

Detecting the Unknown with Snort and the Statistical Packet Anomaly Detection Engine (SPADE)

Simon Biles
Computer Security Online Ltd.

Introduction

SPADE is a pre-processor plug-in for the Snort intrusion detection engine. It detects network traffic that deviates from the “normal” behaviour of your network. This paper will introduce SPADE, it’s conceptual working and a simple overview of its installation and configuration.

History

The SPADE pre-processor plug-in to Snort was created by Stuart Staniford and James Hoagland whilst they were working for Silicon Defence, a company funded by the American Defence Advanced Research Projects Agency (DARPA). Sadly, in 2003 DARPA decided to cut funding for unclassified network research, and Silicon Defence was no more. Fortunately for us however, the creators had the foresight to release SPADE and a number of their other projects under the GPL, so it has been free for us to pick up and continue to work with.

Concept

Each network has “normal” traffic that is expected, for example if you are running a web server, your normal traffic is from a large range of external IP addresses to a single address on port 80 TCP. If you are running an internal DNS server, you expect there to be traffic to and from port 53 UDP and TCP from the machines within your network. This can be characterised as “normal” traffic. It would be “not normal” for your internal DNS server to be receiving requests from external IP addresses to port 80 TCP.

An administrator can see instantly that this is wrong, but a signature based IDS such as Snort would be none the wiser if a signature didn’t exist to match. This is where SPADE steps in. SPADE allows your IDS to detect things that it doesn’t have prewritten signatures for – this is the concept of zero-day detection. You can now protect yourself against things that haven’t been discovered yet.

Function

In order to detect abnormal, anomalous packets SPADE maintains probability tables that contain information regarding the number of occurrences of different kinds of packet over time on the network. It assigns a higher weight to more recent occurrences, and gradually phases out older occurrences.

Using the above example these probability tables could tell us that the probability of a packet to the DNS server on port 53 is 10% ($P(\text{dest_ip}=\text{DNS_SRV}, \text{dest_port}=53) = 10\%$) where as the probability of a packet to the DNS server on port 80 is 0.1% ($P(\text{dest_ip}=\text{DNS_SRV}, \text{dest_port}=80)=0.1\%$). Again from this information it is clear that there is something uncommon about the second packet, but SPADE doesn't stop here. First this probability is converted first to a "raw anomaly score" and then into a "relative anomaly score" – this leaves us with a number between 0 and 1 for easy comparison.

The calculations are quite straightforward, given a probability $P(X)$ for a packet X the "raw anomaly score" $a(X)$ is equal to $-\log_2(P(X))$. To get to the "relative anomaly score", $a(X)$ is then divided by the maximum possible "raw anomaly score" to leave us with $A(X)$ being between 0 and 1, with 0 being completely "normal" and 1 being completely "not normal".

When we are setting up the SPADE sensor we specify an alerting threshold, so a packet X giving an $A(X)$ over a certain value will trigger an alert. Sending a standard Snort alert through the normal Snort alerting mechanisms.

SPADE can't however tell you if the packet is an attack or simply a misconfiguration. It has no concept of the intention of the packet, simply if it is within normal network behaviour or not. It also doesn't (yet – give us time !) examine the packet contents or flags, so it won't pick up someone trying out a new attack against a commonly used service on your network such as a web or mail server if that attack relies on exploiting the normal port.

Just as a parting shot though – you can make use of the tables that SPADE keeps to examine what sort of traffic is normal on your network and see just where your bandwidth goes.

Set Up

SPADE was originally released as a patch to the Snort source tree. One would download the Snort source, unpack it, download the SPADE patch, and apply it. At this point in time, the patches haven't been rewritten for the latest version of Snort, but you can download an integrated tar of the entire source from the SPADE project repository on BleedingSnort.com. Installing Snort with SPADE applied is exactly the same as installing Snort, in the source directory run the *configure* script, *make* and *make install*.

Then setup Snort with signatures as you would normally, for more information on this see the Snort documentation or refer to a good book on the subject – I would thoroughly recommend the "Snort Cookbook" from O'Reilly – there is a section on SPADE in there that I also wrote along with a lot of other good advice on setting up Snort to get the best from it.

To activate SPADE within Snort, you need to insert into your main Snort configuration file (usually snort.conf) the pre-processor option line for SPADE as follows:

```
preprocessor spade: { <optionname>=<value> }
```

You can set any number of the possible options on this line separated by spaces. Possible options are detailed in the following table.

