Global Information Assurance Certification Paper

Cleaning Up the Back Yard

A discussion on your mother's home network security.

GCIA Gold Certification

Author: Wil Knoll, wil.knoll@axia.com
Adviser: Rick Wanner

Contents

My Mom and the Internets

Every Sunday I visit my mother. It has become customary, as I am sure it is with other family geeks, to begin each Sunday visit with a check in on her computer to make sure that everything is patched, updating, and cleaned out. During these check-ins she asks questions off of a list that she creates over the course of the week. The questions are predominately “Should I click/open/use some random button/attachment/program?” She is thankfully savvy enough to know to ask before clicking.

Not only do my parents run a business out of their home, but their computers have become libraries of precious memories and personal documents. Outside of a solid backup regimen, which most average users do not follow, the common home network is barely prepared to deal with the types of threats that exist on the Internet. The largest issues facing the average user today are phishing attempts, viruses and worms. A simple NAT firewall can stop most worms from entering the network unsolicited, but a simple NAT firewall is poorly equipped to deal with much beyond that. And to add, those NAT firewalls have been targeted themselves (Higgins, 2007).

It is possible to clean up the back yard with Free Open Source Software and a little design. Using off the shelf components and Open Source software the family geek can deploy a more multilayered security stance that will provide far more visibility and control over the network. This is not to say that large swaths of the Internet can be cleaned up just by plugging in a box, but to say that

if anything should be a safe haven on the internet, it should be the family network, the backyard. It makes sense to clean up the backyard before taking on the world's trash.

Conventions

The following are the conventions for txt that will be used in this paper.

#sysctl net.inet.ip.forwarding=1	Operating systems commands are represented in this font style, and preceded by a pound sign. This style indicates a command that is entered at a command prompt or shell.
/etc/rc.local	Filenames, paths and directories are represented in this style.
net.inet.ip.forwarding 0 -> 1	The results of a command and other computer output.
http://malwaredatabase.net	URLs will be represented in this style.

The Problem

First, the default configuration on most home Firewalls tend to be set up with two rules. One, block any new traffic in. Two, allow any traffic out. This allows for easy integration with a home network, but does not take into consideration any more advanced configuration. Not all outbound traffic is legitimate, and can point to a compromise. This often goes un-noticed because that traffic is not blocked or alerted on.

This default policy of allowing everything out can allow data loss. Personal information could be leaked off the network by any piece of software, and it could send it anywhere it wishes. Any DNS request can come from any host out to anywhere on the internet. This isn't a problem if there is a trust relationship with the domain name server. But more often there is not, users are unaware of DNS and how they ended up using one server over another. Nor are end users aware of who controls the DNS they are using, or if their computers have changed which DNS they are resolving from. Recently, malware designers have created networks of their own rouge DNS servers to facilitate phishing and ad-revenue click theft, and used their malware to point a victim's computer at those DNS networks. (Hacquebord, 2008).

The fact that DNS is traditionally allowed out in most configurations has even produced projects designed to escape captive portals by tunnelling through DNS (Nussbaum, 2006). As well as allowing DNS out to arbitrary servers, ICMP is also allowed out

traditionally, and this protocol also can provide a route out of a network (Gil, 2005). I mention this to illustrate a point on egress rules and how an attacker can approach navigating a network. The end user should never have to go through such means as tunnelling one protocol over another to get out of their own network.

Lastly, host based security can only take us so far. Antivirus running on a host machine is only as good as its definition file. And it's only a single engine doing a single scan. Antivirus engines themselves have been proven vulnerable, and exploits are publicly available for some (Xue, 2008). A defence-in-depth approach would advocate having multiple layers of antivirus scanning, different engines at different places in line to the end host. The importance of having a host based antivirus service running is still just as important, but having multiple scanning engines running on different devices increases the probability of stopping malicious programs.

