# Global Information Assurance Certification Paper

# SANS GCIA Practical ver 3.5

Alexander Schinner

SANS München
19-24 April 2004

# Contents

# 1. Data mining - Detection and isolation of events using transformations

## 1.1. Abstract

There is a variety of tools to filter packets from a network. One of the most popular ones is the Berkeley Packet Filter (BPF). All such filters are based on static descriptions, e.g., fixed source ports or fixed subnets of IP addresses. These methods work well for most types of network traffic, but there are cases in which a wider variety of applications may be appropriate. In this paper we will introduce a new analysis tool which will allow us to do a time-dependent analysis. One of the advantages of this method is that it enables us to loosen the relationship of the packet to it's IP source address. We will show that we can distinguish between traffic from different machines even if they have the same source address (e.g., NAT router) and that we can detect traffic from the same machine, even if the IP source address had changed. Many other analysis are possible. The basic concept behind this new filter is that it will try to detect any type of linear relationship in the data, independent of nuisance factors as white noise, etc.

## 1.2. Introduction

Due to the TCP/IP network concept, an intrusion detection specialist must handle enormous amounts of data, analyzing a sequence of packets with a lot of header information like IP-addresses, ports, checksums, flags, etc. [1]. How can we find our way through this jungle of data, which is usually formed by different outputs of tcpdump, ethereal, and snort? Typically, we try to find patterns that allow us to categorize the packets into logical groups [2]. Based on our experience, we decide which groups are interesting and need further analysis. Generally, those groupings are described as Berkeley Packet Filters. Those will, e.g. based on the IP address, ask questions of the following type:

1. `tcpdump -r data.dmp dst host 10.0.0.1`
   (Who was accessing my server 10.0.0.1?)

2. `tcpdump -r data.dmp src host 192.168.1.1`
   (How did the user of computer 192.168.1.1 try to attack my server farm?)

These methods allow us to analyze a wide range of aspects like source and destination ports, and patterns, such as illegal values in certain header fields, illegal combinations of header fields, or values which are in principal legal but are highly suspicious. An example of a legal but suspicious expression could be the value $31337$ or the hexadecimal number $0xDEADBEEF$. Of course, an intrusion detection specialist will combine all of these techniques, which will lead to a very good analysis, but as a fact, all the criteria are time independent. What does this mean?

We will try to explain this in a more formal way. We will assume that we have sniffed $N$ packets $p^i$ with $i = 1, \ldots, N$. Each packet is a combination of different values, therefore we might represent it as a vector. However, this complicated syntax gives us no gain in our work. We want to introduce a more simple writing.

In the following, we will index packets $p$ with a filter we want to apply, e.g., $p_{SrcIP}$ if we are interested in the source address. With this, we will define our search as an index function $f(p)$ which will be equal to one if the packet matches our criteria $c$, else the function is zero.

$$f(p) = \begin{cases} 1 & \text{for} \quad p_{criteria} = c \\ 0 & \text{else} \end{cases} \tag{1.1}$$

Using this notation, a search for `port 80` or `host 192.168.1.1` will have the following structure.

$$f_1(p) = \begin{cases} 1 & \text{for} \quad p_{port} = 80 \\ 0 & \text{else} \end{cases} \tag{1.2}$$

$$f_2(p) = \begin{cases} 1 & \text{for} \quad p_{host} = 192.168.1.1 \\ 0 & \text{else} \end{cases} \tag{1.3}$$

Of course, the two index functions $f_1(p)$ and $f_2(p)$ can also be combined to search for `port 80 and host 192.168.1.1`. In this case we will use the following notation.

$$f(p) = f_1(p) \times f_2(p) \tag{1.4}$$

Every BPF can be written as search function. But can every search function be written as BPF?

Figure 1.1 shows an example where this may be difficult. We see from the pattern in the graph that someone tried to contact many ports on our server. Can we write a BPF for filtering only the packets of this port scan? Yes an no. Yes, because in most cases, we will try to identify the source IP (e.g., 10.0.0.10) and then we will filter for the address (`tcpdump src host 10.0.0.10`). No, because in some cases these connects may not originate from one address but from different source addresses. A reason might be that the sender was faking his address. In this case, we will need a search function with a time dependent criterion $c(t)$.

$$f(p) = \begin{cases} 1 & \text{for} \quad p_{criteria} = c(t) \\ 0 & \text{else} \end{cases} \tag{1.5}$$

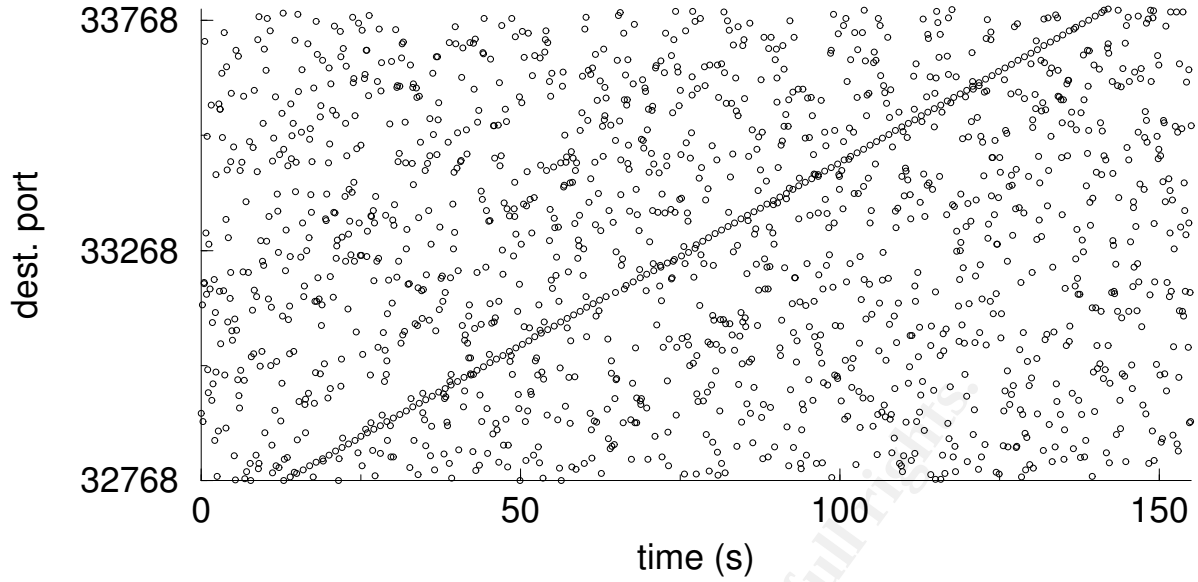



Figure 1.1.: This graph shows a port scan, plotting destination ports versus time.

However, as most networks are very noisy, the exact comparison $p_{criteria} = c(t)$ will be difficult. Some packets will arrive a few milliseconds earlier, some a little bit later. Therefore, we need a term of the form $p_{criteria} \approx c(t)$. Defining a small value $d$ which describes the uncertainty, we write:

$$f(p) = \begin{cases} 1 & \text{for} \quad |p_{criteria} - c(t)| < d \\ 0 & \text{else} \end{cases} \tag{1.6}$$

In the example given by figure 1.1, $c(t)$ would be a linear function of the form $at + b$, where $a$ is the number of connects per second and $b$ is an offset, mostly representing the time the scan started. Two problems arise. First, even if we know $a$ and $b$, it is not possible to translate this criterion into BPF notation. Furthermore, we do not know $a$ and are not very much interested in $b$. The next sections will show how to realize such a time dependent search function without knowledge of $a$ and $b$.

But before we do the math, we have to prepare our data. How this can be achieved easily will be explained in appendix A.

### 1.2.1. Method

In most cases $c(t)$ will follow a linear function as in the example given by figure 1.1. Therefore, within this paper, we will concentrate on linear functions $c(t) = at + b$. But other functions are possible.

The most obvious way to fit data of this form would be linear regression. Unfortunately, our data will most probably not only hold one interesting pattern, but several ones, in addition to white noise, etc. We thus are not interested in fitting $c(t) = at + b$ for all data points, but

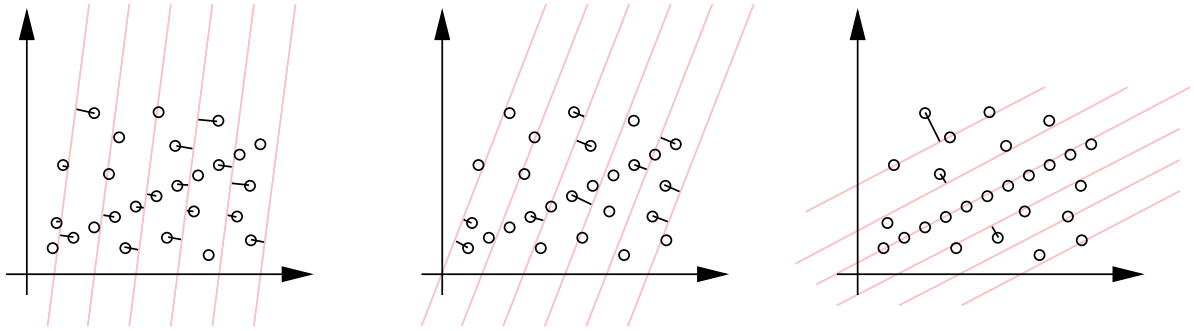


Figure 1.2.: The plots show the distance of the points for three different values of $a_i$.

only for a subset of data points of interest, and therefore will need to work with a different algorithm.

Not knowing the correct values for $a$ and $b$, we will need to apply best guesses.

Trying different values of $a_i$ and $b_i$, we will calculate the shortest distance of each point to the line $c_i(t) = a_i t + b_i$ and will count the number $n_i$ of all points, which are close enough to the line. The maximum allowed distance will be $d$ from equation 1.6. The line $c_i(t)$ with the largest $n_i$ count is the best available result. This method works good, but is slow. We can reduce calculation times by choosing narrow search intervals for $a$ and $b$. Still, the method has limitations.

Equivalent, but faster is the following variation of the above idea. Instead of rotating (parameter $a$) and shifting (parameter $b$) a linear function and fitting the points, we will simply rotate the data set using a rotation matrix [3].

$$\begin{pmatrix} t'_i \\ p'_i \end{pmatrix} = \begin{pmatrix} t_i \\ p_i \end{pmatrix} \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \tag{1.7}$$

We will then distribute the rotated points $(t'_i, p'_i)$ to bins of fixed width. Afterwards, we will calculate the frequency distribution of the points. If we have chosen a good rotation angular $\alpha$, one bin will contain a maximum possible number of points $p_i$ with $f_i(p) = 1$. The advantage is that instead of varying $a$ and $b$ we only need to tune one parameter $\alpha$. Another advantage is that this angle $\alpha$ is only of interest in the range $\alpha = [-\frac{\pi}{2}; \frac{\pi}{2}]$. The idea becomes clearer if we look at figure 1.3.

We are starting with $N$ pairs of values $p_i, t_i$. We can assume, that both $t_{1...N}$ and $p_{1...N}$ are in the intervals $t = [0, t_{max}]$ and $p = [0, p_{max}]$, respectively. If the raw data does not fulfill this restriction, we can simply shift the values without changing the result.

Now, we have to choose a value $d$. In the example above we chose $d = 1$. This means, that the width of a bin is one second.

With this, out algorithm will be of the following form:

**Start main loop** Loop over $\alpha = [-\frac{\pi}{2}; \frac{\pi}{2}]$ with sufficiently small step

**Transformation** Rotate the data points by the rotation angle $\alpha$

$$t'_i = \cos(\alpha) * t_i - \sin(\alpha) * p_i \tag{1.8}$$

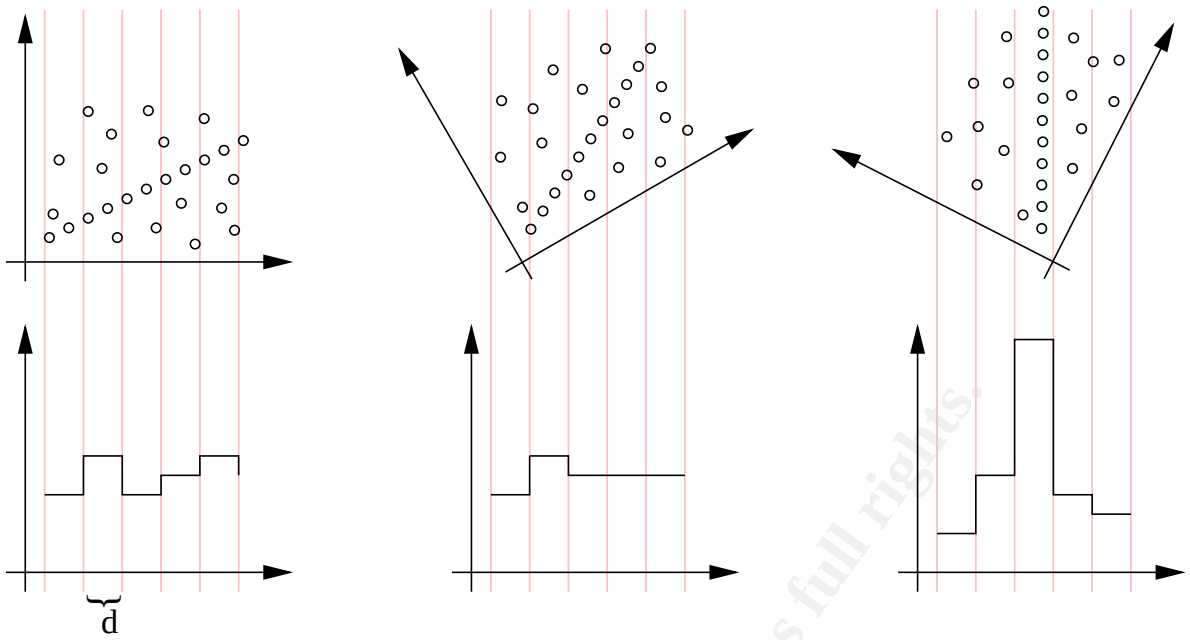



Figure 1.3.: Rotating the points to detect linear pattern

$$p'_i = \sin(\alpha) * t_i + \cos(\alpha) * p_i \tag{1.9}$$

**Calculate bin width** As $d$ is a value from the original system, we have to rotate it, too:

$$d' = \cos(\alpha)d \tag{1.10}$$

**Distribute points to bins**

**Find bin with most points**

**Check for best result until now** If the bin found for this $\alpha$ is better than the best value already found remember $\alpha$ and all other interesting information.

**End main loop**

**Output** Print the dataset for the "best" bin

The above algorithm was designed for linear functions of the form $c(t) = at + b$. If we want to filter other functions, we need to replace the rotation by another appropriate transformation.

One situation we need to pay attention to while working with this algorithm is the case where we observe port scans for very long time periods. This results in an extremely flat rectangle due to the fact that the number of ports is restricted to $65536$ but the time period is large.

In this case the slope $a$ of $c(t)$ becomes very small, which means that the according rotation angular $\alpha$ will be close to $\frac{\pi}{2}$ and $\cos(\alpha)$ will be close to 0, resulting in numerical problems.

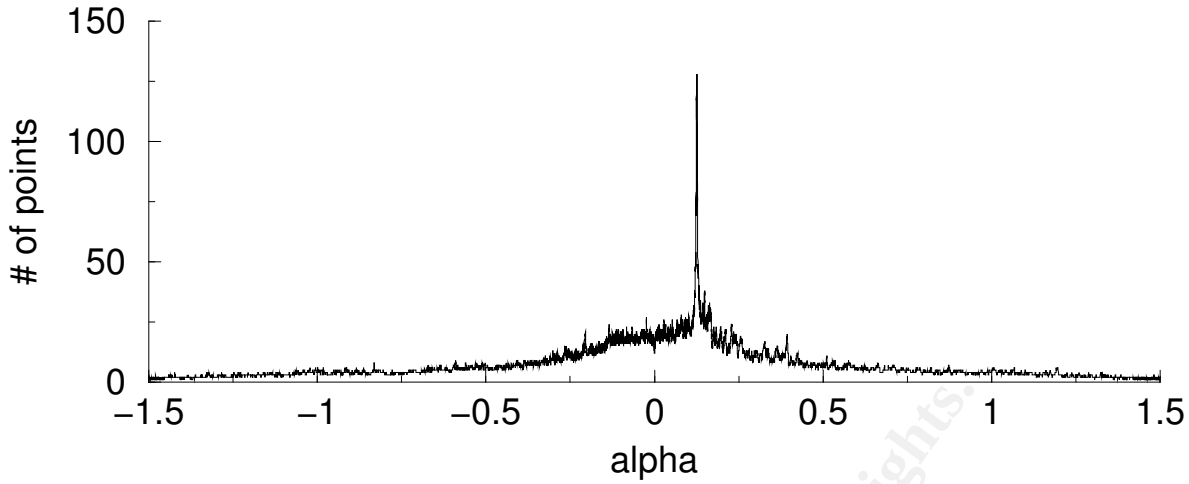



Figure 1.4.: Maximum number of packets found during each step plotted versus $\alpha$

To improve the numerical stability, it is useful to scale the values, so that both axes $t$ and $p(t)$ cover roughly the same intervals. This will not change the results.

After explanation of the basic math, we want to demonstrate how this method works.

We will do so by applying it to the data shown in figure 1.1. While processing the algorithm, we have stored the maximum number of packets found during each step. Plotting this value versus $\alpha$ gives us a good impression of the internal structure of the data. Please find the according plot presented in figure 1.4. The maximum number of packets was seen at $\alpha \approx 0.051$, therefore this is the value of interest. Figure 1.5 shows that our algorithm filtered exactly those packets, that we would have found suspicious after a first manual review of the pattern in figure 1.1.

The data for this example comes from the lab. One computer (10.0.0.10) made 1431 random connections, the other (10.0.0.11) made 129 connects generating the linear pattern. Altogether, we had 1560 connects.

| | | reality | |
|---|---|---|---|
| | | linear pattern | white noise |
| algorithm | linear pattern | 121 (93,7 %) | 7 (6.3 %) |
| | white noise | 8 (0.5 %) | 1424 (99.5 %) |

For the linear pattern 121 correct packets were filtered, and 7 wrong ones. 1424 packets were identified correctly as white noise, and only 8 packets of the linear pattern were failed to be filtered. Using no previous knowledge for the analysis, this is a good result.

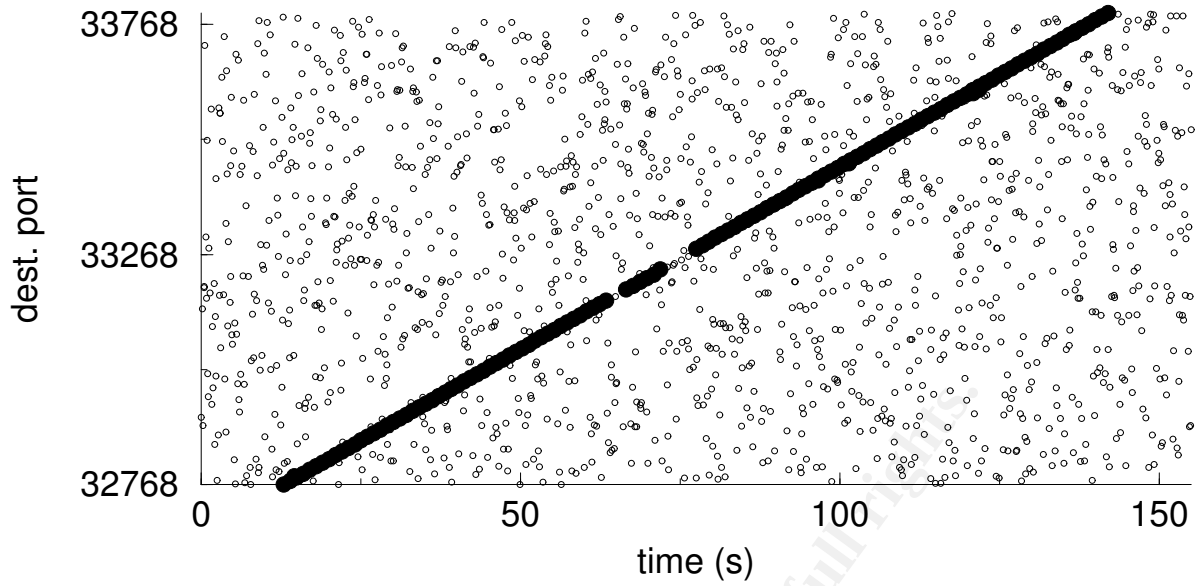



Figure 1.5.: Our algorithm filtered exactly those packets, that we would have found suspicious after a first manual review.

## 1.3. Examples

### 1.3.1. Same source, different computers

Steven Bellovin [4] showed in his paper *A Technique for Counting NATted Hosts* that it is possible to distinguish different computers behind a NAT (see RFC 3022 [5]) device. The same is possible with the here described method, with one big difference. The filter described by Bellovin is as he admits "primarily suitable for analyzing NATs serving networks with comparatively little Intranet traffic". The method within this paper, however, will not be influenced in such a way by nuisance parameters, as white noise. An example of the power of the here described method is given in the analysis as described in figure 1.6, with data from a NATting DSL router. In the top graph, we see that different patterns of potential interest exist in the raw data. Using our fitting algorithm, we have done several runs during which we did select the most intense pattern. After each run, the packets which form this most intense pattern were deleted from the data. As a next step, a new run was started for the remaining data in order to find the next intense pattern. This can be seen in graphs 2 to 4.

