Anti-Malware Tools: Intrusion Detection Systems

Martin Overton
IBM, UK

About Author

Martin Overton, IBM

Martin currently works for IBM [EMEA Managed Security Services Delivery (MSSD) core team] as a malware/anti-malware specialist, and is part of the Global Virus Emergency Team as well as the World-Wide Threat Team.

He is a regular speaker at the Virus Bulletin International Conferences, and has lost count of the many other presentations he has done and is a regular contributor to the Virus Bulletin periodical.

Martin is a charter member of AVIEN, a WildList reporter, a member of the Anti-Phishing Working Group and a founder member of the UK ISS User Group (UKISSUG).

To date he has accumulated over fourteen years of experience in investigating and combating viruses, Trojans and related malicious software (malware).

His hobbies, when time allows, include reading (mainly science fiction and science/technology/history books), astronomy, keeping a number of bugs (tarantulas and scorpions); and is a member of the British Tarantula Society. If this doesn't mark him as being weird enough, he also likes snakes (owning a Californian Kingsnake). Oh yes, and he does some computer programming. Occasionally his wife and son get to see him!

Contact Details: 51Cook Road. Horsham, West Sussex, RH12 5GJ, England, phone: +44 2392 563442, email: overtonm@uk.ibm.com.

Keywords

IDS, Snort, Signature, Rule, Worm, Virus, Trojan, Malware, Spyware, MySQL, BASE

This paper was written for, and presented at, the 2005 EICAR conference at the Westin Dragonara Hotel,

Malta which was held from April 30th until May 3rd 2005.

I would welcome any constructive feedback on this paper and its content.

Abstract

When most people think of tools to combat malware, very few will give a passing thought to Intrusion Detection Systems, why?

Common reasons include:

- *They don't realise that IDS systems can be used against malware (viruses, Trojans, worms, etc.)*
- *They are too difficult to setup, maintain and use.*
- *That they are too prone to false alarms.*

This paper will investigate the use of IDS systems, specifically to counter/block/detect malware. What's more, this paper will focus on SNORT (which is a free IDS system available for both UNIX and Windows).

This paper will include instructions and guidance on the setup of such a system, numerous examples of suitable rules to detect and block malware and useful tools that can make the sifting of logs easier and more palatable as well as configuration and other tools and utilities that may be useful in managing and maintaining SNORT.

The use of an IDS system can be extremely useful in cases of fast burning or very complex malware outbreaks as a stop-gap until the anti-virus vendors manage to get reliable updates out to their customers.

An IDS is also useful in identifying infected systems in your organization that need remedial action before the 'trickle' of infections become a 'torrent' and you are left fighting to keep your head above the rising waters.

This paper is based on the recent two-part article written for Virus Bulletin [October and November 2004] and parts of that article have been used with their permission.

Introduction

This paper will discuss the use of the SNORT IDS (Intrusion Detection System) but with a twist – using it to detect malware by using various signature creation techniques. Before we stick our noses into the trough let us cover a few definitions so that we all know what I mean by the relevant terms used in this paper.

I would strongly suggest that unless you have in-depth knowledge of IDS in general and Snort in particular that you try and obtain copies of the books/papers/articles listed in Appendix A.

What is an Intrusion Detection System

I will use the following definition: "A system that tries to identify attempts to hack or break into a computer system or to misuse it. IDSs may monitor packets passing over the network, monitor system files, monitor log files, or set up deception systems that attempt to trap hackers[1]".

There are two main types, these are: Host-based IDS [HIDS] and Network based IDS [NIDS]. This paper will only cover NIDS focusing on the base SNORT and MySQL versions. Snort-Inline will not be covered.

What is SNORT

For the uninitiated, SNORT is a lightweight Network IDS [NIDS] which works on Windows and *NIX 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 not? I'm using Bayesian Filtering [an anti-SPAM tool] to catch malware with great success.

I'm not saying that virus scanners are useless or that IDS is better, my reasoning for using an IDS to catch malware includes:

- Fast moving threats require quick (and sometimes 'dirty') detection methods.
- Most malware that causes problems are network borne, and therefore an IDS is a suitable tool.
- Many of the signatures I create are created BEFORE some (if not all) AV companies have detection capabilities for a new breaking threat.
- I'm 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 just relying 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 probably many, many, other reasons.

[1] http://myphliputil.pearsoncmg.com/student/bp_hoffer_moderndbmgmt_6/glossary.html

Discussion

This section of the paper will discuss how and where to get SNORT and suggested additions to get the best out of the beast. Then we will move on to how the assembled beast looks, and then we will go past the crackling and get to the real meat; how to train it to catch malware.

Getting the PIG

This section will offer brief guidance on getting the required packages on the following platforms, it will not cover in any real detail how to perform the install of Snort or the components, but links to the required instructions will be supplied:

Windows

The first thing you will need to install is the WinPCap package, this allows Snort to 'sniff' the packets on the network, Snort will not function without it being installed. You can find it here: http://winpcap.polito.it/install/bin/WinPcap_3_0.exe

Next install the Windows binary package: http://www.snort.org/dl/binaries/win32/snort-2_3_0.exe

If you want to log all alerts to a database then you will need to install MySQL.

If you want to be able to query the MySQL database, or even the raw log files [if you decide not to use MySQL] then you will need to install either IIS or Apache, PHP, Adodb, JPGraph and SnortReport [to interrogate the raw logs] or BASE [to query the MySQL database].

If you want a user-friendly front-end to Snort, then get a good one from: http://www.engagesecurity.com/products/idscenter/

A good article covering how to install Snort on Windows can be found here: http://www.giac.org/practical/gsec/Jeff_Richard_GSEC.pdf or see the Snort 2.1 book mentioned in the recommended reading section in Appendix A.

Linux

As with Windows you will need to ensure a number of pre-requisites are installed before attempting to install the SNORT RPMs or compiling it from the latest source tarball. These include:

Pcap [Packet Capture Tool] and Pcre [Perl Compatible Regular Expression Tool]

If you want to log all alerts to a MySQL database then you will need to install MySQL.

If you want to be able to query the MySQL database, or even the raw log files [if you decide not to use MySQL] then you will need to install Apache, PHP, Adodb, JPGraph and SnortReport [to interrogate the raw logs] or BASE [to query the MySQL database].

If you want a user-friendly front-end to Snort, then so far I haven't found a good GPL one for Linux. So, it's the command-line for you! If you do find one then please let me know.

You can find a good document on how to setup Snort, etc. on Linux here: http://www.internetsecurityguru.com/documents/snort_acid_rh9.pdf or see the Snort 2.1 book mentioned in the recommended reading section in Appendix A.

As to which flavour of *NIX to use, it is up to you, personally I prefer RedHat, Suse or Mandrake, others swear by BSD derivatives.

All Operating Systems

Whatever operating system and/or flavour you decide to use make sure that you harden the OS as otherwise you may find your NIDS hosting malware or generating attacks rather than detecting them.

Hardening guidelines can be found for all major *NIX flavours and Windows on the web with a simple search.