Goals

1. Provide fire-walling and other gateway network services to the internal network, including anti-spoof and strong egress rules.
2. Manage DNS requests to protect from phishing attacks.
3. Provide Antivirus scanning at the perimeter for POP3 and web traffic

Egress rules are a very basic way to lock down a network. The benefits of egress rules for the home user are less obvious then from a corporate environment, but not dissimilar. Tighter outbound rules can slow data loss, and break malware trying to do harm on the network. It also helps network administrators keep tabs on what the end users are doing on the network. A more altruistic reason to use egress rules would be to stop the home network from being used to attack others if it was compromised (Hatch, 2003).

Managing DNS requests gives a slight performance boost if there are multiple hosts browsing the network by allowing the server to cache requests locally. But the largest benefit of managing the DNS requests going off network is running those requests through OpenDNS, a free DNS resolver that manages a large anti-phishing database known as PhishTank.

OpenDNS (www.opendns.org) has made quite a name for itself in its few years of operation. Since 2006, OpenDNS has grown from serving 1 billion queries in its first month to 7 billion queries per day in October 2008. OpenDNS maintains a large network of resolvers, spread across the USA with one in the United Kingdom as well. One strength lies in how every URL has been categorised. Any host that uses its

services is able to create an account and set which categories should be resolved or not. For parents and schools, this is a wonderful resource, as sites with questionable content or content that breaks policy is not able to be resolved.

PhishTank was the first community powered phishing information clearinghouse. Over 50,000 members add phishing URLs which are then qualified and tracked. Community members must register with the site and then peer-review each others submissions, which cuts through most of the noise and makes submitters responsible for the content they submit. No phish is verified until at least two other users (based against certain criteria) have vetted it. Several webbrowers use PhishTank for phishing protection, and PhishTank provides API access for integration into other tools.

Using OpenDNS and PhishTank to block phishing URLs is the simplest solution to dealing with phishing. This stops most phishing attacks cold, as the end user can never resolve the IP address of the link they clicked. OpenDNS has also been proactive in securing their servers to threats against the DNS protocol (Danchev, 2008).

Part of the multilayered approach to security is to scan for virus using varying engines at different part of the transmission path. In the corporate environment there is traditionally virus scanning happening on the mail server, and then again on the desktop. Placing a virus scanner on the perimeter that is inspecting files transferred through e-mail or web gives the network a second chance to catch malicious files inbound, using a different scanning engine.

There are many software distributions available that can fulfill most if not all of these goals. There is even an OpenBSD based distribution or two available that would fit the bill. But using one of them would not fulfill the forth goal of this paper, to learn how to put them all together for ourselves.

Deployment

To fulfill our goals, the following software packages have been selected.

1. Squid Proxy and squidclamAV – If there are more than two or three machines behind the firewall that are web-browsing, using a proxy server to cache common images and other data could help speed the browsing experience. At the same time, it provides the tie in to ClamAV for scanning web-browsing through squidclamAV.
2. P3Scan -> A transparent mail proxy that allows us to pass all mail to SPAM and Virus engines. SPAM protection is not detailed here, but an easy extension once P3Scan is in place.
3. ClamAV - As a tool kit ClamAV provides us with several tools to do virus scanning at the perimeter. Automatic database updates, and protection that is widely accepted to rival commercial offerings
4. DNSmasq and OpenDNS – we are going to cache and relay DNS requests to OpenDNS. Not only can this speed browsing using cached DNS responses, but OpenDNS provides a cleaned domain name service. Phishing protection, domain blocking, adult site blocking, all happens, giving us a multi layered approach to protection.

OpenBSD (http://openbsd.org)has been selected as the base operating system for this build. The rational is that the OpenBSD system is open, code audited, and well known for its practical paranoia. The hardware requirement for a home gateway firewall is trivial compared to the contemporary workstation. At a bare minimum, two network

interface cards, a 486 CPU, and 64 megs of RAM should suit most situations.

Part of the reason that OpenBSD is a natural fit is that it ships with the pf Firewall. pf has earned a name for itself for it's simple configuration and human readable configuration, while building in strong features normally only found in corporate sized firewalls.