### 1.3.2. Same computer, different sources

Here an example from the wild. For a small cluster of computers, packet headers were logged for about 7 days. Plotting (figure 1.7) the source ports for each start of a three way handshake we found an interesting, discontinuous line. Using our method, we isolated these data from the rest and had a more detailed look. Destination port for all these connections was a single IP address on port 110 (POP3). However, the source IP address was changing

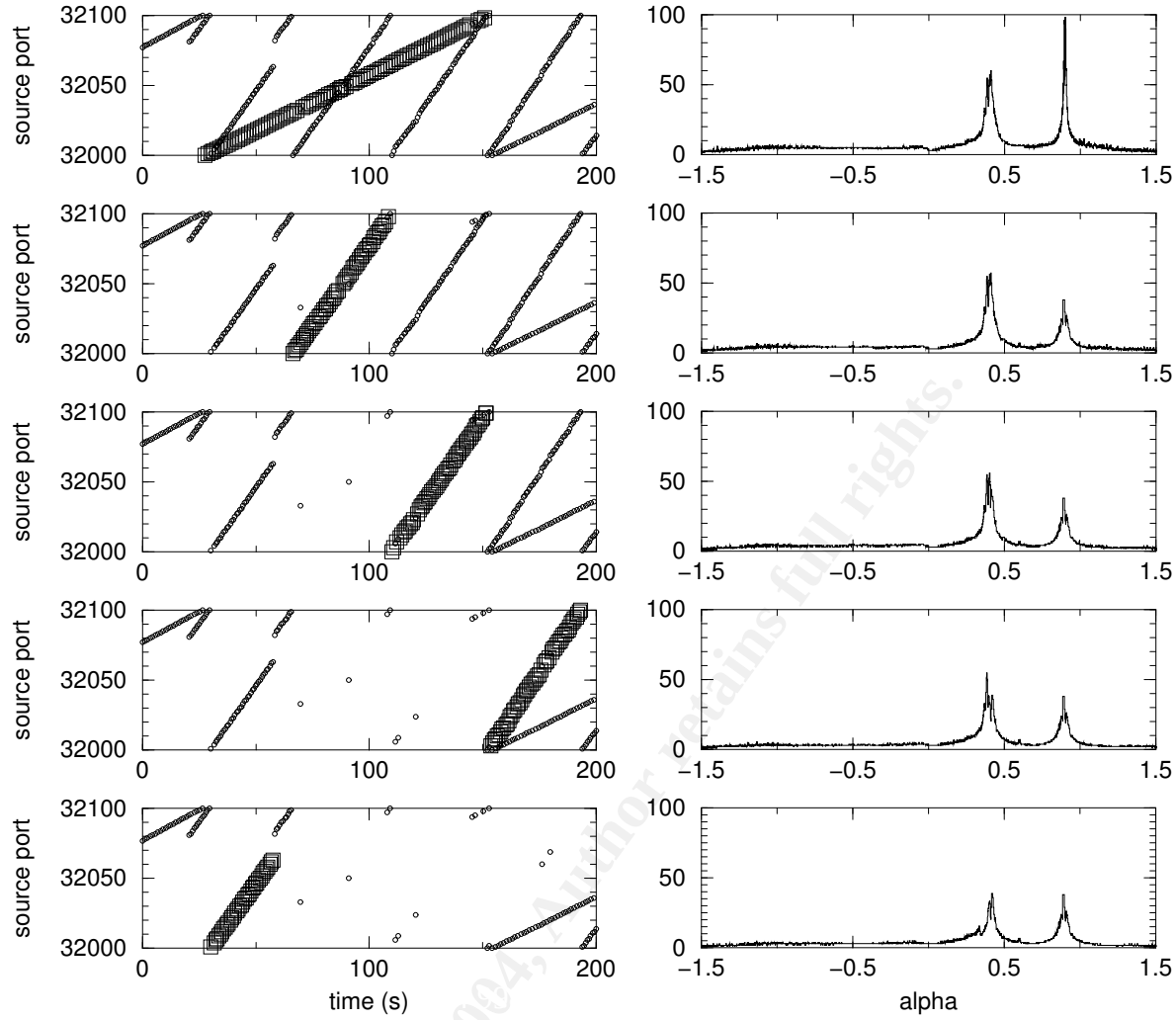



Figure 1.6.: Data from a NATting DSL router

from day to day, but always came from the range of the same ISP.

| Day | Name | IP address |
|---|---|---|
| 1 | pD954FEXX.dip.t-dialin.net | 217.84.254.XX |
| 2 | pD9ED18XX.dip.t-dialin.net | 217.237.24.XX |
| 3 | pD9E43EXX.dip.t-dialin.net | 217.228.62.XX |
| 4 | pD9E43DXX.dip.t-dialin.net | 217.228.61.XX |
| 5 | pD9E438XX.dip.t-dialin.net | 217.228.56.XX |
| 6 | pD954F8XX.dip.t-dialin.net | 217.84.248.XX |

Living in Germany, we know that pXXXXXXXX.dip.t-dialin.net is a typical address for a DSL connection from a T-COM customer. T-COM is disconnecting customers every 24 hours. When reconnecting, which is possible at once, the customer gets another address.

We think, that the source is a DSL router which is always powered on. This would explain the constantly increasing source port. The computer behind this router is switched

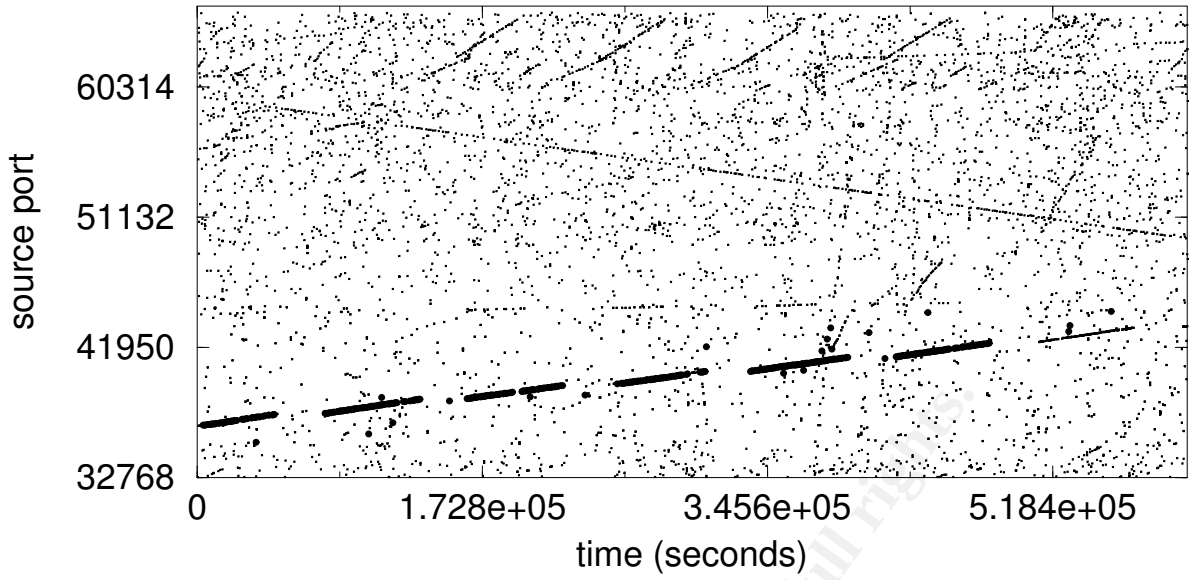



Figure 1.7.: The source ports for each start of a three way handshake were logged for about 7 days

on in the morning, starts to check the POP3 account regularly and is switched off at night. This explains the discontinuity of the line. Then, at night, the router is disconnected and reconnected with another IP address, so we see changing source addresses.

## 1.4. Conclusion

Using this method it is possible to distill more information from the raw data than by using only BPF. Data, simply seen by the human eye, can be isolated and used for further analysis.

## 1.5. Bibliography

[1] W. Richard Stevens. *TCP/IP illustrated (vol. 1): The Protocols*. Addison-Wesley Longman Publishing Co., Inc., 1993.

[2] Stephen Northcutt, Judy Novak *IDS: Intrusion Detection-Systems (german edition)*. mitp-Verlag, Bonn, 2001.

[3] H. Stöcker. *Taschenbuch mathematischer Formeln und moderner Verfahren*. Verlag Harri Deutsch, 1995.

[4] Steven M. Bellovin. A technique for counting NATted hosts. In *Proceedings of the second ACM SIGCOMM Workshop on Internet measurment*, pages 267–272. ACM Press, 2002.

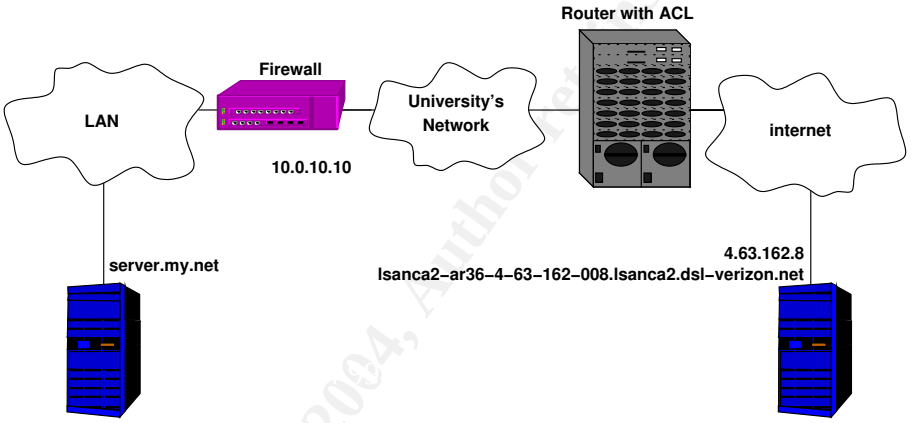
[5] Traditional ip network address translator (traditional nat) http://www.faqs.org/rfcs/rfc3022.html.

# 2. Network detects

## 2.1. . . . egg and spam; egg bacon and spam; egg bacon sausage and spam . . .

### 2.1.1. Source of Trace

The log files to be analyzed were taken from the web server of a workgroup at a German university. The web server is protected by a stateful packet filter firewall and the universities router with active ACLs.




### 2.1.2. Detect was generated by

The here described attack was documented in a log file that was generated by an Apache web server. At the time of the attack Apache 1.3.x was installed (exact version unknown), and, given the local procedures, it is almost sure that the latest Debian package for Apache was installed.

The log file contained the following lines which had been detected by the local administrator who allowed us to use them for this practical:

```
4.63.162.8 - - [31/Aug/2003:12:11:07 +0200] "GET /cgi-bin/FormMail.cgi
 ?realname=yzkkp%20brgoeh&recipient=piscesali@aol.com&email=WantDis@aol.com
 &subject=http://server.workgroup.university.de/cgi-bin/FormMail.cgi
 &message=rvqgd%20ddfyuynatbd%20eanbqjgblgb%20ioffys HTTP/1.1" 404 317
 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)"
4.63.162.8 - - [31/Aug/2003:12:11:07 +0200] "GET /cgi-bin/FormMail.pl
 ?realname=pxyaw%20iqmevw&recipient=piscesali@aol.com&email=WantDis@aol.com
 &subject=http://server.workgroup.university.de/cgi-bin/FormMail.pl
```

```
 &message=hvxej%20uaunbmlzqzs%20uhuaxgeqbnj%20hnvdnh HTTP/1.1" 404 316
 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)"
4.63.162.8 - - [31/Aug/2003:12:11:08 +0200] "GET /cgi-bin/formmail.pl
 ?realname=zlixy%20oybfaq&recipient=piscesali@aol.com&email=WantDis@aol.com
 &subject=http://server.workgroup.university.de/cgi-bin/formmail.pl
 &message=uvuwi%20lpelksuvrel%20iarscqtaypw%20jjwbhv HTTP/1.1" 404 316
 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)"
4.63.162.8 - - [31/Aug/2003:12:11:08 +0200] "GET /cgi-bin/formmail/formmail.cgi
 ?realname=mzndb%20snxsou&recipient=piscesali@aol.com&email=WantDis@aol.com
 &subject=http://server.workgroup.university.de/cgi-bin/formmail/formmail.cgi
 &message=arrkl%20ycjxhwirxrq%20nwvgydgektt%20efjvma HTTP/1.1" 404 326
 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)"
4.63.162.8 - - [31/Aug/2003:12:11:08 +0200] "GET /cgi-bin/formmail/FormMail.pl
 ?realname=gxajo%20iywwvy&recipient=piscesali@aol.com&email=WantDis@aol.com
 &subject=http://server.workgroup.university.de/cgi-bin/formmail/FormMail.pl
 &message=qnqft%20muvxtfujbzt%20dzubgyxrkyj%20pwnwpr HTTP/1.1" 404 325
 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)"
4.63.162.8 - - [01/Sep/2003:22:09:23 +0200] "GET /cgi-bin/FormMail.pl
 ?realname=owxzv%20hwldbv&recipient=piscesali@aol.com&email=WantDis@aol.com
 &subject=http://server.workgroup.university.de/cgi-bin/FormMail.pl
 &message=fbwdi%20aztmalrypyr%20tgazvfdpzmp%20ntucmn HTTP/1.1" 404 316
 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)"
4.63.162.8 - - [01/Sep/2003:22:09:24 +0200] "GET /cgi-bin/formmail.pl
 ?realname=sxujr%20akurmb&recipient=piscesali@aol.com&email=WantDis@aol.com
 &subject=http://server.workgroup.university.de/cgi-bin/formmail.pl
 &message=ggfia%20xaqwweghdpx%20tlcdocelqbi%20buitth HTTP/1.1" 404 316
 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)"
4.63.162.8 - - [01/Sep/2003:22:09:24 +0200] "GET /cgi-bin/formmail/formmail.cgi
 ?realname=qdkgy%20prbpry&recipient=piscesali@aol.com&email=WantDis@aol.com
 &subject=http://server.workgroup.university.de/cgi-bin/formmail/formmail.cgi
 &message=xovhi%20vzguetfobou%20rasdvadbhqx%20bcgspe HTTP/1.1" 404 326
 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)"
4.63.162.8 - - [01/Sep/2003:22:09:32 +0200] "GET /cgi-bin/FormMail.cgi
 ?realname=bimnr%20dapqgk&recipient=piscesali@aol.com&email=WantDis@aol.com
 &subject=http://server.workgroup.university.de/cgi-bin/FormMail.cgi
 &message=txsif%20gfhadhpdckf%20gcpdtsjdnie%20lqhhbb HTTP/1.1" 404 317
 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)"
4.63.162.8 - - [01/Sep/2003:22:09:33 +0200] "GET /cgi-bin/formmail/formmail.pl
 ?realname=uonkt%20sdvadb&recipient=piscesali@aol.com&email=WantDis@aol.com
 &subject=http://server.workgroup.university.de/cgi-bin/formmail/formmail.pl
 &message=hqqbc%20gsjxfwzimhx%20vvuwplpesks%20uwreti HTTP/1.1" 404 325
 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)"
4.63.162.8 - - [05/Sep/2003:15:36:16 +0200] "GET /cgi-bin/formmail.cgi
 ?realname=lgwtu%20tvokod&recipient=piscesali@aol.com&email=WantDis@aol.com
 &subject=http://server.workgroup.university.de/cgi-bin/formmail.cgi
 &message=qjrtu%20qdrhzwrbwsz%20evvpivgnult%20mobvur HTTP/1.1" 404 317
 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)"
4.63.162.8 - - [05/Sep/2003:15:36:16 +0200] "GET /cgi-bin/formmail.pl
 ?realname=hjqvg%20mgznyx&recipient=piscesali@aol.com&email=WantDis@aol.com
 &subject=http://server.workgroup.university.de/cgi-bin/formmail.pl
 &message=lcleg%20tgmjsqcnzvt%20zhpztykhwxn%20xaezpt HTTP/1.1" 404 316
 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)"
```

```
4.63.162.8 - - [05/Sep/2003:15:36:17 +0200] "GET /cgi-bin/formmail/formmail.cgi
 ?realname=uxvbj%20rvvamj&recipient=piscesali@aol.com&email=WantDis@aol.com
 &subject=http://server.workgroup.university.de/cgi-bin/formmail/formmail.cgi
 &message=yzpsc%20gbroooqjmqy%20lltoplymbar%20mwrmtz HTTP/1.1" 404 326
 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)"
4.63.162.8 - - [05/Sep/2003:15:36:17 +0200] "GET /cgi-bin/formmail/formmail.pl
 ?realname=wapcv%20dfgcod&}=piscesali@aol.com&email=WantDis@aol.com
 &subject=http://server.workgroup.university.de/cgi-bin/formmail/formmail.pl
 &message=skicn%20idkphhathsy%20gwfyanhgdmk%20whtwua HTTP/1.1" 404 325
 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)"
```

The above is a typical example of an Apache web server log file format. Let us have a closer look at the last line to understand the structure of this log file format [6].

| | |
|---|---|
| `4.63.162.8` | IP address of the client |
| `- -` | userids from `identd` and from HTTP authentication. |
| `05/Sep/2003:15:36:17 +0200` | Time when the server finished processing the request |
| `"GET ... HTTP/1.1"` | Request from the client |
| 404 | Status code |
| 325 | Bytes sent |
| `"-"` | the Referer |
| `"Mozilla/4.0 (...)"` | The User-Agent HTTP request header |

Now that we have understood the log file format, we will analyze the attack with its different aspects.

The requests came at three different days:

| Script | Date |
|---|---|
| /cgi-bin/FormMail.cgi | 31/Aug/2003:12:11:07 +0200 |
| /cgi-bin/FormMail.pl | 31/Aug/2003:12:11:07 +0200 |
| /cgi-bin/formmail.pl | 31/Aug/2003:12:11:08 +0200 |
| /cgi-bin/formmail/formmail.cg | 31/Aug/2003:12:11:08 +0200 |
| /cgi-bin/formmail/FormMail.pl | 31/Aug/2003:12:11:08 +0200 |
| /cgi-bin/FormMail.pl | 01/Sep/2003:22:09:23 +0200 |
| /cgi-bin/formmail.pl | 01/Sep/2003:22:09:24 +0200 |
| /cgi-bin/formmail/formmail.cg | 01/Sep/2003:22:09:24 +0200 |
| /cgi-bin/FormMail.cgi | 01/Sep/2003:22:09:32 +0200 |
| /cgi-bin/formmail/formmail.pl | 01/Sep/2003:22:09:33 +0200 |
| /cgi-bin/formmail.cgi | 05/Sep/2003:15:36:16 +0200 |
| /cgi-bin/formmail.pl | 05/Sep/2003:15:36:16 +0200 |
| /cgi-bin/formmail/formmail.cg | 05/Sep/2003:15:36:17 +0200 |
| /cgi-bin/formmail/formmail.pl | 05/Sep/2003:15:36:17 +0200 |

Each of those requests accesses a CGI script and transfers a number of parameters. In order to increase comprehensibility, it will be attempted to decode these parameters. Each of the requests had the same parameters but (sometimes) different parameter values.

realname= The sent realnames are different for every request. It seems as if they were generated at random.

| Script | realname |
|---|---|
| /cgi-bin/FormMail.cgi | yzkkp brgoeh |
| /cgi-bin/FormMail.pl | pxyaw iqmevw |
| /cgi-bin/formmail.pl | zlixy oybfaq |
| /cgi-bin/formmail/formmail.cg | mzndb snxsou |
| /cgi-bin/formmail/FormMail.pl | gxajo iywwvy |
| /cgi-bin/FormMail.pl | owxzv hwldbv |
| /cgi-bin/formmail.pl | sxujr akurmb |
| /cgi-bin/formmail/formmail.cg | qdkgy prbpry |
| /cgi-bin/FormMail.cgi | bimnr dapqgk |
| /cgi-bin/formmail/formmail.pl | uonkt sdvadb |
| /cgi-bin/formmail.cgi | lgwtu tvokod |
| /cgi-bin/formmail.pl | hjqvg mgznyx |
| /cgi-bin/formmail/formmail.cg | uxvbj rvvamj |
| /cgi-bin/formmail/formmail.pl | wapcv dfgcod |

recipient= The sent recipient is identical for all requests. It is piscesali@aol.com.

email= The sent parameter 'email' is identical for all requests and has the value WantDis@aol.com.

subject= The sent subjects are following a fixed pattern. They always contain the URL for the script which the attacker tried to exploit. From the returned answer the attacker will know which URL was successful.

```
http://server.workgroup.university.de/cgi-bin/FormMail.cgi
http://server.workgroup.university.de/cgi-bin/FormMail.pl
http://server.workgroup.university.de/cgi-bin/formmail.pl
http://server.workgroup.university.de/cgi-bin/formmail/formmail.cgi
http://server.workgroup.university.de/cgi-bin/formmail/FormMail.pl
http://server.workgroup.university.de/cgi-bin/FormMail.pl
http://server.workgroup.university.de/cgi-bin/formmail.pl
http://server.workgroup.university.de/cgi-bin/formmail/formmail.cgi
http://server.workgroup.university.de/cgi-bin/FormMail.cgi
http://server.workgroup.university.de/cgi-bin/formmail/formmail.pl
http://server.workgroup.university.de/cgi-bin/formmail.cgi
http://server.workgroup.university.de/cgi-bin/formmail.pl
http://server.workgroup.university.de/cgi-bin/formmail/formmail.cgi
http://server.workgroup.university.de/cgi-bin/formmail/formmail.pl
```

message= The sent messages are different for every request but have a similar structure. It seems as if they were generated at random.

| Script | Message |
|---|---|
| /cgi-bin/FormMail.cgi | rvqgd ddfyuynatbd eanbqjgblgb ioffys |
| /cgi-bin/FormMail.pl | hvxej uaunbmlzqzs uhuaxgeqbnj hnvdnh |
| /cgi-bin/formmail.pl | uvuwi lpelksuvrel iarscqtaypw jjwbhv |
| /cgi-bin/formmail/formmail.cg | arrkl ycjxhwirxrq nwvgydgektt efjvma |
| /cgi-bin/formmail/FormMail.pl | qnqft muvxtfujbzt dzubgyxrkyj pwnwpr |
| /cgi-bin/FormMail.pl | fbwdi aztmalrypyr tgazvfdpzmp ntucmn |
| /cgi-bin/formmail.pl | ggfia xaqwweghdpx tlcdocelqbi buitth |
| /cgi-bin/formmail/formmail.cg | xovhi vzguetfobou rasdvadbhqx bcgspe |
| /cgi-bin/FormMail.cgi | txsif gfhadhpdckf gcpdtsjdnie lqhhbb |
| /cgi-bin/formmail/formmail.pl | hqqbc gsjxfwzimhx vvuwplpesks uwreti |
| /cgi-bin/formmail.cgi | qjrtu qdrhzwrbwsz evvpivgnult mobvur |
| /cgi-bin/formmail.pl | lcleg tgmjsqcnzvt zhpztykhwxn xaezpt |
| /cgi-bin/formmail/formmail.cg | yzpsc gbroooqjmqy lltoplymbar mwrmtz |
| /cgi-bin/formmail/formmail.pl | skicn idkphhathsy gwfyanhgdmk whtwua |

### 2.1.3. Probability the source address was spoofed

The spamer/attacker had established a TCP three-way-handshake. Therefore, we know for sure that address 4.63.162.8 is not spoofed.

### 2.1.4. Description of attack

This attack targets web servers with faulty versions of the formmail script (for example [7]). Different vulnerabilities for the different versions of this class of scripts are known:

| Name | Description |
|---|---|
| CVE-1999-0172 | FormMail CGI program allows remote execution of commands. |
| CVE-1999-0173 | FormMail CGI program can be used by web servers other than the host server that the program resides on. |
| CVE-2000-0411 | Matt Wright's FormMail CGI script allows remote attackers to obtain environmental variables via the env_report parameter. |
| CAN-2001-0357 | FormMail.pl in FormMail 1.6 and earlier allows a remote attacker to send anonymous email (spam) by modifying the recipient and message parameters. |
| CAN-2004-0259 | The check_referer() function in Formmail.php 5.0 and earlier allows remote attackers to bypass access restrictions via an empty or spoofed HTTP Referer, as demonstrated using an application on the same web server that contains a cross-site scripting (XSS) issue. |

The attackers main goal is to find one of those scripts which he can manipulate in such a way that he is able to send emails to arbitrary addresses. In general, this is done by sending an extra parameter (e.g., recipient) that was not foreseen by the script. This parameter may

overwrite internal variables. If the attacker has found a vulnerable victim, he will use it to send spam mails. The emails will come from the victims IP address, so the spammer has built a barrier between himself and the spam recipients.

