FEATURE 2

MALWARE IN A PIG PEN – PART 1

Martin Overton
Independent Researcher, UK

PIG TALES

It is often stated that pigs are very intelligent animals – more so than dogs and cats. In fact, pigs are considered to be the fourth most intelligent animals (excluding humans) on the planet [1]. Only chimpanzees, dolphins and elephants (in that order) rate higher in the intelligence stakes.

This article will discuss the use of the SNORT Intrusion Detection System (IDS) with a twist – using it to detect malware by employing various signature creation techniques.

For the uninitiated, SNORT is an IDS that works on *Windows* and UNIX systems and is free (apart from the hardware and manpower costs), very flexible and widely used and respected.

WHY USE AN IDS TO CATCH MALWARE?

Why use an IDS to catch malware? Well, why not? Currently I'm using Bayesian filtering to catch malware too – with great success. I am not saying that virus scanners are useless, or that the use of an IDS is a better method. My reasons for using an IDS to catch malware are as follows:

- Fast-moving threats require quick (and sometimes 'dirty') detection methods.
- Most malware threats that cause problems are network-borne, and therefore an IDS is a suitable tool.
- Many of the signatures I use are created *before* some (if not all) AV companies have detection capabilities for a new breaking threat.
- I am a great believer in defence-in-depth and multi-layered defences against malware. Using an IDS as well as virus-scanning tools offers better overall coverage for a network than merely relying on one or more virus scanners.
- Using an IDS to detect malware propagating across your network means that you have the SOURCE IP address, which will allow faster resolution and clean-up. This is particularly important with mass-mailing worms that forge mail headers as well as fast-spreading/attacking network worms such as Nimda, Slammer, Blaster, etc.

There are other reasons, but in order to list them all this article would need to be several times longer.

FIRST, CATCH YOUR PIG!

You can download, catch and install your pig (SNORT), as well as find more details about its habits, needs, feeding, care instructions and its preferred stabling (most UNIX flavours as well as *Windows*) at http://snort.org/.

Installation on UNIX is very straightforward; even on *Windows* only the additional step of installing PCAP (available at http://winpcap.polito.it/) is required. All other parts, such as reporting, database storage of alerts and management tools, are optional extras.

"These rules are going away. We don't care about virus rules any more."

THIS LITTLE PIGGY CAUGHT MALWARE

When I was first introduced to SNORT I was intrigued to find that, from the early days, SNORT was supplied with a set of virus detection rules (a signature/rule file known as 'virus.rules') which were useful – however, some time ago the following was placed in the virus.rules file: "These rules are going away. We don't care about virus rules any more."

So, many security specialists who used SNORT because it could be used to detect viruses felt somewhat aggrieved that their favourite animal could no longer sniff out new malware. As usual, just as I started to look at using a new tool to detect malware the vendor dropped the very 'feature' that drew me to it in the first place. It seems to be the story of my life!

What could be done to re-educate our pigs? Since no one else (at the time [2]) seemed to be creating rules/signatures for SNORT to detect new malware, I took it upon myself to learn how to create them and make them available to like-minded security professionals.

SNIFFING OUT THE TRUFFLES/MALWARE

So, what can you do to re-train your piggy to sniff out the malware travelling across your networks?

A couple of years ago I decided to try to use SNORT to catch new malware. I installed PCAP and SNORT onto a *Windows* box (I have also installed it on *Linux*) along with the nice front-end known as *IDScenter* from engage security [3], and fired the pig up. Then I installed *ACID* [4], which required a web server (*Microsoft IIS* or *Apache*), *MySQL* [5] and *PHP* [6].

Next came the difficult bit: how to create SNORT signatures, especially for new malware.

EENY, MEENY, MINEY, MO, HOW TO CATCH A MALWARE USING SNORT

Take your freshly caught malware sample, open it in a hex editor (if it is a binary sample) or a text editor (if it is still MIME-encoded in an email). Now examine its entrails and see what they can predict for its future:

"*… I see a teenage nerd, hunched over his/her computer in a room painted entirely in black. Wax-gnarled candle stubs burn fitfully, casting a spectral glow over the proceedings and all the while the teenager is muttering incantations in C/C++, threatening to put some hex on you …*"

A word of caution is needed here: this is not a task for the general end-user population. Only trained and knowledgeable staff who are used to dealing with, working with and handling live malware samples should attempt this – preferably on a dedicated system that is *not* networked, just in case the unthinkable should happen and they should accidentally launch it!

If you cannot justify a dedicated PC, you could use *VMware* instead (remember to disable network support in the virtual machine that you use).

Unless the malware under the knife is polymorphic, pads itself out (with random garbage instructions/code) to fool MD5 hashes, or is encrypted (such as in a password-protected zip file), only a single signature will usually be required to detect it – more will be required if it spreads using other vectors, such as email or peer-to-peer (P2P) networks.

FIRST TAKE YOUR MALWARE ENTRAILS

Let us look at W32/Netsky.p@MM [7] as an example of a typical modern email-based worm that also travels via P2P. How do I create signatures to detect it?

First I find out whether the malware sample is static by hashing (MD5 or SHA1) all the samples I have of it. This may also require me to decode the attachments if they were MIME-encoded when received and perform the same steps on the decoded attachment. If they are zipped, I must unzip them and repeat the steps again on the unzipped, decoded samples.

MIME IF I HAVE A PEEK?

Netsky.p is a static binary image, which means that the MIME-encoded binary image is also static. The next step is to view the sample in a hex editor and select a suitable MIME string to be used to detect the worm when it arrives via email.