One note on P3Scan. The version of P3Scan that will be deployed is quite old. P3Scan 0.9.3 was released several years ago, and the project was orphaned. Recently, development has begun again but has not yielded a stable release. Because this new release was not considered ready for prime time, I decided to stay with a stable version.

The Build

A quick thought about the build process. Every system administrator likes to have boxes built out their own way. In enterprise environments, where corporate standards and compliance come into play, more and more considerations are made about the file system layout to patch levels. This paper is targeted as a guide for the home administrator to help clean up the backyard, and is more concerned with the higher level configuration of services and not worried about how much space an individual likes to have for swap or what PS1 exported prompt they like, or if bash and screen are installed.

Having said that, the following configs were made with an internal interface that was set to 172.20.0.1/24, with the external interface getting DHCP from the ISP. OpenBSD 4.3 was current at the time of writing, which was used for this document.

The OpenBSD install process is one of the most caustic to the new user. Beyond the documentation available directly from the OpenBSD site directly (http://openbsd.org) there are many sites online that have OpenBSD install walkthroughs (http://www.openbsd101.com/installation.html). The CD liner of purchased OpenBSD CDs has the full printout of the install script as well. For brevity sake, we will leave getting OpenBSD installed as an exercise in competency for the reader.

Once OpenBSD has been installed, changes must be made to the /etc/sysctl.conf and to the running kernel state of the system. Specifically, I needed to set up the machine to forward packets. By

default, OpenBSD's kernel is not configured to forward packets. The `sysctl` command is used to change the running state of the machine.

```
#sysctl net.inet.ip.forwarding=1
net.inet.ip.forwarding: 0 -> 1
```

To make this change permanent, the /etc/sysctl.conf file needs to be edited so that the net.inet.ip.forwarding=0 is changed to net.inet.ip.forwarding=1. This way at boot, the Firewall knows it is to forward packets. As for all the other options, if we are not going to be running them then why expose more of the firewall?

The pf firewall

pf's configuration file works great with macros, and will accept a macro for almost anything. Traditionally pf.conf files begin with naming the interfaces and hosts. The "not_private" list macro can be helpful when crafting rule sets for traffic out to the Internet. I have specified traffic not (!) destined to an RFC 1918 private network as a catch all destination, which is tighter than a 0.0.0.0 or ANY route by specifying only the addresses that could be routed on the Internet.

```
ext_if="fxp0"
int_if="xl0"
loop_if="lo0"
lan1="172.20.0.0/24"
not_private="{ !192.168.0.0/16, !172.16.0.0/12, !127.0.0.0/8, !10.0.0.0/8, !0.0.0.0/8, !255.255.255.255/32 }"
```

Using groups of services instead of writing rules for each service saves time and makes it easy to track what set of ports are being allowed to whom. The "fw_" ports will be terminated on the firewall itself, and the "internet_" ports will be passed through. Also, SSH,HTTP and HTTP proxy are added to the list of services on the

firewall itself. To allow administration from the network SSH is allowed, and HTTP is allowed for the virus landing page. I have SSH and SSL-VPN allowed out to the internet for my own purposes, but some networks may not need any direct connections at all.

An "isp_smtp" address is created, to only allow outbound mail connections to the ISP's SMTP server. Some networks already do this, blocking outbound TCP/25 connections to stop the spread of spam, but others do not.

```
fw_tcp="{ 22, 80, 3128 }"
fw_udp="{ 53, 123 }"
internet_tcp="{ 22, 10443 }"
isp_smtp="24.71.223.43"
```

"grind" option will be used on sensitive services where outsiders grinding logins should not be allowed, brute forcing SSH or MYSQL logins for example. It translates to say that any source can only have a total of three connections, and they may not create them at a rate faster than two every five minutes. If they do, they will be added to the abusers table and every packet/session will be globally dropped. "grind" is only the name of the macro and could be any string desired.

```
grind="(max-src-conn 3, max-src-conn-rate 2/5, overload <abusers> flush global)"
```