### 2.1.5. Attack mechanism

As stated before, in the current case, attacks were run at 3 different days. In total, 14 attempts took place with 7 different scripts. The below table describes the number of attempts per script. It springs into mind that the scripts were tested with different frequencies.

| Script | # of tries |
|---|---|
| /cgi-bin/formmail.pl | 3 |
| /cgi-bin/formmail/formmail.cg | 3 |
| /cgi-bin/FormMail.cgi | 2 |
| /cgi-bin/FormMail.pl | 2 |
| /cgi-bin/formmail/formmail.pl | 2 |
| /cgi-bin/formmail.cgi | 1 |
| /cgi-bin/formmail/FormMail.pl | 1 |

From the log files we know the source address of the attacker who came from a DSL account (lsanca2-ar36-4-63-162-008.lsanca2.dsl-verizon.net) of Verizon Online. Details about the source network 4.63.160.0/21 can be obtained with whois.

```
OrgName:    GTE Intelligent Network Services
OrgID:      GINS
Address:    5525 MacArthur Blvd.
Address:    Suite 320
City:       Irving
StateProv:  TX
PostalCode: 75038
Country:    US

NetRange:   4.63.160.0 - 4.63.167.255
CIDR:       4.63.160.0/21
NetName:    GTEINS-63-160-15
NetHandle:  NET-4-63-160-0-1
Parent:     NET-4-0-0-0-1
NetType:    Reassigned
Comment:    The information for POC handle VOH1-ARIN has been
Comment:    reported to be invalid. ARIN has attempted to obtain updated
Comment:    data, but has been unsuccessful. To provide current contact
Comment:    information, please email hostmaster@arin.net.
RegDate:    2002-05-16
Updated:    2003-06-03

TechHandle: VOH1-ARIN
TechName:   Hostmaster, Verizon Online
TechPhone:  +1-800-927-3000
```

```
TechEmail:  hostmaster@bizmailsrvcs.net

OrgAbuseHandle: VOH1-ARIN
OrgAbuseName:   Hostmaster, Verizon Online
OrgAbusePhone:  +1-800-927-3000
OrgAbuseEmail:  hostmaster@bizmailsrvcs.net

OrgNOCHandle: VOH1-ARIN
OrgNOCName:   Hostmaster, Verizon Online
OrgNOCPhone:  +1-800-927-3000
OrgNOCEmail:  hostmaster@bizmailsrvcs.net

OrgTechHandle: VOH1-ARIN
OrgTechName:   Hostmaster, Verizon Online
OrgTechPhone:  +1-800-927-3000
OrgTechEmail:  hostmaster@bizmailsrvcs.net
```

According to Apache's error log the attacker failed in each of the 14 attempts simply because the web server is not providing one of those scripts.

```
[Sun Aug 31 12:11:07 2003] [error] [client 4.63.162.8]
 script not found or unable to stat: /usr/lib/cgi-bin/FormMail.cgi
[Sun Aug 31 12:11:07 2003] [error] [client 4.63.162.8]
 script not found or unable to stat: /usr/lib/cgi-bin/FormMail.pl
[Sun Aug 31 12:11:08 2003] [error] [client 4.63.162.8]
 script not found or unable to stat: /usr/lib/cgi-bin/formmail.pl
[Sun Aug 31 12:11:08 2003] [error] [client 4.63.162.8]
 script not found or unable to stat: /usr/lib/cgi-bin/formmail
[Sun Aug 31 12:11:08 2003] [error] [client 4.63.162.8]
 script not found or unable to stat: /usr/lib/cgi-bin/formmail
[Mon Sep  1 22:09:23 2003] [error] [client 4.63.162.8]
 script not found or unable to stat: /usr/lib/cgi-bin/FormMail.pl
[Mon Sep  1 22:09:24 2003] [error] [client 4.63.162.8]
 script not found or unable to stat: /usr/lib/cgi-bin/formmail.pl
[Mon Sep  1 22:09:24 2003] [error] [client 4.63.162.8]
 script not found or unable to stat: /usr/lib/cgi-bin/formmail
[Mon Sep  1 22:09:32 2003] [error] [client 4.63.162.8]
 script not found or unable to stat: /usr/lib/cgi-bin/FormMail.cgi
[Mon Sep  1 22:09:33 2003] [error] [client 4.63.162.8]
 script not found or unable to stat: /usr/lib/cgi-bin/formmail
[Fri Sep  5 15:36:16 2003] [error] [client 4.63.162.8]
 script not found or unable to stat: /usr/lib/cgi-bin/formmail.cgi
[Fri Sep  5 15:36:16 2003] [error] [client 4.63.162.8]
 script not found or unable to stat: /usr/lib/cgi-bin/formmail.pl
[Fri Sep  5 15:36:17 2003] [error] [client 4.63.162.8]
 script not found or unable to stat: /usr/lib/cgi-bin/formmail
[Fri Sep  5 15:36:17 2003] [error] [client 4.63.162.8]
 script not found or unable to stat: /usr/lib/cgi-bin/formmail
```

### 2.1.6. Correlations

This kind of search for vulnerable formmail scripts is well known. Some websites even provide formmail scripts that were especially prepared for spammers. These script versions act like a vulnerable formmail script. However, in fact, they do not send email but do only generate statistics to count the connects. For example the page *xrea.com* [8] got different similar connects on September 26th, 2003 with identical `recipient`(piscesali@aol.com) and `email` (WantDis@aol.com) parameters. Here, the source address was 4.63.166.246 (lsanca2-ar36-4-63-166-246.lsanca2.dsl-verizon.net) which is also belonging to Verizon's DSL pool.

The *FormMail hall of shame* [9] got the same connect on September 4th, 2003 from the source 4.63.162.8 (lsanca2-ar36-4-63-162-008.lsanca2.dsl-verizon.net). Once again, a computer with DSL coming from Verizon.

The maintainers of the site Attrition.org tried to contact Verizon about this problem. Their page [10] documents their unsuccessful tries to get response. Finally, they banned Verizon from accessing Attrition.org.

### 2.1.7. Evidence of active targeting

As mentioned in the last section the spammer tried to access Formmail scripts on many different servers. The log entries are part of a general scan of large parts of the Internet.

### 2.1.8. Severity

Severity = (criticality + lethality) - (system countermeasures + network countermeasures)

**Severity=5** The targeted web server is a very valuable target. It is the only computer accessible through the firewall. This server is also providing many important services for the workgroup.

**Lethality=4** If the system had a vulnerable formmail script, high network traffic would be generated. If this will stay undetected, chances are high that the server (also working as regular mail server) or even the complete university will be included in blackmail lists as spammer. On the other hand, the data on the server itself will not be affected.

**System countermeasures=4** During installation, the system was hardened, and daily checks for new patches from Debian take place. The only reason, why I will not assign a 5 is, that this server combines too much functionality in one machine.

**Network countermeasures=3** The network is secured with a stateful packet filter and a Router with ACLs. However, knowing the university's network outside of the workgroup, network countermeasures are quite low.

Severity = (5+4)-(4+3)=2

### 2.1.9. Defensive recommendation

Considering the university's attitude towards network security, only local actions can be taken.

1. Services, accessible through the firewall should not stay any longer on the central server of the workgroup. Adding a DMZ to the firewall will enhance the security.

2. In the current case, access to cgi scripts is not really necessary. Access to those directories should be blocked. Especially, http://server.workgroup.university.de/cgi-bin/apcupsd/multimon.cgi is of no public interest.

3. It might be interesting to have a look at mod_security for the Apache [11].

### 2.1.10. Multiple choice test question

You are the system administrator for a workgroup at a university. Someday, you get an angry email, that your sever is sending spam. Which of the following lines from your Apache log tells you that you are really having a problem.

a) access.log:

```
1.2.3.4 - - [some date] "GET /cgi-bin/formmail.pl?
 realname=wapcv%20dfgcod&recipient=piscesali@aol.com&email=WantDis@aol.com
 &subject=http://server.workgroup.university.de/cgi-bin/formmail.pl
 &message=skicn%20idkphhathsy%20gwfyanhgdmk%20whtwua HTTP/1.1"  404
 325 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)"
```

b) access.log:

```
1.2.3.4 - - [some date] "GET /cgi-bin/formmail/formmail.pl?
 realname=wapcv%20dfgcod&recipient=piscesali@aol.com&email=WantDis@aol.com
 &subject=http://server.workgroup.university.de/cgi-bin/formmail/formmail.pl
 &message=skicn%20idkphhathsy%20gwfyanhgdmk%20whtwua HTTP/1.1"  200
 325 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)"
```

c) error.log:

```
[some date] [error] [client 1.2.3.4]
 script not found or unable to stat: /usr/lib/cgi-bin/formmail.pl
```

d) error.log:

```
[some date] [error] [client 1.2.3.4]
request failed: erroneous characters after protocol string: GET
 /cgi-bin/formmail.pl?email=f2%40aol%2Ecom&subject=server%2Eworkgroup
 %2Euniversity%2Ede%2Fcgi%2Dbin%2Fformmail%2Epl&recipient=davidsbabe61301
 %40aol%2Ecom&msg=w00t HTTP/1.1Content-Type: application/x-www-form-urlencoded
```

The correct answer is b)

a) The return code is 404, so this access was not successful.

b) The return code is 200, so this access is successful. It is high time to check the formmail.pl script.

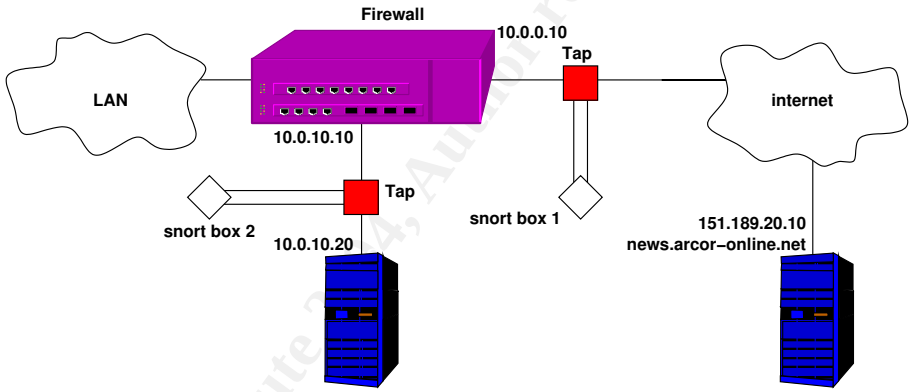
c) No script found, so there is no problem.

d) The request contained an error this is not a problem.

## 2.2. He does not like snort?

### 2.2.1. Source of trace

The second detect is from a productive IDS of a company. Two computers running snort are connected to the network with Ethernet taps. One box is sniffing the traffic between the firewall and the Internet. The other box is monitoring the traffic between the firewall and the DMZ. In order to be able to correlate the detects, it must be granted that the two snort boxes are working on a synchronized time base, which, in this case, was achieved by using NTP, or in detail, by using the same NTP server.




Alerts from both systems are collected in the same MySQL Database and analyzed using ACID v0.9.6b22, a PHP-based GUI.

### 2.2.2. Detect was generated by

Both intrusion detection systems generated an alert following the below rule:

```
alert ip any any -> any any (msg:"ATTACK-RESPONSES id check returned root";
 content:"uid=0|28|root|29|"; classtype:bad-unknown; sid:498; rev:6;)
```

Snort box 1 (Internet) generated the following alert:

```
Generated by ACID v0.9.6b22 on Day Month XX, 2004 XX:XX:XX

------------------------------------------------------------------------------
#(1 - 114797) [2004-YY-YY XX:XX:XX.9364+02]
      [snort/498]  ATTACK-RESPONSES id check returned root
IPv4: 151.189.20.10 -> 10.0.0.10
      hlen=5 TOS=0 dlen=1500 ID=14912 flags=0 offset=0 TTL=60 chksum=XXXXX
TCP:  port=24732 -> dport: 119  flags=***A**** seq=1021705143
      ack=1013121476 off=8 res=0 win=17152 urp=0 chksum=XXXX
      Options:
       #1 - NOP len=0
       #2 - NOP len=0
       #3 - TS len=8 data=XXXXXXXXXXXXXXXX
Payload:  length = 1448

000 : 54 41 4B 45 54 48 49 53 20 3C 32 41 71 46 46 2D   TAKETHIS <2AqFF-
010 : 31 7A 4A 2D 31 35 40 67 61 74 65 64 2D 61 74 2E   1zJ-15@gated-at.
020 : 62 6F 66 68 2E 69 74 3E 0D 0A 50 61 74 68 3A 20   bofh.it>..Path:
030 : 6E 65 77 73 31 2E 64 74 61 67 2E 64 65 21 74 61   news.arcor.de!ta
040 : 6B 65 6D 79 2E 6E 65 77 73 2E 74 65 6C 65 66 6F   kemy.news.telefo
050 : 6E 69 63 61 2E 64 65 21 74 65 6C 65 66 6F 6E 69   nica.de!telefoni
060 : 63 61 2E 64 65 21 62 6F 72 69 75 6D 2E 62 6F 78   ca.de!borium.box
070 : 2E 6E 6C 21 6E 65 77 73 68 75 62 33 2E 68 6F 6D   .nl!newshub3.hom
080 : 65 2E 6E 6C 21 68 6F 6D 65 2E 6E 6C 21 6E 65 77   e.nl!home.nl!new
090 : 73 32 2E 74 65 6C 65 62 79 74 65 2E 6E 6C 21 6E   s2.telebyte.nl!n
0a0 : 65 77 73 2E 6E 65 77 73 6C 61 6E 64 2E 69 74 21   ews.newsland.it!
0b0 : 6E 65 77 73 66 65 65 64 2E 6E 65 74 74 75 6E 6F   newsfeed.nettuno
0c0 : 2E 69 74 21 65 72 6F 64 65 2E 62 6F 66 68 2E 69   .it!erode.bofh.i
0d0 : 74 21 62 6F 66 68 2E 69 74 21 6E 65 77 73 2E 6E   t!bofh.it!news.n
0e0 : 69 63 2E 69 74 21 72 6F 62 6F 6D 6F 64 0D 0A 46   ic.it!robomod..F
0f0 : 72 6F 6D 3A 20 53 61 6D 20 48 6F 63 65 76 61 72   rom: Sam Hocevar
100 : 20 3C 73 61 6D 40 7A 6F 79 2E 6F 72 67 3E 0D 0A    <sam@zoy.org>..
110 : 4E 65 77 73 67 72 6F 75 70 73 3A 20 6C 69 6E 75   Newsgroups: linu
120 : 78 2E 64 65 62 69 61 6E 2E 62 75 67 73 2E 64 69   x.debian.bugs.di
130 : 73 74 0D 0A 53 75 62 6A 65 63 74 3A 20 42 75 67   st..Subject: Bug
140 : 23 32 36 39 37 39 39 3A 20 6C 69 62 68 69 67 68   #269799: libhigh
150 : 67 75 69 30 2E 39 2D 35 3A 20 53 65 67 6D 65 6E   gui0.9-5: Segmen
160 : 74 61 74 69 6F 6E 20 66 61 75 6C 74 20 69 6E 20   tation fault in
170 : 7E 43 76 76 49 6D 61 67 65 46 69 6C 74 65 72 73   ~CvvImageFilters
180 : 0D 0A 44 61 74 65 3A 20 46 72 69 2C 20 30 33 20   ..Date: Fri, 03
190 : 53 65 70 20 32 30 30 34 20 32 30 3A 31 30 3A 30   Sep 2004 20:10:0
1a0 : 37 20 2B 30 32 30 30 0D 0A 4D 65 73 73 61 67 65   7 +0200..Message
1b0 : 2D 49 44 3A 20 3C 32 41 71 46 46 2D 31 7A 4A 2D   -ID: <2AqFF-1zJ-
1c0 : 31 35 40 67 61 74 65 64 2D 61 74 2E 62 6F 66 68   15@gated-at.bofh
1d0 : 2E 69 74 3E 0D 0A 52 65 66 65 72 65 6E 63 65 73   .it>..References
1e0 : 3A 20 3C 32 41 6F 6B 4A 2D 38 62 37 2D 34 35 40   : <2AokJ-8b7-45@
1f0 : 67 61 74 65 64 2D 61 74 2E 62 6F 66 68 2E 69 74   gated-at.bofh.it
200 : 3E 0D 0A 58 2D 4F 72 69 67 69 6E 61 6C 2D 54 6F   >..X-Original-To
210 : 3A 20 51 69 6E 67 6E 69 6E 67 20 48 75 6F 20 3C   : Qingning Huo <
220 : 71 69 6E 67 6E 69 6E 67 68 40 6C 61 6E 77 61 72   qingningh@lanwar
230 : 65 2E 63 6F 2E 75 6B 3E 2C 20 32 36 39 37 39 39   e.co.uk>, 269799
240 : 40 62 75 67 73 2E 64 65 62 69 61 6E 2E 6F 72 67   @bugs.debian.org
```

```
250 : 0D 0A 4F 6C 64 2D 52 65 74 75 72 6E 2D 50 61 74   ..Old-Return-Pat
260 : 68 3A 20 3C 64 65 62 62 75 67 73 40 62 75 67 73   h: <debbugs@bugs
270 : 2E 64 65 62 69 61 6E 2E 6F 72 67 3E 0D 0A 52 65   .debian.org>..Re
280 : 70 6C 79 2D 54 6F 3A 20 53 61 6D 20 48 6F 63 65   ply-To: Sam Hoce
290 : 76 61 72 20 3C 73 61 6D 40 7A 6F 79 2E 6F 72 67   var <sam@zoy.org
2a0 : 3E 2C 20 32 36 39 37 39 39 40 62 75 67 73 2E 64   >, 269799@bugs.d
2b0 : 65 62 69 61 6E 2E 6F 72 67 0D 0A 52 65 73 65 6E   ebian.org..Resen
2c0 : 74 2D 54 6F 3A 20 64 65 62 69 61 6E 2D 62 75 67   t-To: debian-bug
2d0 : 73 2D 64 69 73 74 40 6C 69 73 74 73 2E 64 65 62   s-dist@lists.deb
2e0 : 69 61 6E 2E 6F 72 67 0D 0A 52 65 73 65 6E 74 2D   ian.org..Resent-
2f0 : 43 63 3A 20 53 61 6D 20 48 6F 63 65 76 61 72 20   Cc: Sam Hocevar
300 : 28 44 65 62 69 61 6E 20 70 61 63 6B 61 67 65 73   (Debian packages
310 : 29 20 3C 73 61 6D 2B 64 65 62 40 7A 6F 79 2E 6F   ) <sam+deb@zoy.o
320 : 72 67 3E 0D 0A 58 2D 44 65 62 69 61 6E 2D 50 72   rg>..X-Debian-Pr
330 : 2D 4D 65 73 73 61 67 65 3A 20 72 65 70 6F 72 74   -Message: report
340 : 20 32 36 39 37 39 39 0D 0A 58 2D 44 65 62 69 61    269799..X-Debia
350 : 6E 2D 50 72 2D 50 61 63 6B 61 67 65 3A 20 6C 69   n-Pr-Package: li
360 : 62 68 69 67 68 67 75 69 30 2E 39 2D 35 0D 0A 58   bhighgui0.9-5..X
370 : 2D 44 65 62 69 61 6E 2D 50 72 2D 4B 65 79 77 6F   -Debian-Pr-Keywo
380 : 72 64 73 3A 20 70 61 74 63 68 0D 0A 4D 49 4D 45   rds: patch..MIME
390 : 2D 56 65 72 73 69 6F 6E 3A 20 31 2E 30 0D 0A 43   -Version: 1.0..C
3a0 : 6F 6E 74 65 6E 74 2D 54 79 70 65 3A 20 74 65 78   ontent-Type: tex
3b0 : 74 2F 70 6C 61 69 6E 3B 20 63 68 61 72 73 65 74   t/plain; charset
3c0 : 3D 75 73 2D 61 73 63 69 69 0D 0A 43 6F 6E 74 65   =us-ascii..Conte
3d0 : 6E 74 2D 44 69 73 70 6F 73 69 74 69 6F 6E 3A 20   nt-Disposition:
3e0 : 69 6E 6C 69 6E 65 0D 0A 4D 61 69 6C 2D 43 6F 70   inline..Mail-Cop
3f0 : 69 65 73 2D 54 6F 3A 20 6E 65 76 65 72 0D 0A 58   ies-To: never..X
400 : 2D 4E 6F 2D 43 63 3A 20 49 20 72 65 61 64 20 6D   -No-Cc: I read m
410 : 61 69 6C 69 6E 67 2D 6C 69 73 74 73 3B 20 64 6F   ailing-lists; do
420 : 20 6E 6F 74 20 43 43 20 6D 65 20 6F 6E 20 72 65    not CC me on re
430 : 70 6C 69 65 73 2E 0D 0A 58 2D 53 6E 6F 72 74 3A   plies...X-Snort:
440 : 20 75 69 64 3D 30 28 72 6F 6F 74 29 20 67 69 64    uid=0(root) gid
450 : 3D 30 28 72 6F 6F 74 29 0D 0A 55 73 65 72 2D 41   =0(root)..User-A
460 : 67 65 6E 74 3A 20 4D 75 74 74 2F 31 2E 35 2E 34   gent: Mutt/1.5.4
470 : 69 0D 0A 58 2D 52 63 2D 56 69 72 75 73 3A 20 32   i..X-Rc-Virus: 2
480 : 30 30 34 2D 30 37 2D 32 30 5F 30 31 0D 0A 58 2D   004-07-20_01..X-
490 : 52 63 2D 53 70 61 6D 3A 20 32 30 30 34 2D 30 38   Rc-Spam: 2004-08
4a0 : 2D 32 39 5F 30 31 0D 0A 58 2D 4D 61 69 6C 69 6E   -29_01..X-Mailin
4b0 : 67 2D 4C 69 73 74 3A 20 3C 64 65 62 69 61 6E 2D   g-List: <debian-
4c0 : 62 75 67 73 2D 64 69 73 74 40 6C 69 73 74 73 2E   bugs-dist@lists.
4d0 : 64 65 62 69 61 6E 2E 6F 72 67 3E 20 0D 0A 4C 69   debian.org> ..Li
4e0 : 73 74 2D 49 44 3A 20 3C 64 65 62 69 61 6E 2D 62   st-ID: <debian-b
4f0 : 75 67 73 2D 64 69 73 74 2E 6C 69 73 74 73 2E 64   ugs-dist.lists.d
500 : 65 62 69 61 6E 2E 6F 72 67 3E 0D 0A 41 70 70 72   ebian.org>..Appr
510 : 6F 76 65 64 3A 20 72 6F 62 6F 6D 6F 64 40 6E 65   oved: robomod@ne
520 : 77 73 2E 6E 69 63 2E 69 74 0D 0A 4C 69 6E 65 73   ws.nic.it..Lines
530 : 3A 20 31 38 0D 0A 4F 72 67 61 6E 69 7A 61 74 69   : 18..Organizati
540 : 6F 6E 3A 20 6C 69 6E 75 78 2E 2A 20 6D 61 69 6C   on: linux.* mail
550 : 20 74 6F 20 6E 65 77 73 20 67 61 74 65 77 61 79    to news gateway
560 : 0D 0A 53 65 6E 64 65 72 3A 20 72 6F 62 6F 6D 6F   ..Sender: robomo
570 : 64 40 6E 65 77 73 2E 6E 69 63 2E 69 74 0D 0A 58   d@news.nic.it..X
580 : 2D 4F 72 69 67 69 6E 61 6C 2D 43 63 3A 20 44 65   -Original-Cc: De
```

```
590 : 62 69 61 6E 20 42 75 67 20 54 72 61 63 6B 69 6E   bian Bug Trackin
5a0 : 67 20 53 79 73 74 65 6D                           g System
```

