Studying IDS signatures using botnet infected honey pots

Johannes Hassmund
Email: johannes.hassmund@liu.se
Supervisor: Nahid Shahmehri, *nsh@ida.liu.se*
Project Report for Information Security Course
Linköping University, Sweden

Abstract

In this report we explore botnet malware using an isolated network of intentionally infected honeypots. The honeypots are placed on an isolated network designed to protect the Internet from our infected hosts. By a passive study of the network traffic between the isolated network and the Internet we suggest an IDS signature that successfully discovers the malware FakeAlert.JB. We also analyze available signatures used for detection of the Conficker.C botnet. We suggest that the study of intentionally infected honeypots bears an important role in the analysis of botnets. To fully qualify as a method for the development of IDS signatures and to obtain a deeper understanding of sophisticated malware, the black box approach used in this project needs to be complemented with analysis using reverse engineering techniques such as disassembly of malware binaries.

1. Introduction

In this section we present the objectives of *LiU IRT* (Linköping University Incident Response Team) and the motivation and goals of the project described in this report. We also present the methods used and the limitations of the project.

1.1 Background

Linköping University Incident Response Team handles intrusions, intrusion attempts, spam, malware incidents, complaints regarding copyright infringement and other IT security related matters within Linköping University. Between 2006 and 2008 in average 44 incidents per year regarding virus and other malware have been recorded. We suspect that all incidents are neither discovered, nor reported so the number of total infections is probably somewhat larger than this number. The majority of recorded incidents regard students connected to the university's WLAN.

It is important to disconnect infected hosts from the network as promptly as possible for several reasons, mainly to prevent further infections and to uphold the university's reputation in the Internet community.

Infections of hosts connected to *LiU-Net* (Linköping University Network) are mainly discovered through the university's *Intrusion Detection System* (IDS), due to anomalies in the use of certain ports, among whose port 25 (SMTP) and the Windows RPC ports (137-139 and 445) are the most prominent. Malware are also discovered due to complaints from external parties or notifications from *Sunet CERT* (Swedish University Network Computer Emergency Response Team) and by the use of antivirus software. In general infected computers have already been disconnected by the IDS when external warnings arrive.

The network traffic patterns that are used to notice suspected infections are mainly the consequence of either a worm that is trying to spread to other hosts on the network, or *botnet* participants sending large amounts of spam emails.

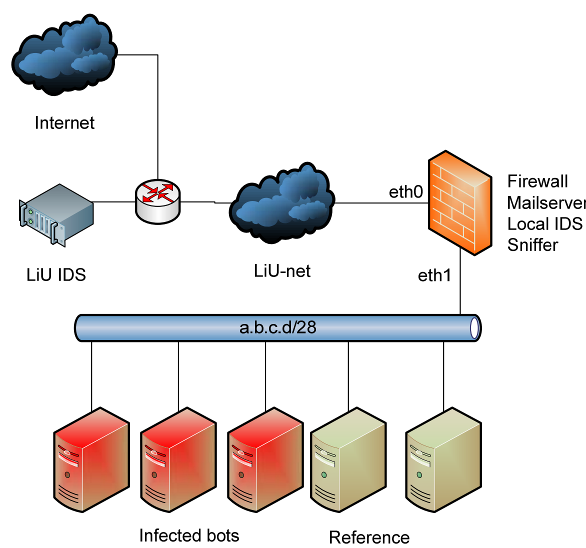

Figure 1. Logical setup of infected honeypots

1.2 Purpose

Even though the current approach used at Linköping University for detection of malware, as described in the previous section, has proven to be efficient, it is not satisfactory since an anomaly does not occur until malicious activity have been ongoing for some time.

The main goal of this project is to identify or develop signatures to be used in Linköping university's IDS in order to detect the control traffic of infected bots rather than the consequences of infection (e.g. massive spam senders). By moving from a reactive to proactive approach we aim to minimize time from infection to time of detection.

A secondary goal is to gain experience from this type of malware study and to establish a platform on which further studies can be performed in a safe manner.

1.3 Method

This project has been pursued as a part of a university course in Information Security at the Department of Computer Science in association with the IT Incident Response Team at Linköping University. The theoretical part of the project is based on a literature study introducing concepts of control channels of botnets. The actual study of IDS signatures has been performed on a network of honeypots connected to a firewall protecting the Internet from the infected bots. This setup is illustrated in Figure 1 and is further described in Chapter 3.

1.4 Limitations

Due to time constraints focus will be put exclusively to three malware binaries; *FakeAlert.JB*, *Conficker.B* and *Conficker.C*. These malwares were chosen since they all have been active on LiU-Net during the time of this project.

2. Botnets overview

A botnet is a group of compromised computers, *bots*, under control by a malicious individual; a *botmaster*. Botnets commonly include mechanisms for self propagation, for example by exploiting security vulnerabilities over the network like a worm, or sending virus infected spam. What distinguishes botnets from other kinds of malware is the ability to establish a command and control channel with the botmaster.

Botnets can range in size from a handful to several hundred thousands of cooperating computers [1]. The most prominent threats of botnets are spamming and *DDoS* (distributed denial of service) attacks.

2.1 History

The first bots appeared in the *IRC* (Internet Relay Chat) community and were designed to perform administrative duties like providing logging capabilities and help channel operators to fight abuse. The bot Eggdrop, initially developed with these purposes in 1993, is considered to be one of the first bots [2]. Development of Eggdrop is still in progress as an open source project.

Computer viruses have been known since the 1970s [3] and the first malicious network worm, the Morris Worm, appeared in 1988 [4]. Even though these self propagation techniques were well known at the time of the first IRC bot appearance, it was not until around the year 2000 that malware authors combined the techniques constructing self propagating botnets [5].

2.2 Control channels

Botnets traditionally use a command and control structure as illustrated in Figure 2. Early bots like Agobot and SDBot [6] utilized the IRC protocol. Infected hosts connect to an IRC server through which the owner of the botnet can issue commands that the hosts carry out. There also exists bots controlled by HTTP, making the control traffic harder to differentiate from normal network traffic patterns and even DNS [7] which might increase chances of control traffic to get through firewalls.

Today peer to peer protocols seem to be bot malware authors' preferred choice since this technique makes tracking harder [7] and has the potential to make the botnet more robust, whereas a static controller host might be shut down, hence pacifying the botnet. The notorious Storm and Conficker botnets both use peer to peer techniques [8] [9].