I would also strongly suggest that you use a separate network card *without* an IP address for your IDS box as this will make it harder for an attacker to detect it and/or attack/disable it.

The Finished Article

Right, now you have downloaded and installed the relevant packages, add-ons and pre-requisites, what should the finished installation look like when using a web browser?

Well, if you have installed SNORT without MySQL support and are just using the raw log files, then you should be using SnortReport. An example from an old live installation appears below:

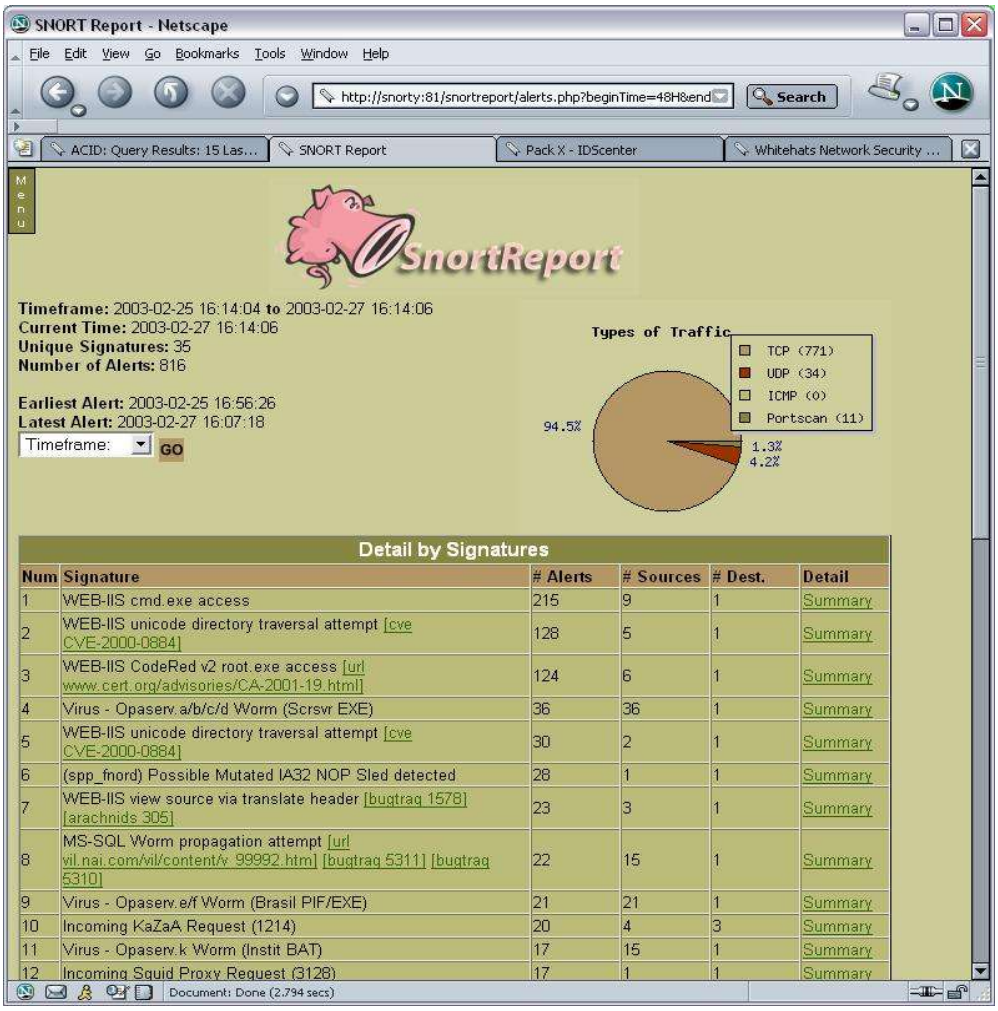

Figure 1 SnortReport

However, if you have installed MySQL and SNORT with MySQL support then you should have also installed either ACID or BASE. The first screenshot below shows the front page from a live installation of ACID.

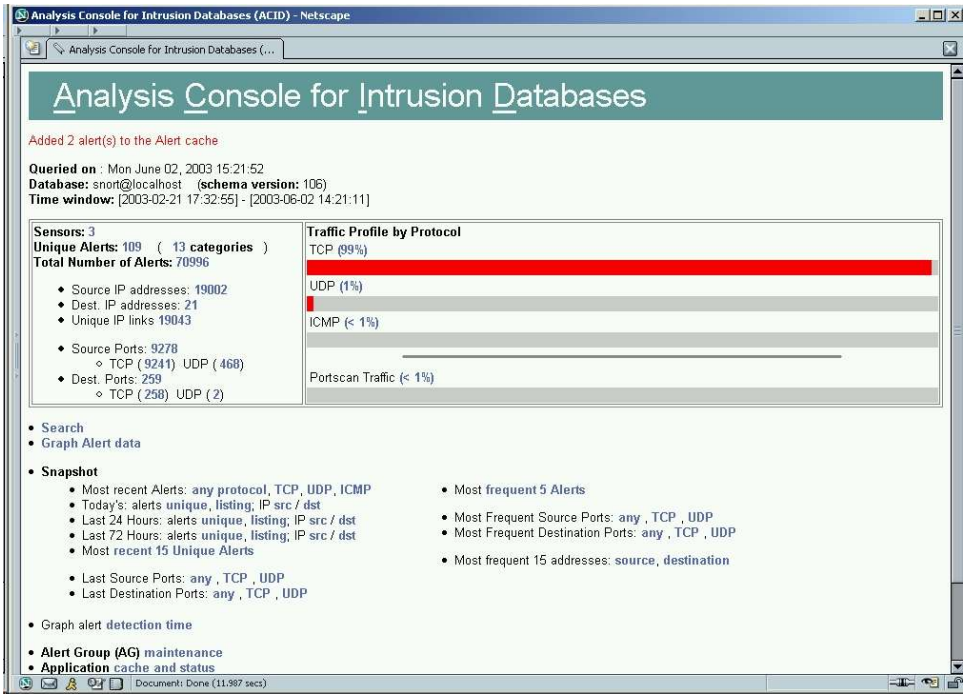

Figure 2 ACID

The next screenshot shows the 'Last 15 Alerts' page from a live installation of BASE. You may have noticed the similarity of the screenshots between ACID and BASE. Well, that is intentional as BASE is the replacement for ACID and maintained by several of the same developers.