Snort box 2 (DMZ) generated the following alert:

```
Generated by ACID v0.9.6b22 on Day Month XX, 2004 XX:XX:XX

------------------------------------------------------------------------------
#(2 - 14595) [2004-YY-YY XX:XX:XX.941616+02]
      [snort/498]  ATTACK-RESPONSES id check returned root
IPv4: 10.0.10.10 -> 10.0.10.20
      hlen=5 TOS=0 dlen=1500 ID=13856 flags=0 offset=0 TTL=63 chksum=XXXX
TCP:  port=36005 -> dport: 119  flags=***A**** seq=2024033782
      ack=1356143190 off=8 res=0 win=33580 urp=0 chksum=XXXXX
      Options:
       #1 - NOP len=0
       #2 - NOP len=0
       #3 - TS len=8 data=XXXXXXXXXXXXXXXX
Payload:  length = 1448

000 : 54 41 4B 45 54 48 49 53 20 3C 32 41 71 46 46 2D   TAKETHIS <2AqFF-
010 : 31 7A 4A 2D 31 35 40 67 61 74 65 64 2D 61 74 2E   1zJ-15@gated-at.
020 : 62 6F 66 68 2E 69 74 3E 0D 0A 50 61 74 68 3A 20   bofh.it>..Path:
030 : 6E 65 77 73 31 2E 64 74 61 67 2E 64 65 21 74 61   news.arcor.de!ta
040 : 6B 65 6D 79 2E 6E 65 77 73 2E 74 65 6C 65 66 6F   kemy.news.telefo
050 : 6E 69 63 61 2E 64 65 21 74 65 6C 65 66 6F 6E 69   nica.de!telefoni
060 : 63 61 2E 64 65 21 62 6F 72 69 75 6D 2E 62 6F 78   ca.de!borium.box
070 : 2E 6E 6C 21 6E 65 77 73 68 75 62 33 2E 68 6F 6D   .nl!newshub3.hom
080 : 65 2E 6E 6C 21 68 6F 6D 65 2E 6E 6C 21 6E 65 77   e.nl!home.nl!new
090 : 73 32 2E 74 65 6C 65 62 79 74 65 2E 6E 6C 21 6E   s2.telebyte.nl!n
0a0 : 65 77 73 2E 6E 65 77 73 6C 61 6E 64 2E 69 74 21   ews.newsland.it!
0b0 : 6E 65 77 73 66 65 65 64 2E 6E 65 74 74 75 6E 6F   newsfeed.nettuno
0c0 : 2E 69 74 21 65 72 6F 64 65 2E 62 6F 66 68 2E 69   .it!erode.bofh.i
0d0 : 74 21 62 6F 66 68 2E 69 74 21 6E 65 77 73 2E 6E   t!bofh.it!news.n
0e0 : 69 63 2E 69 74 21 72 6F 62 6F 6D 6F 64 0D 0A 46   ic.it!robomod..F
0f0 : 72 6F 6D 3A 20 53 61 6D 20 48 6F 63 65 76 61 72   rom: Sam Hocevar
100 : 20 3C 73 61 6D 40 7A 6F 79 2E 6F 72 67 3E 0D 0A    <sam@zoy.org>..
110 : 4E 65 77 73 67 72 6F 75 70 73 3A 20 6C 69 6E 75   Newsgroups: linu
120 : 78 2E 64 65 62 69 61 6E 2E 62 75 67 73 2E 64 69   x.debian.bugs.di
130 : 73 74 0D 0A 53 75 62 6A 65 63 74 3A 20 42 75 67   st..Subject: Bug
140 : 23 32 36 39 37 39 39 3A 20 6C 69 62 68 69 67 68   #269799: libhigh
150 : 67 75 69 30 2E 39 2D 35 3A 20 53 65 67 6D 65 6E   gui0.9-5: Segmen
160 : 74 61 74 69 6F 6E 20 66 61 75 6C 74 20 69 6E 20   tation fault in
170 : 7E 43 76 76 49 6D 61 67 65 46 69 6C 74 65 72 73   ~CvvImageFilters
180 : 0D 0A 44 61 74 65 3A 20 46 72 69 2C 20 30 33 20   ..Date: Fri, 03
190 : 53 65 70 20 32 30 30 34 20 32 30 3A 31 30 3A 30   Sep 2004 20:10:0
1a0 : 37 20 2B 30 32 30 30 0D 0A 4D 65 73 73 61 67 65   7 +0200..Message
1b0 : 2D 49 44 3A 20 3C 32 41 71 46 46 2D 31 7A 4A 2D   -ID: <2AqFF-1zJ-
1c0 : 31 35 40 67 61 74 65 64 2D 61 74 2E 62 6F 66 68   15@gated-at.bofh
1d0 : 2E 69 74 3E 0D 0A 52 65 66 65 72 65 6E 63 65 73   .it>..References
1e0 : 3A 20 3C 32 41 6F 6B 4A 2D 38 62 37 2D 34 35 40   : <2AokJ-8b7-45@
1f0 : 67 61 74 65 64 2D 61 74 2E 62 6F 66 68 2E 69 74   gated-at.bofh.it
200 : 3E 0D 0A 58 2D 4F 72 69 67 69 6E 61 6C 2D 54 6F   >..X-Original-To
```

```
210 : 3A 20 51 69 6E 67 6E 69 6E 67 20 48 75 6F 20 3C   : Qingning Huo <
220 : 71 69 6E 67 6E 69 6E 67 68 40 6C 61 6E 77 61 72   qingningh@lanwar
230 : 65 2E 63 6F 2E 75 6B 3E 2C 20 32 36 39 37 39 39   e.co.uk>, 269799
240 : 40 62 75 67 73 2E 64 65 62 69 61 6E 2E 6F 72 67   @bugs.debian.org
250 : 0D 0A 4F 6C 64 2D 52 65 74 75 72 6E 2D 50 61 74   ..Old-Return-Pat
260 : 68 3A 20 3C 64 65 62 62 75 67 73 40 62 75 67 73   h: <debbugs@bugs
270 : 2E 64 65 62 69 61 6E 2E 6F 72 67 3E 0D 0A 52 65   .debian.org>..Re
280 : 70 6C 79 2D 54 6F 3A 20 53 61 6D 20 48 6F 63 65   ply-To: Sam Hoce
290 : 76 61 72 20 3C 73 61 6D 40 7A 6F 79 2E 6F 72 67   var <sam@zoy.org
2a0 : 3E 2C 20 32 36 39 37 39 39 40 62 75 67 73 2E 64   >, 269799@bugs.d
2b0 : 65 62 69 61 6E 2E 6F 72 67 0D 0A 52 65 73 65 6E   ebian.org..Resen
2c0 : 74 2D 54 6F 3A 20 64 65 62 69 61 6E 2D 62 75 67   t-To: debian-bug
2d0 : 73 2D 64 69 73 74 40 6C 69 73 74 73 2E 64 65 62   s-dist@lists.deb
2e0 : 69 61 6E 2E 6F 72 67 0D 0A 52 65 73 65 6E 74 2D   ian.org..Resent-
2f0 : 43 63 3A 20 53 61 6D 20 48 6F 63 65 76 61 72 20   Cc: Sam Hocevar
300 : 28 44 65 62 69 61 6E 20 70 61 63 6B 61 67 65 73   (Debian packages
310 : 29 20 3C 73 61 6D 2B 64 65 62 40 7A 6F 79 2E 6F   ) <sam+deb@zoy.o
320 : 72 67 3E 0D 0A 58 2D 44 65 62 69 61 6E 2D 50 72   rg>..X-Debian-Pr
330 : 2D 4D 65 73 73 61 67 65 3A 20 72 65 70 6F 72 74   -Message: report
340 : 20 32 36 39 37 39 39 0D 0A 58 2D 44 65 62 69 61    269799..X-Debia
350 : 6E 2D 50 72 2D 50 61 63 6B 61 67 65 3A 20 6C 69   n-Pr-Package: li
360 : 62 68 69 67 68 67 75 69 30 2E 39 2D 35 0D 0A 58   bhighgui0.9-5..X
370 : 2D 44 65 62 69 61 6E 2D 50 72 2D 4B 65 79 77 6F   -Debian-Pr-Keywo
380 : 72 64 73 3A 20 70 61 74 63 68 0D 0A 4D 49 4D 45   rds: patch..MIME
390 : 2D 56 65 72 73 69 6F 6E 3A 20 31 2E 30 0D 0A 43   -Version: 1.0..C
3a0 : 6F 6E 74 65 6E 74 2D 54 79 70 65 3A 20 74 65 78   ontent-Type: tex
3b0 : 74 2F 70 6C 61 69 6E 3B 20 63 68 61 72 73 65 74   t/plain; charset
3c0 : 3D 75 73 2D 61 73 63 69 69 0D 0A 43 6F 6E 74 65   =us-ascii..Conte
3d0 : 6E 74 2D 44 69 73 70 6F 73 69 74 69 6F 6E 3A 20   nt-Disposition:
3e0 : 69 6E 6C 69 6E 65 0D 0A 4D 61 69 6C 2D 43 6F 70   inline..Mail-Cop
3f0 : 69 65 73 2D 54 6F 3A 20 6E 65 76 65 72 0D 0A 58   ies-To: never..X
400 : 2D 4E 6F 2D 43 63 3A 20 49 20 72 65 61 64 20 6D   -No-Cc: I read m
410 : 61 69 6C 69 6E 67 2D 6C 69 73 74 73 3B 20 64 6F   ailing-lists; do
420 : 20 6E 6F 74 20 43 43 20 6D 65 20 6F 6E 20 72 65    not CC me on re
430 : 70 6C 69 65 73 2E 0D 0A 58 2D 53 6E 6F 72 74 3A   plies...X-Snort:
440 : 20 75 69 64 3D 30 28 72 6F 6F 74 29 20 67 69 64    uid=0(root) gid
450 : 3D 30 28 72 6F 6F 74 29 0D 0A 55 73 65 72 2D 41   =0(root)..User-A
460 : 67 65 6E 74 3A 20 4D 75 74 74 2F 31 2E 35 2E 34   gent: Mutt/1.5.4
470 : 69 0D 0A 58 2D 52 63 2D 56 69 72 75 73 3A 20 32   i..X-Rc-Virus: 2
480 : 30 30 34 2D 30 37 2D 32 30 5F 30 31 0D 0A 58 2D   004-07-20_01..X-
490 : 52 63 2D 53 70 61 6D 3A 20 32 30 30 34 2D 30 38   Rc-Spam: 2004-08
4a0 : 2D 32 39 5F 30 31 0D 0A 58 2D 4D 61 69 6C 69 6E   -29_01..X-Mailin
4b0 : 67 2D 4C 69 73 74 3A 20 3C 64 65 62 69 61 6E 2D   g-List: <debian-
4c0 : 62 75 67 73 2D 64 69 73 74 40 6C 69 73 74 73 2E   bugs-dist@lists.
4d0 : 64 65 62 69 61 6E 2E 6F 72 67 3E 20 0D 0A 4C 69   debian.org> ..Li
4e0 : 73 74 2D 49 44 3A 20 3C 64 65 62 69 61 6E 2D 62   st-ID: <debian-b
4f0 : 75 67 73 2D 64 69 73 74 2E 6C 69 73 74 73 2E 64   ugs-dist.lists.d
500 : 65 62 69 61 6E 2E 6F 72 67 3E 0D 0A 41 70 70 72   ebian.org>..Appr
510 : 6F 76 65 64 3A 20 72 6F 62 6F 6D 6F 64 40 6E 65   oved: robomod@ne
520 : 77 73 2E 6E 69 63 2E 69 74 0D 0A 4C 69 6E 65 73   ws.nic.it..Lines
530 : 3A 20 31 38 0D 0A 4F 72 67 61 6E 69 7A 61 74 69   : 18..Organizati
540 : 6F 6E 3A 20 6C 69 6E 75 78 2E 2A 20 6D 61 69 6C   on: linux.* mail
```

```
550 : 20 74 6F 20 6E 65 77 73 20 67 61 74 65 77 61 79    to news gateway
560 : 0D 0A 53 65 6E 64 65 72 3A 20 72 6F 62 6F 6D 6F   ..Sender: robomo
570 : 64 40 6E 65 77 73 2E 6E 69 63 2E 69 74 0D 0A 58   d@news.nic.it..X
580 : 2D 4F 72 69 67 69 6E 61 6C 2D 43 63 3A 20 44 65   -Original-Cc: De
590 : 62 69 61 6E 20 42 75 67 20 54 72 61 63 6B 69 6E   bian Bug Trackin
5a0 : 67 20 53 79 73 74 65 6D                           g System
```

The above is a typical example of the output that ACID generates while emailing the events from the web based GUI.

```
#(2 - 14595) [2004-YY-YY XX:XX:XX.941616+02]
      [snort/498]  ATTACK-RESPONSES id check returned root
```

#(6 - 14595) is an internal identifier, followed by time and date of the alert. The event was triggered by snort rule 498 ([snort/498])

```
IPv4: 10.0.10.10 -> 10.0.10.20
      hlen=5 TOS=0 dlen=1500 ID=13856 flags=0 offset=0 TTL=63 chksum=XXXX
```

This is the IPv4 header. It contains source (10.0.10.10) and destination (10.0.10.20) address, followed by the other parts of the IPv4 headers.

```
TCP:  port=36005 -> dport: 119  flags=***A**** seq=2024033782
      ack=1356143190 off=8 res=0 win=33580 urp=0 chksum=XXXXX
      Options:
       #1 - NOP len=0
       #2 - NOP len=0
       #3 - TS len=8 data=XXXXXXXXXXXXXXXX
```

This is the TCP header with source and destination port, flags (only ACK set) and other header information.

Although the header of the two alerts is completely different, the payload is exactly identical.

```
TAKETHIS <2AqFF-1zJ-15@gated-at.bofh.it>
Path: news.arcor.de!takemy.news.telefonica.de!telefonica.de!
 borium.box.nl!newshub3.home.nl!home.nl!news2.telebyte.nl!
 news.newsland.it!newsfeed.nettuno.it!erode.bofh.it!bofh.it!
 news.nic.it!robomod
From: Sam Hocevar <sam@zoy.org>
Newsgroups: linux.debian.bugs.dist
Subject: Bug#269799: libhighgui0.9-5: Segmentation fault in ~CvvImageFilters
Date: Fri, 03 Sep 2004 20:10:07 +0200
Message-ID: <2AqFF-1zJ-15@gated-at.bofh.it>
References: <2AokJ-8b7-45@gated-at.bofh.it>
X-Original-To: Qingning Huo <qingningh@lanware.co.uk>, 269799@bugs.debian.org
Old-Return-Path: <debbugs@bugs.debian.org>
Reply-To: Sam Hocevar <sam@zoy.org>, 269799@bugs.debian.org
Resent-To: debian-bugs-dist@lists.debian.org
Resent-Cc: Sam Hocevar (Debian packages) <sam+deb@zoy.org>
X-Debian-Pr-Message: report 269799
X-Debian-Pr-Package: libhighgui0.9-5
```

```
X-Debian-Pr-Keywords: patch
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Disposition: inline
Mail-Copies-To: never
X-No-Cc: I read mailing-lists; do not CC me on replies.
X-Snort: uid=0(root) gid=0(root)
User-Agent: Mutt/1.5.4i
X-Rc-Virus: 2004-07-20_01
X-Rc-Spam: 2004-08-29_01
X-Mailing-List: <debian-bugs-dist@lists.debian.org>
List-ID: <debian-bugs-dist.lists.debian.org>
Approved: robomod@news.nic.it
Lines: 18
Organization: linux.* mail to news gateway
Sender: robomod@news.nic.it
X-Original-Cc: Debian Bug Tracking System
```

From the key word `TAKETHIS` and from the destination ports we can conclude that snort has observed the communication between two NNTP-Servers exchanging a message. However, only a part of the NNTP header is visible. The content that triggered both alerts starts at byte 441 and reads `uid=0(root)`. This might be the result of successfully exploiting the NNTP-server. We will check this in more detail.

The source of the first packet is the NNTP-server of Arcor, one of the largest German Internet Providers. The recipient is the firewall. The second packet was sent from the firewall to the companies NNTP-Server, located in the DMZ. In both cases, the source port is a high port and the destination port is 119, the port assigned to NNTP service.

The first question to be resolved is, whether the two packets are referring to the same event. Comparing both headers it seems, that these two packets are belonging to two separate TCP connections. As mentioned before, the payload is identical. In addition, the time difference between both detects ($.941616 - 0.9364 = 5.2ms$) is so small, that it seems unreasonable that there is a normal NNTP-server working on the firewall, receiving the message and sending it to the NNTP-server 10.0.10.20.

What helps in understanding the above is some basic details we know of the firewall. As the great differences between the two TCP/IP headers suggest, this firewall is not a packet filter with Network Address Translation. Instead, this firewall is an Application Gateway. Any TCP connection from the Internet ends at the firewall (packet 1 from 151.189.20.10 to the external address 10.0.0.10). Furthermore, another connection is established between the firewall 10.0.10.10 and the "real" destination 10.0.10.20, the NNTP-server. This means, we have to investigate only one incident, both snort boxes had seen the same thing.

### 2.2.3. Probability the source address was spoofed

Due to the internal structure of the firewall, the TCP-connection between the DMZ-interface and the internal NNTP-server is only initiated after the three-way-handshake between the NNTP-server of ARCOR and the external interface is established. So, it is extremely unlikely that the source addresses 151.189.20.10 and 10.0.10.10 are spoofed.

## 2.2.4. Description of attack

Snort itself declares rule 498 as `ATTACK-RESPONSES`. This rule is not for a specific attack, but tries to detect a typical behavior of a hacker after a successful attack. After the exploit is sent, the attacker checks his id. Other snort rules of this category are searching for strings like `1 file(s) copied` if the hacker tried to copy a file or `Directory of` as a response to a `dir`-command. Depending on the type of traffic, these rules can trigger false positives, if a user is reading those advisories mentioned below. On the other hand, if you happen to see such kind of traffic on connections which are normally encrypted it is high time to get your companies incidence response procedure manual.

Let us have a look at some examples of "Advisories for hackers" found by google. The below is an example of the comment of an ssh exploit [12]:

```
% drop-root -v24 localhost
* ½%.2022u%24$hn@localhost's password:
* Connection closed by 127.0.0.1
*
* % telnet localhost 10275
* Trying 127.0.0.1...
* Connected to localhost.
* Escape character is '^]'.
* id; exit;
* uid=0(root) gid=0(root) groups=0(root)
* Connection closed by foreign host.
```

Even advisories [13] use this command as "proof" that the exploit was successful.

```
elguapo@gentoo tmp $ ./0x82-BRU_overformat 1

0x82-BRU_overformat - backup and restore utility (BRU) local root exploit.
Target package: BRU-15.1-3.i386.rpm

[*] shellcode: 0xbffff9e
 [*] It's my message:
KFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFK...
KFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFKFthanks!!ÿÿ>
sh-2.05b# id
uid=0(root) gid=0(root) groups=100(users),10(wheel)
```

As rule 498 is just searching for the string `uid=0(root)`, downloading this practical as HTML might trigger your companies snort system.

## 2.2.5. Attack mechanism

The suspicious text `uid=0(root)` is part of the line `X-Snort: uid=0(root) gid=0(root)`. This line itself is part of the header of an NNTP message. Obviously, this is not a usual NNTP header line. To evaluate whether this line is legal traffic it is advisable, to read the according RFC. RFC 850 defines the "Standard for Interchange of USENET Messages" and defines the following:

```
[...]
An article consists of several header lines, followed by a
blank line, followed by the body of the message. The
header lines consist of a keyword, a colon, a blank, and
some additional information. [...]
Certain headers are required, certain headers are
optional. Any unrecognized headers are allowed, and will
be passed through unchanged. [...]
```

The most important sentence is: "Any unrecognized headers are allowed, and will be passed through unchanged." This means, that the `X-Snort` line is absolutely correct, and covered by the RFC. We can only speculate about the senders motives. According to the `From:` header the sender was Sam Hocevar. He is maintainer of several debian packages and had found a buffer overflow in the rinetd daemon. But we cannot be sure, that this header was added by him.

1. The sender can be easily forged.
2. The header line could have been added by any NNTP server the message had passed.

What are the effects of this header? First, nearly every snort-based IDS will be triggered, and thus, time of the analysts will be wasted. Second, this message might be blocked if it is sent through an inline-snort based Intrusion Prevention System.

### 2.2.6. Correlations

No information was found about this kind of provocation.

### 2.2.7. Evidence of active targeting

Due to the structure of the NNTP protocol, this message will be distributed all over the world. The nasty sender has no chance to direct this traffic towards a specific destination. Instead, he is bothering every SNORT based IDS which is monitoring NNTP-traffic.

### 2.2.8. Severity

Severity = (Criticality + Lethality) - (System Countermeasures + Network Countermeasures)

**Severity=1** The NNTP system hosts no other services. Loosing this server will not be a great problem.

**Lethality=1** The lethality is extremely low, as this is a (provoked) false positive.

**System countermeasures=4** The NNTP-system is regularly patched and tested and the firewall is accepting NNTP-messages only from a certain peer.

**Network countermeasures=5** he companies network is designed for In-depth defense and is protected by a professional and up-to-date firewall. We can calculate with a 5.

Severity$= (1+1) - (4+5) = -7$

The severity is extremely low, although the IDS manager was quite angry when he had seen this packet for the first time.

## 2.2.9. Defensive recommendation

It is very hard to defend this kind of traffic. And, as this traffic has a very low severity, fighting this traffic might cost more time and money than ignoring it. But, if you really want to take measures, you can do the following:

1. Contact the possible sender in the hope that he will stop this. This will only work if he is really the sender and if he is willing to stop.
2. Change the snort rule. One can, for example add `content: !"X-Snort";` to the rule.
3. Try to filter the traffic at the firewall either by removing this annoying header or by rejecting every message with this header. This will of course not work, if the sender decides to change the string.
4. Try to filter the traffic and let only those headers pass, which are really necessary according to RFC 850.

## 2.2.10. Multiple choice test question

In your companies network, snort has alerted four packets with the following signature

```
alert ip any any -> any any (msg:"ATTACK-RESPONSES id check returned root";
content:"uid=0|28|root|29|"; classtype:bad-unknown; sid:498; rev:6;)
```

Which of theses packets is the most severe?

a)
```
22:59:37 IP external.www > internal.3142: ...
0000  4e 6f 77 20 65 6e 74 65  72 20 69 64 20 61 6e 64  Now enter id and
0010  20 79 6f 75 20 77 69 6c  6c 20 73 65 65 20 77 68   you will see wh
0020  65 74 68 65 72 20 74 68  65 20 65 78 70 6c 6f 69  ether the exploi
0030  74 20 77 61 73 20 73 75  63 63 65 73 73 66 75 6c  t was successful
0040  6c 3c 62 72 3e 68 6f 73  74 23 20 69 64 3c 62 72  l<br>host# id<br
0050  3e 75 69 64 3d 30 28 72  6f 6f 74 29 20 67 69 64  >uid=0(root) gid
0060  3d 30 28 72 6f 6f 74 29  3c 62 72 3e              =0(root)<br>
```

b)
```
16:31:12 IP internal.telnet > external.32772: ...
0000  75 69 64 3d 30 28 72 6f  6f 74 29 20 67 69 64 3d  uid=0(root) gid=
0010  30 28 72 6f 6f 74 29 20  67 72 6f 75 70 73 3d 30  0(root) groups=0
0020  28 72 6f 6f 74 29 0d 0a  69 6e 74 65 72 6e 23     (root)..intern#
```

c)
```
10:32:91 IP internal.ssh > external.32772: ...
0000  75 69 64 3d 30 28 72 6f  6f 74 29 20 67 69 64 3d  uid=0(root) gid=
0010  30 28 72 6f 6f 74 29 20  67 72 6f 75 70 73 3d 30  0(root) groups=0
0020  28 72 6f 6f 74 29 0d 0a  69 6e 74 65 72 6e 23     (root)..intern#
```

d)
```
15:37:64 IP external.4367 > internal.nntp: ...
0000  4d 61 69 6c 2d 43 6f 70  69 65 73 2d 54 6f 3a 20  Mail-Copies-To:
0010  6e 65 76 65 72 0a 58 2d  53 6e 6f 72 74 3a 20 75  never.X-Snort: u
0020  69 64 3d 30 28 72 6f 6f  74 29 20 67 69 64 3d 30  id=0(root) gid=0
0030  28 72 6f 6f 74 29 0a 55  73 65 72 2d 41 67 65 6e  (root).User-Agen
0040  74 3a 20 4d 75 74 74 2f  31 2e 35 2e 34 69 0a     t: Mutt/1.5.4i.
```

a) (wrong) The traffic is part of a normal web-page. Maybe a user is reading how to become a hacker.

b) (wrong) Working as root with telnet is not advisable, but may be necessary for some reason.

c) (correct) You see plain text on an encrypted connection and the text is typical for hacker behavior after starting an exploit. Here, chances are very high that your ssh server was exploited.

d) (wrong) An example of a correct, but nasty header for NNTP-Traffic. Although this is extremely impolite, it must be accepted.

## 2.3. Just looking!

This detect was posted to intrusionsincidents.org on September, $18^{th}$, 2004. We did not receive any response untill the submission of the practical.

```
Message-Id: <200409180933.57286.schinner@acm.org>
X-OriginalArrivalTime: 18 Sep 2004 12:45:55.0766 (UTC)
  FILETIME=[70882560:01C49D7D]
```

### 2.3.1. Source of trace

The original tcpdump log file `2002.10.13` was downloaded from the given site [14]. According to SANS Institute, the network traffic was captured using an unknown version of snort with an unknown rule set. The data has been sanitized.