Option	Meaning
logfile	This specifies the logging file for SPADE. This file is regenerated on each reload and restart of Snort, and contains data which may be useful in tuning the detectors.
statefile	SPADE maintains probability tables, this specifies the file that these tables are written to for recovering the probability tables on process restart. (Defaults to "spade.rcv")
cpfreq	This specifies the frequency that the state table file set above is updated with the current state. This is done every N updates to the state. (Defaults to 50000)
dest	This allows you to specify the location that SPADE messages are sent to. The possible values here are : "alert", "log" or "both". "Alert" sends messages to the Snort alert facilities that have been configured, "Log" to the configured Snort logfiles. "Both" – to both.
adjdest	This configures SPADE to output "Threshold Adjusted" messages to a different destination to other messages. It takes the three options above, and also a "none" option which causes SPADE to be silent about threshold adjustment.

An example of the above options in use to configure SPADE with the /var/log/spade/spade.log log file, the /var/log/spade/state.rcv state file, saving state every 25000 updates, sending messages to the Snort alert facility except for threshold adjustments which are ignored would look like this :

```
preprocessor spade: logfile=/var/log/spade/spade.log
statefile=/var/log/spade/state.rcv cpfreq=25000 dest=alert
adjdest=none
```

SPADE will also function better if you let it know what it should expect to be inside it's "own" network. This is done using the following line :

```
preprocessor spade-homenet: <network> <network> ...
```

Where <network> is replaced with a network in CIDR notation, a single IP address or "any" for everything – it will default to "any" if nothing else is specified. For example :

```
preprocessor spade-homenet: 192.168.0.0/16 172.17.10.74
```

(N.B. This is independent to the Snort HOMENET variable).

You now need to set up some detectors for SPADE to work with. Detectors are set up with the following line, and the options detailed in the table below.

```
preprocessor spade-detect: { <optionname>=<value> }
```

You can use any combination of the following options. Where required they are covered in more detail in following tables.

Option	Action
type	This indicates the detector type. You can choose from closed-dport , dead-dest , odd_dport or odd-typecode .
to	Sets the direction of traffic home is traffic with destinations in the earlier specified homenet, nothome is everything else, and any is both directions.
from	from is the same as to except for the source rather than the destination.
proto	Specifies which protocol the detector is for, can be "tcp", "udp" or "icmp".
tcpflags	Specifies flags that are set for TCP packets. Possible values are "synonly", "synack", "setup", "established", "teardown" or "weird".
icmptype	Specifies the type of ICMP packet to look for, can be "err", "noterr" or "all".

thresh	This is the initial threshold for packets to be reported based upon their anomaly score.
minobs	This is Minimum Observations: how many packets need to be observed before alerts are sent. This covers the start up of the system, when all packets look like anomalies.
wait	The number of seconds that a message is held in the waiting queue before timing out.
Xdips	Exclude reports from this detector about certain destination IP addresses.
Xdports	Exclude reports from this detector about certain destination ports.
Xsips	Exclude reports from this detector about certain source addresses.
Xsports	Exclude reports from this detector about certain source ports.
id	A label for the detector, must start with a letter, and can contain only alphanumeric characters, “-“ and “ ” —
revwaitrpt	If response waiting is enabled for this detector, this causes the conditions for the detection to be reversed.
scalefreq	This is how often in minutes that the existing observations are decayed in favor of newer observations.
scalefactor	This is the relative weight that should be given to old data at each “scalefreq” reweighing.
scalehalflife	This option helps to attain a certain half life for the weight of an item of traffic. It can be created through “scalefreq” and “scalefactor”, but is easier to specify as an exact time in seconds.

scalecutoff	This is the point below which an item of traffic will be removed from the active dataset.
-------------	---

Each of the detector types makes use of the other options in slightly different ways some are inappropriate for use with a specific detector type.

closed-dport

This detector type looks at TCP and UDP traffic for attempts to connect to closed ports. This is common behavior for port scanners, which attempt to connect to all ports to determine what is open. There is an option to wait for the rejection of the packet before issuing an alert to see if the port was open or not, which removes alerts caused through the use of passive FTP. This will create one of three types of alert. Without the response wait option enabled it gives “Rare dest port used”. If response waiting is enabled and a RST or ICMP unreachable response is sent then it gives “Closed dest port used”. Finally, if response waiting is enabled and the port is open it gives “Rare but open dest port used”. The normal options are as follows.

Option	Action
to,id	As normal
protocol	“tcp” or “udp” only.
tcpflags	“synonly”, “synack”, “established”, “teardown” and “weird” available.
wait	How long to wait for the response packet.
revwaitrpt	Wait for response.

dead-dest

This detector type scans for traffic that is being sent to IP addresses that are not in use. This will detect the typical behavior of network scanners and worms that are unaware of the internal layout of your network. The alert given is “Non-live dest used”. The normal options are as follows.