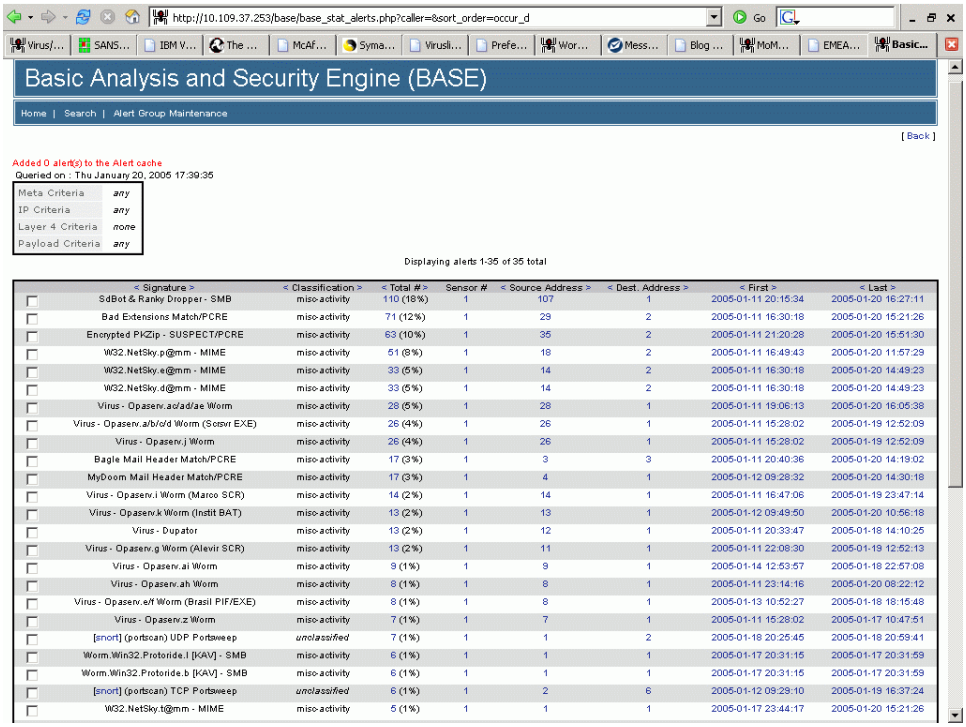

Figure 3 BASE

Snorting Malware

When I was first introduced to SNORT (about 4 years ago, although I have known about it for much longer) 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.

The supplied rules were rather basic and tended to use filenames and other non-binary or non-specific (generic) signatures which made them prone to false positives. Furthermore it was simple for a malware author to defeat these signatures; they simply renamed the files. So now we've gone from a problem of possible false-positives to one of false-negatives!

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 anymore."

When the many security specialists who regularly used SNORT because it could be used to detect viruses saw this notice in the rule file the quite rightfully felt slightly annoyed. It seemed from that rather cavalier statement in the rules files that they could no longer use their favourite animal to sniff out new malware...what could be done to re-educate our faithful malware hunting pigs?

However, I recently raised this issue with SourceFire [the company who own and maintain Snort] and they have informed me that they do still create virus rules/signatures. However, these are usually only available to their 'commercial' customers who pay for their service.

At that time no-one else seemed to be creating rules/signatures for SNORT to detect new malware. So as I'd just started using SNORT for this very purpose I decided to take on the challenge of learning how to create rules/signatures and furthermore to make them available to like minded security professionals.

Rules and Signatures

However all was not lost, occasionally not only were malware related signature posted on the Snort website [individually] but also Symantec would occasionally post them in some of the descriptions for malware they had classified.

Just before I wrote my article for Virus Bulletin I was approached by a new group that were interested in using my malware signatures for Snort. This group now produce the 'Bleedingsnort' signature/rule sets, not only for malware but new exploits, etc. However the signatures/rules they produce are indeed 'bleeding-edge' and therefore may be more likely to cause false positives or even worse false-negatives. Of course it all depends on the quality of the signatures/rules produced and the level of testing they go through to minimise both false-positives and false-negatives alike.

Bleedingsnort can be found here: http://www.bleedingsnort.com

As I was completing this paper SourceFire [the company behind Snort] announced that they would be working with Bleedingsnort.

My own Snort signatures can be found on my own personal website here: http://arachnid.homeip.net. However access is not available without registering on the site and requesting access to the relevant section. This is due to the fact that some of the signatures are very effective and if the malware authors got hold of them then they could learn how to defeat them.

How to Catch Malware Using SNORT

Let us now move on to the most important part of this paper, how to create new malware rules/signatures for use with Snort…so put your malware aprons on [this is a cross between cookery and the practical side of dissection in a biology class] and let us begin. No malware were harmed during the writing of this paper.

Let us begin:

Take one 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 e-mail).

Now let us examine its entrails and see what we can use as a detection string or pattern.

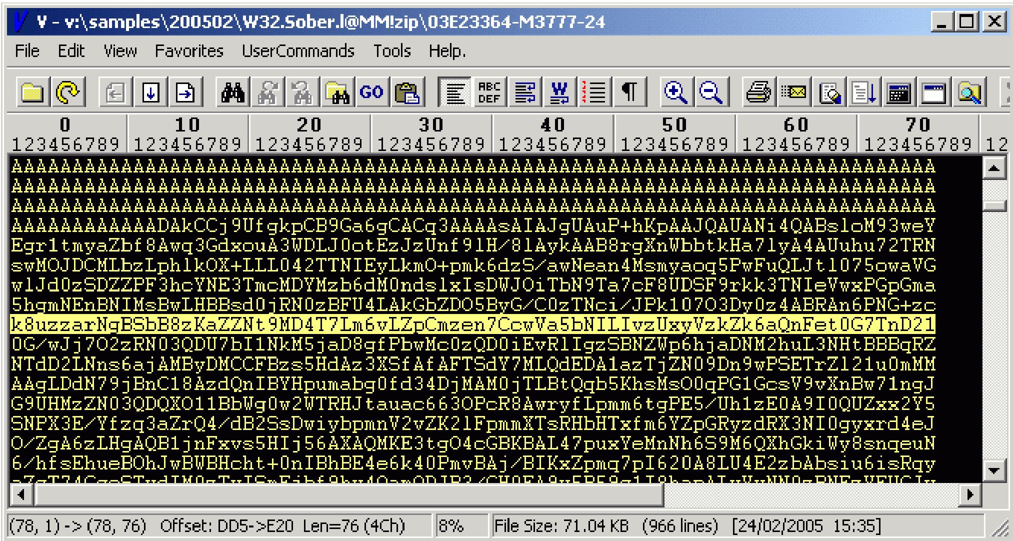

Figure 4: Screenshot of MIME encoded W32/Sober.L infected e-mail attachment

I think a word of caution is needed at this point, before we go any further:

This is not a task for the general end-user population; *only* trained and knowledgeable staff that are used to dealing with, handling and working with live malware samples should attempt this. This should be preferably undertaken on a dedicated system that is NOT networked, just in case the unthinkable should happen and they 'accidentally' launch it!

However, if you can't justify a dedicated PC then you could use VMWare instead (remember to disable network support in the Virtual Machine that you use). Bear in mind that some malware can detect it is inside a VMWare virtual system and change its behaviour; this is only a problem if you actually launch the malware you are examining.

Unless the malware 'under-the-knife' is polymorphic or pads itself out (with random garbage instructions/code) to fool MD5 hashes, or is encrypted (such as in a password protected zip file), then only a single signature is usually required to detect it. Several signatures may be required if the malware you are examining uses other vectors, such as e-mail or peer-to-peer (P2P) to spread, as this may require you to write a rule/signature to detect it in a specific encoded state.

Tasty Malware Entrails.

As an example of a typical modern multi-vector worm, let us take a look at W32/Netsky.p@MM[2] as it propagates via e-mail and also via P2P. In this case how do I create signatures to detect it?