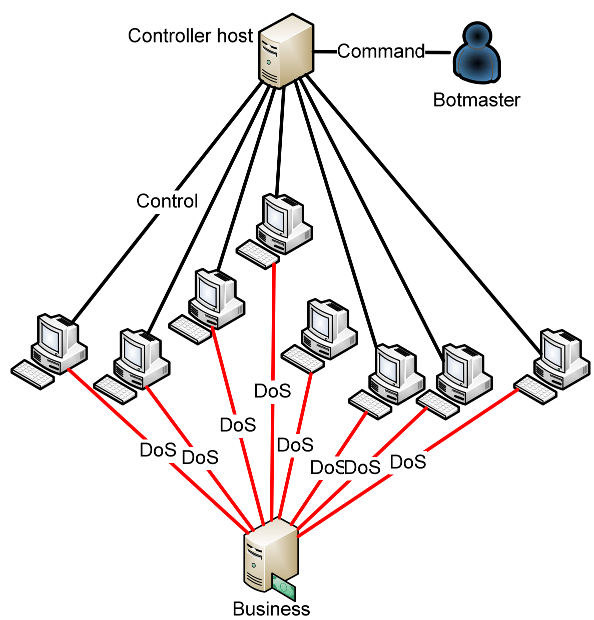

Figure 2. Command and Control structure of a traditional botnet

2.3 Threats

Botnets can be used for numerous malicious purposes. As stated in section 1.1; most bots identified within LiU-Net are found due to the large number of SMTP connections that are initiated. Researchers suggest that botnets is the number one method of choice for spammers [10] [11] and that sending spam is currently the most prominent use of botnets [12].

DDoS attack is another area where Botnets appear to be the perfect tool. To successfully pacify the victim of a DDoS attack, it is desirable for the attacker to utilize a greater amount of bandwidth than what is available for the victim. The effect of several thousands of bots initiating DoS attacks at a coordinated time has the potential to be devastating.

DDoS attacks have been used for extortions of Internet businesses [7] as well as attacks towards business competitors [13]. Recent attacks against Estonia [12] and Georgia [14] show that botnets have the capacity to substantially disturb small countries.

Other threats include hosting of *phishing* web sites [15] and *privacy theft* [16]. The latter has gained increased attention during 2009 with the reveal of GhostNet; a botnet claimed to target Tibetan officials [17]. Privacy theft can be performed by for example downloading documents or the installation of *key loggers* (software that records key strokes) on infected hosts.

The capacity of botnets is not restricted to the individuals or organizations developing them. There are commercial botnets where capacity is sold and charged by the minute of use [12]. We can expect new threats to emerge as new business concepts surface.

3. Implementation of the honeypots

This section describes how the honeypots were set up in an isolated network environment and which actions were taken to protect innocent hosts on the Internet from our honeypots.

3.1 Logical setup

The logical setup of the infected bots is shown in Figure 1. The compromised computers are physical machines installed with Windows XP SP2 (no further patches) acting as full interaction honeypots. These computers are connected to a separate network partly isolated by a firewall. On the same network two reference computers are installed. One of the reference machines is configured with Windows XP SP3, fully patched and the other one carries the same configuration as the infected computer; namely Windows XP SP2 and no patches. After the experiment the reference machines were inspected to conclude if the bots were able to spread within the network.

3.2 Generic firewall configuration

In order to study control traffic of the infected bots we had no choice but to connect the laboratory network to the Internet. This entails some inevitable risks. First we have a major risk of our botnet disturbing and attacking other computers on the Internet, both internal and external to LiU-net. To manage this risk we configured the firewall to block all outgoing traffic to LiU-net. This may seem egoistic, but is necessary since the infected network is a part of LiU-net and the bots are placed inside the university's defense perimeters.

To protect external organizations and Internet users, the firewall was configured to block all traffic on the notorious TCP ports 135, 137-139 and 445. Further raw blocking was considered but was not used since we do not want to make assumptions on how the control traffic will flow. Instead we opted for a thorough monitoring of the traffic, never running the system unless we were sure we could respond to alarms within 15 minutes.

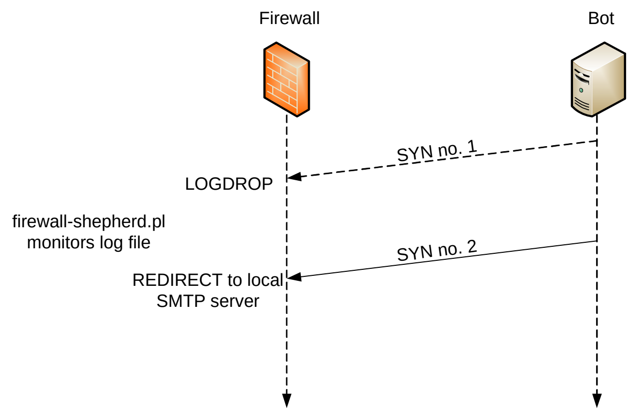

Figure 3. First SYN-packet is log-dropped and analyzed, deciding the faith of further packets

3.3 Simulating successful spam bots

Even though we wanted to stop external attacks and spam we strongly needed the bots to perceive a normal networked environment. This was accomplished by the following setup.

Initially all outgoing connections to TCP port 25 are dropped by the firewall and logged by the *syslog* (system log) as illustrated in Figure 3. A Perl script `firewall-shepherd.pl` is continuously monitoring the syslog. Whenever `firewall-shepherd.pl` discovers a previously unknown IP address, which has been logged due to a connection attempt to port 25, it will try to connect to this address. If there is no reply or the reply does not follow the SMTP standard [18] the IP address will be added to the known list of IP addresses, marked as non-responding and no further actions will be taken. The firewall will continue to drop traffic destined to port 25 of the address. If there is a SMTP server responding on the given address, `firewall-shepherd.pl` will send a polite SMTP `HELO`- and `QUIT`-message. The script will then parse the reply of the server and make a request to another Perl script, the `destination-manager.pl`. Now, the destination manager has knowledge of how the specified IP address should present itself. Finally, `firewall-shepherd.pl` will allow and redirect traffic destined to port 25 of the given address to a local Postfix SMTP server. Accordingly, upcoming SYN-packets (see Figure 3) will be accepted and rerouted to the local Postfix server.