Option	Action
to,id	As normal
tcpflags	“synonly”, “synack”, “established”, “teardown” and “weird” available.
icmptype	As normal

odd-dport

This detector type looks for use of ports which differ from the normal usage patterns. This is often a symptom of a compromised host running something new. This

can be applied to local or remote sources, and is reported with an alert of “Source used odd dest port”. The normal options are as follows.

Option	Action
from, id	As normal
protocol	“tcp” or “udp” only.
wait	How long to wait for a response packet if you specify “revwaitrpt”.
revwaitrpt	Wait for a response before alerting.

odd-port-dest

This detector looks for anomalous behavior in the way of connections being made to normal ports on unusual machines. For example if e-mail usually goes to a specific host and this changes suddenly, it may be that the host has been compromised in some way. The alert given is “Source used odd dest for port”. The normal options are as follows.

Option	Action
from, id	As normal
protocol	“tcp” and “udp” only.
maxentropy	This is a measure of the variation that should be expected for the destination IP of a given port. 0 indicates that there should be only one port expected, and increasingly higher numbers indicate an increasingly higher variation.

odd-typecode

This detector reports odd ICMP packets on the network. The alert given is “Odd ICMP type/code found”. The normal options are as follows.

Option	Action
to, id	As normal
icmptype	As normal – defaults to “any”

All of the above are covered again in the Usage.Spade file included in the distribution. I would recommend a read of this before installing, as this will be the first place that any changes are detailed in.

There are four facilities to help you to find a threshold that works for you so that you don’t get flooded by alerts about innocuous packets, but so that you aren’t missing out on things that you should be seeing. The first three of these facilities are called

threshold adapting. Method 1 is the simplest, it periodically takes a weighted average of the current threshold and the recently observed ideal. To activate it you add the following line to the Snort configuration file :

```
preprocessor spade-adapt: {<optionname>=<value>}
```

Where <optionname> is one of “id”, “target”, “obsper”, “newweight” or “bycount”. The “id” option is required and is the ID of the detector whose threshold you are adapting. The target packet count is given by the “target” argument, the length of time during which that count is aimed for and the refresh rate of the threshold is given by “obsper”, “newweight” is the weight given to the new component of the weighted average. Setting the “bycount” variable to 1 will use a packet count, rather than a period of time to determine the new threshold.

Method 2 is more involved, it takes into account the averages on short, medium and long term components. To include it, add the following line to the Snort configuration file :

```
preprocessor spade-adapt2: {<optionname>=<value>}
```

The options for this method are : “id” (again mandatory), “target”, “obsper”, “NS”, “NM” or “NL”. “Target” is the specification of the number of alerts wanted. If it is greater than or equal to 1, it is an hourly alert rate. If it is less than 1, then it is a fraction of considered packets to report, based on the best estimate of your packet rate. The latter might be better if your network might experience a shift in packet rates. “obsper” is the number of minutes in an observation window. This is the frequency with which the threshold is updated. “NS” times “obsper” determines the length of the short term, “NM” is the number of short term periods there are in the medium term, and “NL” is the number of medium term periods there are in the long term.

Method 3 is fairly simple in comparison. The reporting threshold is based on an average of the ideal threshold values from the last N observation periods. This mode is invoked with the following line the Snort configuration file :

```
preprocessor spade-adapt3: {<optionname>=<value>}
```

The options for the third method again contain the obligatory “id” and also : “target”, “obsper” or “numper”. “Target” is the specification of the target alert rate and takes the same form as that of Method 2. “Obsper” is the number of minutes in a period of observation, and “numper” is the number of periods of observation to take the average over.

You can use multiple adaptors at the same time, but they must refer to different detectors.

The other method of threshold finding is called “Threshold Advising”. This will report to the log file the threshold that would have been needed to produce the required

number of alerts during a given period of time. You can enable this method by entering the following line in the Snort configuration file :

```
preprocessor spade-threshadvise: {<optionname>=<value>}
```

Apart from the already familiar “id” we also have “target” to specify the number of alerts that we are aiming for in a given period, and “obsper” to specify the length of this period.

Conclusion

Hopefully this has given you a feel for the reasoning behind the SPADE project, and enough to get yourselves setup to add it to your IDS repertoire. If you have any questions you can either direct them to me at simon@computersecurityonline.com or through one of the forums on <http://www.bleedingsnort.com>. We are also looking for help from anyone who would be interested in taking part, or if you would like to make a donation to help with future development. We are also very interested to hear from people who have implemented SPADE in their environment, how they are getting on with it, and any changes that they would like to see.