Firstly, I will examine and confirm if the malware sample is static by using hashing algorithms (such as MD5 or SHA1) to hash all the samples I have at hand for it.

This may also require me to decode the attachments if they were received as MIME encoded [via e-mail]. I would then perform the same steps on the decoded attachment. Likewise if the decoded attachments are zipped, I will unzip them and repeat the steps again on the now unzipped and MIME decoded samples.

Think of this as being like a set of those Russian wooden dolls, inside the first one is a smaller one, inside that is yet a smaller one, etc. until we cannot open any more.

MIME without Marcel Marceau

In the case of our example using W32/Netsky.p, the malware is a static binary image and this means that the MIME encoded binary image is also static. This makes our job so much easier. We could have useful Snort signatures within 5-10 minutes.

The next step is to view the sample in a hex/text editor/viewer and select a suitable MIME string to be used to detect the worm when it arrives MIME-encoded via e-mail.

A suitable string should be available within the first thirty lines of the MIME encoded attachment in the e-mail. Try and find a line that is complex, not one that has mainly 'A's in as otherwise the rule/signature will trigger on perfectly harmless and uninfected e-mail traffic, which we want to avoid if at all possible.

I usually select at least one full line (72 characters, sometime I will use over 100 characters instead) to ensure that the chances of a false positive or negative is minimised.

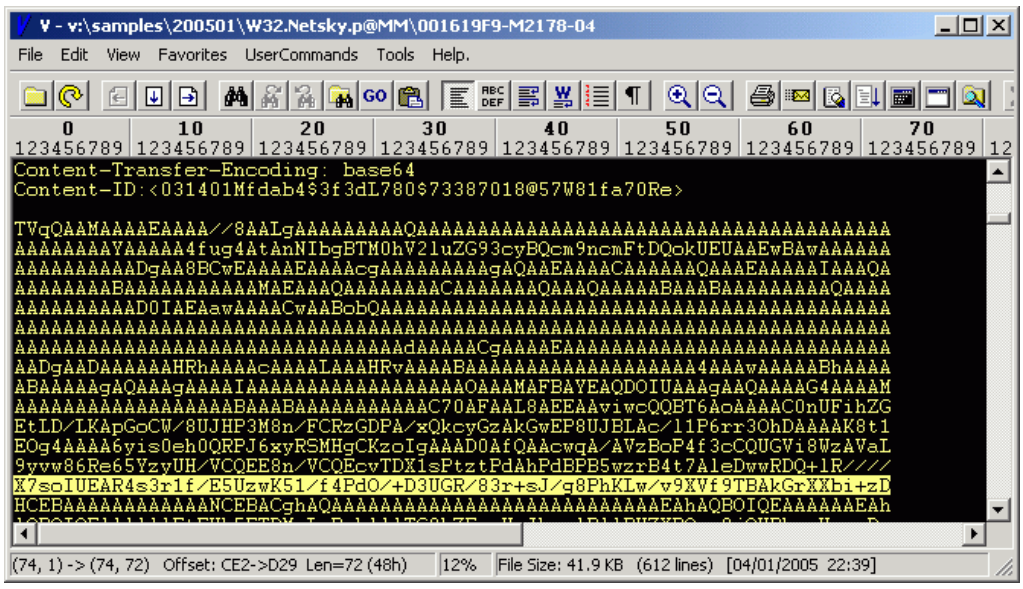

Figure 5: Screenshot of Text/Hex Viewer showing MIME encoded Netsky.p sample

[2]See http://vil.nai.com/vil/content/v_101119.htm for details about this worm. First seen on 21st March 2004.

Once a likely MIME signature string is found [see the highlighted line in Figure 5] this is then tested in a simple virus scanner I created for testing the suitability of SNORT signatures.

This scanner; known as 'MyScan', is then run against all captured samples of Netsky.p.

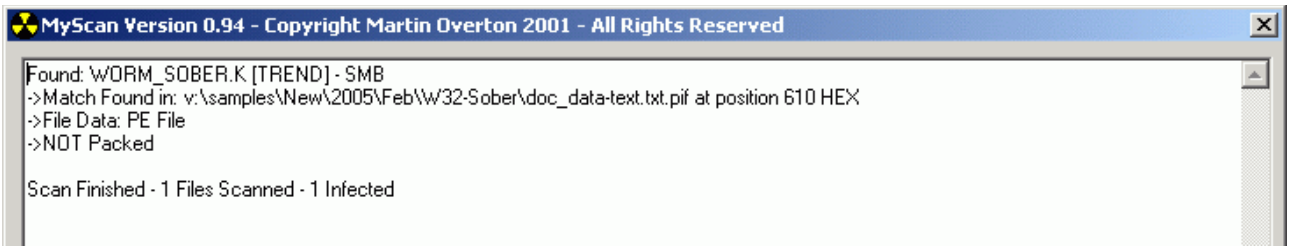

Figure 6: Screenshot showing MyScan detecting Netsky.p in an infected sample

I also test this 'new' signature against previous members of the same malware family to try and ensure that they will not false alarm, in some cases this is not possible, and in these cases I will modify the original rule to state that it also detects this new variant.

I also check all my existing signatures in the 'MyScan' database against the new variant that I'm creating a signature for, again this is for false positive testing.

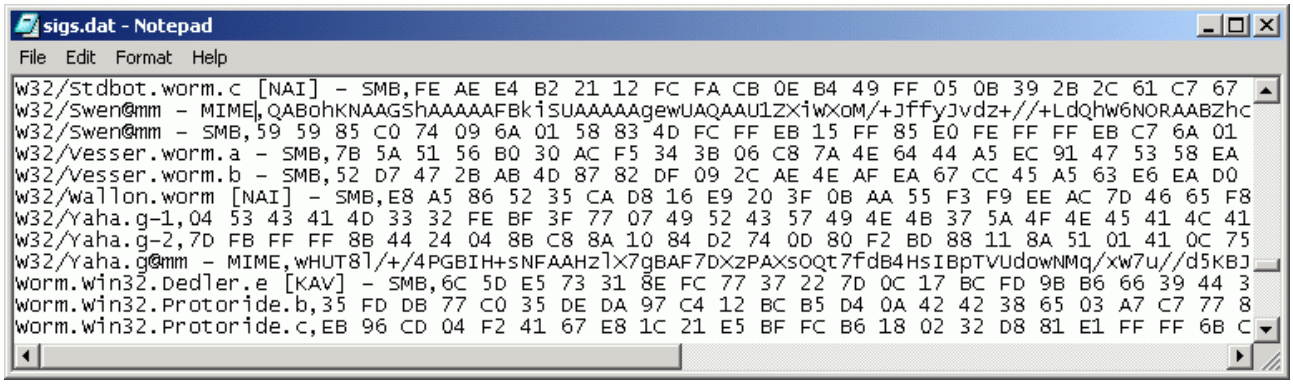

Figure 7: Screenshot of a portion of the MyScan database of signatures

Once the new signature(s) have been tested and any issues ironed out, they are then placed into the rule set which I maintain, known as ‘malware.rules’. You may have noticed that the signatures in the MyScan database [sigs.dat] are not formatted as Snort signatures, only the ‘content’ and proposed ‘msg:’ data is used.