The local Postfix server has been modified to make a connection to `destination-manager.pl` in order to fetch data on how it should present itself towards the clients. As a consequence of this, all clients inside the firewall will perceive that they are communicating with any real SMTP server that is available, but will in reality only communicate with the local Postfix server running on the firewall machine. Figure 4 illustrates the cooperation of the Perl scripts and Postfix.

The postfix server accepts all destinations but delivers all mail to a local spam trap, effectively hindering spam from reaching the Internet.

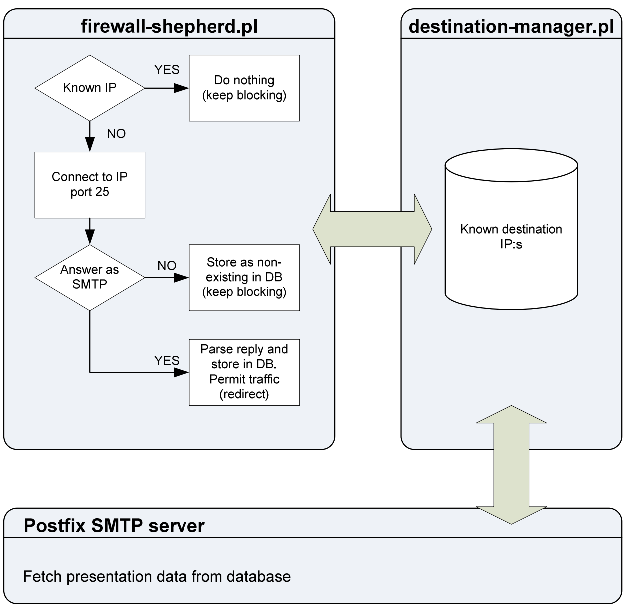

Figure 4. Mechanisms to stealthy capture spam

3.4 Risk of provoking the botnet to DDoS us

Apart from the risk of the infected bots launching attacks towards external machines of the Internet there is also a risk of provoking the botnet to launch a DDoS attack against ourselves. Some botnets defend themselves in this way upon detection of probing or reverse engineering attempts [1]. The ideal situation would be to have a separated research network for the purpose of this project. Since this is not possible no attempt to inject traffic into the botnet will be made; all analysis will be purely passive and we make efforts to be perceived as a normal network of clients.

3.5 Initial IDS configuration

The server running the firewall of the laboratory network was augmented with a local IDS. The purpose of this IDS was to increase the probability to detect attacks towards innocent hosts on the Internet, originating from our honeypots. The local IDS was also used as a platform for experimenting with IDS signatures without disturbing the main IDS of the university. Candidate signatures proven efficient on the local IDS would later be deployed on the main IDS to analyze the impact of false positives; the latter can only be done in a ‘live’ environment.

3.6 Activity recording

Network traffic between the honeypots and the Internet was recorded using `tcpdump` [19] on the firewall server. By using the syntax “`tcpdump –w` *`filename`*”, traffic was recorded in raw format allowing later study in the Wireshark Network Protocol Analyzer [20].

3.7 Client infection

Initially spam e-mails captured by antivirus filters on the university’s e-mail gateway were studied to retrieve appropriate botnet malware binaries for the project. This would however turn out to be a less appropriate source since the malware found this way have not been seen active on the university network. Instead we opted for a selection of malware based on warning e-mails sent to the university’s Incident Response Team by *Sunet CERT* (Swedish university network Computer Emergency Response Team).

The first malware chosen was a botnet binary identified by the AVG antivirus software as “Trojan Horse FakeAlert.JB”. This malware was easily retrieved from the website adorelyric.com shown in Figure 6.

As a second malware we choose to study the Conficker botnet which we have seen some infections of on the university network. An actual binary was somewhat hard to get hands on but eventually an archive of malware where found on the web site www.offensivecomputing.net [21].

4. Analysis of malware and evaluation of IDS signatures

This section presents the malware and IDS signatures chosen for study; namely FakeAlert.JB and the Conficker botnet.

4.1 FakeAlert.JB (adorelyric.com)

During spring 2009 Linköping University received a warning stating that a computer belonging to a department of the university was infected with the “Fast-flux botnet adorelyric.com”. The binary supplied by this web page was identified as “Trojan Horse FakeAlert.JB”.

Fast-flux is a technique used by phishing attackers to make it harder to get rid of malicious sites hosted on compromised machines by rapidly changing the IP-address that the domain name points to [22]. This technique makes it extremely difficult to shut down a phishing site by contacting the ISPs of the hosting web servers. The remaining option is to get the registrar to suspend the domain name. To illustrate the technique a sequence of eight DNS queries of adorelyric.com is shown in Figure 5.

```
# for i in `seq 8` ; do
dig  adorelyric.com +short ; done
[redacted].109.15
[redacted].59.136
[redacted].59.171
[redacted].78.226
[redacted].224.198
[redacted].193.43
[redacted].135.142
[redacted].11.76
```

Figure 5. Repeated lookups of a fast-flux domain

adorelyric.com was one of a number of domain names that, at the time, supplied the web page shown in Figure 6. The web page announces a truly amazing application, allowing the user to secretly read other individuals' SMS messages without access to their cell phone. Upon executing the binary supplied it appears that nothing happens, but without noticing the user the computer starts sending a lot of traffic to various web servers and shortly also receiving HTTP traffic on port 80.

A typical HTTP request and server response made to our infected host is shown in Appendix A. We have not put any significant amount of effort into decoding or decrypting the traffic intercepted. The rest of the captured traffic shows that this is a typical pattern of the communication and as humans it is fairly easy to recognize similar requests. To make the IDS do this matching we focus on parts of the communication deviating from standard HTTP requests. Notice that restricting focus to the beginning of packets minimizes the load on the IDS server.

The request made to the malicious server on our infected host is a HTTP POST request on the form "`POST /coxbgxe.png HTTP/1.1`", the content specification says "`Content-Type: application/x-www-form-urlencoded`". This is a very strange specification; if a remote browser would post form data the receiving URL would hardly be a png-image. Would a HTTP POST-request specifying a png-image ever have this Content-Type specification under normal circumstances?

In order to answer this question an IDS signature was written with the intent to capture these circumstances. This signature was then deployed on the main IDS of the university network. The suggested signature, shown in Figure 8, has so far given zero false positives, still detecting all known instances of the malware studied. Knowing that the signature is deployed give attackers the opportunity to spam the IDS with false positives since it is trivial to craft traffic that triggers the given rule. It is however easy for a security analyst to inspect this traffic and deduce if the host really is infected, studying the response of the HTTP request (shown in Figure A-4, Appendix A).