Flags used to add new TCP sessions to the state table, but not TCP sessions with illegal options for the first packet in the hand shake.

```
tcp_flags="flags S/SA keep state"
```

pf can be configured to allow other programs on the firewall to dynamically add rules if needed. To set this up, and anchor is required. This can be used for a number of functions, from allowing Snort IPS to feed rules into pf creating an IDS (http://snort2c.sourceforge.net/), to setting up two level authentication for wireless segments (http://www.openbsd.org/faq/pf/authpf.html). Here an anchor is set to redirect FTP traffic to the ftp-proxy that OpenBSD ships with.

```
nat-anchor "ftp-proxy/*"
rdr-anchor "ftp-proxy/*"
```

Redirections need to be made next. As squid is not a transparent proxy, clients will either be directly configured or automatically configured by DHCP and the wpad.dat file. This means it will not need a port redirection as the web clients know what port to speak to it on. But P3Scan, the POP3 mail scanner will require one. The mail clients are going to attempt to connect out to the Internet on TCP/110, but the redirection will have pf capture that traffic and pass it to the port that P3Scan is listening on.

Also, I created one redirect that will take a high TCP port and pass it in to sshd. When writing the “pass in” rules later on, we must remember that the redirection will happen first, so to allow in the outside SSH traffic, we have to allow traffic to port 22, not 40443.

```
rdr on $int_if proto tcp from $lan1 to $not_private port 110 -> $loop_if port 8110
rdr on $ext_if proto tcp from $not_private to $ext_if port 40443 -> $ext_if port 22
```

Setting up basic NAT functions is simple. This line directs the firewall to NAT on the external interface from Lan1 to anything, using the IP of the external interface.

```
nat on $ext_if from $lan1 to any -> $ext_if
```

The next line places the ftp-proxy anchor. FTP does something a little bit different than most other protocols on the Internet, starting up a second TCP session back to a client. Sitting behind a firewall that connection back is often dropped, as it seem to be some unsolicited connection. Using the anchor ftp-proxy can update the running configuration of pf with an inbound rule to allow the data connection back from the server.

```
anchor "ftp-proxy/*"
```

Anti-spoof is essentially a pre-canned rule that ships with pf. It basically makes a list of the IP networks and their associated interfaces. If 192.168.1.0/24 is hanging off of interface X, then packets sourced from it should not be seen inbound on interface Y. The Anti-spoof option drops them if such a situation arises.

```
antispoof quick for { $loop_if, $int_if }
```

Unlike Cisco and other network devices, in pf the Default Deny can come first on the rule chain. The following translates as “Drop anything from an IP in the abusers table. Block any packet trying to enter the external interface.”

```
block log quick from <abusers>
block in log on $ext_if all
```

Allowing SSH to run exposed on the Internet is a decision left to each administrator. If SSH is required, then using a high TCP port and obscuring the service slightly is a good idea. Adding the grind option will make sure that bots and brute force login attempts will only get the IP of the perpetrator blocked for any type of connection type. The redirection created earlier is parsed first, so the rule must reflect that and is written for the destination port after rewriting.

```
pass in on $ext_if inet proto tcp from $not_private to $ext_if port 22 $tcp_flags $grind
```

"Allow traffic to leave the external interface as long as it is TCP/UDP or ICMP". This allows the firewall to do required operations without worrying about the egress rules that are applied against the internal hosts. Protocols beyond TCP/UDP and ICMP are not used by the firewall so are not allowed at all. The internal interface will have the ingress rules set for the local interface which perform the role of "egress" rules, or "out to the Internet".

```
pass out on $ext_if inet proto tcp all $tcp_flags
pass out on $ext_if inet proto { udp, icmp } all keep state
```

Here all packets are blocked both on the ingress and egress of the internal interface. This is the default deny on the interface.

```
block in log on $int_if all
block out log on $int_if all
```