Before analyzing any network traffic it is highly recommended to have some basic knowledge about the network itself. Without this knowledge, a lot of questions cannot be answered and it will be hard to judge the severity of an attack. As the only data provided by GIAC is the tcpdump log file, we will at first try to make good, reasonable guesses on the network.

Using the program ipanalyze, we see that 245 IP-addresses are contributing to this log:

```
host> ipanalyze -r ~/SANS/raw/2002.10.13 -o '%d\n%s' | sort -u | wc -l
245
```

The next step of our little network analysis is to check the frequency of the different IP addresses grouped by class B networks. With some luck, the most frequent among those should be the local network.

```
207.166.0.0/16    3109
64.154.0.0/16     1086
66.159.0.0/16     663
205.188.0.0/16    430
209.11.0.0/16     117
63.111.0.0/16     107
209.10.0.0/16     71
64.12.0.0/16      48
255.255.0.0/16    35
207.68.0.0/16     30
```

The biggest block of addresses occurs in the subnet 207.166.0.0/16 with 68 addresses ranging from 207.166.11.232 to 207.166.252.249. By checking the MAC address, we try to prove that this is the local network.

If the MAC addresses in network 207.166.0.0/16 are mostly belonging to NIC manufacturers, the computer is most probably part of a LAN. This would be another strong hint for the assumption that 207.166.0.0/16 is the local network.

```
> ipanalyze -r 2002.10.13 -o '%m\n%M' | sort -u
00:00:0c:04:b2:33
00:03:e3:d9:26:c0
```

Bad luck, only two MAC addresses were found, both belonging to CISCO. This means, we are sniffing between two switches, routers or firewalls. Now we can try to get some information by mapping the MAC addresses to the IP addresses.

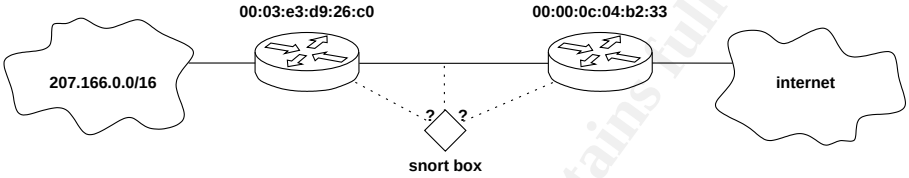
```
ipanalyze -r 2002.10.13 -o '%M %s\n%m %d' | sort -u

00:00:0c:04:b2:33 12.11.133.5
00:00:0c:04:b2:33 12.111.47.194
00:00:0c:04:b2:33 12.47.193.41
[... deleted some lines ...]
00:00:0c:04:b2:33 80.4.53.36
00:00:0c:04:b2:33 80.6.223.204
00:00:0c:04:b2:33 81.98.104.191
00:03:e3:d9:26:c0 207.166.103.217
00:03:e3:d9:26:c0 207.166.104.170
[... deleted some lines ...]
00:03:e3:d9:26:c0 207.166.87.53
00:03:e3:d9:26:c0 207.166.95.105
00:03:e3:d9:26:c0 207.166.98.123
```

Obviously, all computers in the subnet 207.166.0.0/16 can be reached through the MAC address 00:03:e3:d9:26:c0, all other addresses are connected to 00:00:0c:04:b2:33. The fact that computers in Parsippany, NJ (2.11.133.5) and Tokyo, Japan (219.163.126.118) can be accessed over the same MAC address, belonging to CISCO, tells us, that this device is a router or firewall connected to the Internet.

The local network seems to be behind a CISCO device, which will have the following MAC address 00:03:e3:d9:26:c0. However, as this data has been sanitized by GIAC, it can be assumed that these addresses are not the original ones. The device could be a router or a firewall, but we can not be sure.

Important for an analysis is where and with what kind of method we are capturing the network traffic. Can the dump tell us how the snort box is sniffing? The IDS can be connected using a SPAN port, a Hub or an Ethernet tap. All these methods do not influence the network, therefore we do not gain any knowledge about the type of connection.




## 2.3.2. Detect was generated by

The detect was generated by snort version 2.2.0 (build 30) using the command line:

```
snort -c /etc/snort/snort.conf -k none -r 2002.10.13
```

Using the option `-t` was important as due to the sanitizing the header checksums are incorrect. Only with this option snort is willing to analyze packets with bad checksums. The generated alerts had been saved in a MySQL database and were viewed with ACID v0.9.6b20-5.1. We will analyze the following three detects in more detail:

```
Generated by ACID v0.9.6b20-5.1 on Sat, 11 Sep 2004 19:49:09 +0200

(3 - 3780) [2002-11-13 01:21:55] url[snort/615]  SCAN SOCKS Proxy attempt
IPv4: 66.159.18.49 -> 207.166.87.157
      hlen=5 TOS=0 dlen=60 ID=49542 flags=0 offset=0 TTL=53 chksum=21100
TCP:  port=48451 -> dport: 1080  flags=******S* seq=2275796995
      ack=0 off=10 res=0 win=5840 urp=0 chksum=41682
      Options:
       #1 - MSS len=2 data=05B4
       #2 - SACKOK len=0
       #3 - TS len=8 data=01F1298500000000
       #4 - NOP len=0
       #5 - WS len=1 data=00
Payload: none
(3 - 3781) [2002-11-13 01:21:55] [snort/620]  SCAN Proxy Port 8080 attempt
IPv4: 66.159.18.49 -> 207.166.87.157
      hlen=5 TOS=0 dlen=60 ID=18547 flags=0 offset=0 TTL=53 chksum=52095
```

```
TCP:  port=48452 -> dport: 8080  flags=******S* seq=2272616959
      ack=0 off=10 res=0 win=5840 urp=0 chksum=3495
      Options:
       #1 - MSS len=2 data=05B4
       #2 - SACKOK len=0
       #3 - TS len=8 data=01F1298C00000000
       #4 - NOP len=0
       #5 - WS len=1 data=00
Payload: none
-----------------------------------------------------------------------------
#(3 - 3782) [2002-11-13 01:21:55] [snort/618]  SCAN Squid Proxy attempt
IPv4: 66.159.18.49 -> 207.166.87.157
      hlen=5 TOS=0 dlen=60 ID=3550 flags=0 offset=0 TTL=53 chksum=1557
TCP:  port=48453 -> dport: 3128  flags=******S* seq=2282038346
      ack=0 off=10 res=0 win=5840 urp=0 chksum=24091
      Options:
       #1 - MSS len=2 data=05B4
       #2 - SACKOK len=0
       #3 - TS len=8 data=01F1299400000000
       #4 - NOP len=0
       #5 - WS len=1 data=00
Payload: none
```

The notation of the output format is described in the analysis of the last detect. Please refer to section 2.2.

These above alerts had been triggered by the following snort signatures:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 8080
 (msg:"SCAN Proxy Port 8080 attempt"; flags:S,12; flow:stateless;
 classtype:attempted-recon; sid:620; rev:10;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 3128
 (msg:"SCAN Squid Proxy attempt"; flags:S,12; flow:stateless;
 classtype:attempted-recon; sid:618; rev:9;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 1080
 (msg:"SCAN SOCKS Proxy attempt"; flags:S,12; flow:stateless;
 reference:url,help.undernet.org/proxyscan/; classtype:attempted-recon;
 sid:615; rev:9;)
```

The only functional difference between these three signatures is the destination port.

### 2.3.3. Probability the source address was spoofed

The alert showed, that there were three attempts to connect to different ports. Most probably, the attacker wanted to check whether the ports are accessible. Spoofing the source address would be useless, as in this case the scanner would get no answer. An exception might be, if the sender was able to monitor the traffic going to the forged source address. However, chances are high, that the address was not spoofed.

### 2.3.4. Description of attack

The three connects follow the same scheme. The attacker tried to initiate a three way handshake to proxy services which should normally be accessible only by users of the local network. Then, after he had found a proxy accepting his requests, he had the following different possibilities.

1. Attack hosts on the LAN in the hope to use some trusted relationships
2. Attack or connect hosts on the outside, hiding behind the proxies IP-address
3. Attack the proxy service itself by using some exploits.

### 2.3.5. Attack mechanism and correlations

In the current case, all three connects came within 0.15 seconds. This means the attacker was using an automated tool. Was the connect successful? It is hard to decide this directly from the tcpdump file, as we see no answer from 207.166.87.157 either accepting or rejecting the connection. However, there is some evidence that this connect was not successful:

1. We see no other packet originating from or going to the attackers address (66.159.18.49).
2. No other traffic from outside the network is directed to ports 3128 and 1080. There are only two other connects to port 8080. One can assume, that an open proxy might have attracted much more traffic which would have been recorded and reported by snort.

On the other hand, 207.166.87.157 might be the local proxy. Every connection going to port 80 on the outside network is coming from 207.166.87.157.

What can we say about the tool that was used by the attacker? The first guess would be that the connects had not been initiated by a human attacker but by a Trojan, the well known ZeroRing. Three connects to port 1080, 8080, and 3128 are typical for this malware. Everything we need to know about this Trojan was already described by Steven Northcut [15] so I do not want to bother the reader by repeating his words. However, also according to S. Northcut [16], some attackers are hiding other attacks behind a typical ZeroRing behavior.

A search of the CVE database for keywords like squid, proxy or socks results in such a tremendous lot of entries that it might in fact be a good idea to hide behind a well known pattern while attempting new attacks.

What can we say about the attacker himself? Searching dnsstuff.com for 66.159.18.49 we get the following information:

```
OrgName:    IIC Internet
OrgID:      IICINT
Address:    17905 Vista Court
City:       Santa Clarita
StateProv:  CA
PostalCode: 91387
Country:    US
```

```
NetRange:   66.159.16.0 - 66.159.20.255
CIDR:       66.159.16.0/22, 66.159.20.0/24
NetName:    WLCO-TWC874610-IICINT
NetHandle:  NET-66-159-16-0-1
Parent:     NET-66-159-0-0-1
NetType:    Reassigned
NameServer: STLDNS1.WCG.NET
NameServer: TULDNS1.WCG.NET
Comment:
RegDate:    2002-07-19
Updated:    2002-07-19

TechHandle: CH1236-ARIN
TechName:   Herzig, Chris
TechPhone:  +1-661-298-1438
TechEmail:  chris@iicinternet.com
```

The company itself, according to their homepage, seems to work in the web-hosting business. We searched dshield.org and mynetwatchman.com to see if there is activity reported for 66.159.18.49. Both did not return any hit.

The program p0f, a passive OS fingerprinting utility, tells us, that the attacker is using Linux.

```
p0f - passive os fingerprinting utility, version 2.0.3
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on '2002.10.13', 207 sigs (12 generic), rule: 'host 66.159.18.49'.
66.159.18.49:48451 - Linux 2.4/2.6 (up: 90 hrs)
  -> 207.166.87.157:1080 (distance 11, link: ethernet/modem)
66.159.18.49:48452 - Linux 2.4/2.6 (up: 90 hrs)
  -> 207.166.87.157:8080 (distance 11, link: ethernet/modem)
66.159.18.49:48453 - Linux 2.4/2.6 (up: 90 hrs)
  -> 207.166.87.157:3128 (distance 11, link: ethernet/modem)
```

As the Trojan is running on Windows systems but the source seems to be a Linux box, the theory about the ZeroRing attack might be improper.

Another piece of information is the link [17] which we find in the snort signature "Proxy attempt" (sid:615). According to this page the UnderNet server is scanning everybody who is connecting to their service:

> Due to the overwhelming abuse of misconfigured Wingate, Socks and Proxy servers being exploited daily, the UnderNet network is now checking all users upon connection to any of the UnderNet IRC Servers. This check is ONLY DONE if a user attempts to establish a connection to an UnderNet IRC server.This should not be considered an attack on your system. Be aware that this sort of connection to your system is probably common if you use services such as free IRC networks, game servers, etc...

Unfortunately, no connection to UnderNet IP-address 209.198.2.21 was made, which would have explained the detect. The last sentence in the message taken from the URL [17] gives

us a next hint for our search. Maybe our target 207.166.87.157 had made connections which might have triggered this traffic. ACID gives us the following alerts for the host:

| | |
|---|---|
| CHAT IRC nick change | 718 |
| SHELLCODE x86 NOOP | 25 |
| SCAN nmap TCP | 18 |
| CHAT MSN message | 3 |
| SCAN Proxy Port 8080 attempt | 1 |
| SCAN SOCKS Proxy attempt | 1 |
| SCAN Squid Proxy attempt | 1 |

Great, there is IRC traffic, don't forget UnderNet is an IRC server, too. Most of the traffic (680 packets) is going to server.iicinternet.com (66.159.18.68). And this IP belongs to the same range, the proxy scan came from. So this is another strong evidence that the origin was not a ZeroRing but triggered by IRC traffic.

Unfortunately, we have a problem with the timestamps. The alerts based on the IRC traffic appeared between 2002-11-13 20:26:49 and 2002-11-13 20:22:57. The proxy scan was 19 hours earlier, at 2002-11-13 01:21:55. The tcpdump does not tell us whether there was IRC traffic before 20:26:49. The type of IRC traffic that we would need to proove the above assumptions would normally not be registered by snort. The registered IRC traffic, however, is suspicious in itself. 680 Nick changes to `R00teD` would need another analysis not given here. Everything points to the fact that the attack was not triggered by ZeroRing but by the target itself.

### 2.3.6. Evidence of active targeting

If we assume that the attacker was infected by the ZeroRing Trojan, targeting 209.198.2.21 was only a random hit. Alternatively, host 207.166.87.157 might have attracted this scan by entering some shady areas of the Internet. The is no evidence from this data (ignoring the nick changes) that 207.166.87.157 is under attack.

### 2.3.7. Severity

Severity = (Criticality + Lethality) - (System Countermeasures + Network Countermeasures)

**Severity=3** Host 207.166.87.157 seems to be in heavy use on the victims network. However, without complete information it is hard to decide its criticality.

**Lethality=4** Here we should differentiate between the two possible sources. If it was a ZeroRing, the severity would be quite high, maybe even a 5. If it was the reaction of the IRC-server, there will be no harm to the system, so maybe a 1. As chances are very high, we are not dealing with ZeroRing, we choose 4.

**System countermeasures=2** Judging from the other traffic of this host, there is no draconian system administrator. It is hard to decide how well the network is patched.

**Network countermeasures=2** Due to the estimated network, it seems reasonable to assume, that there is at least a packet filter firewall or ACLs on the router.

Severity$= (3 + 4) - (2 + 2) = 3$

## 2.3.8. Defensive recommendation

The first steps should be at least a stateful packet filter firewall in front of the organizations network. Judging from the seen traffic, there are no restrictions for the users, resulting in highly suspicious traffic. The management of this organization must also face the fact that users are surfing for hard core porn. Checking this and the other log files with a patched[1] version of driftnet (driftnet is comparable to dsniff, however, it sniffs for pictures and not for passwords), one sees pictures showing every detail of the human anatomy striking very explicit poses.

## 2.3.9. Multiple choice test question

According to some IDS specialists, the Trojan ZeroRing tries to connect the ports 1080, 8080 and 3128 of a possible victim in always exactly this order.

In your log files, you found three connects in the following order: 8080, 1080 and 3128. Which statement is not correct?

a) Maybe it is a modified version of ZeroRing

b) It can't be ZeroRing, as this is the wrong order

c) It might be an original ZeroRing, but the packets arrived in another order than they had been sent.

d) Maybe someone is hiding behind a behavior very similar to ZeroRing.

The correct answer is b)

a) This is possible, of course.

b) A wrong statement, see a) and c)

c) Today not very likely, but possible

d) This is possible, see http://www.sans.org/y2k/050300-1100.htm

[1]The original version of driftnet cannot read tcpdump files. In this case, a combination of driftnet and tcpreplay mostly works.

# 3. Analyze this

## 3.1. Overview

All files for the below analysis were downloaded from the site mentioned by the GCIA Practical Assignment Version 3.5. The task of this report is to improve the network security at the university and to show up acute problems. To achieve this, we analyze the provided data and suggest suitable actions to improve the security. The only information provided to us were the IDS system log files of 5 subsequent days.

No information about network structure, operating system, etc. was available. All according facts need to be guessed from the log-files. Diverse university net services can be accessed via Internet. Among those, there are official hosts, like mail server, Novell server, and more than 340 web servers.

Several of the alarms reported by the IDS are analyzed in detail. Depending on the type of alarm, we have to deal with different problems. Some alerts point to the fact that the IDS was also used as a monitoring tool. Other alerts show computers which have been attacked and were taken over by the attacker. Furthermore, some alerts were found that suggest a misuse of university IP-addresses. Some problems regarding the configuration of the network and the functioning of the DHCP-server are also discussed.

Due to the huge amount of data, it is impossible to analyze each and every single alarm. In order to classify the computers, top-talker lists are generated for different aspects. In addition, a link graph and the registration information of several important computers outside the university should simplify the classification of the diverse alarms.

Afterwards, specific suggestions are made to improve the network security. We separate short, middle, and long-term actions and discuss the responsibilities for the different actions.

At the end of the report, the applied methods are summarized for transparency.

## 3.2. A list of the files

Three different file types are available and were investigated. All files originate from an unknown version of snort. Moreover, the available data is not the original but a modified one. The GIAC page says:

> The logs themselves have been sanitized. All of the IP addresses of the protected network space have been "munged". Additionally, the checksums have been modified to prevent clever people from discovering the original IP addresses. You will find that certain keywords within the packets have been replaced with "X"s. All ICMP, DNS, SMTP and Web traffic has also been removed.

This will have some effects on the analysis of the data, as some errors were added to the files during this procedure.

The following files were downloaded from `http://isc.sans.org/logs/` and cover the time from May, 27-31, 2004.

| Date | Alerts | OOS Report | Scans |
|---|---|---|---|
| 27.3.2004 | alert.040327.gz | oos_report_040323 | scans.040327.gz |
| 28.3.2004 | alert.040328.gz | oos_report_040324 | scans.040328.gz |
| 29.3.2004 | alert.040329.gz | oos_report_040325 | scans.040329.gz |
| 30.3.2004 | alert.040330.gz | oos_report_040326 | scans.040330.gz |
| 31.3.2004 | alert.040331.gz | oos_report_040327 | scans.040331.gz |

Please pay attention to the fact that the files for the OOS reports have different timestamps than the alert and scan files. Also, the timestamps inside the OOS reports differ from the date, the filename suggests. For this analysis, it was assumed that the timestamps inside the files are correct and the filenames are faulty.

Before data could be analyzed, it needed some slight preparation. Please consult appendix A for more detailed information.

## 3.3. Relationships between the different computers

To estimate the size of the university network the number of unique addresses from the alerts and OOS files were counted. The scan files where ignored, because an IP-address appearing in a scan log does not necessarily mean, the according system does exist.

In total, 268 IP addresses appeared as source address of an alert or OOS event. Ignoring the fact that these addresses might be forged, we assume that these systems do really exist.

1052 IP addresses were found as destination address for alert and OOS events. Here, chances are higher that not all addresses found are belonging to a real machine. However, as we had ignored scans from the alert files, this number seems reasonable. Altogether, 1320 unique addresses of the university were found.

A first estimation of the function of different computers can be made based on the used well-known ports. The usual suspects are http (80), https (443), dns (53), mail (25), ssh (22), ftp (21), and telnet (23). High volume ports must also be investigated.

Nowadays, web servers are the most used services in the Internet. In total, 347 web servers could be identified. Hosts like userpages.MY.NET (MY.NET.24.44) with 4096 hits and www.MY.NET (MY.NET.24.34) with 1600 hits look like official servers. Printers like ss513-printer1.MY.NET (MY.NET.10.203), geography-printer4.MY.NET (MY.NET.10.24) or lib-hc-printer3.MY.NET (MY.NET.150.33) or management systems like webadmin2.MY.NET (MY.NET.24.58) should not be available over the Internet. There is no need to configure a printer from outside of the university.

The number of HTTPS-servers is much lower than the number of HTTP web servers.

| Name | IP | Hits |
|---|---|---|
| webmail.MY.NET | MY.NET.24.74 | 24 |
| webauth.MY.NET | MY.NET.12.7 | 12 |
| lan2.MY.NET | MY.NET.30.4 | 7 |
| lan1.MY.NET | MY.NET.30.3 | 7 |
| your.MY.NET | MY.NET.24.48 | 4 |
| my.MY.NET | MY.NET.24.33 | 2 |

Identifying the name server is tricky. As all alerts for connections to port 53 are scans, we cannot say for sure, behind which addresses we can find DNS-servers. However, two servers have much more traffic than all other machines

| Name | IP | Hits |
|---|---|---|
| MY3.MY.NET | MY.NET.1.3 | 472 |
| MY4.MY.NET | MY.NET.1.4 | 59 |

The DNS names suggest that they are official university mail servers.

| Name | IP | Hits |
|---|---|---|
| mxin.MY.NET | MY.NET.12.6 | 366 |
| listproc.MY.NET | MY.NET.24.20 | 21 |
| mdx.MY.NET | MY.NET.60.38 | 3 |

Ssh connections are registered for only four addresses, exchanging only few packets. These are extremely low numbers. Normally, in a university environment, remote connections over ssh are very common. Of course, snort does not register every ssh connection. Therefore, it is difficult to achieve more detailed knowledge on the ssh server.

| Name | IP | Hits |
|---|---|---|
| lan1.MY.NET | MY.NET.30.3 | 4 |
| linux2.gl.MY.NET | MY.NET.60.16 | 4 |
| lan2.MY.NET | MY.NET.30.4 | 3 |
| linux3.gl.MY.NET | MY.NET.60.39 | 1 |

For only two hosts lan1.MY.NET and lan2.MY.NET 6 telnet connections were detected. As telnet is a very insecure protocol, such a low number is good.

FTP connections are registered for eight addresses. According to some signatures the hosts MY.NET.53.29, MY.NET.70.49, and MY.NET.70.50 are belonging to the HelpDesk.

| Name | IP | Hits |
|---|---|---|
| ftp1.MY.NET | MY.NET.24.47 | 133 |
| ragnarok.MY.NET | MY.NET.24.27 | 41 |
| lan2.MY.NET | MY.NET.30.4 | 12 |
| lan1.MY.NET | MY.NET.30.3 | 11 |
| ecs020pc06.ucslab.MY.NET | MY.NET.53.29 | 5 |
| ? | MY.NET.42.1 | 4 |
| ecs020pc-15.ucs.MY.NET | MY.NET.70.50 | 3 |
| ecs020pc-14.ucs.MY.NET | MY.NET.70.49 | 2 |

MY.NET.24.8 seems to be the university Usenet server.

| IP | Hits |
|---|---|
| MY.NET.24.8 | 123 |
| MY.NET.30.4 | 3 |
| MY.NET.30.3 | 3 |

The last server, which could easily be detected, is the NTP time-server MY3.MY.NET (MY.NET.1.3) with 2301 hits.

Up to now, we had searched for well known services. It is of course possible that other services are also important for this site. So, we should have a look at the most frequent destination ports, maybe we can identify another server and service.

| Name | IP | Dest. Port | Hits |
|---|---|---|---|
| lan1.MY.NET.edu | MY.NET.30.3 | 524 | 17516 |
| lan2.MY.NET.edu | MY.NET.30.4 | 524 | 3619 |
| lan2.MY.NET.edu | MY.NET.30.4 | 51443 | 13582 |
| lan1.MY.NET.edu | MY.NET.30.3 | 3019 | 6730 |
| ? | MY.NET.97.82 | 1122 | 5460 |
| ecs021pc34.ucslab.MY.NET.edu | MY.NET.53.111 | 3658 | 1228 |
| eds-lin1.engr.MY.NET.EDU | MY.NET.110.72 | 12203 | 450 |

The above analysis has shown us that the servers lan1.MY.NET.edu and lan2.MY.NET.edu are playing an important role in the university infrastructure. For nearly all of the seen ports it was difficult to identify the network protocol. But for a few we can make a first guess.

**MY.NET.30.4:51443** According to different sites found in google, Novell NetWare 6.0 is available through this port.

**MY.NET.110.72:12203** Searching for this port, google gives us the hint that this port is used for Online Gaming. A server for *Medal Of Honor* seems to be bound to this port.

**MY.NET.30.3:524**

**MY.NET.30.4:524** Once again, google is pointing us to Novell NetWare [18]. NDS communicates on port 524 TCP and UDP.

We thus know that the university is using Novell network. Important servers for this service are lan1.MY.NET.edu and lan2.MY.NET.edu. These systems are highly valuable targets. Also important are the servers MY3.MY.NET (MY.NET.1.3) and MY4.MY.NET (MY.NET.1.4) which have handmade snort rules to monitor the complete traffic. Some web servers (http and https) and other services (e.g., NTP) could also be identified from the data.

## 3.4. A list of detects

Ignoring port scans, we found 53 unique alerts from the alert files. Different combinations of flags ranging from a NULL scan to a full XMAS scan add another 110 different alerts for the out-of-spec files. Additionally, more than 22.500.000 port scans are contributing to the data we have to analyze. So, only a small selection of the data can be reviewed in detail. The analyzed signatures were selected for three reasons:

**Frequency** Two signatures, producing more than 54% of all alerts must not be ignored.

**Severity** A signature, pointing to a highly dangerous event must be investigated, even if we see only few according packets.

**Custom signatures** We assume, that the maintainer of the university IDS has written the custom signatures for certain reasons.

The most frequent signatures cover nearly 95% of all triggered alerts.

| | Signature | Hits |
|---|---|---|
| 1 | MY.NET.30.3 activity | 28207 |
| 2 | MY.NET.30.4 activity | 21296 |
| 3 | High port 65535 tcp - possible Red Worm - traffic | 13423 |
| 4 | EXPLOIT x86 NOOP | 9343 |
| 5 | Incomplete Packet Fragments Discarded | 5357 |
| 6 | SMB Name Wildcard | 5164 |
| 7 | Null scan! | 1304 |
| 8 | High port 65535 udp - possible Red Worm - traffic | 1239 |
| 9 | TFTP - Internal UDP connection to external TFTP server | 1157 |
| 10 | Traffic from port 53 to port 123 | 1154 |

As the original snort signatures were not provided by SANS Institute, one can only guess which signatures were changed from the original snort.org version. However, some of the signatures are containing strings, which tell us that these signatures are customized. 15 signatures were found which were containing the string `MY.NET` (Abbreviation of the university name) or references to IP-addresses of the university network (e.g., MY.NET.53.29).

| | Signature | Hits |
|---|---|---|
| 1 | MY.NET.30.3 activity | 28207 |
| 2 | MY.NET.30.4 activity | 21296 |
| 3 | [MY.NET NIDS IRC Alert] IRC user /kill detected & possible trojan. | 616 |
| 4 | [MY.NET NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC | 479 |
| 5 | [MY.NET NIDS] External MiMail alert | 133 |
| 6 | [MY.NET NIDS IRC Alert] XDCC client detected attempting to IRC | 52 |
| 7 | [MY.NET NIDS IRC Alert] Possible drone command detected. | 11 |
| 8 | [MY.NET NIDS IRC Alert] Possible Incoming XDCC Send Request Detected. | 10 |
| 9 | [MY.NET NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot | 9 |
| 10 | External FTP to HelpDesk MY.NET.53.29 | 5 |
| 11 | External FTP to HelpDesk MY.NET.70.50 | 3 |
| 12 | [MY.NET NIDS] Internal MiMail alert | 2 |
| 13 | External FTP to HelpDesk MY.NET.70.49 | 2 |
| 14 | [MY.NET NIDS IRC Alert] K\:line'd user detected & possible trojan. | 1 |
| 15 | [MY.NET NIDS IRC Alert] User joining Warez channel detected. Possible XDCC bot | 1 |

The first 4 of the following sections were chosen based on their number of occurrences. The other detects were chosen for their severity or importance on the network configuration.

### 3.4.1. MY.NET.30.3 activity and MY.NET.30.4 activity

These two signatures are custom signatures with a very high frequency. None of the two signatures seems to be very selective about the traffic. In both cases, the according IP-address must be the destination. As more than 1350 different destination ports for MY.NET.30.3 and 1330 for MY.NET.30.4 were found, the signature might accept any destination port. No source address from the university network MY.NET.0.0/16 was found. Each system was contacted both on typical tcp ports (www, 80) and typical udp (tftp, 69) ports. We assume that the signature has the below structure.