As far as we can tell, FakeAlert.JB does only spread in the form of a Trojan horse deceiving users to install an 'awesome' program.

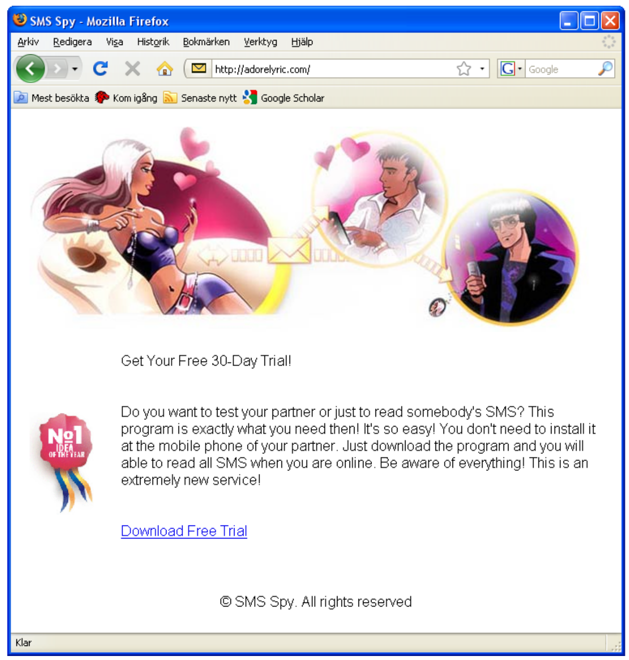

Figure 6. Malicious web page promising an exciting application

4.2 Conficker.B and Conficker.C

Conficker (also known as Downadup, Downup, Conflicker and Kido) [23] is a worm based botnet which has gained quite some attention during the spring of 2009. The original Conficker binary exploits a vulnerability in the Windows *RPC* (Remote Procedure Call) protocol announced by Microsoft on October 23rd, 2008 [24]. Even though Microsoft released patches for the vulnerability at the time of the announcement, Conficker which was first observed about a month later [25] [26], is said to have given rise to "the most dominating infection outbreak since Sasser in 2004" [9].

The first confirmed infection of Conficker at Linköping University was noticed on February 23rd, 2009. Since then about 20 confirmed or suspected infections have been noticed, which can be compared to a total of 34 incidents involving suspected botnet malware during the same period.

Conficker is an interesting piece of malware and seems to differ from the traditional botnets in the sense that the worm rather than the traditional Command and Control structure updates itself with new versions of the binary using a peer to peer approach [9]. A thorough study of the worm and its variants could probably fill a master thesis on its own. In this project we focus exclusively on the ability to discover Conficker by the use of IDS.

4.3 Observations of honeypot infected with Conficker.B

Upon infection of a host with Conficker.B, it shortly starts TCP-scanning the Internet, looking for hosts which have port 445 open. The packets sent are ordinary TCP SYN-packets which by themselves cannot be used as signatures for an IDS. If every try to initiate a TCP connection on port 445 were to be interpreted as a host infected with malware the number of false positives would be unbearable.

The massive amount of connections made however provide an excellent mean to eliminate false positives. Figure 7 shows a warning from the IDS indicating a large number of SYN-packets destined to port TCP/445 on various addresses, all packets originate from a single infected host. Even though we cannot know that the host is infected with Conficker we can be sure that it is performing some malicious activity and should be disconnected from our network promptly.

There are other patterns that can be used for detection of Conficker.B infected clients; for example before starting the SYN-scan the hosts infected with Conficker.B checks their external IP addresses by contacting the web sites www.getmyip.com, www.whatismyip.org, www.whatsmy-ipaddress.com, and checkip.dyndns.org. However, only checking for numerous connection attempts of port 445 has the potential to detect other malware as well as Conficker.B and it is desirable to keep the signatures as simple as possible. We have noticed that the SYN-scan starts immediately after the DNS lookups mentioned, meaning that there is no significant amount of time to be gained regarding by implementing further signatures. Consequently we opt for continued use of this simple approach and not investigating Conficker.B further.

Port 445 alert 4 for 130.236.[redacted]
root [root@[redacted]]
Till: [redacted]
Kopia:

```
 Total connections:       13600  (ignore limit 0)
 Unique destinations:       800 (overflow, could be more) (ignore limit 0

 Current alert level:        4
 Previous alert level:       0

 First lines seen:

 04/21/09-16:47:43.040739 130.236.[redacted]:1065 -> [redacted].118.[redacted]:445
 04/21/09-16:47:43.040765 130.236.[redacted]:1064 -> [redacted].32.[redacted]:445
 04/21/09-16:47:43.040770 130.236.[redacted]:1066 -> [redacted].31.[redacted]:445
 04/21/09-16:47:43.040797 130.236.[redacted]:1067 -> [redacted].44.[redacted]:445
 04/21/09-16:47:43.040805 130.236.[redacted]:1068 -> [redacted].124.[redacted]:445
 04/21/09-16:47:43.040985 130.236.[redacted]:1069 -> [redacted].120.[redacted]:445
 04/21/09-16:47:43.041722 130.236.[redacted]:1072 -> [redacted]77.[redacted]:445
 04/21/09-16:47:43.041735 130.236.[redacted]:1071 -> [redacted].36.[redacted]:445
 04/21/09-16:47:43.041975 130.236.[redacted]:1073 -> [redacted]25.[redacted]:445
 04/21/09-16:47:43.042975 130.236.[redacted]:1080 -> [redacted].54.[redacted]:445

 Last lines seen:

 04/21/09-16:50:01.894706 130.236.[redacted]:2580 -> [redacted].43.[redacted]:445
 04/21/09-16:50:02.113921 130.236.[redacted]:2586 -> [redacted].52.[redacted]:445
 04/21/09-16:50:02.113941 130.236.[redacted]:2592 -> [redacted].126.[redacted]:445
 04/21/09-16:50:02.113946 130.236.[redacted]:2585 -> [redacted].7.[redacted]:445
 04/21/09-16:50:02.113950 130.236.[redacted]:2590 -> [redacted].85.[redacted]:445
 04/21/09-16:50:02.113982 130.236.[redacted]:2584 -> [redacted].101.[redacted]:445
```

Figure 7. IDS warning upon a large number of tries to initiate connections on port 445, originating from an infected host

An observation worth mentioning is that Conficker.B spreads aggressively on USB memories, but was not able to spread to vulnerable computers within the isolated network during an eight hour period.

