```
#DELETE statement injection:
alert ip any any -> $SIP_PROXY_IP $SIP_PROXY_PORTS \
(msg:"SQL Injection – Injection of DROP statement"; \
pcre:"/\'delete/ix"; \
sid: 1000520; rev:1;)
```

```
#SELECT statement injection:
alert ip any any -> $SIP_PROXY_IP $SIP_PROXY_PORTS \
(msg:"SQL Injection – Injection of DROP statement"; \
pcre:"/\'select/ix"; \
sid: 1000530; rev:1;)
```

```
#INSERT statement injection:
alert ip any any -> $SIP_PROXY_IP $SIP_PROXY_PORTS \
(msg:"SQL Injection – Injection of DROP statement"; \
pcre:"/\'insert/ix"; \
sid: 1000530; rev:1;)
```

```
#UPDATE statement injection:
alert ip any any -> $SIP_PROXY_IP $SIP_PROXY_PORTS \
(msg:"SQL Injection – Injection of DROP statement"; \
pcre:"/\'update/ix"; \
sid: 1000540; rev:1;)
```

```
#UNION statement injection:
alert ip any any -> $SIP_PROXY_IP $SIP_PROXY_PORTS \
(msg:"SQL Injection – Injection of DROP statement"; \
pcre:"/\'union/ix"; \
sid: 1000550; rev:1;)
```

4 Conclusion

Snort IDS can be relatively easily configured with sets of rules for detection of unusual and suspicious activity in the VoIP network based on the SIP protocol and thus can be used for the detection of attack on major SIP network components, especially on SIP proxies. Unfortunately, Snort rules provide only limited facilities for attack signature specification, so for more sophisticated detection, development of Snort preprocessors seems to be needed. Furthermore, there are other complications in the usage of TLS when Snort is not able to do packet payload inspection.

Experiences from real life application shows that proposed set of rules can also be used for the detection of various misconfigurations in SIP networks and thus help defend parts of a SIP network against unwanted and accidental overloads or even outages as well.

References

[1] SNOCER project team: General Reliability and Security Framework for VoIP Infrastructures.