```
alert ip $EXTERNAL_NET any -> MY.NET.30.3 any {
msg: "MY.NET.30.3 activity"; }

alert ip $EXTERNAL_NET any -> MY.NET.30.4 any {
msg: "MY.NET.30.4 activity"; }
```

There are two possibilities why the university might have chosen the above signatures. One could be that these rules are not used for network based intrusion detection but for generating statistics. Such a time profile can be seen in figure 3.1. Another might be that MY.NET.30.3 and MY.NET.30.4 are honeypots which are monitored with this IDS. However, in this case it is unclear, why only traffic directed to these addresses is logged but no traffic coming from them.

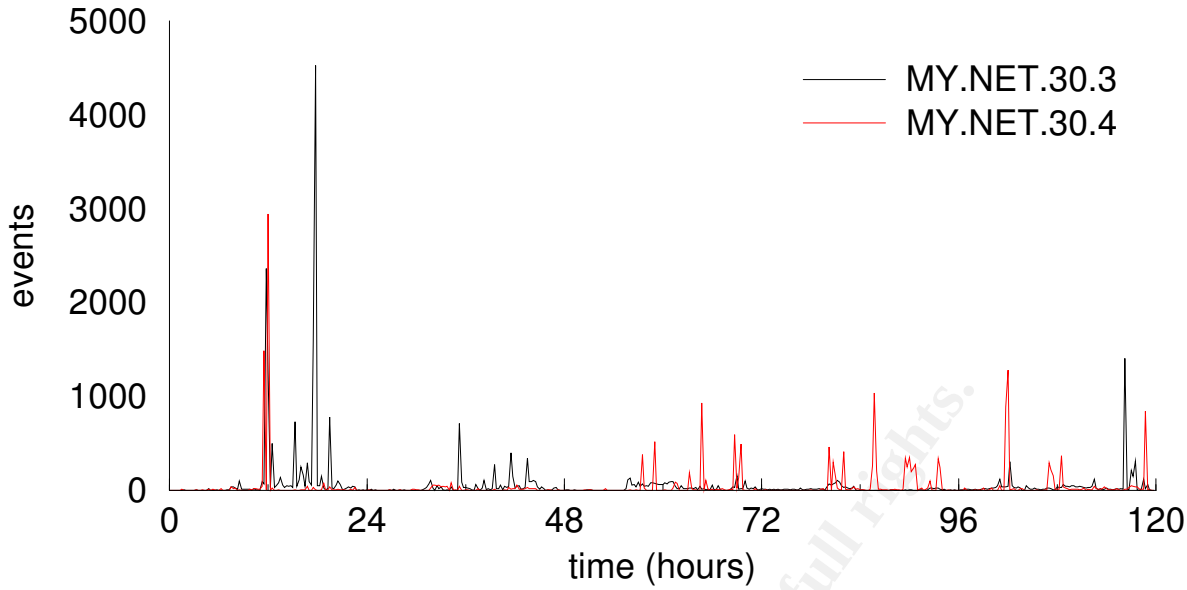



Figure 3.1.: Time profile of events for MY.NET.30.3 and MY.NET.30.4

### 3.4.2. High port 65535 tcp - possible Red Worm - traffic High port 65535 udp - possible Red Worm - traffic

These two messages point towards the Red Worm also known as Adore. None of these signatures are part of the default rules set by snort.org. Red Worm, not to be mix up with Code Red, a worm for Windows systems, is targeting Linux systems with vulnerable wu-ftpd, named, rpc.statd and lpd services [19, 20]. For both signatures, traffic is registered coming from and going to our university. Obviously, it does not matter, whether port 65536 is source or destination port.

We assume that the signature has the below structure.

```
alert tcp any any <> any 65535 {
msg: "High port 65535 tcp - possible Red Worm - traffic"; }

alert udp any any <> MY.NET.30.4 any {
msg: "High port 65535 udp - possible Red Worm - traffic"; }
```

Doug Kite showed some increasing destination port numbers for traffic originating from port 65535. He came to the conclusion that this is typical of traceroute or other mapping tools [21]. In our case, no increasing port numbers can be found.

Glenn Lareratt [22] came to the conclusion, that many systems at MY.NET university are infected with Adore. Philip.ljungberg@kbc.be supposed a connection between AFS, Adore, TFTP and ICMP alerts [23]. We cannot comment this, as no AFS correlated alerts were found. Maybe the snort rule set has changed since his analysis.

Another aspect is, that traffic from port 65535 is generally not forbidden and may appear in daily traffic. For example, the communication between client 68.55.121.177:65535 and web server MY.NET.29.3:80 could be regular, unsuspicious traffic. Red worm traffic should

mainly be directed against ports udp/53, tcp/21, and tcp/515. However, no packet to ports 515 or 21 was found; only 13 packets to port 53 were detected. This does not fit in the picture of an actively searching Red Worm. This traffic might be the combination of regular traffic and some scanning tools.

### 3.4.3. EXPLOIT x86 NOOP

For snort version 2.2.0RC1 the according rules would be

```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any
(msg:"SHELLCODE x86 NOOP";
content:"|90 90 90 90 90 90 90 90 90 90 90 90 90 90|";
depth:128; reference:arachnids,181;
classtype:shellcode-detect; sid:648; rev:7;)

alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any
(msg:"SHELLCODE x86 NOOP"; content:"aaaaaaaaaaaaaaaaaaaa";
classtype:shellcode-detect; sid:1394; rev:5;)
```

These signatures try to detect buffer overflows based on the fact, that the necessary shellcode is often padded with NOOP instructions [24, 25]. As David Oborn [26] pointed out in his analysis, false positives are quite usual for this kind of events. These sequences are quite common in different files like images, sounds, documents etc. In our case, only 73 out of 9343 packets are coming from port 80, but in 8216 cases the traffic is going to port 80. There might be two reasons for this. Either, many pictures are uploaded to the web server (e.g., as attachment for webmail) or someone is firing exploits on our university.

For better insight, we will have a look at the time distribution of theses events (figure 3.2). There is a huge peak on March, $28^{\text{th}}$. Most of the traffic is coming from dial-up hosts.

| Name | Address | Hits |
|---|---|---|
| pool-141-157-60-104.balt.east.verizon.net | 141.157.60.104 | 614 |
| nr14-66-161-196-103.fuse.net | 66.161.196.103 | 148 |
| VDSL-130-13-111-49.PHNX.QWEST.NET | 130.13.111.49 | 143 |
| danielo21.campus.luth.se | 130.240.193.238 | 116 |
| pool-151-197-41-243.phil.east.verizon.net | 151.197.41.243 | 116 |
| adsl-208-191-120-5.dsl.snantx.swbell.net | 208.191.120.5 | 116 |
| adsl-223-177-47.mia.bellsouth.net | 68.223.177.47 | 90 |
| host84.adamsmark.com | 65.245.150.84 | 87 |
| | 219.233.3.171 | 85 |
| Toronto-HSE-ppp3664999.sympatico.ca | 65.95.162.70 | 85 |

As the the $28^{\text{th}}$ is a Sunday and students are at home, this supports the idea that file uploads are responsible for most of these alerts. For a more detailed analysis of these events, it is necessary to know the packet's payload.

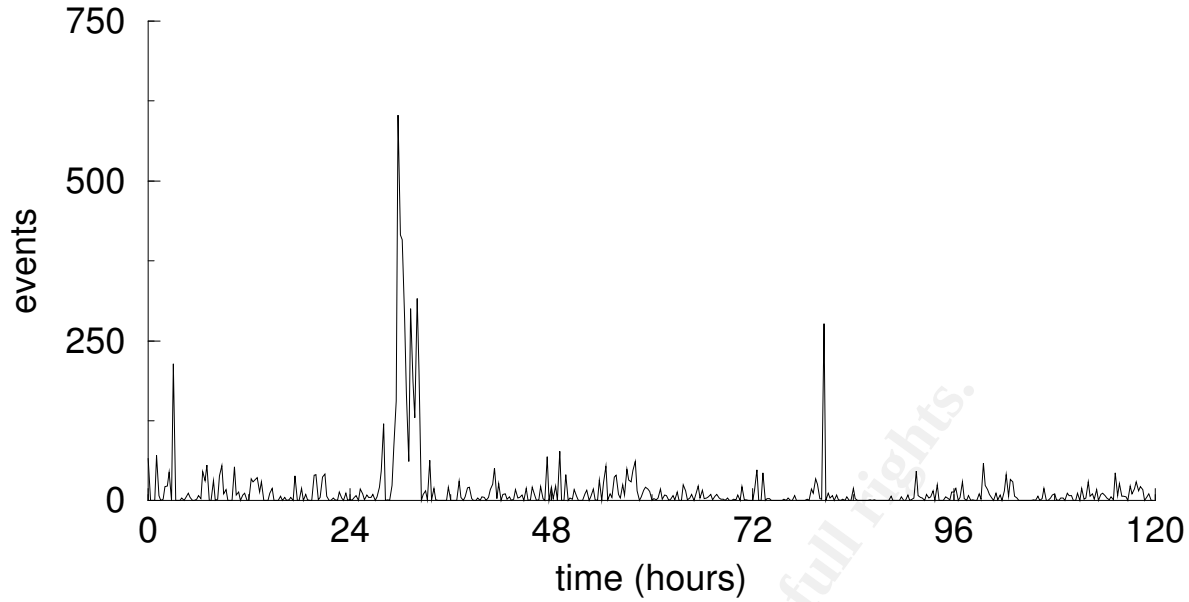



Figure 3.2.: Time distribution of `EXPLOIT x86 NOOP` events. There is a huge peak on March, 28th.

### 3.4.4. SMB Name Wildcard

In general, most SMB Wildcard alerts are generated by windows machines surfing the net. If one of those machines is accessing a web server, it also tries to build up an SMB (Windows networking) connection.

In our university NIDS, all alerts are coming from MY.NET.0.0/16. This either means that snort was configured in a way that only these attempts are logged or that the university firewall is blocking SMB traffic from outside.

It is interesting to see traffic pointing to the 192.168.0.0/16 network. Depending on the local network this might be normal. There might also be a problem with the university DHCP server. We can conclude this from the traffic pointing towards the 169.254.0.0/16 network. RFC 3330 defines this subnet as "link local" and states "Hosts obtain these addresses by auto-configuration, such as when a DHCP server may not" be found.

Without further knowledge, we must assume that traffic from port 137 to 137 is acceptable. Traffic from other source ports, however, might be active scanning. There are two source addresses, which are generating at least 535 (MY.NET.150.44) and 437 (MY.NET.150.198) alerts. In most cases the source port was between 1051 and 1119, but 137 packets had source port 80. Chances are high that someone on these machines is doing active scanning.

### 3.4.5. TFTP - Internal UDP connection to external tftp server

TFTP (RFC1350) is an extremely simple, trivial file transfer protocol. Nowadays, TFTP is used only in local communication. In most cases, it is only used for configuration exchange with CISCO routers or for PXE network booting. However, lately the use of TFTP has tremendously increased. Different worms like Nimda [27] or Blaster [28] are using this pro-

tocol to download their core components to the victim's machine. If TFTP connections are seen across the boundaries of the local network, chances are high that a worm is spreading. We assume that the signature has the below structure.

```
alert udp any any <> $EXTERNAL_NET 69{
msg: "TFTP - Internal UDP connection to external tftp server"; }
```

Not having the payload or any other information, one cannot distinct, whether the rules search for GET or PUT requests. 1157 contacts in 5 days seem too little to support the suspicion of a new virus that successfully starts spreading. In our case, the connects are equally distributed. By searching for IP addresses we see something interesting. 99% (1148) of all alerts are coming from a single host 65.107.99.68 belonging to the network of XO Communications, a telecommunication provider.

```
03/31-08:29:03.011350
  [**] TFTP - Internal UDP connection to external tftp server [**]
  65.107.99.68:69 -> MY.NET.1.3:123
```

This means, that someone is accessing the university NTP server from port 69. Start time is 03/30-16:45:49.942108, stop time is 03/31-08:29:03.011350. As other alerts for these signatures show packets coming from MY.NET and going to external addresses, we can be sure that MY.NET.1.3 did not answer these requests.

Now we want to have a look at the 9 alerts not associated with 65.107.99.68.

```
03/27-06:39:58.336027   MY.NET.84.235:4672 -> 83.32.103.133:69
03/27-06:40:00.440533   MY.NET.84.235:4672 -> 83.32.103.133:69
03/28-21:11:07.045300   MY.NET.84.235:5877 -> 217.81.50.124:69
03/31-05:18:03.445148   MY.NET.84.235:5877 -> 213.37.180.145:69
```

If an external host can manipulate an internal host in such a way, that the internal host is fetching data using TFTP, chances are very high that the local host is vulnerable for certain worms. It is highly recommended that MY.NET.84.235 should be isolated from the network.

Even worse are the following alerts.

```
03/28-06:44:58.330847   221.10.89.48:69 -> MY.NET.1.115:53
03/31-05:18:03.440393   213.37.180.145:69 -> MY.NET.84.235:5877
03/31-07:47:44.292488   66.250.188.23:69 -> MY.NET.69.211:33477
03/31-10:57:21.517111   63.250.197.21:69 -> MY.NET.81.108:21186
03/31-12:09:07.883560   66.250.188.23:69 -> MY.NET.66.29:53957
```

Here, an external host tried to download something from a local host. This indicates an already successful, now spreading infection of the local system.

Neither knowing details on the used snort signature nor the local network, we cannot judge whether there is really a worm spreading. The internal hosts should be investigated, or maybe even a forensic analysis might be done.

Al Williams found for TFTP network traffic analyzed by him, that it was associated with known Peer to Peer ports [29]. The data analyzed here does not give any support for this fact.

### 3.4.6. NIMDA - Attempt to execute root from campus host NIMDA - Attempt to execute cmd from campus host

Apache logs next to always show expressions similar to the below example, were someone is testing, whether he can execute a Windows shell to take over an IIS server [30].

```
"GET /scripts/..%255c%255c../winnt/system32/cmd.exe?/c+dir"
"GET /scripts/root.exe?/c+dir HTTP/1.0"
```

For this kind of traffic, there are well known snort signatures.

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
 (msg:"WEB-IIS CodeRed v2 root.exe access";
 flow:to_server,established; uricontent:"/root.exe"; nocase;[...])

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
 (msg:"WEB-IIS cmd.exe access";
 flow:to_server,established; content:"cmd.exe"; nocase; [...])
```

The problem with these signatures is, that they only trigger alerts for attacks against the local network. This leads to an extremely high rate of alerts, which are unnecessary because everybody knows that this kind of attacks is an ongoing thread. Much more interesting is, whether a local host is attacking a remote host. This seems to be the mission of the found alerts. Probably, they were generated by the original rules by exchanging source and destination IP addresses.

```
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS
(msg:"NIMDA - Attempt to execute root from campus host";
flow:to_server,established; uricontent:"/root.exe"; nocase; [...])

alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS
(msg:"NIMDA - Attempt to execute cmd.exe from campus host";
flow:to_server,established; content:"cmd.exe"; nocase;  [...])
```

Five internal hosts were registered during the 5 day period.

| Addresses | Attacks |
|---|---|
| MY.NET.97.242 | 37 |
| MY.NET.97.12 | 9 |
| MY.NET.97.248 | 3 |
| MY.NET.97.16 | 2 |
| MY.NET.97.28 | 1 |

False positives for this kind of rule are well known. If, for example, someone searches google.com for cmd.exe, the above signature will trigger an alarm. In our case, no search engine was targeted. The five systems need to be isolated from the network and should be investigated in detail.

In order to achieve a more efficient distribution, worms generally prefer to infect machines with an IP-address "similar" to the address of the host they are coming from. 39 out of 53 target systems are in network 130.0.0.0/8.

### 3.4.7. TCP SRC and DST outside network
### ICMP SRC and DST outside network

The alert messages `SRC and DST outside network` suggest, that the person who set up the NIDS for the university was trying to detect suspicious network traffic. Based on the local router configuration, he expects that all packets are either directed to or coming from the network MY.NET/16. Packets which do not have a university IP address as source or destination are therefore suspicious. We assume that the signature has the below structure.

```
alert tcp $EXTERNAL_NET any -> $EXTERNAL_NET any
(msg:"TCP SRC and DST outside network"; );

alert icmp $EXTERNAL_NET any -> $EXTERNAL_NET any
(msg:"ICMP SRC and DST outside network"; );
```

It is interesting, that either no signature for UDP traffic was activated or no according traffic exists. There might be different reasons for the above alert messages. The first idea is, that someone is faking his source address to hide it's identity. This, however, is not very likely, as the attacker will not get any answer to his request. Only fire-and-forget attacks would make sense. Jamell Creque [31] found in his analysis more than 1.4 million alerts of this kind. Our data, however, has a completely different profile with only 224 alerts. We do not think that a DOS attack is on the run.

Daniel Martin raised another idea about this kind of traffic, which he described in his posting to the incidents mailing list [32]. According to him, while connecting web-servers, Windows tries to build SMB connections with every IP address the local machine is bound to. In our case, none of the packets has a port 137.

It is very likely that most of these alerts are based on misconfigured computers. To proove this theory, we will have a detailed look at the source addresses. 42 packets came from the networks 192.168.0.0/32, 192.168.1.0/32, or 192.168.2.0/32. These are RFC 1918 addresses. Depending on point of view, these addresses are not really "outside network". 46 addresses with 137 packets had a source IP-address in a 172.X.0.0/8 network. We performed a reverse DNS lookup for each of these addresses, they all belong to AOL. Another 2 addresses with 44 packets came from Comcast. Probably, some students had connected their laptop to their home ISP (AOL or Comcast). Then, at the university, they possibly did not get an address from the DHCP server (we had seen in one of the last sections, that the university has problems with the DHCP server) and their system used the old address. Going more into detail, we see rather different traffic profiles:

Here, an AOL user is (unsuccessfully?) connecting Bank of America's Online system:

```
03/31-06:45:59.154505  172.166.255.157:1431 -> 66.77.116.80:443
03/31-06:46:00.738000  172.166.255.157:1431 -> 66.77.116.80:443
03/31-06:46:04.981062  172.166.255.157:1431 -> 66.77.116.80:443
03/31-06:46:12.680059  172.166.255.157:1431 -> 66.77.116.80:443
03/31-06:46:28.674166  172.166.255.157:1431 -> 66.77.116.80:443
```

The following Comcast customer is trying to build connections to P2P servers. According to IANA [33], this port number is mainly used by the gnutella network. 25 connects within 5 seconds with nearly sequential source ports are seen:

```
03/28-22:41:53.475518 67.160.1.251:1084 -> 65.163.60.244:6346
03/28-22:41:53.577277 67.160.1.251:1092 -> 140.192.175.165:6346
03/28-22:41:59.509779 67.160.1.251:1095 -> 68.70.174.229:6349
03/28-22:41:59.734260 67.160.1.251:1096 -> 24.7.169.9:6346
03/28-22:41:59.655603 67.160.1.251:1097 -> 68.38.67.133:6346
03/28-22:41:59.717945 67.160.1.251:1105 -> 69.137.102.4:6346
03/28-22:41:59.638144 67.160.1.251:1106 -> 24.44.200.190:6346
03/28-22:41:59.703119 67.160.1.251:1107 -> 209.152.84.103:6346
03/28-22:41:59.622833 67.160.1.251:1108 -> 64.231.120.32:6346
03/28-22:41:59.607098 67.160.1.251:1109 -> 68.12.35.4:6346
03/28-22:41:59.718273 67.160.1.251:1112 -> 217.233.223.3:6346
03/28-22:41:59.734417 67.160.1.251:1113 -> 64.146.145.228:6346
03/28-22:41:59.701894 67.160.1.251:1114 -> 24.100.10.109:6348
03/28-22:42:05.577241 67.160.1.251:1117 -> 12.216.113.153:6346
03/28-22:42:05.642378 67.160.1.251:1118 -> 80.180.70.100:6346
03/28-22:42:05.562617 67.160.1.251:1120 -> 24.74.50.187:6346
03/28-22:41:53.577298 67.160.1.251:1121 -> 80.4.214.91:6346
03/28-22:42:05.578839 67.160.1.251:1121 -> 80.4.214.91:6346
03/28-22:42:05.658749 67.160.1.251:1122 -> 81.240.225.88:6346
03/28-22:41:53.651231 67.160.1.251:1124 -> 68.80.121.112:6346
03/28-22:42:05.754915 67.160.1.251:1124 -> 68.80.121.112:6346
03/28-22:41:53.651301 67.160.1.251:1125 -> 12.218.173.165:6346
03/28-22:42:05.770197 67.160.1.251:1125 -> 12.218.173.165:6346
03/28-22:41:53.667205 67.160.1.251:1126 -> 64.228.76.28:6346
03/28-22:42:05.786977 67.160.1.251:1126 -> 64.228.76.28:6346
```

As defensive recommendation Glenn Larratt [22] suggests to use CISCO's `"ip verify unicast reverse-path"` feature, an idea we emphatically support.