This data is then correctly formatted as a ‘real’ Snort signature in the malware.rules file, using the correct syntax. Finally the malware.rules file and the individual signature/rule is made available to other security professionals and researchers, including AVIEN members and Virus Bulletin subscribers.

The signature which appears below is the one I created for W32/Netsky.p and it has been very successful. As at the end of February 2005 over 5,200 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;)
```

Let's break down the above signature/rule into its component parts so that you can see how it works.

```
alert tcp
```

The first part tells SNORT to send an 'alert' when the signature is matched/triggered. This means when Snort sees a 'tcp' packet that contains the ‘signature’ we chose previously that contains the signature for the malware, in this case W32/Netsky.p.

Snort can also test UDP and ICMP traffic as well as TCP, other protocols may well be added.

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

The next part specifies that we only want it to trigger when the traffic is coming from an IP address that is NOT one of ours [$EXTERNAL_NET any] (but we don’t care which TCP port it arrives from), but is being sent to one of our IP address range [$HOME_NET any] (it can be received on any TCP port).

$EXTERNAL_NET and $HOME_NET are user-defined variables used by SNORT. 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. This is useful if you want to watch traffic which may originate from inside or outside of your network rather than just inside or just outside.

You can also tie down the detection to specific ports instead of using the 'any' port keyword, such as '25' or '110'

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

When a signature match [MIME encoded] occurs then send the alert text “W32.Netsky.p@mm – MIME” to the console, log or database (whatever you use or have setup). This alert can also be setup to call external programs [such as a pager tool] or use network message functions to alert you immediately.

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

So if the above is found in the TCP packet then alert.

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

Furthermore, log it as the 'classtype' of 'misc-activity'. This could be any registered classtype, so you could set the classtype to 'malware' ‘worm-traffic’ or ‘nasty-traffic’if you prefer.

The final part of the signature/rule is the 'rev' statement, this is just used to allow revision control so that you can keep track of how many changes you have made to a rule. If you are maintaining rules it is a good idea to use the ‘rev’ statement to keep track of things.

Just BIN There

Now we have seen how to handle MIME encoded malware arriving via e-mail, let us move swiftly on to looking at malware that arrives as a binary, such as share-crawling worms.

Not surprisingly we follow almost the same steps 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. In this case we do need a hex viewer/editor to view the internals of the sample which will allow us to find a suitable string of hex values to use as a signature.

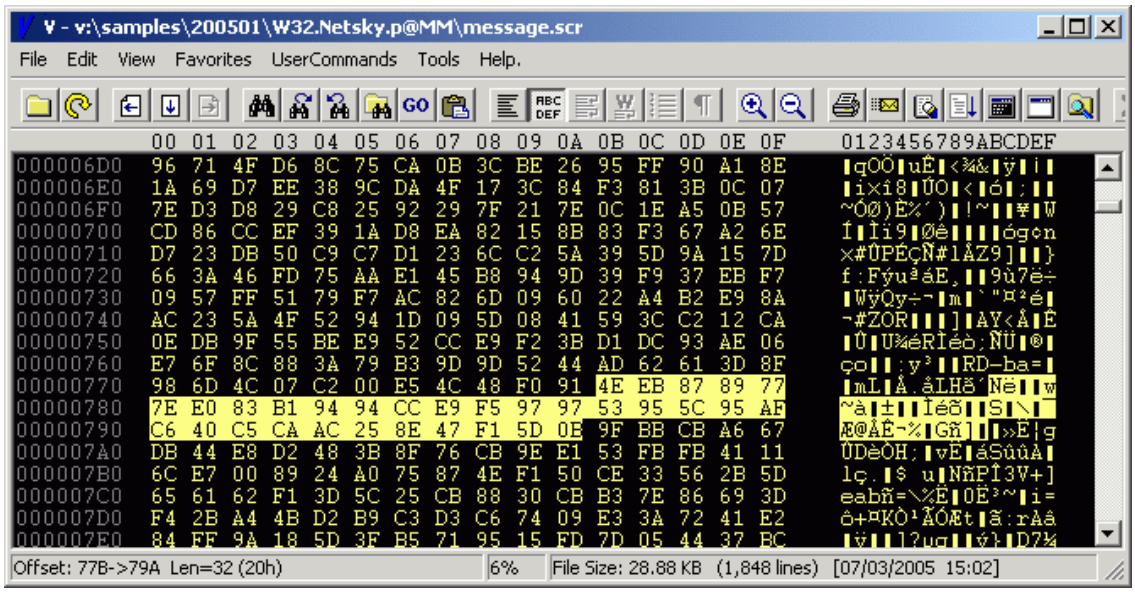

Figure 8: Hex view of a W32/Netsky.p binary sample

As you can see from the screenshot above, I have highlighted a section of code to use as a signature for W32/Netsky.p in its binary form. Here is the completed Snort signature using the selected string as a 'signature'.

```
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.

The main difference in the 'content' section of the signature, when comparing them to the MIME signature is that hex signatures must also be prefixed and suffixed by the | (broken pipe) character inside the "(double quotes) whereas MIME signatures are just enclosed in " (double quotes).

For the more adventurous of you 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 too.....and lots more besides.

Reversal Techniques

The rule/signature for W32/netsky.p we have created so far in this paper show how to detect this malware coming from an external network [mainly the internet] to your internal network. What if you want to reverse this test or even test both directions at the same time, can it be done?

Sure it can, but there are several things to be aware of, I'll cover these next.

Simply change the original part of the signature/rule from:

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

Which you would use to detect inbound packets (from an IP not on your internal network).

To the following:

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

This would be way to detect outbound packets (from an IP on your internal network).

This will reverse the direction of the test, however you may think that toy can do this another way, but you can't just use '<-' as this is NOT supported by SNORT.

Or you could use the following:

```
$EXTERNAL_NET any <> $HOME_NET any
```

Which is what you would use if you want to test data going in either direction with a single signature/rule (both inbound and outbound).

Flow With Me

Another useful keyword is the 'flow' directive. This can be used to limit rules to client or server traffic, such as:

```
alert tcp $EXTERNAL_NET 110 -> $HOME_NET any (msg:"VIRUS Klez Incoming";
flow:to_server,established; dsize:>120; content:"MIME"; content:"VGhpcyBwcm9";
classtype:misc-activity; sid:1800; rev:2;)
```

The above rule/signature will only trigger once a client has connected to a server (in this case a POP3 server) and act on the data received from the server. This example is one of the original Snort 'virus.rules' entries. As you can see it uses several other directives (dsize and sid which I will cover later in this paper).

Web Content

Let us say that you want SNORT to alert on web traffic that meets a specific signature.

```
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"WEB-MISC readme.eml download
attempt"; flags:A+; uricontent:"/readme.eml"; nocase; classtype:attempted-user; sid:1284;
reference:url,www.cert.org/advisories/CA-2001-26.html; rev:8;)
```

This rule will trigger when a URL contains '/readme.eml' in any case (upper, lower or mixed).