Using the "not_private" list, we allow traffic destined to the Internet and not other private subnets through the firewall. This

only works if those packets are destined to the list of ports within "internet_tcp".

```
pass in on $int_if inet proto tcp from $lan1 to $not_private port $internet_tcp $tcp_flags
```

To allow a quick connectivity check from behind the firewall, without logging into it, pings are allowed through to one known IP address. This allows any internal host to check for connectivity out to a known good host (OpenDNS in this case), by allowing only ICMP echo requests out and only ICMP echo replies back in.

```
pass in on $int_if inet proto icmp from $lan1 to 208.67.220.220 icmp-type echoreq keep state
```

The next rules are the ones that allow the traffic from the network into the firewall itself for webproxy, NTP, and DNS. Also, SSH to the firewall itself is allowed here for administration.

```
pass in quick on $int_if inet proto tcp from $lan1 to $int_if port $fw_tcp $tcp_flags
pass in quick on $int_if inet proto udp from $lan1 to $int_if port $fw_udp keep state
```

DHCP receives its own special case rule, as the destination address in DHCP requests is broadcast, and not the address of the firewall. UDP traffic addressed to the broadcast address will be allowed into the internal interface. Setup of dhcpd is outside the scope of this paper but quite simple (http://www.openbsdsupport.org/OpenBSD-as-a-DHCP-Server.html).

```
pass in quick on $int_if inet proto udp from any to 255.255.255.255 port 67 keep state
```

I allow ICMP against the internal interface on the firewall to help with network setup.

```
pass in quick on $int_if inet proto icmp from $lan1 to $int_if icmp-type echoreq keep state
```

P3Scan also has its own rule, as traffic is redirected at the loopback address of the firewall (lo0), and not the internal interface.

```
pass in quick on $int_if inet proto tcp from $lan1 to loo port 8110 $tcp_flags
```

And lastly, the only type of SMTP traffic allowed out should be the traffic destined for the ISP's SMTP servers.

```
pass in quick on $int_if inet proto tcp from $lan1 to $isp_smtp port 25 $tcp_flags
```

After saving the changes to the config file, normally /etc/pf.conf, we would use pfctl –e to enable the firewall. This will also spit out any parse errors. It is possible to use pfctl –s all to output from pf all rules, sessions, counts, and other info.

pf also creates a pseudo network interface for its log output. It is possible to use tcpdump to sniff packets from that interface, pflog0, to see what is getting logged. To make this useful, we must only add log to the rules that we are interesting in seeing output for at that time. As a rule of thumb it is a good start to use logging on internal block rules, then add log to others as needed for debugging services. In the setup provided above I have logging set for the block rules on the internal interface and the block rule for the <abusers> table.

The /etc/rc.conf file must be changed to start pf at boot time. I changed pf=NO to pf=YES and saved the file.

So now the network has a NAT/Firewall in place that is using strong egress and ingress rules to decide how traffic flows through it. But right now, web browsing and POP3 mail are not being passed through as the firewall doesn't know how to cleanse them yet.