Using tcpdump with the command line option -e will give more information on this problem. Knowing the MAC addresses, it is easy to track this kind of traffic. Concluding, this kind of traffic points more to network problems than to an attack.

### 3.4.8. IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize

The message belongs to a custom signature at our university. Most probably, this signature is based on snort rule 1242 (see below) and not 1243, which is similar.

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"WEB-IIS ISAPI .ida access";  uricontent:".ida"; nocase;
reference:arachnids,552;
reference:bugtraq,1065;
reference:cve,2000-0071; sid:1242; [...])
```

Everybody, who reads his apache log files, has seen requests, which can trigger this alert.

```
GET /default.ida?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 [...]
 %u8b00%u531b%u53ff%u0078%u0000%u00=a HTTP/1.0
```

Probably, the signature `NIMDA - Attempt to execute * from campus host`, which triggered the alerts, was generated in the same intention as the above mentioned snort rule

1242. The author of this signature was searching for computers on the local network infected by worms which try to exploit *Microsoft IIS UNC Path Disclosure Vulnerability* [34, 35]. We assume that the signature has the below structure.

```
alert tcp $INTERNAL_NET any -> $EXTERNAL_NET $HTTP_PORTS
(msg:"IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize";
uricontent:".ida"; nocase; )
```

Only two addresses are responsible for 89 alerts. The hosts MY.NET.97.242 (68 alerts) and MY.NET.97.12 (21 alerts) appeared also as top talkers for the NIMDA related signatures. These two hosts must be isolated from the local network as fast as possible.

### 3.4.9. Conflicting DNS registration information

Searching the oos_report.files we found the following packet:

```
03/29-03:05:10.608065 66.218.55.183:4649 -> MY.NET.82.55:80
TCP TTL:112 TOS:0x0 ID:30244 IpLen:20 DgmLen:338 DF
******** Seq: 0xADD78D73  Ack: 0x6EE1CA61  Win: 0xC418  TcpLen: 0
0C 43 C0 00 00 00 02 38 70 00 00 00 00 62 0C 00  .C.....8p....b..
00 00 00 0C 41 80 00 00 65 73 2F 70 72 6F 6A 5F  ....A...es/proj_
41 2E 6A 70 67 20 48 54 54 50 2F 31 2E 31 0D 0A  A.jpg HTTP/1.1..
41 63 63 65 70 74 3A 20 2A 2F 2A 0D 0A 52 65 66  Accept: */*..Ref
65 72 65 72 3A 20 68 74 74 70 3A 2F 2F 77 65 62  erer: http://web
2E 6B 69 6C 6C 65 72 68 65 61 64 2E 6E 65 74 2F  .killerhead.net/
6C 61 76 65 6E 64 61 72 74 69 6E 74 65 64 2F 70  lavendartinted/p
72 6F 6A 65 63 74 73 2E 68 74 6D 6C 0D 0A 41 63  rojects.html..Ac
63 65 70 74 2D 4C 61 6E 67 75 61 67 65 3A 20 65  cept-Language: e
6E 2D 75 73 0D 0A 41 63 63 65 70 74 2D 45 6E 63  n-us..Accept-Enc
6F 64 69 6E 67 3A 20 67 7A 69 70 2C 20 64 65 66  oding: gzip, def
6C 61 74 65 0D 0A 55 73 65 72 2D 41 67 65 6E 74  late..User-Agent
3A 20 4D 6F 7A 69 6C 6C 61 2F 34 2E 30 20 28 63  : Mozilla/4.0 (c
6F 6D 70 61 74 69 62 6C 65 3B 20 4D 53 49 45 20  ompatible; MSIE
36 2E 30 3B 20 57 69 6E 64 6F 77 73 20 4E 54 20  6.0; Windows NT
35 2E 31 29 0D 0A 48 6F 73 74 3A 20 77 65 62 2E  5.1)..Host: web.
6B 69 6C 6C 65 72 68 65 61 64 2E 6E 65 74 0D 0A  killerhead.net..
43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 4B 65 65 70  Connection: Keep
2D 41 6C 69 76 65 0D 0A 0D 0A                    -Alive....
```

This packet was alerted, because not flags were set (NULL SCAN). But the real value of these packet lies in the payload. We see an HTTP/1.1 GET-request. These requests must always have the `HOST:` header. In this case the host is web.killerhead.net.

```
host> nslookup web.killerhead.net
[...]
Non-authoritative answer:
Name:   web.killerhead.net
Address: MY.NET.82.55
[...]
```

This IP-address is resolved into a university address. But, as killerhead.net is not a likley name for a university, we try a reverse lookup for MY.NET.82.55:

```
host> nslookup MY.NET.82.55 MY.NET.edu
[...]
Server:         MYNET3.MY.NET
Address:        MY.NET.1.3#53

55.82.NET.MY.in-addr.arpa       name = oit-82-55.pooled.MY.NET.
```

The fact that `nslookup MY.NET.82.55` returns the name `oit-82-55.pooled.MY.NET` means, that someone registered a new domain for one of the university computers. Looking up killerhead.net, using whois at dnsstuff.com, we can even get name and mail address of this person.

```
domain:       killerhead.net
status:       production
organization: Studio Psychowerks
owner:        Kendrick Hernandez
email:        kherna1@MYNET.edu
address:      7208 Johnnycake Road
city:         Baltimore
postal-code:  21228
country:      US
admin-c:      kherna1@MYNET.edu#0
tech-c:       kherna1@MYNET.edu#0
billing-c:    kherna1@MYNET.edu#0
nserver:      a.ns.joker.com 194.176.0.2
nserver:      b.ns.joker.com 194.245.101.19
nserver:      c.ns.joker.com 194.245.50.1
registrar:    JORE-1
created:      2003-09-24 21:25:03 UTC JORE-1
modified:     2004-06-29 14:11:55 UTC JORE-1
expires:      2006-09-24 17:24:49 UTC
source:       joker.com
```

Asking the whois database for the IP address MY.NET.82.55 returns the university we did investigate:

```
OrgName:    University of MY.NET
OrgID:      MYNET
Address:    ******************
City:       *********
StateProv:  **
PostalCode: *****
Country:    US

NetRange:   MY.NET.0.0 - MY.NET.255.255
CIDR:       MY.NET.0.0/16
NetName:    MYNETET
NetHandle:  NET-130-85-0-0-1
Parent:     NET-130-0-0-0-0
[...]
```

This means Mr. Kendrick Hernandez registered www.killerhead.com at the authorized registrar joker.com using a university IP address. Another look in the alert log file shows, that this host is also responsible for a certain number of `EXPLOIT x86 NOOP`, `Possible trojan server activity`, and `Null scan!` alerts. We think, that this behavior is against the policy of the university and should be investigated.

## 3.5. A "top talkers" list

As only parts of the network traffic can be reconstructed, it is hard to decide which host is victim or target. Thus, in calculating the top ten talkers from the alert and oos_report files, we decided to treat source and destination address as equivalent.

| | Alert | | OOS | |
|---|---|---|---|---|
| | Address | Hits | Address | Hits |
| 1. | MY.NET.30.3 | 28210 | MY.NET.6.7 | 1244 |
| 2. | MY.NET.30.4 | 21298 | 68.54.84.49 | 1204 |
| 3. | 80.181.112.186 | 10485 | MY.NET.12.6 | 553 |
| 4. | MY.NET.97.82 | 10484 | MY.NET.42.7 | 359 |
| 5. | 68.55.174.94 | 7590 | MY.NET.24.44 | 355 |
| 6. | 67.31.152.200 | 6635 | MY.NET.42.5 | 287 |
| 7. | MY.NET.153.176 | 5180 | 66.75.122.52 | 280 |
| 8. | 68.55.178.168 | 3127 | 68.5.196.199 | 247 |
| 9. | 140.142.8.73 | 3074 | 66.225.198.20 | 145 |
| 10. | 69.136.228.63 | 2988 | 24.48.220.79 | 122 |

Analyzing the port scan log file, source and destination were treated separately. Here, the role of the victim and of the attacker is much clearer.

| | Source | | Destination | |
|---|---|---|---|---|
| | Address | Hits | Address | Hits |
| 1. | 69.6.57.7 | 87774 | MY.NET.190.92 | 10262212 |
| 2. | 69.6.57.9 | 87672 | MY.NET.111.51 | 3895474 |
| 3. | 192.26.92.30 | 86777 | MY.NET.1.3 | 3811712 |
| 4. | 192.48.79.30 | 71308 | MY.NET.1.4 | 752202 |
| 5. | MY.NET.25.68 | 71164 | MY.NET.84.235 | 472229 |
| 6. | MY.NET.190.92 | 63602 | MY.NET.34.14 | 217639 |
| 7. | 192.5.6.30 | 57163 | MY.NET.110.72 | 203591 |
| 8. | 4.13.52.66 | 56418 | MY.NET.153.174 | 188530 |
| 9. | 203.20.52.5 | 54350 | MY.NET.97.108 | 133766 |
| 10. | 128.194.254.5 | 47847 | MY.NET.97.103 | 87814 |

It is also possible to check other categories than IP addresses. The participating ports tell us, which service is most threatened. It is quite unusual, that port 80 (http) is not the top port.

| | total | | Source | | Destination | |
|---|---|---|---|---|---|---|
| | Port | Hits | Port | Hits | Port | Hits |
| 1. | 524 | 21135 | 65535 | 7887 | 524 | 21135 |
| 2. | 65535 | 14799 | 1078 | 5488 | 51443 | 13582 |
| 3. | 51443 | 13582 | 1122 | 5098 | 80 | 10801 |
| 4. | 80 | 11908 | 137 | 4192 | 65535 | 6912 |
| 5. | 1122 | 10558 | 1033 | 2126 | 3019 | 6730 |
| 6. | 137 | 9356 | 69 | 1403 | 1122 | 5460 |
| 7. | 3019 | 6756 | 53 | 1388 | 137 | 5164 |
| 8. | 1078 | 5488 | 1077 | 1254 | 123 | 2326 |
| 9. | 123 | 2334 | 80 | 1107 | 110 | 1457 |
| 10. | 1033 | 2129 | 3658 | 862 | 3658 | 1228 |

In addition, it might be interesting, where the participating attackers are from. Here, only IP addresses not from the MY.NET.0.0/16 subnet are counted.

| | | Country | Hits |
|---|---|---|---|
| 1. | US | United States | 76233 |
| 2. | IT | Italy | 10609 |
| 3. | BE | Belgium | 1051 |
| 4. | CN | China | 944 |
| 5. | DE | Germany | 891 |
| 6. | FI | Finland | 782 |
| 7. | GB | Great Britain | 774 |
| 8. | JP | Japan | 730 |
| 9. | SE | Sweden | 723 |
| 10. | CA | Canada | 465 |

## 3.6. Five selected external source addresses and registration information

The host 65.107.99.68 is heavily contributing to the `TFTP - Internal UDP connection to external tftp server` alerts.

```
OrgName:    XO Communications
OrgID:      XOXO
Address:    Corporate Headquarters
Address:    11111 Sunset Hills Road
City:       Reston
StateProv:  VA
PostalCode: 20190-5339
Country:    US

ReferralServer: rwhois://rwhois.eng.xo.com:4321/
```

```
NetRange:   65.104.0.0 - 65.107.255.255
CIDR:       65.104.0.0/14
NetName:    XOXO-BLK-15
NetHandle:  NET-65-104-0-0-1
Parent:     NET-65-0-0-0-0
NetType:    Direct Allocation
NameServer: NAMESERVER1.CONCENTRIC.NET
NameServer: NAMESERVER2.CONCENTRIC.NET
NameServer: NAMESERVER3.CONCENTRIC.NET
NameServer: NAMESERVER.CONCENTRIC.NET
Comment:
RegDate:
Updated:    2003-08-08

OrgAbuseHandle: XCNV-ARIN
OrgAbuseName:   XO Communications, Network Violations
OrgAbusePhone:  +1-866-285-6208
OrgAbuseEmail:  *****@xo.com

OrgTechHandle: XCIA-ARIN
OrgTechName:   XO Communications, IP Administrator
OrgTechPhone:  +1-703-547-2000
OrgTechEmail:  *******@eng.xo.com
```

| GeoIP Information | |
|---|---|
| TARGET: | 65.107.99.68 |
| CITY: | SAN JOSE |
| STATE: | CALIFORNIA |
| COUNTRY: | US |
| LAT: | 37.32 |
| LONG: | -121.92 |

The top #1 scanning host 69.6.57.7 was chosen to get more detailed information.

```
OrgName:    WholesaleBandwidth, Inc.
OrgID:      WHOLE
Address:    1416 S Main St.
Address:    220-152
City:       Adrian
StateProv:  MI
PostalCode: 49221
Country:    US

NetRange:   69.6.0.0 - 69.6.79.255
CIDR:       69.6.0.0/18, 69.6.64.0/20
NetName:    WHOLE-2
NetHandle:  NET-69-6-0-0-1
Parent:     NET-69-0-0-0-0
NetType:    Direct Allocation
NameServer: NS1.WHOLESALEBANDWIDTH.COM
NameServer: NS2.WHOLESALEBANDWIDTH.COM
```

```
Comment:
RegDate:    2002-11-21
Updated:    2004-02-03

OrgAbuseHandle: ABUSE71-ARIN
OrgAbuseName:   Abuse Department
OrgAbusePhone:  +1-866-444-8419
OrgAbuseEmail:  *****@wholesalebandwidth.com

OrgNOCHandle: NOC197-ARIN
OrgNOCName:   Network Operations Center
OrgNOCPhone:  +1-866-444-8419
OrgNOCEmail:  ***@wholesalebandwidth.com

OrgTechHandle: SUPPO14-ARIN
OrgTechName:   Customer Support
OrgTechPhone:  +1-866-444-8419
OrgTechEmail:  *******@wholesalebandwidth.com
```

| GeoIP Information | |
|---|---|
| TARGET: | 69.6.57.7 |
| CITY: | |
| STATE: | |
| COUNTRY: | AU |
| LAT: | -25.00 |
| LONG: | 135.00 |

For the alerts, the first two talkers are from the internal network. The first external address appearing is 80.181.112.186.

```
inetnum:      80.181.112.0 - 80.181.141.255
netname:      TELECOM-ADSL
descr:        Telecom Italia
descr:        Accesso ADSL
country:      IT
admin-c:      BS104-RIPE
tech-c:       BS104-RIPE
status:       ASSIGNED PA
remarks:      Please send abuse notification to *****@telecomitalia.it
notify:       **********@telecomitalia.it
mnt-by:       TIWS-MNT
changed:      ******@telecomitalia.it 20030805
source:       RIPE

route:        80.181.0.0/16
descr:        INTERBUSINESS
origin:       AS3269
notify:       *******@cgi.interbusiness.it
mnt-by:       TIWS-MNT
mnt-routes:   INTERB-MNT
changed:      ******@telecomitalia.it 20021001
```

```
source:       RIPE

person:       BBBEASYIP STAFF
address:      Via Val Cannuta, 250
address:      I-00100 Roma
address:      Italy
phone:        +39 06 36881
e-mail:       **********@telecomitalia.it
nic-hdl:      BS104-RIPE
notify:       **********@telecomitalia.it
changed:      ******@telecomitalia.it 20001019
source:       RIPE
```

| GeoIP Information | |
|---|---|
| TARGET: | 80.181.112.186 |
| CITY: | AMSTERDAM |
| STATE: | NORTH HOLLAND (province) |
| COUNTRY: | NL |
| LAT: | 52.35 |
| LONG: | 4.90 |

The next two external hosts were chosen, because, according to the IDS, they were talking to each other. 172.166.255.157 is trying to connect 66.77.116.80 on port 443 (https).

```
OrgName:    America Online
OrgID:      AOL
Address:    22000 AOL Way
City:       Dulles
StateProv:  VA
PostalCode: 20166
Country:    US

NetRange:   172.128.0.0 - 172.191.255.255
CIDR:       172.128.0.0/10
NetName:    AOL-172BLK
NetHandle:  NET-172-128-0-0-1
Parent:     NET-172-0-0-0-0
NetType:    Direct Allocation
NameServer: DAHA-01.NS.AOL.COM
NameServer: DAHA-02.NS.AOL.COM
NameServer: DAHA-07.NS.AOL.COM
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    2000-03-24
Updated:    2003-08-08

TechHandle: AOL-NOC-ARIN
TechName:   America Online, Inc.
TechPhone:  +1-703-265-4670
TechEmail:  *******@aol.net

OrgAbuseHandle: AOL382-ARIN
```

```
OrgAbuseName:   Abuse
OrgAbusePhone:  +1-703-265-4670
OrgAbuseEmail:  *****@aol.net

OrgNOCHandle: AOL236-ARIN
OrgNOCName:   NOC
OrgNOCPhone:  +1-703-265-4670
OrgNOCEmail:  ***@aol.net

OrgTechHandle: AOL-NOC-ARIN
OrgTechName:   America Online, Inc.
OrgTechPhone:  +1-703-265-4670
OrgTechEmail:  *******@aol.net
```

| GeoIP Information | |
|---|---|
| TARGET: | 172.166.255.157 |
| CITY: | VIENNA |
| STATE: | VIRGINIA |
| COUNTRY: | US |
| LAT: | 38.93 |
| LONG: | -77.26 |

```
CustName:   Douglas-Danielle.com
Address:    226 W. Ontario St. Suite 500B
City:       Chicago
StateProv:  IL
PostalCode: 60601
Country:    US
RegDate:    2003-04-18
Updated:    2003-04-18

NetRange:   66.77.116.64 - 66.77.116.127
CIDR:       66.77.116.64/26
NetName:    QWEST-CEC-DDNIE
NetHandle:  NET-66-77-116-64-1
Parent:     NET-66-77-0-0-1
NetType:    Reassigned
Comment:
RegDate:    2003-04-18
Updated:    2003-04-18

TechHandle: DW820-ARIN
TechName:   Wysocki, David
TechPhone:  +1-201-770-4133
TechEmail:  ********@qis.qwest.net

OrgAbuseHandle: QIA2-ARIN
OrgAbuseName:   Qwest IP Abuse
OrgAbusePhone:  +1-877-886-6515
OrgAbuseEmail:  *****@qwest.net
```

```
OrgTechHandle: QIA-ARIN
OrgTechName:   Qwest IP Admin
OrgTechPhone:  +1-877-886-6515
OrgTechEmail:  *******@qwest.com
```

| GeoIP Information | |
|---|---|
| TARGET: | 66.77.116.80 |
| CITY: | CAMBRIDGE |
| STATE: | MASSACHUSETTS |
| COUNTRY: | US |
| LAT: | 42.36 |
| LONG: | -71.10 |

## 3.7. Link graph

For representation of an interesting link graph (figure 3.3), we chose to plot a part of the traffic from the analysis of the `TCP SRC and DST outside network` events. Only source addresses in the networks 192.168.0.0/24, 192.168.1.0/24, or 192.168.2.0/24 were chosen for plotting. The source addresses were grouped by these /24 networks. For each destination the number of hits and the destination ports were added to the link. The arrow shows the direction of the alerted communication.

## 3.8. Defensive recommendations

*"Intellectuals solve problems; geniuses prevent them."*
*Albert Einstein*

The investigated university has several acute problems we have to solve soon, but we must also think about the future. Many of the detected problems will reappear sooner or later if no fundamental changes are performed. Different actions are necessary to achieve an improved security for the network.

**Short-term actions** These kind of actions should take place as soon as this report is available to the IT department. It should be in the authority of the administrators to start such actions without additional instances.

**Mid-term actions** Within the next 3-6 months different actions should be taken to increase the overall security. Also, different policies should be worked out to reach the long term goals.

**Long-term actions** Basic changes to the university infrastructure should take place to increase the overall security. These changes can be realized only in collaboration with the university management.

The defensive recommendations are not only based on the problems described in the above report, but are also based on other insights gained by working with the log files.

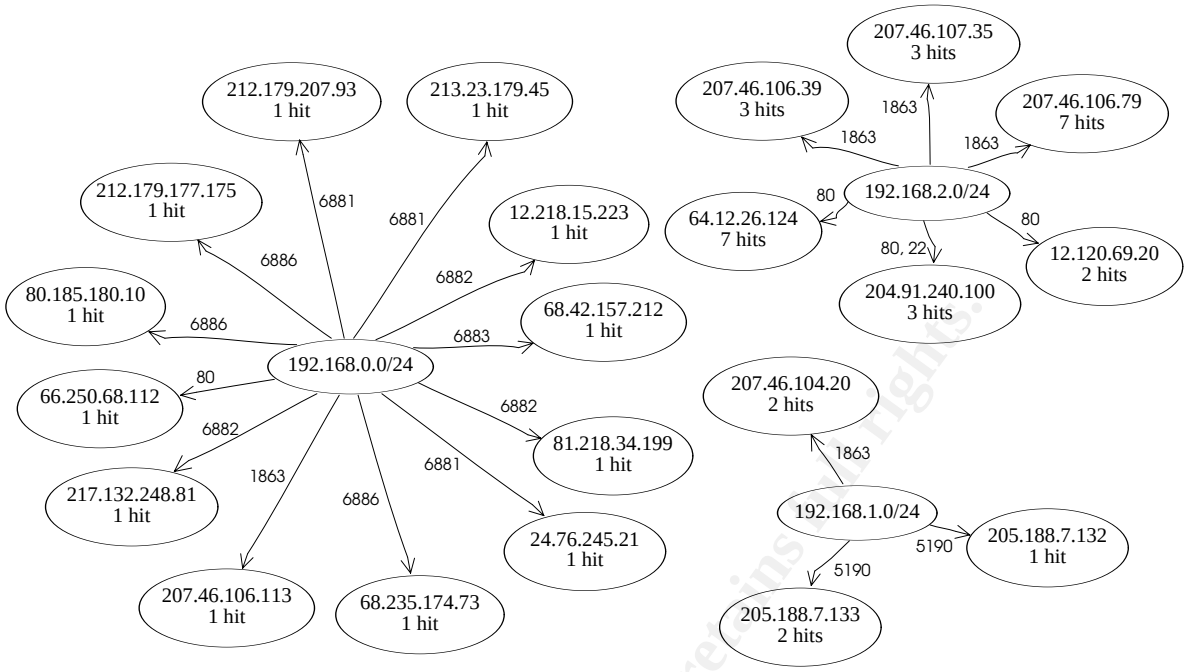



Figure 3.3.: Link graph for a part of the traffic from the analysis of the `TCP SRC and DST outside network` events

### 3.8.1. Short-term actions

Most of these actions will be simple (e.g., removal of a worm) but will have limited effects (another worm will appear).

1. Check all computers, which might be infected by different worms or viruses. MY.NET.97.242, MY.NET.97.12 and MY.NET.84.235 should be isolated from the network. If one of these machines is really infected, possible victims of the machines should be contacted, too.

2. Talk to Mr. Kendrick Hernandez about killerhead.net.

3. If Peer-to-Peer networking is not allowed at the university, typical ports should be blocked at the perimeter router. A first step would be to block packets to the ports tcp/1214 (Kazaa), TCP/6346, UDP/6346, TCP/6347, UDP/6347(gnutella), and TCP/[6881:6889] (BitTorrent).

4. Normally, Windows networking is not needed across the Internet. Connections going to the typical Windows ports 135, 137, 139, 445, and 5000 should be blocked.

5. Check if the DHCP server is running. Use a program which is simple to setup (e.g., monit http://www.tildeslash.com/monit/) to check and restart this service.

6. Block traffic from RFC 1918 address on the perimeter.

### 3.8.2. Mid-term actions

Normally, many of these actions must be authorized by the IT-department manager. These actions can solve many of the university's problems and prevent some new ones.

1. The installed IDS shows, that network security is a topic at the university. However, the system is not used optimally. Signatures like `MY.NET.30.3 activity` are good for traffic analysis but not for intrusion detection. Changing the log format so that detailed information is stored will make analysis much more efficient.

2. Check the IDS for signatures like `External FTP to HelpDesk MY.NET.53.29`. Instead of detecting this kind of traffic, just prevent it. Use iptables, `/etc/hosts.deny`, bridging firewalls, or Router ACLs, whichever is the easiest fix.

3. Replacing rules like `MY.NET.30.3 activity`, Nagios might help to monitor the network and critical services like DHCP.

4. Push the use of virus scanners, personal firewalls, and other kind of security software at the university. A local mirror for this kind of software, good instructions on how to install and use it, and an article in the university's newspaper can help.

5. A simple announced security scan with Nessus might show up many vulnerabilities. Then issue an ultimatum to the owners of these possible targets: Either they fix their computers within 5 days or they will be disconnected.

6. Get official statements from university management whether Peer-to-Peer networking, IRC, instant messaging, etc. are allowed. If not, block them on the perimeter routers or firewalls.

7. Allow sending of mail to the outside only from certain, well known hosts.

8. Setup network policies which must be signed by the university officials.

### 3.8.3. Long-term actions

These actions are the most vague ones. Depending on the sort of policy, on the willingness to collaborate and the available money, different goals can be reached. Many problems can be prevented or detected very early.

1. Install a system performing security scans and automatically inform the user of the results.

2. Install proxies and reverse proxies for different protocols. Whenever possible, try to reduce the number of hosts either reachable from the Internet or communicating directly with the Internet. Remember: to perform good science work, gnutella is not necessary.

3. Start segmentation of the network. Define and control traffic between the different segments.

4. Enforce the policies.

## 3.9. Description of the analysis process

Certain steps were performed before the data was analyzed.

1. The log files were downloaded and the local size was compared to the one seen on the server. Unfortunately, no md5sums or similar are available.

2. Where necessary, we unzipped the log files and concatenated them. So we ended with three files alert.all, oos_report.all and scans.all.

3. Presumably, due to the sanitizing process, some lines of the log file were corrupted. We removed these lines.

4. Information on portscans were redundant in the scans.all and alert.all files. We removed the protscans from alert.all and generated a new file alert.noscan.

Having read many warnings in other GCIA alumni's papers [36], we decided not to use tools like snortsnarf for the analysis. Instead we wanted to use a database. To import the data to MySQL, we used some modified versions of the script found in Daniel Clarks practical [37] which is originally based on the work of Jeremy Chartier. We added some simple error checking and switched to Perl::DBI. For the oos_event, we added some fields for further analysis.