This is an ideal solution to handling malware that downloads components and updates from the web, such as W32/Bagle.az@MM[i] or Downloader-PU[ii]. As you can see this rule uses a number of other directives that haven't yet been covered in this paper (nocase, uricontent, reference and flags).

Let me quickly cover those new directives:

Nocase: Does what you would expect, it tells Snort to ignore the case of the previous content test, be it upper, lower or even mixed case.

Uricontent: This tells Snort to only look in the NORMALISED request URI field, useful for looking for simple text strings in URLs. Any non-standard characters are ignored, such as '%c0', etc.

Reference: This allows you to include a link to a webpage with more information on the threat/exploit/malware or whatever you choose.

Flags: Just as you would expect this directive will check to see if the specified TCP flag bits are present.

Dsize: This is used to test the packet payload size, it's main use is in checking for oversize packets such as in buffer overflow attacks.

Sid is Here!

The 'sid' keyword is used to uniquely identify a specific rule/signature. However, before numbering your own signatures the following information needs to be digested and understood:

Sid No.	Meaning
<100	Reserved for future use.
100-1,000,000	Reserved: ONLY for rules included with SNORT (i.e. 'Official' rules).
>1,000,000	Free for use for local rules on a free-for-all basis.

Multiple content

Earlier in this paper I showed a SNORT signature/rule that contained just one 'content' section (signature) to be matched against incoming data.

However, as mentioned previously, you are not limited to a single 'content' section and you can even mix content types, such as binary and text strings as in the rule below that will detect the well known SQL Slammer worm, please note that this rule is looking at UDP traffic rather than TCP:

```
Alert udp $EXTERNAL_NET any -> $HOME_NET 1434 (msg:"W32.SQLEXP.Worm propagation (1434)";
content:"|68 2E 64 6C 6C 68 65 6C 33 32 68 6B 65 72 6E|"; content:"|04|"; offset:0;
depth:1;)
```

You can even mix content types in the same content section, as in this rule for detecting Admin access being attempted via Netbios [SMB]:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS SMB ADMIN$access";
flow:to_server,established; content:"\\ADMIN$|00 41 3a 00|"; reference:arachnids,340;
classtype:attempted-admin; sid:532;  rev:4;)
```

Hide and Seek

Now that I've covered relatively simple to detect malware and how to select and create suitable signature strings from both binary and MIME encoded samples let us now move on to malware that are somewhat more challenging:

Obfuscated malware samples are often packed (sometimes as many as ten different packers are used) or have some mild polymorphism, such as adding random text or other garbage to the file to fool MD5 or other hash functions….they can be as slippery as a greased pig and about as hard to catch!

Encrypted samples, in the context of this paper, refer to the password protected zips that have been seen in many of the Bagle variants, both those with plain text password in the e-mail body and also those that use the graphic password 'trick' to try and slowdown or stop the scanner from scanning the encrypted file held in the password protected zip.

As with encrypted samples to reliably detect samples that use obfuscation may require multiple signatures or you may be able to find another way to detect them altogether.

Let us now look at a different way of detecting worms, not by the attachment but by the manufactured e-mail headers.

Snorting with PCRE (PERL Compatible Regular Expressions)

As hinted at above, there are sometimes other ways to reliably detect obfuscated or encrypted malware e-mails not by the body or attachment data but by the manufactured mail headers that they use (or do not use).

For example, both the MyDoom and Bagle families use manufactured headers that can be used as a reliable method of detecting them, without the need to create signatures to detect the attachment, such as the one below:

```
alert tcp $EXTERNAL_NET any -> any any (msg:"MyDoom Mail Header Match/PCRE"; pcre:"/X-
MIMEOLE\: Produced By Microsoft MimeOLE V6\.00\.2600\.0000/"; pcre:"/boundary=["][-]{4}
\=\_NextPart\_000\_\d{4}\_.{8}\..{8}/"; pcre:"/filename=["]\S{1,}[.](bat|scr|com|cmd|exe|
pif|zip)/";classtype:misc-activity; rev:1;)
```

The above will reliably detect MyDoom constructed e-mails, even if they are corrupted, non-viable or truncated. So far the authors of MyDoom haven't yet changed the mail headers, but that may well happen soon. Let me make this very clear this will detect NEW [unkown] variants without modification.

As you may have noticed this rule/signature uses multiple 'pcre:' directive, the rule will only fire if all of these pcre tests are met and results in a 'true' value.

Let me break this down into the relevant regular expression tests and explain exactly what each use of the directive is looking for and how it works. Please note that all pcre strings should be enclosed in '/', see the examples in this paper as it shows all pcre strings/tests used in Snort signatures/rules use this notation. If you want to test these pcre strings/test just using the stand-alone pcre executable then remove them.

Let's start with the first regular expression:

```
"/X-MIMEOLE\: Produced By Microsoft MimeOLE V6\.00\.2600\.0000/"
```

This tells Snort to look for the following string 'X-MIMEOLE: Produced By Microsoft MimeOLE V6.00.2600.0000', notice the use of '\' [escape]; this is used to tell the pcre engine that the next character should be treated as a character and not as a directive.

Why is this important? Well in the case of '.' if we didn't 'escape' it with the '\' then the pcre engine would treat the next character as any single character, when what we want it to find is the character '.' in this example. Likewise the ':' is treated in a special way by pcre and to remove the possibility of this being incorrectly read I have '\'[escaped] it so that it is treated as a normal character.

It is possible that this string could be handled as a standard text string using a normal 'content:' directive instead. E.g: content: "X-MIMEOLE: Produced By Microsoft MimeOLE V6.00.2600.0000". I used a pcre: directive instead as this offers me the option of using wildcards

and other techniques should the malware authors change the mail header or add some randomisation.

```
"/boundary=["][-]{4}\=\_NextPart\_000\_\d{4}\_.{8}\..{8}/"
```

The line above tells Snort to use the pcre function to find lines that contain 'boundary=' and have at least 4 '-' and then contains '=_NextPart_000_' which is followed by any 4 digits '\d{4}' followed by '_' and any 8 characters followed by '.'and any 8 characters.

```
"/filename=["]\S{1,}[.](bat|scr|com|cmd|exe|pif|zip)/"
```

The line above now tells Snort to use the pcre function to find lines that contain 'filename=' followed by a '"'then followed by any character that is not a whitespace, and at least 1 character long with not maximum number of character limit which is followed by a '.' which has any one of the listed three-letter extensions.

It looks and can be quite complex to do this right, and there are many different ways to search for strings, substrings, etc. So my pcre tests shown in this paper should not be considered as the correct or only way to do this. You can find lots of useful data on pcre and pcre expressions and their usage on the following website: http://www.pcre.org .

Let us now move on to a similar solution to detecting e-mails generated by the Bagle mass-mailing worm.