4.4 Conficker.C

The C-variant of Conficker behaves in quite different ways than the earlier variants. A host freshly infected with Conficker.C neither seem to spread the malware by USB memories, nor probe for open TCP ports 445. Instead it tries to synchronize to the botnet using a UDP based peer to peer protocol. The IP-addresses chosen to scan for is decided by an algorithm involving the current date [9]. A suggested signature to match this synchronization traffic is available at [27], shown in Figure 9.

This signature successfully detects Conficker.C but causes a significant amount of false positives. We suspect the Internet phone application Skype as one of the sources of these, making the signature less appropriate for a large network's IDS.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg: "BOTNET TESTING RULE:
Candidate to detect adorelyric.com-like malware"; flow:to_server; content:"POST
/"; depth: 10; content:".png HTTP/1.1"; depth: 30; content: "Content-Type:
application/x-www-form-urlencoded"; depth: 200; sid: 1100001; rev:1;)
```

Figure 8. Candidate signature to detect control traffic of FakeAlert.JB/adorelyric.com-like malware

```
alert udp $HOME_NET [!1720,!1722,!2427,!5060,1024:] -> $EXTERNAL_NET [!1720,!1722,
!2427,!5060,1024:] (msg:"ET CURRENT_EVENTS Possible Downadup/Conficker-C P2P encrypted
traffic UDP Ping Packet (bit value 1)"; dsize:>19; byte_test:1, &, 1, 19; threshold:
type both, track by_src, count 95, seconds 50; classtype:trojan-activity; reference:
url,mtc.sri.com/Conficker/addendumC/ ; reference:url,
www.emergingthreats.net/cgi-bin/cvsweb.cgi/sigs/CURRENT_EVENTS/CURRENT_Conficker ;
sid:666661; rev:3;)
```

Figure 9. Signature that detects Conficker.C P2P traffic, but give rise to numerous false positives

Yegneswaran [28] has suggested an IDS plug in based on Conficker.C's internal algorithms for calculating IPs of peers to sync with. The plug in has been tried out on the university's network and has only given reason to a small, manageable number of false positives. The plug in however needs to be rewritten slightly, as well as analyzed in terms of load impact, before being deployed.

5. Related Work

The work presented in this report touches upon a great amount of previously conducted research. In this section we present a small selection of such work.

Gu et al. have studied methods to recognize botnet command and control channels using network anomaly detection. They study correlations of network traffic, rather than signatures, thus enabling detection of previously unknown botnets even if the payload of the control traffic is encrypted. Their focus is however put exclusively on HTTP and IRC based control channels. [29]

The P2P botnet Storm has been studied by Holz et al.. They have conducted their research by gathering botnet binaries using *spam traps* (e-mail addresses set up solely for the purpose of receiving spam) and installing these on honeypots. Finally they have successfully infiltrated the Storm botnet and injected their own commands into the botnet control channels, thereby disturbing and measuring the botnet. [8]

Rajab et al. have presented an overview of botnet techniques and tracking methods. Their work was conducted before the significant rise in popularity of P2P techniques. [16]

The use of honeypots has been given an extensive presentation by Provos et al. Although their book "Virtual Honeypots" focus on virtualization techniques they also cover aspects relevant for standalone honeypots. [7]

6. Conclusion and Further Work

We conclude that a protected network of honeypots has proven to be a great tool for the security department or a security analyst who wishes to get hands-on experience of malware. We expect such experience to provide required insights when determining which signatures that may be appropriate for use with an IDS or to benchmark an antivirus software.

The honeypot setup used included a somewhat sophisticated mechanism for capturing outgoing spam. In retrospect we can conclude that this mechanism was never needed since the malware studied did not initiate any spam sending sessions. It was however a safety net that we would not have wanted to be without. A simpler approach to this matter may have been appropriate but we strongly advice against experiments with malware on Internet connected hosts without at least some basic approach towards spam redirection.

As shown in the case of FakeAlert.JB, passive monitoring of botnet control traffic can provide sufficient basis for the design of IDS signatures. We can also use this approach to confirm that current techniques of malware discovery are successful and appropriate, as in the case of Conficker.B.

However, as malware gets more sophisticated we expect limited success in the quest of designing IDS signatures when experiments are restricted to the black box approach used in this project. In addition to passive study of network traffic in isolation we believe that a more in-depth understanding requires analysis of the malware binaries themselves. The results of our study of available signatures for detection of Conficker.C are an example of this. The first signature tested gave rise to far too many false positives to be useful for our purposes. The more successful approach of detection, based on work by Yegneswaran [28], requires analysis of the binary itself.

We believe that reverse engineering and disassembly of malware binaries will keep proving to bear an important role in further research of specific botnets. It might however be in more sophisticated traffic correlation analysis that we will find the most efficient techniques in the quest of disarming botmasters.

7. References

[1] **Sandep, Sarat, Sandeep and Terzis, Andreas.** *Measuring the Storm Worm Network, HiNRG Technical Report: 01-10-2007.* 2007.

[2] **Grizzard, Julian B., et al.** *Peer-to-Peer Botnets: Overview and Case Study.* Cambridge, Massachusetts : USENIX Association, 2007.

[3] **Kaspersky Lab.** Viruslist.com. *History of Malware.* [Online] [Cited: 04 15, 2009.] http://www.viruslist.com/en/viruses/encyclopedia?chapter=153310937.

[4] **Orman, Hilarie.** The Morris Worm: A Fifteen-Year Perspective. *Security & Privacy, IEEE.* 2003, Vol. 1, 5.

[5] **McLaughlin, Laurianne.** Bot Software Spreads, Causes New Worries. *IEEE Distributed Systems Online.* 2004, Vol. 5, 6.

[6] **Barford, Paul and Yegneswaran, Vinod.** An Inside Look at Botnets. [book auth.] Mihai Christodorescu, et al. *Malware Detection (Advances in Information Security).*

[7] **Provos, Niels and Holz, Thorsten.** *Virtual Honeypots, From Botnet tracking to Intrusion Detection.* Boston, Massachusetts : Addison-Wesley Professional, 2007. ISBN 978-0321336323.

[8] **Holz, Thorsten, et al.** *Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm.* San Francisco, California : USENIX Association, 2008.

[9] **Porras, Phillip, Saidi, Hassen and Yegneswara, Vinod.** An Analysis of Conficker's Logic and Rendezvous Points. *Malware Threat Center.* [Online] [Cited: 04 23, 2009.] http://mtc.sri.com/Conficker/.

[10] **Ramachandran, Anirudh and Feamster, Nick.** Understanding the network-level behavior of spammers. *ACM SIGCOMM Computer Communication Review.* 2006, Vol. 36, 4.

[11] **Geer, David.** Malicious bots threaten network security. *Computer.* 2005, Vol. 38, 1.

[12] **Lesk, Michael.** The New Front Line: Estonia under Cyberassault. *Security & Privacy, IEEE.* 2007, Vol. 5, 4.

[13] **Freiling, Felix C., Holz, Thorsten and Wicherski, Georg.** Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks. *Computer Security – ESORICS 2005.* Berlin : Springer, 2005.

[14] **Keizer, Gregg.** Cyber attacks knock out Georgia's Internet presence. *MIS Asia.* [Online] 08 12, 2008. [Cited: 05 03, 2009.] http://mis-asia.com/news/articles/cyber-attacks-knock-out-georgias-internet-presence.

[15] **Ianell, Nicholas and Hackworth, Aaron.** Botnets as a Vehicle for Online Crime. *The International Journal of Forensic Computer Science.* 2007, Vol. 2, 1.

[16] **Rajab, Moheeb Abu, et al.** *A Multifaceted Approach to Understanding the Botnet Phenomenon.* Rio de Janeriro, Brazil : ACM, 2006. 1-59593-561-4.

[17] **Information Warfare Monitor.** *Tracking GhostNet.* s.l. : Information Warfare Monitor, 2009.

[18] **Klensin, J.** Simple Mail Transfer protocol. *The Internet Engineering Task Force.* [Online] [Cited: 03 26, 2006.] http://tools.ietf.org/html/rfc5321.

[19] **tcpdump.org.** tcpdump/libpcap. [Online] [Cited: 04 26, 2009.] http://www.tcpdump.org/.

[20] **Wireshark Foundation.** *Wireshark.* [Online] [Cited: 04 28, 2009.] http://www.wireshark.org/.

[21] **Offensive Computing, LLC.** Offensive Computing. [Online] [Cited: 04 17, 2009.] http://www.offensivecomputing.net/.

[22] **Moore, Tyler and Clayton, Richard.** *Examining the impact of website take-down on phishing.* Pittsburgh, Pennsylvania : ACM, 2007. ISBN:978-1-59593-939-8.

[23] **Conficker Working Group.** Conficker Working Group . [Online] http://www.confickerworkinggroup.org/wiki/.

[24] **Microsoft corporation.** Microsoft Security Bulletin MS08-067 – Critical. *Microsoft TechNet.* [Online] [Cited: 04 24, 2009.] http://www.microsoft.com/technet/security/Bulletin/MS08-067.mspx.

[25] **Symantec.** W32.Downadup. *Symantec Security Response.* [Online] [Cited: 04 26, 2009.] http://www.symantec.com/security_response/writeup.jsp?docid=2008-112203-2408-99&tabid=2.

[26] **McAfee Inc.** W32/Conficker.worm. *McAfee Avert® Labs Threat Library.* [Online] http://vil.nai.com/vil/content/v_153464.htm.

[27] **Conficker Working Group.** Network Detection. *Conficker Working Group.* [Online] http://www.confickerworkinggroup.org/wiki/pmwiki.php/ANY/NetworkDetection.

[28] **Yegneswaran, Vinod.** Conficker C Peer-to-peer Detector. *SRI International.* [Online] [Cited: 04 29, 2009.] http://mtc.sri.com/Conficker/contrib/plugin.html.

[29] **Gu, Guofei, Zhang, Junjie and Lee, Wenke.** *BotSniffer: Detecting Botnet Command and Control Channels.* San Diego, CA : s.n., 2008.

Appendix A