```
CREATE TABLE alert_event (aid INT UNSIGNED NOT NULL,
timestamp DATETIME NOT NULL,
signature VARCHAR(255) NOT NULL,
ip_src INT UNSIGNED NOT NULL,
ip_dst INT UNSIGNED NOT NULL,
l4_sport INT UNSIGNED NOT NULL,
l4_dport INT UNSIGNED NOT NULL,
PRIMARY KEY (aid),
INDEX ip_src (ip_src),
INDEX ip_dst (ip_dst),
INDEX signature (signature));

CREATE TABLE oos_event (aid INT UNSIGNED NOT NULL,
timestamp DATETIME NOT NULL,
ip_src INT UNSIGNED NOT NULL,
ip_dst INT UNSIGNED NOT NULL,
```

```
l4_sport INT UNSIGNED NOT NULL,
l4_dport INT UNSIGNED NOT NULL,
ttl INT UNSIGNED NOT NULL,
tos INT UNSIGNED NOT NULL,
id INT UNSIGNED NOT NULL,
ip_len INT UNSIGNED NOT NULL,
dgm_len INT UNSIGNED NOT NULL,
flags CHAR(8) NOT NULL,
PRIMARY KEY (aid),
INDEX ip_src (ip_src),
INDEX ip_dst (ip_dst));
```

While trying to import the data for the scan log files, the computer became heavily overloaded. Therefore, we split the data into three tables: UDP scans, SYN scans and the rest. Now, the database could handle this huge amount of data and was answering fast enough for efficient working.

```
CREATE TABLE scan_event_udp (aid INT UNSIGNED NOT NULL, ...

CREATE TABLE scan_event_syn (aid INT UNSIGNED NOT NULL, ...

CREATE TABLE scan_event_other (aid INT UNSIGNED NOT NULL,
timestamp DATETIME NOT NULL,
ip_src INT UNSIGNED NOT NULL,
ip_dst INT UNSIGNED NOT NULL,
l4_sport INT UNSIGNED NOT NULL,
l4_dport INT UNSIGNED NOT NULL,
PRIMARY KEY (aid),
INDEX ip_src (ip_src),
INDEX l4_dport (ip_src)
);
```

Then, most of the analysis was done using SQL commands on the command line, combined with certain small helpers. Of course, tools like awk, grep, and perl are essential for this work, too.

This following script tries to replace ip addresses, coming from the database, with the dotted notation:

```
#!/usr/bin/perl -p
# filename ip.pl
BEGIN{ use Socket; }
s/(\d{6,})/inet_ntoa(pack "N",$1)/ge;

host> mysql -B  -e "select ip_src from alert_event limit 2" sans
ip_src
67356234
67359336
host> mysql -B  -e "select ip_src from alert_event limit 2" sans | ip.pl
ip_src
4.3.198.74
4.3.210.104
```

Having the ip address, we often want to know the DNS name.

```
#!/usr/bin/perl -p
# filename dns.pl
BEGIN{ use Socket; }
s/(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})/gethostbyaddr(inet_aton($1), AF_INET)."($1)"/ge;
```

We use this script in combination with ip.pl.

```
host> mysql -B  -e "select ip_src from alert_event limit 2" sans | ip.pl | dns.pl
lsanca1-ar54-4-3-198-074.lsanca1.dsl-verizon.net(4.3.198.74)
lsanca2-ar35-4-3-210-104.lsanca2.dsl-verizon.net(4.3.210.104)
```

Using geoip, we get the country, the ip is located in.

```
#!/usr/bin/perl -p
# filename geo.pl
BEGIN{ use Geo::IP;
$gi = Geo::IP->new(GEOIP_STANDARD);
}
s/(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})/$gi->country_code_by_addr($1)/ge;
```

In combination with another short script, which calculates the frequency of lines from the input, we can generate very simple top talker lists.

```
#!/usr/bin/perl
# filename cu.pl
use strict;
my %count;

while (<>){
 s/[\s\n]//g;
 $count{$_}++;
}

foreach my $key (sort {$count{$b} <=> $count{$a};} (keys(%count))) {
   print "$key \t $count{$key}  \n";
}
```

```
host> mysql -B -e "select ip_src from alert_event" sans | ip.pl | geo.pl | cu.pl | head -10
US       80294
IT       5520
FI       762
DE       754
SE       701
CN       699
JP       555
BE       538
GB       503
BR       297
```

For the time plots, we used the following script. It expects two input parameters. The first is the database and the second is an SQL snippet.

```
#!/usr/bin/perl
#filename timeprofile.pl
use Mysql;
use DBI;
use Socket;
use strict;
my $arg  = @ARGV[0];
my $arg2 = @ARGV[1];

my $dbh  =
  DBI->connect( 'DBI:mysql:' . 'database' . ':' . 'host', 'user', 'passwd',
    { RaiseError => 1, AutoCommit => 1 } );
my $timestamp1;
my $timestamp2;

exit if ( !defined $dbh );

my $select =
  $dbh->prepare(
    "select count(*) from $arg where timestamp between ? and ? $arg2 ");

for ( my $day = 27 ; $day <= 31 ; $day++ ) {
    for ( my $hour = 0 ; $hour < 24 ; $hour++ ) {
        for ( my $quarter = 0 ; $quarter < 4 ; $quarter++ ) {
            if ( $quarter == 0 ) {
                $timestamp1 = "2004-03-$day $hour:00:00";
                $timestamp2 = "2004-03-$day $hour:14:59";
            } elsif ( $quarter == 1 ) {
                $timestamp1 = "2004-03-$day $hour:15:00";
                $timestamp2 = "2004-03-$day $hour:29:59";
            } elsif ( $quarter == 2 ) {
                $timestamp1 = "2004-03-$day $hour:30:00";
                $timestamp2 = "2004-03-$day $hour:44:59";
            } else {
                $timestamp1 = "2004-03-$day $hour:45:00";
                $timestamp2 = "2004-03-$day $hour:59:59";
            }
            $select->execute( $timestamp1, $timestamp2 );
            my @sum = $select->fetchrow_array;
            print 1.0 * ( $hour + 24.0 * ( $day - 27 ) ) + 0.25 * $quarter, " ",
              $sum[0], "\n";
        }
    }
}
exit 0;
```

A time profile of the traffic towards port 80 can be obtained by:

```
host> timeprofile.pl alert_event "and l4_dport='80'" > profile.dat
```

All the scripts are without error checking and are far from bullet proof. However, they are fast enough and if one knows how to use them, they can be very efficient and comfortable.

# A. ipanalyze

The obvious choice for analyzing network traffic is tcpdump. But have a look at the typical output of tcpdump and imagine you have to plot the destination port versus time.

```
20:30:41.171711 IP 127.0.0.1.32771 > 127.0.0.1.22: S
 2155951624:2155951624(0) win 32767
 <mss 16396,sackOK, timestamp 787190 0,nop,wscale 0>
20:30:41.171731 IP 127.0.0.1.22 > 127.0.0.1.32771: S
 2159986720:2159986720(0) ack 2155951625 win 32767
 <mss 16396,sackOK,timestamp 787190 787190,nop,wscale 0>
20:30:41.171752 IP 127.0.0.1.32771 > 127.0.0.1.22: .
 ack 1 win 32767 <nop,nop,timestamp 787190 787190>
20:30:41.198703 IP 127.0.0.1.22 > 127.0.0.1.32771: P
 1:44(43) ack 1 win 32767 <nop,nop,timestamp 787217 787190>
```

For gnuplot, xmgr and similar tools you need an input file of the following structure:

```
1094063441.171711 22
1094063441.171731 32771
1094063441.171752 22
1094063441.198703 32771
```

In order to get such an input file, we could either isolate the necessary data from the tcpdump output by using some quickly written but lengthy regular expressions or write a specific program for this task. We have decided to do the latter, and wrote ipanalyze, a small Perl program, which can produce nearly every tabular output format. The program accepts a subset of the command line options of tcpdump. Additionally, there is the option -o for the description of the output format. For the example above, the following command line will generate a table with comma separated values.

```
alex@host> ipanalyze -i input.dmp -o '"%E","%D"'
"1094063441.171711","22"
"1094063441.171731","32771"
"1094063441.171752","22"
"1094063441.198703","32771"
```

## A.1. Name

ipanalyze - dump traffic on a network in tabular format

## A.2. Synopsis

```
ipanalyze [ -pdh] [ -c count ] [ -i interface ]
                  [ -w file ] [ -r file ] [ -n string ]
                  [ -s snaplen ] [ -o output format ]
                  [ expression ]
```

## A.3. Description

The program ipanalyze prints a tabular view of packets on a network interface that match the boolean `expression`. An optional output format can be specified, so that different table formats can be generated. The program can also be run with the `-w` flag, in which case the table is saved to a file for later analysis with gnuplot or other tools. Please note that the output will not be readable by tcpdump any more. The `-r` flag causes the program to read from a saved packet file rather than to read packets from a network interface. In all cases, only packets that match `expression` will be processed by ipanalyze. If not run with the `-c` flag, ipanalyze will continue capturing packets until it is interrupted.

## A.4. Options

- -c Exit after receiving `count` packets
- -p Do not put the interface into promiscuous mode
- -r Read packets from `file` (e.g., created by tcpdump)
- -s Snarf `snaplen` bytes of data from each packet rather than the default of 135
- -w Write the output to `file` rather than printing it to standard output
- -i Listen on `interface`
- -n Specify a `string` which is printed for missing header values. Default is -
- -h Hide prefixes for hexadecimal, octal and binary numbers
- -d Lead zeros can be disabled with this option. Warning: In combination with -h this may give confusing results.
- -o Define output `format`. Default is `"%Z: %E\t %s:%S \t->%d:%D \tLength: %l \tProtocol: %p"`

## A.5. Examples

Below, you see the default behavior of ipanalyze.

```
host> ./ipanalyze not port 22
1: 1094409296.161171 10.0.0.10:34266 -> 10.0.0.11:25    Length: 54 Protocol: 6
2: 1094409296.162161 10.0.0.11:25    -> 10.0.0.10:34266 Length: 78 Protocol: 6
3: 1094409296.162299 10.0.0.10:34266 -> 10.0.0.11:25    Length: 52 Protocol: 6
```

Use the following commands, if you want to plot time vs. destination port for host 10.0.0.1.

```
host> ./ipanalyze -c 100 -w output1.dat -o "%E %D" dst host 10.0.0.1
host> gnuplot
gnuplot> plot "output1.dat"
```

If you want to create a CSV file, you must escape the quotation marks.

```
host> ./ipanalyze -o  '\"%E\",\"%D\"' src host 10.0.0.1
"1094410109.784112","22"
"1094410109.784802","22"
"1094410110.164480","22"
```

If the output should have more than one line, you can use \n. Using this feature, we can easily determine unique MAC addresses for a tcpdump file.

```
host> ./ipanalyze -c 1000 -o "%m\n%M" | sort -u
00:04:57:ae:93:fb
00:04:67:dd:ff:29
08:00:64:63:a6:d6
```

# A.6. Output format

## A.6.1. Meta Information

| Format | Default Format | Description |
|---|---|---|
| %Z | decimal | Packet Counter |
| %z | decimal | Time, relative to first packet |
| %E | decimal | Time |

## A.6.2. Layer 2

| Format | Default Format | Description |
|---|---|---|
| %M | 11:22:33:44:55:66 | Source MAC |
| %m | 11:22:33:44:55:66 | Destination MAC |
| %e | hexadecimal | Type |

### A.6.3. Layer 3

| Format | Default Format | Description |
|---|---|---|
| %H | hexadecimal | Header Checksum |
| %I | hexadecimal | IHL |
| %O | decimal | Fragment Offset |
| %T | decimal | TTL |
| %d | 192.168.1.1 | Destination Address |
| %f | decimal | Fragmentation (More Fragments) |
| %i | hexadecimal | Identification |
| %l | decimal | Total Length |
| %n | decimal | Fragmentation (Don't Fragment) |
| %p | decimal | Protocol |
| %s | 192.168.1.1 | Source Address |
| %t | hexadecimal | TOS |
| %v | hexadecimal | Version |
| %x | decimal | Fragmentation (Reserved) |

### A.6.4. Layer 4

| Format | Default Format | Description |
|---|---|---|
| %A | decimal | A |
| %C | decimal | Code |
| %D | decimal | Destination Port |
| %F | decimal | F |
| %L | decimal | Length |
| %P | decimal | P |
| %Q | hexadecimal | Sequence Number |
| %R | decimal | R |
| %S | decimal | Source Port |
| %U | decimal | U |
| %Y | decimal | S |
| %a | hexadecimal | Acknowledgment Number |
| %c | hexadecimal | Checksum |
| %o | decimal | Offset |
| %r | hexadecimal | Reserved |
| %u | decimal | Urgent Pointer |
| %w | decimal | Window |
| %y | decimal | Type |

Nearly all output formats can be changed using modifiers.

16 hexadecimal output
10 decimal output
8 octal output
2 binary output

For example, the following command lines will print the destination port with the four different modifiers. Depending on the flag `-h` you will see prefixes, `-d` will remove lead zeros.

```
host> ipanalyze -c 1 -o "%16D %10D %8D %2D"
0x0019 25 0o31 0b00000000000000000000000000011001
host> ipanalyze -c 1 -h -o "%16D %10D %8D %2D"
0019 25 31 00000000000000000000000000011001
host> ipanalyze -c 1 -d -o "%16D %10D %8D %2D"
0x19 25 0o31 0b11001
host> ipanalyze -c 1 -d -h -o "%16D %10D %8D %2D" port 25
19 25 31 11001
```

Only MAC addresses, times and the counter do not have these modifiers.

## A.7. See also

tcpdump(8), perl(1)

## A.8. Authors

The original authors is: Alexander Schinner
The man-page was inspired by the tcpdump man page.

## A.9. Bugs

Bugs? Yes!

# B. Patch for driftnet

The program `driftnet` written by Chris Lightfoot[1] is a nice tool to sniff pictures directly from the net. However, the program is missing a feature to read the network traffic from a tcpdump file. This simple patch extends the program with this feature. The patch has already been mailed to the author.

```
*** driftnet.c Tue Jul  9 21:26:41 2002
--- driftnet-patched.c Sun Sep  5 19:46:00 2004
***************
*** 51,56 ****
--- 51,57 ----
  int tmpdir_specified;
  char *tmpdir;
  int max_tmpfiles;
+ char *input_filename=NULL;

  enum mediatype extract_type = m_image;

***************
*** 458,464 ****
  /* main:
   * Entry point. Process command line options, start up pcap and enter capture
   * loop. */
! char optstring[] = "hi:psSMvam:d:x:";

  int main(int argc, char *argv[]) {
      char *interface = NULL, *filterexpr;
--- 459,465 ----
  /* main:
   * Entry point. Process command line options, start up pcap and enter capture
   * loop. */
! char optstring[] = "hi:psSMvam:d:x:r:";

  int main(int argc, char *argv[]) {
      char *interface = NULL, *filterexpr;
***************
*** 523,528 ****
--- 524,533 ----
                  tmpdir = optarg;
                  tmpdir_specified = 1; /* so we don't delete it. */
                  break;
+
+      case 'r':
+                  input_filename = optarg;
+                  break;

  #ifndef NO_DISPLAY_WINDOW
              case 'x':
***************
*** 662,678 ****
  #endif /* !NO_DISPLAY_WINDOW */
```

[1] http://www.ex-parrot.com/~chris/

```
      /* Start up pcap. */

!     pc = pcap_open_live(interface, SNAPLEN, promisc, 1000, ebuf);
!     if (!pc) {
!         fprintf(stderr, PROGNAME": pcap_open_live: %s\n", ebuf);
!
!         if (getuid() != 0)
!             fprintf(stderr, PROGNAME": perhaps you need to be root?\n");
!         else if (!interface)
!             fprintf(stderr, PROGNAME": perhaps try selecting an interface with the -i option?\n");
!
!         return -1;
      }

      if (pcap_compile(pc, &filter, (char*)filterexpr, 1, 0) == -1) {
--- 667,690 ----
  #endif /* !NO_DISPLAY_WINDOW */

      /* Start up pcap. */
+     if (input_filename){
+       pc = pcap_open_offline(input_filename, ebuf);
+        if (!pc) {
+          fprintf(stderr, PROGNAME": pcap_open_live: %s\n", ebuf);
+  return -1;
+      }
+     }else{
+       pc = pcap_open_live(interface, SNAPLEN, promisc, 1000, ebuf);
+      if (!pc) {
+          fprintf(stderr, PROGNAME": pcap_open_live: %s\n", ebuf);
+
+          if (getuid() != 0)
+              fprintf(stderr, PROGNAME": perhaps you need to be root?\n");
+          else if (!interface)
+              fprintf(stderr, PROGNAME": perhaps try selecting an interface with the -i option?\n");

!         return -1;
!      }
      }

      if (pcap_compile(pc, &filter, (char*)filterexpr, 1, 0) == -1) {
```

# Bibliography


[1] W. Richard Stevens. *TCP/IP illustrated (vol. 1): the protocols*. Addison-Wesley Longman Publishing Co., Inc., 1993.

[2] Judy Novak Stephen Northcutt. *IDS: Intrusion Detection-Systeme*. mitp-Verlag, Bonn, 2001.

[3] H. Stöcker. *Taschenbuch mathematischer Formeln und moderner Verfahren*. Verlag Harri Deutsch, 1995.

[4] Steven M. Bellovin. A technique for counting natted hosts. In *Proceedings of the second ACM SIGCOMM Workshop on Internet measurment*, pages 267–272. ACM Press, 2002.

[5] Traditional ip network address translator (traditional nat) http://www.faqs.org/rfcs/rfc3022.html.

[6] http://httpd.apache.org/docs-2.0/logs.html .

[7] http://www.scriptarchive.com/formmail.html .

[8] http://iandu.s7.xrea.com/unimama/fm_scanners/200309.html .

[9] http://www.softwolves.pp.se/internet/formmail_hall_of_shame/0309 .

[10] http://www.attrition.org/postal/verizon/ .

[11] http://www.modsecurity.org/documentation/ .

[12] http://www.k-otik.com/exploits/09.21.0x333hztty.c.php.

[13] http://www.security-corporation.com/articles-20030718-001.html.

[14] http://www.incidents.org/logs/raw/2002.10.13.

[15] http://www.sans.org/resources/idfaq/ring_zero.php.

[16] http://www.sans.org/y2k/050300-1100.htm.

[17] http://help.undernet.org/proxyscan/.

[18] http://support.novell.com/cgi-bin/search/searchtid.cgi?/10056600.htm.

[19] http://www.europe.f-secure.com/v-descs/adore.shtml.

[20] http://www.sans.org/y2k/adore.htm.

[21] http://www.giac.org/practical/gcia/doug_kite_gcia.pdf.

[22] http://is.rice.edu/ glratt/practical/glenn_larratt_gcia.html.

[23] http://cert.uni-stuttgart.de/archive/intrusions/2002/08/msg00300.html.

[24] nearly each issue of phrack http://www.phrack.org/.

[25] http://www.mmshannon.net/docs/mike_shannon_gcia.pdf.

[26] http://www.giac.org/practical/david_oborn_gcia.html.

[27] http://us.mcafee.com/virusinfo/default.asp?id=description&virus_k=99209.

[28] http://www.microsoft.com/germany/technet/servicedesk/bulletin/blaster.mspx.

[29] http://www.whitehats.ca/main/members/herc_man/files/al_williams_gciapractical.pdf.

[30] http://www.cert.org/advisories/ca-2001-19.html.

[31] http://www.giac.org/practical/gcia/jamell_creque_gcia.pdf.

[32] http://lists.jammed.com/incidents/2001/05/0034.html.

[33] http://www.iana.org/assignments/port-number.

[34] http://www.securityfocus.com/bid/1065/credit/.

[35] http://www.whitehats.com/cgi/arachnids/show?_id=ids552.

[36] http://www.giac.org/gcia.php.

[37] http://www.giac.org/practical/gcia/daniel_clark_gcia.pdf.