```
alert tcp $EXTERNAL_NET any -> any any (msg:"Bagle Mail Header Match/PCRE";
pcre:"/Message-ID\:\W{1,}[<][a-z]{19}[@]/"; pcre:"/boundary=["][-]{8}[a-z]{20}/";
classtype:misc-activity; rev:1;)
```

The above will reliably detect Bagle constructed e-mails under the same conditions as with MyDoom and even the latest ones are still reliably detected.

You can also use this technique to detect/block unwanted attachments in e-mail:

```
alert tcp $EXTERNAL_NET any -> any any (msg:"Bad Extensions Match/PCRE";
pcre:"/attachment\;\W{1,}filename=["]\S{1,}[.](scr|com|exe|cpl|pif|hta|vbs)/";
classtype:misc-activity; rev:1;)
```

The signature above does not include all recommended 'bad extensions' to block, just a small subset, feel free to add ones you want to include. This clearly shows the beauty of Snort and its open architecture, try doing this with some of the closed commercial products and see how far you get.

```
alert tcp $EXTERNAL_NET any -> any any (msg:"Encrypted PKZip - SUSPECT/PCRE";
flow:to_server,established; pcre:"/UEsDBAoAA\S{10,}[A]{4,}/"; classtype:misc-activity;
rev:1;)
```

The final signature/rule above will usually only trigger on password protected zip files created by e-mail worms. To date it has not triggered on password protected zips that contain samples from other researchers for me to use to create new SNORT signatures/rules. However, this is still a 'test' rule and should be used with care. So in internet acronym speak YMMV [Your mileage may vary].

A quick word of warning, using lots of 'pcre:' directives will require significantly more processor time than using simple content: matching directive, so use them sparingly.

Network Worms and Exploits

SNORT is extremely useful for detecting, tracing and blocking (I'll cover this later in this paper) many of the network worms that have now become part of the background noise on the internet, as well as the vast array of bot families and their increasingly numerous offspring.

Although the SNORT maintainers no longer supply (or support) the 'virus.rules' signature set for the product, they do offer signatures that can still be used to identify the use of most of the exploits that a reasonable percentage of worms, viruses and bots depend on to allow them to auto-run when previewed in Outlook, or getting on to a system via a known exploit in DCOM, LSASS or GDI for example:

```
alert tcp any any -> any 135 (msg:"DCOM Exploit (MS03-026) targeting Windows XP SP1";
content:"|BA 26 E6 77 CC E0 FD 7F CC E0 FD 7F|"; classtype:attempted-admin; sid:1100007;
reference:URL,www.microsoft.com/security/security_bulletins/ms03-026.asp;
reference:URL,jackhammer.org/rules/1100007; rev:1;)
```

The above will detect the DCOM vulnerability being attempted on a Windows XP SP1 based system.

IM Worming my way in

Instant messaging from a malware author's point of view is really starting to take off. Malware taking advantage of Instant Messaging client and/or services are not new, they have been around since IM function were first included in mass-mailing worms like W32/Goner[iii] and W32/Aplore[iv]. These worms just used IM as one of their infection vectors, they have now evolved into true IM worms that we have today.

New threats that have appeared such as the Bropia family of IM malware and the recent emergence of several new families like Kelvir and Fatso [Sumom or Serflog] have shown that this infection vector is viable and I expect it to be used more widely in the months to come. The good news is that theses threats are trivial to detect using Snort. Signatures/rules were created within five minutes of samples arriving in my inbox from a suspicious user. An example appears below:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"WORM_KELVIR.B [TREND]"; content:"|66
76 47 65 24 C6 64 11 16 88 6C 73 88 88 A8 67 87 88 68 68 78 87 78 68 76 77 64 66 46 7C 64
11|"; classtype: misc-activity;rev:1;)
```

As you can see this will detect the recent Kelvir.B instant messaging worm which targets Microsoft's own Instant Messenger application.

When SNORT tells Porky-Pies [Lies]

This section will cover some of the possible problems you may encounter when using SNORT. These are not SNORT issues, but issues you may encounter with the signatures/rules themselves usually due to poor signature selection or implementation.

False Positives

A false positive is when a rule is triggered on a file that is not malicious, but flagged as if it were.

Personally, as with anti-virus these do occur, especially when signatures are selected in haste and these are not sufficiently tested and so erroneously flag innocent files as malware. To date I have found very few false positive issues with the signatures/rules I create. This I attribute to the level of testing I carry out before making them available. However, they do occur from time to time and should be expected.

So, to minimise the possibilities of a false positive here is a list of things to avoid:

- MIME/Binary signatures less than 20 characters.
- Signatures made from the very start of the MIME or Binary file.
- Limited or no testing on real samples and harmless files of a similar structure.
- Single PCRE directives with common text patterns.
- File or attachment names.

And here are some things to do to help reduce the false-alarm problem:

- Create long signatures for MIME or Binary files, at least 32 [Binary] and 72 [MIME].
- Test, test and test again with real infected files and clean files too.
- If possible use multiple 'content:' statements or use other directives to limit the search.
- If you use PCRE based rules/signatures they require significantly more testing as false-positives are more likely to occur.

Try to think of signatures in the same way as keywords used for searching for web sites, etc. on Google. Too few or too common keywords will result in millions of hits, but over 90% will not be relevant to what you are actually searching for. The same applies to Snort signatures/rules.

If you use the 'Flexible Response' features (flexresp) in SNORT you could end up with a self-inflicted DoS [Denial of Service], so do be careful when using this feature.

False Negatives

A false negative is when a rule is not triggered on a file that really is malicious and should be correctly detected by an existing Snort rule/signature.

This is a more serious problem, as it means that the signature is flawed and misses 'real' infected files/content that should have been identified. In some cases this is difficult to resolve, especially with complex obfuscated or encrypted malware. To resolve this issue in these cases it usually requires multiple signatures/rules to be created or a different approach, such as using header information rather than MIME body data.

The same techniques should be applied as discussed with false-positives in the previous section of the paper.

Blocking instead

Earlier in this paper I covered the 'alert' directive, which will send an alert to the SNORT logs, Syslog, a database or other configured storage options when a signature is matched.

There are other options you can use when using the flexresp [Flexible Response] features on what action to take when a signature is matched; these include the ability to terminate the session, either at the originator, the destination end of the conversation or both at the same time.

The advantage of this is that you can stop an infection attempt dead in its tracks!