```
No.     Time        Source                     Destination           Protocol Info
    830 170.820551  [redacted]          130.236.[redacted]         TCP      [TCP segment of a
reassembled PDU]

0000  00 13 21 06 84 cd 00 15 c5 5d 71 f1 08 00 45 00   ..!......]q...E.
0010  00 ee 79 f1 40 00 73 06 8f e7 [redacted] 82 ec   ..y.@.s......m..
0020  [redacted] 11 e5 00 50 09 a6 da b1 38 e0 db 48 50 18   .....P....8..HP.
0030  b4 00 f3 bc 00 00 50 4f 53 54 20 2f 63 6f 78 62   ......POST /coxb
0040  67 78 65 2e 70 6e 67 20 48 54 54 50 2f 31 2e 31   gxe.png HTTP/1.1
0050  0d 0a 52 65 66 65 72 65 72 3a 20 4d 6f 7a 69 6c   ..Referer: Mozil
0060  6c 61 0d 0a 41 63 63 65 70 74 3a 20 2a 2f 2a 0d   la..Accept: */*.
0070  0a 43 6f 6e 74 65 6e 74 2d 54 79 70 65 3a 20 61   .Content-Type: a
0080  70 70 6c 69 63 61 74 69 6f 6e 2f 78 2d 77 77 77   pplication/x-www
0090  2d 66 6f 72 6d 2d 75 72 6c 65 6e 63 6f 64 65 64   -form-urlencoded
00a0  0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 4d 6f   ..User-Agent: Mo
00b0  7a 69 6c 6c 61 0d 0a 48 6f 73 74 3a 20 31 33 30   zilla..Host: 130
00c0  2e 32 33 36 2e 31 2e 32 35 33 0d 0a 43 6f 6e 74   .236.1.253..Cont
00d0  65 6e 74 2d 4c 65 6e 67 74 68 3a 20 39 37 38 0d   ent-Length: 978.
00e0  0a 43 61 63 68 65 2d 43 6f 6e 74 72 6f 6c 3a 20   .Cache-Control:
00f0  6e 6f 2d 63 61 63 68 65 0d 0a 0d 0a               no-cache....
```

Figure A-1. Request from a remote party to our infected host, part I

```
No.     Time        Source                     Destination           Protocol Info
    832 170.835875  [redacted]          130.236.[redacted]         HTTP     POST /coxbgxe.png
HTTP/1.1  (application/x-www-form-urlencoded)

Frame (1032 bytes):

