SECInfy

SNORT – First Line of Defense for Web application attacks

Introduction

As described in SNORT official web site, "*SNORT is an open source network intrusion prevention and detection system utilizing a rule-driven language, which combines the benefits of signature, protocol and anomaly based inspection methods*".
SNORT is most widely used open source IDS till date. Many UTMs uses SNORT to provide services to their clients. SNORT has introduced inline mode which can be used to drop packets. Thus using inline mode, SNORT can be used as firewall as well. Let us go over all modes in which SNORT can be run

SNORT Modes

SNORT can be configured to run in one of the following modes

- **Sniffer mode** – If SNORT is ran in sniffer mode, it captures all the packets and shows it to screen. By redirecting screen output, it is possible to capture all packets in file. This mode is widely used to troubleshoot network problems.
- **Packet Logger mode**: This mode is similar to sniffer mode, In stead of showing packets to screen, SNORT logs all the packets to log file if running in Packet logger mode.
- **Network Intrusion Detection System (NIDS) mode** – NDIS is most widely used mode for SNORT. When SNORT is running in NDIS mode, It allows to write rules (using regex). SNORT checks each packet with matching expression and performs predefined operation.
- **Inline mode** – This is most important mode for writing any firewall rule. When SNORT is running in inline mode, it captures packets, analyze it and drop/pass packet depending on rule. It uses iptables to drop packets.

Typical SNORT Rule

Any typical SNORT rule can be divided in two different parts – Rule Header and Rule Options

Rule Header

Rule header consists of following variables

- Rule Action - This variable indicates what operation needs to perform when matching packet has arrived. Possible values are drop, pass, reject
    - Drop – When specified, SNORT drops packet and does not perform any action. It does not send reset packet back to client
    - Pass – When specified, SNORT passes packet to its destination
    - Reject – When specified, SNORT drops packet and sends reset packet back to client.
- Protocol – This variable indicates which protocol SNORT has to look for. Possible values are TCP and UDP.
- IP Address – It is possible to block requests from certain IP addresses only. SNORT will analyze only packets from specified IP in this variable. If value "any" is specified in this variable, SNORT will analyze all the packets for the rule
- Port– It is possible to block requests for certain Port only. SNORT will analyze only packets for specified Port in this variable. If value "any" is specified in this variable, SNORT will analyze all the packets.
- Direction operator – It is possible to configure SNORT to check only either inbound traffic or inbound/outbound traffic. To specify this, there are two different operators which need to use. "->" can be used to specify all inbound traffic (Requests), **"<-"** can be used to specify inbound/outbound traffic.

Rule Options

Writing rule options are equally important as writing rule headers. Following are some key information about rule options.

- All SNORT rule options are separated from each other using ";"
- Rules argument are supplied using ":"
- Word "PCRE" is used to specify support for PERL style regex
- In case of web application rules, when PCRE is used without a "uricontent", SNORT evaluates the first URI only thus it is advisable to use either content or uricontent to search pattern in all URI. (Chances of bypassing is possible if not used)

Rule for Web Application attacks

With the above knowledge, we are good to start with application rules. It is important to understand that all attacks can not be genetic in case of applications. In case of application vulnerability, vulnerability is different in each application. Writing SNORT rules and stopping attacks at network level can not be full proof solution but it can be one more line of defense for the attacker like web application firewall.

SECInfy

Sample Rules

We have enough background about SNORT and it is capabilities. Let us see some examples to understand how SNORT can be used to protect against web applications.