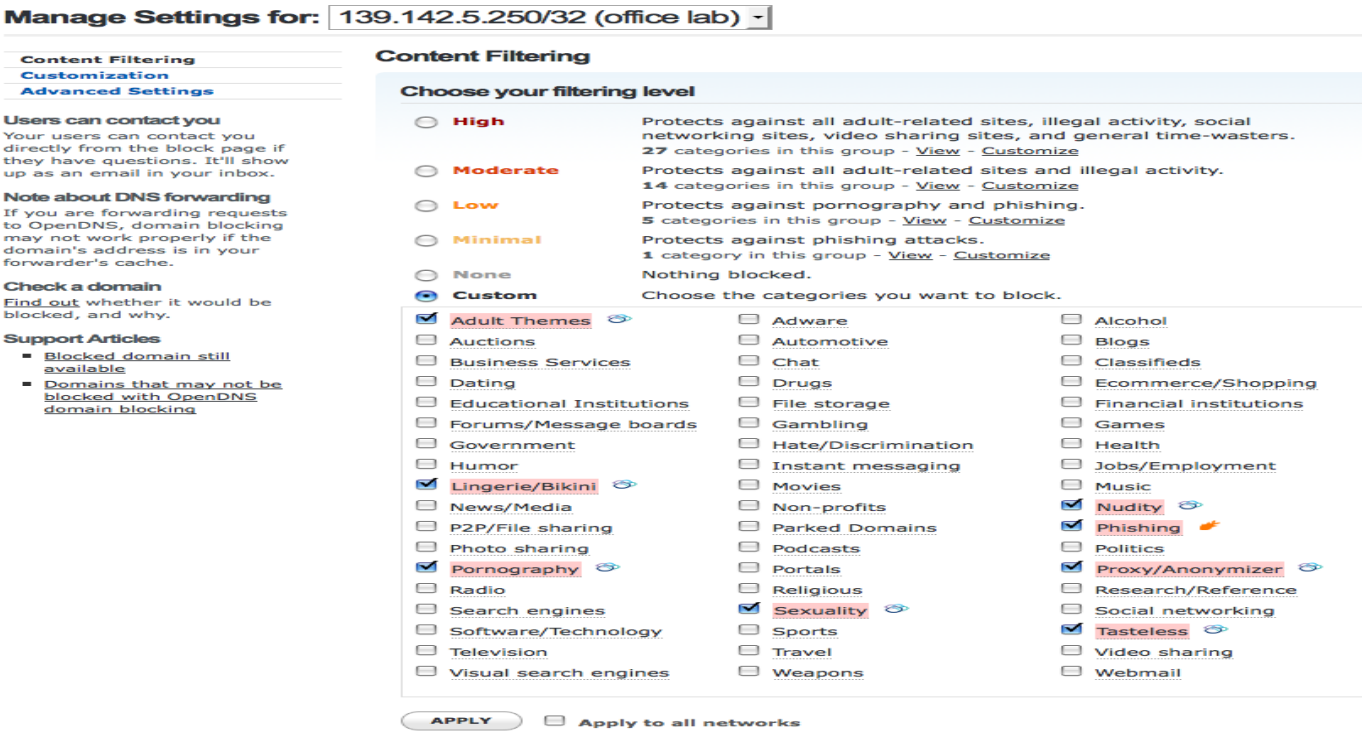
Appendix A contains the entire pf configuration file without comments.

OpenDNS

The use of OpenDNS can be accomplished by pointing a domain name client at it. However, the real power lies in creating an account and tailoring the responses that are resolved or not. To really benefit it, an account must be created. OpenDNS will create one or many Network records, and each can have its own settings. The Network records are basically the external IP of a system using OpenDNS. They also offer a dynamic DNS client that will keep a particular network record aligned with a dynamically assigned address.

After creating an account, I tailored my settings for the network. The following is a screen grab of the web interface for OpenDNS. More than just phishing and malware related sites can be filtered out, making this a great tool for parents as well as network administrators. I have elected to block adult content, as well as proxy sites. If I chose to do web filtering against content in the future, blocking proxy sites will cut down on clients avoiding the

filters. Shown below are the settings for the network “Office Lab”, where the test box was built.



Once the account has been created and configured, the DHCP client needs to be told not to overwrite the DNS settings when it renews DHCP. I edited the /etc/dhclient.conf file to do so.

request subnet-mask, broadcast-address, routers, domain-name,
domain-name-servers, host-name;

Should read

request subnet-mask, broadcast-address, routers, domain-name,
host-name;

After removing the domain-name-severs option, the DHCP client needs to be restarted following that as well.

The final step in is to point the DHCP client to OpenDNS's servers. /etc/resolv.conf dictates what name servers should be used when looking up addresses.

nameserver X.X.X.X

with

nameserver 208.67.220.220
nameserver 208.67.222.222

Ports

Although the OpenBSD FAQ prefers that users stick to using the pre-compiled packages on any OpenBSD system, the packages do lag behind the ports tree to some extent. Even then, the ports tree is only as current as the time the community can donate. To get all the software playing together, I found that the ports tree worked best. Almost no software complained about the versioning of any other software's install when everything was built from ports.