0000  00 13 21 06 84 cd 00 15 c5 5d 71 f1 08 00 45 00   ..!......]q...E.
0010  03 fa 79 f2 40 00 73 06 8c da [redacted] 82 ec   ..y.@.s......m..
0020  01 fd 11 e5 00 50 09 a6 db 77 38 e0 db 48 50 18   .....P...w8..HP.
0030  [redacted] b3 56 00 00 61 3d 5f 77 41 41 41 73 52 77   ...V..a=_wAAAsRw
0040  6b 61 7a 71 70 6b 57 52 48 42 68 79 4a 30 46 4a   kazqpkWRHBhyJ0FJ
0050  33 71 30 50 55 78 75 36 46 34 6e 2d 77 33 62 51   3q0PUxu6F4n-w3bQ
0060  39 59 4c 69 42 42 71 5a 43 57 36 71 66 44 58 63   9YLiBBqZCW6qfDXc
0070  43 4b 4b 4d 45 36 2d 68 44 31 4e 36 39 49 6a 61   CKKME6-hD1N69Ija
0080  59 45 2d 6f 4d 42 6d 33 44 53 37 77 66 63 7a 32   YE-oMBm3DS7wfcz2
0090  43 74 52 48 6e 47 4c 57 7a 4a 38 32 4f 44 76 54   CtRHnGLWzJ82ODvT
00a0  42 4a 73 68 52 76 34 55 6f 51 54 62 59 31 31 48   BJshRv4UoQTbY11H
00b0  52 6f 72 44 75 46 45 5a 5f 51 66 47 48 6c 66 53   RorDuFEZ_QfGHlfS
00c0  32 39 4b 38 50 4d 6f 65 31 50 31 2d 33 47 38 31   29K8PMoe1P1-3G81
00d0  59 58 33 54 39 63 6e 52 6e 43 61 68 38 66 42 74   YX3T9cnRnCah8fBt
00e0  32 68 5f 47 41 6c 45 6d 68 6a 41 43 67 44 43 4e   2h_GAlEmhjACgDCN
00f0  6c 57 5f 47 38 5a 64 5f 39 32 58 53 68 45 33 42   lW_G8Zd_92XShE3B
0100  71 52 62 39 66 36 32 39 38 41 34 77 57 6a 6d 43   qRb9f6298A4wWjmC
0110  71 46 6b 6a 55 76 63 54 6a 4a 32 44 53 41 78 4d   qFkjUvcTjJ2DSAxM
0120  4d 6e 44 74 34 59 47 5a 6f 70 53 77 73 54 4f 74   MnDt4YGZopSwsTOt
0130  54 33 6d 55 6f 2d 6c 47 48 6a 4b 50 68 67 41 65   T3mUo-lGHjKPhgAe
0140  42 45 64 4f 6c 31 56 56 45 76 6b 48 2d 48 69 62   BEdOl1VVEvkH-Hib
0150  53 56 76 34 49 6b 74 33 54 4e 63 6a 34 57 7a 6f   SVv4Ikt3TNcj4Wzo
0160  44 7a 4d 72 47 43 64 61 51 65 48 43 4a 70 67 72   DzMrGCdaQeHCJpgr
0170  49 6f 5f 6e 4d 6a 68 2d 46 33 62 5a 4b 51 34 76   Io_nMjh-F3bZKQ4v
0180  76 65 61 73 45 34 79 44 71 51 4a 4b 51 50 36 35   veasE4yDqQJKQP65
```

```
0190  6c 31 6c 2d 4d 57 64 35 56 55 4d 64 4d 69 4e 43   l1l-MWd5VUMdMiNC
01a0  69 35 33 36 65 47 33 73 50 6a 45 5a 58 6e 46 36   i536eG3sPjEZXnF6
01b0  56 31 34 59 4e 45 4a 79 59 56 6f 36 64 75 73 69   V14YNEJyYVo6dusi
01c0  75 46 73 69 6a 4d 51 64 6a 76 4b 6d 35 33 45 6a   uFsijMQdjvKm53Ej
01d0  30 31 34 55 6a 41 79 6a 4e 4e 67 74 6d 6d 39 77   014UjAyjNNgtmm9w
01e0  7a 4f 66 47 42 57 73 30 49 66 50 46 57 49 55 31   zOfGBWs0IfPFWIU1
01f0  4f 47 4b 44 56 6b 69 64 70 63 70 50 75 36 43 36   OGKDVkidpcpPu6C6
0200  70 79 76 5a 56 74 67 39 30 69 54 75 43 42 47 72   pyvZVtg90iTuCBGr
0210  78 45 31 77 2d 59 46 4d 72 43 37 79 4a 64 4c 35   xE1w-YFMrC7yJdL5
0220  4a 4e 6f 78 70 54 38 69 38 73 37 63 45 56 46 62   JNoxpT8i8s7cEVFb
0230  5a 76 7a 73 66 65 30 47 4d 46 30 4d 30 33 71 4e   Zvzsfe0GMF0M03qN
0240  4a 59 30 65 55 35 6d 59 37 4d 77 56 6d 73 34 48   JY0eU5mY7MwVms4H
0250  35 59 78 62 6a 55 79 32 68 79 73 61 6f 72 4f 7a   5YxbjUy2hysaorOz
0260  32 55 7a 52 70 51 73 6e 7a 4f 64 79 34 74 71 6c   2UzRpQsnzOdy4tql
0270  72 77 73 68 39 53 5a 4c 5a 65 58 41 5a 6b 51 50   rwsh9SZLZeXAZkQP
0280  45 6d 38 59 35 6a 57 31 50 63 56 33 78 71 45 43   Em8Y5jW1PcV3xqEC
0290  50 66 35 70 6f 4c 78 6e 78 57 63 77 62 79 6c 69   Pf5poLxnxWcwbyli
02a0  30 35 59 75 43 56 72 53 6f 31 5f 32 46 4a 69 78   05YuCVrSo1_2FJix
02b0  33 46 6e 32 6a 72 62 6e 57 6a 5f 6d 47 76 71 76   3Fn2jrbnWj_mGvqv
02c0  6d 50 4e 56 47 67 6c 72 64 35 4a 33 78 61 52 49   mPNVGglrd5J3xaRI
02d0  54 32 79 53 71 61 68 6f 4a 53 68 67 6e 70 66 6b   T2ySqahoJShgnpfk
02e0  70 34 71 34 4d 76 4e 6a 78 47 53 31 54 6f 6b 62   p4q4MvNjxGS1Tokb
02f0  64 4e 30 65 5f 47 70 36 43 70 71 59 6d 46 4f 57   dN0e_Gp6CpqYmFOW
0300  4f 6b 30 6f 71 74 77 66 53 56 55 6a 7a 56 6d 68   Ok0oqtwfSVUjzVmh
0310  56 62 6c 7a 7a 71 30 2d 63 6f 64 62 73 30 57 4f   Vblzzq0-codbs0WO
0320  6c 6d 44 78 4e 50 59 65 54 4c 50 70 52 46 75 51   lmDxNPYeTLPpRFuQ
0330  37 77 47 62 56 4a 30 61 64 5a 67 4e 61 65 73 6a   7wGbVJ0adZgNaesj
0340  56 4f 61 66 5f 41 38 6e 71 67 6c 6c 46 43 36 75   VOaf_A8nqgllFC6u
0350  59 46 34 34 64 6d 72 30 33 47 31 6b 6b 58 33 6e   YF44dmr03G1kkX3n
0360  65 67 70 50 55 36 32 6d 56 69 34 35 6e 52 53 62   egpPU62mVi45nRSb
0370  75 2d 6a 61 47 76 46 67 77 72 52 56 62 78 44 46   u-jaGvFgwrRVbxDF
0380  6b 69 49 45 55 54 5a 67 43 4a 32 37 78 71 49 6e   kiIEUTZgCJ27xqIn
0390  4d 50 36 63 2d 50 6b 65 59 66 71 63 30 46 65 4f   MP6c-PkeYfqc0FeO
03a0  70 52 6c 2d 44 66 47 74 6a 6b 5f 37 5a 70 76 59   pRl-DfGtjk_7ZpvY
03b0  30 78 6a 45 6b 75 44 6f 71 78 49 4f 65 70 58 47   0xjEkuDoqxIOepXG
03c0  76 4c 78 75 51 30 30 47 38 75 62 64 36 4b 47 70   vLxuQ00G8ubd6KGp
03d0  37 76 77 71 2d 39 71 71 6c 49 75 4a 59 6b 61 54   7vwq-9qqlIuJYkaT
03e0  34 47 4c 70 38 30 34 6a 38 4c 38 4a 41 54 6c 31   4GLp804j8L8JATl1
03f0  62 42 70 74 72 6e 36 74 49 4c 58 76 57 4f 63 26   bBptrn6tILXvWOc&
0400  62 3d 41 41 41 41 41 41                           b=AAAAAA
```

Figure A-2. Request from a remote party to our infected host, part II