Example - Blocking Special characters for SQL Injection

```
drop TCP any any -> 10.18.19.201 80 (flow:to_server; content:!"GET"; nocase;
pcre="[\'\"\;\:\|\&\$\%\@\\\/<>()+,]")
```

Above rule will

- **Drop** all request matching criteria
- Performs analysis for **TCP** packets
- From **any** Source IP
- From **any** source Port
- Destination IP is **10.18.19.201**
- Destination Port is **80** (HTTP)
- Content should have value "**GET** " – This rule will only work on GET requests, similarly rule can be written for POST request as well.
- Word "**nocase**" specifies matching is not case sensitive
- Regular expression mentioned in "**pcre="[\'\"\;\:\|\&\$\%\@\\\/<>()+,]**" will not allow ' " ; : | & @ \ / < > ( ) + , characters in request.

Thus above rule will drop all packets which has GET request and ' " ; : | & @ \ / < > ( ) + , characters in URL.

Example - Blocking Special words used for SQL Injection – insert, update, delete, select or xp_cmdshell

```
drop TCP any any -> 10.18.19.201 80 (flow:to_server; content:!"GET"; nocase;
pcre="\(insert|update|delete|select|xp_cmdshell\)")
```

Above rule will

- **Drop** all request matching criteria
- Performs analysis for **TCP** packets
- From **any** Source IP
- From **any** source Port
- Destination IP is **10.18.19.201**
- Destination Port is **80** (HTTP)
- Content should have value "**GET** " – This rule will only work on GET requests, similarly rule can be written for POST request as well.
- Word "**nocase**" specifies matching is not case sensitive
- Regular expression mentioned in "**pcre="\(insert|update|delete|select|xp_cmdshell\)**" will not allow insert, update, delete, select and xp_cmdshell words in request.

Thus above rule will drop all packets which has GET request and "insert, update, delete, select or xp_cmdshell" characters in URL.

Example - Blocking Special characters used for SQL Injection in particular field

Till now, in both previous examples, we saw generic rules to stop SQL Injection attacks. It is not possible to stop any application attack just by generic rules. It is important to have application specific, page specific and field specific rules. Also, in some cases, application may need to

accept some of special characters in one or other fields. In such cases, it is important to write page specific rules. In example below, rule checks page name and field name to validate special characters.

```
drop TCP any any -> 10.18.19.201 80 (flow:to_server; content:"POST"; content:"Login.aspx"; nocase; pcre:".*username=.*[\'\"\;\:\|\&\$\%\@\\\/<>()+,]")
```

Above rule will

- **Drop** all request matching criteria
- Performs analysis for **TCP** packets
- From **any** Source IP
- From **any** source Port
- Destination IP is **10.18.19.201**
- Destination Port is **80** (HTTP)
- Content should have value "**POST**" – This rule will only work on POST requests, similarly rule can be written for GET request as well.
- Content should have value "**Login.aspx**" – This is page name. Also note here that for multiple ANDING in content matching, multiple "content" is used.
- Word "**nocase**" specifies matching is not case sensitive
- Regular expression mentioned in "**pcre:".*username=.*[\'\"\;\:\|\&\$\%\@\\\/<>()+,]**" will not allow ' " ; : | & @ \ / < > ( ) + , characters in username field.

Thus above rule will drop all packets for Login.aspx page which has POST request and ' " ; : | & @ \ / < > ( ) + , characters in username field.

Limitations of SNORT

As we saw, Generic and application specific rules can be written in SNORT to provide defense for web application. Up to certain extend, SNORT can replace your paid web application firewall. But SNORT has following limitations

- It is not possible to send error page to client as SNORT does not work on HTTP(S).
- It is not possible just to write rule on outbound traffic. Rule can be written either for both (inbound/outbound) or inbound traffic.

Conclusion

As we saw, SNORT can work as web application firewall with few limitations. But it is important to understand that application based vulnerabilities are different in each application and can not be resolved by any generic rule which is possible in network security. Also, there is no alternative of secure coding; adding such generic rules and protecting application is just one more line of defense and it can not be considered as alternative of proper input validation. The best approach for any organization is perform penetration testing for application, write SNORT rules to protect temporary against attacks and start modifying code for proper implementation of security.

Reference:

http://www.SNORT.org/docs/