After downloading the ports tarball from the OpenBSD ftp site (ftp://ftp.openbsd.org/pub/OpenBSD/4.3/ports.tar.gz) and untaring it to it's home in /usr/ports, I began by installing the following ports to make sure that everything required for other software is available: gettext wget bash gmake unarj unrar unzip arc lha zoo lzo gmp autoconf curl pcre python php5-core

After php5-core is installed, php needs to be initialized and a config file for it placed in the chroot for Apache. Otherwise, Apache would not be able to serve the virus info page that the firewall will redirect hosts too upon detection. PHP also needs temp directory that it can write too. Apache needs a restart to read in the php5 config as well.

```
ln -s /var/www/conf/modules.sample/php5.conf /var/www/conf/modules
mkdir /var/www/tmp
chown www /var/www/tmp
apachectl restart
```

DNSmasq

DNSmasq is a lightweight DHCP server and DNS forwarder. It is one of three pieces of software that will be installed from source, as the ports tree has not caught up to the version released to fix the DNS vulnerability that Dan Kaminsky disclosed (US-CERT TA08-190B) (Kaminsky, 2008). Although using OpenDNS mitigates the attack, there is no reason to have a vulnerable application publicly exposed.

One part of the Kamisky DNS flaw revolves around the lack of source port randomisation for DNS clients. If the source port was not randomised enough then the number of spoofed packets required to fool or poison a DNS client was quite small. Although patching the DNS client fixed this issue, in some cases the firewalls in front of those clients could strip away the randomness in the source ports (Cross, 2008). This firewall will not strip away the randomness however, as the service will make requests directly off of the external port and is therefore not NATed. If DNS resolution was allowed from behind the firewall this would still not be much of a

concern. pf implements source port randomisation properly, and so is not effected by the NAT concern. (Hart, 2008).

With DNSmasq 2.45 downloaded (http://www.thekelleys.org.uk/DNSmasq/) and built, it's configuration must be tailored to the network. As DNSmasq is a lightweight program, it also has a very lightweight configuration file. The comments have been removed to leave only the configuration options set. Of note is the dhcp-option=252 setting. This is the dhcp option that points at a proxy configuration file for dynamically addressed web clients. Not all browsers/dhcp-clients can support this directly. Those that do support this will not need to have any proxy set after squid is put onto the network.

```
domain-needed
local=/test.ca/
interface=xlo
bind-interfaces
domain=unit2.ca
dhcp-range=172.20.0.200,172.20.0.220,255.255.255.0,12h
dhcp-option=42,0.0.0.0
dhcp-option=252,http://172.20.0.1/wpad.dat
dhcp-authoritative
log-queries
log-dhcp
```

The `wpad.dat` file is very simple.

```
function FindProxyForURL(url, host)
{
  return "PROXY 172.20.0.1:3128";
}
```

This simply points a dynamic client at the proper port for web browsing.

Squid can be configured to run transparently, and pf can rewrite packets to the proper port without configuring the web clients themselves. But through testing the most stable platform that worked with antivirus scanning and proxying was a non-transparent set up.

DNSmasq will be started at boot time by the /etc/rc.local script, Apendix B.

Apache

Apache is part of the base install on OpenBSD. To have Apache started at boot time, /etc/rc.conf must be edited. httpd_flags=NO should be changed to httpd_flags="". Apache will serve the wpad.dat file to any client that requests it.

A redirection page is placed in /var/www/htdocs/, to which the user is redirected by squid to if a detection is returned by ClamVM. The page will let the user know why the page they have requested is not allowed. A simple php page is crafted to dynamically generate the page with information about the detection. This file was saved as /var/www/htdocs/virus.php.

```
<?php
/* A Simple Virus Detection info page */

$u=htmlspecialchars($_GET['url']); // used to sanitize the code
?>
<html>
<head><title>Infection Detected</title>
</head>
<body>
```

This page was blocked because of something found on <?php echo($u); ?>. Please contact your administrator.