```
POST /coxbgxe.png HTTP/1.1
Referer: Mozilla
Accept: */*
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla
Host: 130.236.[redacted]
Content-Length: 978
Cache-Control: no-cache
```


a=_wAAAsRwkazqpkWRHBhyJ0FJ3q0PUxu6F4n-w3bQ9YLiBBqZCW6qfDXcCKKME6-hD1N69IjaYE-
oMBm3DS7wfcz2CtRHnGLWzJ82ODvTBJshRv4UoQTbY11HRorDuFEZ_QfGHlfS29K8PMoe1P1-
3G81YX3T9cnRnCah8fBt2h_GAlEmhjACgDCNlW_G8Zd_92XShE3BqRb9f6298A4wWjmCqFkjUvcTjJ2DSAxMMnDt4Y
GZopSwsTOtT3mUo-lGHjKPhgAeBEdOl1VVEvkH-HibSVv4Ikt3TNcj4WzoDzMrGCdaQeHCJpgrIo_nMjh-
F3bZKQ4vveasE4yDqQJKQP65l1l-
MWd5VUMdMiNCi536eG3sPjEZXnF6V14YNEJyYVo6dusiuFsijMQdjvKm53Ej014UjAyjNNgtmm9wzOfGBWs0IfPFWI
U1OGKDVkidpcpPu6C6pyvZVtg90iTuCBGrxE1w-
YFMrC7yJdL5JNoxpT8i8s7cEVFbZvzsfe0GMF0M03qNJY0eU5mY7MwVms4H5YxbjUy2hysaorOz2UzRpQsnzOdy4tq
lrwsh9SZLZeXAZkQPEm8Y5jW1PcV3xqECPf5poLxnxWcwbyli05YuCVrSo1_2FJix3Fn2jrbnWj_mGvqvmPNVGglrd
5J3xaRIT2ySqahoJShgnpfkp4q4MvNjxGS1TokbdN0e_Gp6CpqYmFOWOk0oqtwfSVUjzVmhVblzzq0-
codbs0WOlmDxNPYeTLPpRFuQ7wGbVJ0adZgNaesjVOaf_A8nqgllFC6uYF44dmr03G1kkX3negpPU62mVi45nRSbu-
jaGvFgwrRVbxDFkiIEUTZgCJ27xqInMP6c-PkeYfqc0FeOpRl-
DfGtjk_7ZpvY0xjEkuDoqxIOepXGvLxuQ00G8ubd6KGp7vwq-
9qqlIuJYkaT4GLp804j8L8JATl1bBptrn6tILXvWOc&b=AAAAAA

Figure A-3. The same request as in Figure A-1 and Figure A-2 in pure ASCII. The shaded data is used in the proposed signature of Figure 8

```
Server Response in ASCII (not shown as raw data):
HTTP/1.1 200 OK
Server: nginx/0.6.34
Date: Sun, 26 Apr 2009 12:40:09 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/5.2.8

172
```


_wAAAQ_4znIvP1ISxjAbYZWIlmzuM4VYuLUBN1RYxMnC8nPQcHv_RiwCdUneNxKlt1rxkof42TjDnNaEA0cYiY2DeXT2O3
cg6-kmyFDh-EpYgPTGvfD5bIGFVGbp7To-
LUBP3OWNCdJWcAZmx4IGEHPZV1jw2XNRV6t9jQ5B3ZWps4K0otzoVAAvWTZM887cVwl2kQMylwWIy05cP5r5OZ-
DS5JbeTmAOntBuHtAijp-0KjoW_lOKSdkLfZiy2zhPFLufCSEFQ9eaM4dJuR_rBSIRvHgWRCFONxb6r-
_3ATN6k8MHSZf15gHcp0_5mlpmH5uwfJ6MoN9XVZ-E2OlD3AeG4-re0Gk17nae7U7s5L9k1kw1g

Figure A-4. The response (in ASCII) from our infected host to the request above. The pattern of this response has the potential to track down false positives.