Below is an example:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"Backdoor.GoBot.p [KAV] - SMB";
content: "|B6 B9 ED ED CD 77 5E 11 75 1B 8B BB 01 7E 05 29 54 BF 0D B6 F0 83 7B 0C 3F 44
64 EB 96 0A 8B 72|"; classtype: misc-activity; resp:rst_all;)
```

This would terminate the connection between the source and destination IP addresses when the signature was matched. Obviously, this 'power' should be used with caution as it can cause problems with some applications, especially if there is a false positive problem with the signature itself. As Peter Parker is told in Spiderman "With great power comes great responsibility", so use this feature sparingly and wisely. The key directive in the above example that specifies what to terminate is the 'resp:' directive. In the example this is set to 'rst_all', but it could just as easily be 'rst_snd' [reset sending socket] or 'rst_rcv' [reset receiving socket].

However, using this feature effectively turns SNORT into a so-called IPS (Intrusion Prevention System) rather than an IDS. Another way is to use the Snort-inline version as it has been designed to be used in exactly this way.

Keeping the Rules

So, how do you manage and keep all the rules up-to-date?

For the 'official' Snort rules I suggest that you install and use the wonderful 'Oinkmaster' perl script. This will work on both *NIX and Windows systems and is easy to setup and maintain.

For those of you that like 'GUIs' then you are in luck, for Windows use the front-end I recommended at the beginning of this paper, for *NIX, the latest version of Oinkmaster has an experimental GUI which can be found in the contrib directory of the tar.gz file you downloaded and extracted.

For the Bleeding-Snort rule sets; the simplest way to get them and install them is to use wget. However, you can use a modified configuration file for Oinkmaster instead. Instructions on how to achieve this can be found on the Bleeding-snort website.

For my rules, I will be making available a BETA set which will be the latest updated set that I am currently using, this will be available from my website, but you will need to sign-up for an account and be authorised to access the Snort sigs section of the site before you will be able to get them.

Results

Let us have a look at some of the statistics I've managed to acquire when using SNORT for malware detection.

To date I have created over 500 malware signatures/rules for use with Snort.

Over the years I have been running Snort as a malware-detection tool I have logged over 500,000 malware related alerts from the signatures/rules I have created.

Here is a screenshot from BASE of Snort installation that has been up and running on my home aDSL link since early January of 2005.

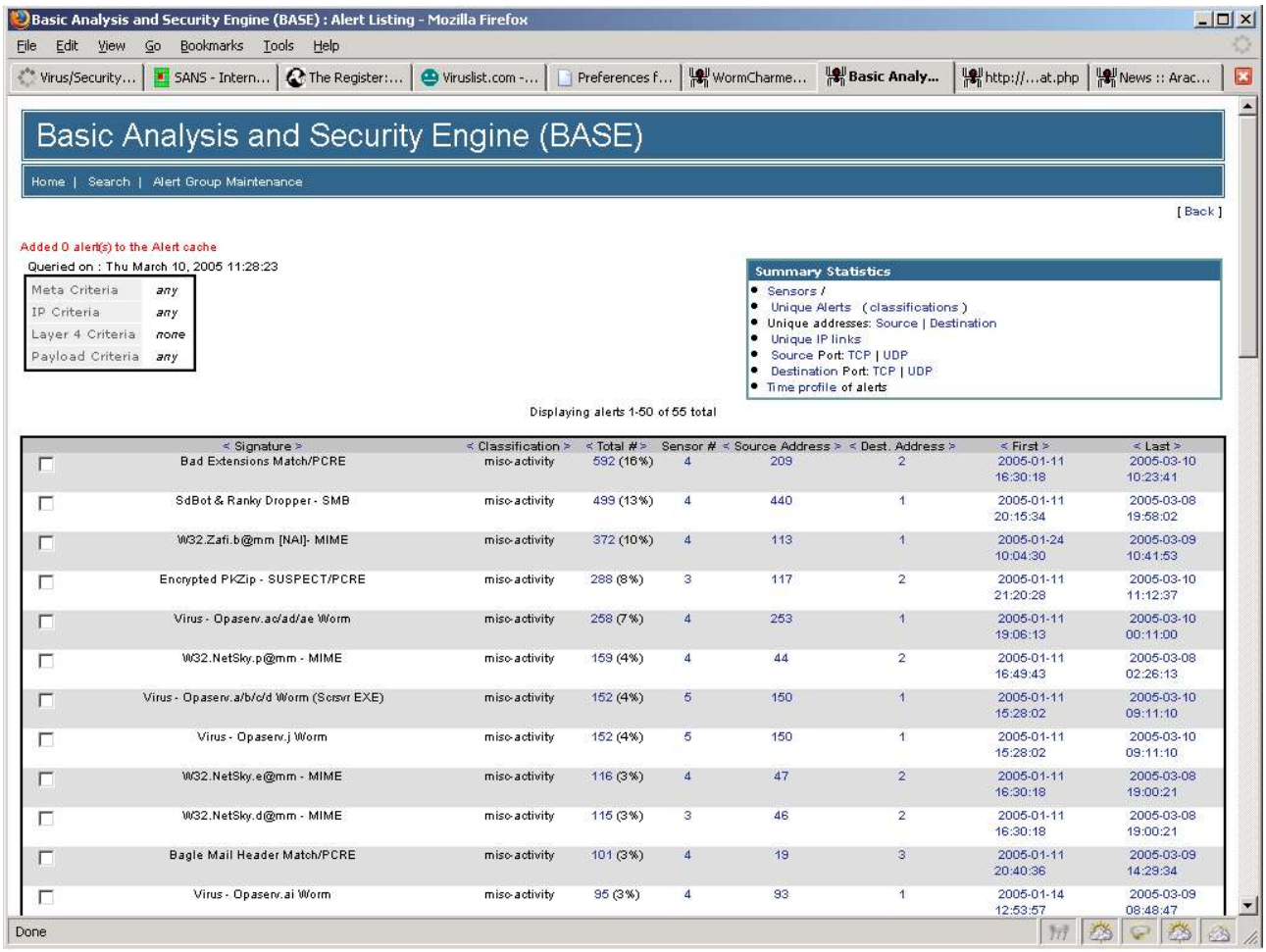

Figure 9: Screenshot of BASE showing number of alerts by each signature name

You can see from the above screenshot just how well my 'pcre' signatures are working as well as my malware specific signatures.

Conclusions

Hopefully I have whet your appetite and shown that SNORT does indeed have its place in the anti-malware toolbox. This is increasingly true when we consider the merging of many technologies between the spammers, scammers, malware and hacking (cracking) communities.

The current trend in Instant Messaging worms will make the need for fast network-based detection and protection almost mandatory for any self-respecting organisation, as otherwise they may well disappear under the swathe of worms that will come. E-mail worms may be on their way out [according to a number of researchers and anti-virus firms], only to be replaced by other faster moving malware threats.

Certainly in the three years that I have been actively using Snort as part of my anti-malware defence strategy it has constantly surprised me with its ease of use, throughput on low-spec systems and the quality of product for what is a effectively free IDS. Many of the commercial products could learn a thing or two from Snort and its openness.

Please do not see this paper as an exhaustive or complete look at SNORT. I have merely scratched the surface of the pig and showed some of the juicy flesh beneath. There is plenty more goodness under the crackling.....dig in and pig out!

Appendix A – Suggested Reading

Snort 2.1 Intrusion Detection, Second Edition published by Syngress, ISBN 1-931836-04-3

Malware in a Pig Pen - Part 1, (Overton, Martin) - Virus Bulletin, October 2004 pp 10-12

Malware in a Pig Pen - Part 2, (Overton, Martin) - Virus Bulletin, November 2004 pp 10-12

Canning More Than SPAM with Bayesian Filtering, (Overton, Martin) - Virus Bulletin International Conference 2004

[i] http://vil.nai.com/vil/content/v_128582.htm

[ii] http://vil.nai.com/vil/content/v_128464.htm

[iii] http://vil.nai.com/vil/content/v_99272.htm

[iv] http://vil.nai.com/vil/content/v_99437.htm