A suitable string should be available within the first 30 lines of the MIME-encoded attachment in the email. Try to find a line that is complex, not one that contains mainly 'A's, as otherwise the rule/signature will trigger on perfectly harmless and uninfected email traffic. I usually select at least one full line (72 characters, although sometimes I will use over 100 characters) to ensure that the chances of false positives/negatives are minimised.

Once a likely MIME signature string has been found, this is tested in a simple virus scanner which was created for testing the suitability of SNORT signatures. This scanner – let's call it 'MyScan' – is then run against all captured samples of Netsky.p.

I also test the 'new' signature against earlier members of the same malware family to try to ensure that they will not trigger a false alarm. I also check all the existing signatures in the 'MyScan' database against the new variant – again, this is for false positive testing.

Once the new signature(s) have been tested and any issues ironed out, they are placed into a rule set known as 'malware.rules', which I maintain. These are then made available to other security professionals and researchers, including AVIEN members.

The signature below is one I created and has been very successful. At the end of August 2004 over 3,800 samples have been detected coming to my (personal) mail server from infected hosts.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any
(msg:"W32.NetSky.p@mm - MIME"; content:
"X7soIUEAR4s3r1f/E5UzwK51/f4PdO/+D3UGR/83r+sJ/
g8PhKLw/v9XVf9T"; classtype: misc-activity; rev 1;)
```

Let's break the signature/rule down into its component parts.

The first part is:

```
alert tcp
```

This tells SNORT to send an 'alert' when the signature is matched/triggered – when it sees a 'tcp' packet (you can also test UDP and ICMP traffic too) that contains the signature for the malware.

The next part is:

```
$EXTERNAL_NET any -> $HOME_NET any
```

This specifies that we want SNORT to trigger only when the traffic is coming from an IP address that is *not* one of ours (on any port), but is being sent to one in our IP address range (again, on any port).

$EXTERNAL_NET and $HOME_NET are user-defined variables. You can use the keyword 'any' in place of them to allow the rule to trigger on traffic originating on your network as well as traffic from outside your network address ranges. You can also tie down the detection to specific ports, such as '25' or '110', instead of using the 'any' port keyword.

The next part is:

```
(msg:"W32.NetSky.p@mm - MIME"; content:
```

This tells SNORT to send the alert text "W32.Netsky.p@mm - MIME" to the console, log or database (whatever you use) when the following malware signature (MIME-encoded) is found in the TCP packet:

```
"X7soIUEAR4s3r1f/E5UzwK51/f4PdO/+D3UGR/83r+sJ/
g8PhKLw/v9XVf9T"
```

The final part is:

```
; classtype: misc-activity; rev 1;)
```

This tells SNORT to log the signature match as the 'classtype' of 'misc-activity'. This could be any registered classtype, so you could set the classtype as 'malware' if you prefer. The last part of the signature/rule is the 'rev' statement; this is used to allow revision control so that you can keep track of how many changes you have made to a rule.

BIN HERE BEFORE?

The same steps are followed when rummaging around in a binary sample (EXE, COM, SCR, etc.) looking for a suitable hex signature which can be used to detect the worm as it travels across the network in its P2P (file sharing) mode of operation.

```
alert tcp $EXTERNAL_NET any -> any any
(msg:"W32.NetSky.p@mm - SMB";content:"|4E EB 87 89 77
7E E0 83 B1 94 94 CC E9 F5 97 97 53 95 5C 95 AF C6 40
C5 CA AC 25 8E 47 F1 5D 0B|"; classtype:misc-
activity;rev:1;)
```

A suitable signature is usually at least 32 'hex' characters, each separated by a space. Again, in some cases I will use a longer signature instead.

As you can see, the main difference in the 'content' section of the signature is that hex signatures must also be prefixed and suffixed by the 'l' (broken pipe) character inside the double quotes, whereas MIME signatures are enclosed only in double quotes.

MIME THE BIN

For the more adventurous security professionals out there, you can have multiple 'content' sections within the same rule, and you can even have both binary (HEX) and text (MIME) signatures in the same rule – and lots more besides, but that's another story.

BRINGING HOME THE BACON

The signatures I create are available to all *Virus Bulletin* subscribers (and other selected parties) on my home webserver at: http://arachnid.homeip.net/snort/index.htm. You will need to use the following login credentials (they are case-sensitive):

```
ID = VB2003
Pass = worm!charmer
```

Any assistance is very welcome – either by creating and sharing your own rules or by reporting success or issues with the rules I have created.

This concludes part one of this article as I have covered 'simple' signatures for non-polymorphic or otherwise non-obfuscated malware. The next part of the article will deal with how to write rules/signatures to detect more difficult (obfuscated and encrypted) malware, as well as looking at the use of PCRE (Perl-Compatible Regular Expressions) instead of static binary or MIME strings, in addition to multiple 'content' rules.

For those who are interested, I use my own SNORT rules, as well as running virus scanners, my Worm Charmer, various honeypots, Bayesian filtering, etc. So, you could say that I eat my own pig food.

NOTES AND REFERENCES

[1] Source: http://www.veganpeace.com/AnimalFacts/Pigs.htm.

[2] Now there are several individuals and groups, most notably the maintainers of Bleedingsnort.

[3] http://www.engagesecurity.com/.

[4] http://www.andrew.cmu.edu/user/rdanyliw/snort/snortacid.html.

[5] http://www.mysql.com/.

[6] http://www.php.net/.

[7] First seen 21 March 2004. See http://vil.nai.com/vil/content/v_101119.htm for details.

Photograph of Luther the pig courtesy of Farm Sanctuary, http://www.farmsanctuary.com/.