</body></html>

Squid

From ports, I installed the Squid 2.6 STABLE18 package from all the other versions available. There are numerous configuration directives that can be used. Squid itself could have several papers written about it, and has.

The line that is of most concern is

```
url_rewrite_program /usr/local/bin/squidclamav
```

.

This invokes squidclamAV, which is the conduit from squid to the ClamAV engine. As well, domain name servers should be declared.

```
dns_nameservers 208.67.220.220 208.67.222.222
```

This is to make sure that squid is also using OpenDNS if it does a lookup. The full config is located in Apendix C. Before squid can be run for the first time, several files that it expects must be touched, and the work directories ownership changed to _squid.

```
#touch /var/squid/logs/access.log
#touch /var/squid/logs/cache.log
#touch /var/squid/logs/store.log
#chown -R _squid /var/squid
```

Squid is then called with squid -z to build it's cache. Then squid is started with simply squid. The server will be started at boot time by the /etc/rc.local script, Appendix B.

ClamAV

ClamAV is available from ports. The lines that are most important in its config are

```
LogFile /var/log/clamd.log
LogTime yes
TemporaryDirectory /tmp
DatabaseDirectory /var/db/clamav
TCPSocket 3310
TCPAddr 127.0.0.1
MaxThreads 20
User _clamav
```

There is a scanmail section of the configuration file, which is to facilitate integration with other mail programs and a local mail queue. Because this machine is not hosting mail, just capturing it with the firewall and pushing it through P3Scan to be scanned, the scanmail section can be ignored. Simply put, the ClamAV scanning deamon is listening on TCP port 3310. The pf rules also do not allow outside connections to that port. Only processes on the local box connecting over the loopback address can connect.

Not only does the scanning engine need to be configured, but also freshclam, which is ClamAV's update script. A cron job must be configured to run freshclam to make sure that we have the most recent definition files. The /etc/freshclam.conf file looks like the following.

```
DatabaseDirectory /var/db/clamav
UpdateLogFile /var/log/freshclam.log
LogTime yes
DatabaseOwner _clamav
DNSDatabaseInfo current.cvd.clamav.net
DatabaseMirror db.ca.clamav.net
DatabaseMirror database.clamav.net
MaxAttempts 5
Checks 24
```

All that needs to be added to the root user's crontab is

```
30 * * * * /usr/local/bin/freshclam >/dev/null 2>&1
```

ClamAV needs to be added to the rc.local script to start at boot time. Please see Apendix B for that. Also, the log files must be touched and their ownership changed before running the update.

```
#touch /var/log/clamd.log
#touch /var/log/freshclam.log
#chown _clamav /var/log/clamd.log
#chown _clamav /var/log/freshclam.log
#freshclam
```

squidclamAV

squidclamAV is the conduit that will pass webpages from squid to the ClamAV scanner. It can be downloaded from http://www.samse.fr/GPL/squidclamAV/, and 3.5 was the most recent version at the time of writing. BSD's file system layout can be slightly different from other systems, so to compile properly on OpenBSD, squidclamAV requires the auto-configure script be called in the following manner so the linker knows where to find the libraries that will be loaded at run time.

```
#env LDFLAGS=-L/usr/local/lib/ CPPFLAGS=-I/usr/local/include/ ./configure
```

The /etc/squidclamav.conf file's important lines are the following.

```
redirect http://172.20.0.1/virus.php
proxy http://127.0.0.1:3128/
clamd_ip 127.0.0.1
clamd_port 3310
```

This instructs squidclamAV to pass from Squid to ClamAV through the localhost TCP port. When ClamAV finds a virus in a URL, it will send back a status code which will tell squidclamAV to rewire the URL. This can cause a slight delay when downloading files, as everything must be downloaded onto the system itself, decompressed, scanned, and then passed to the client that originated the request. However, it is an understandable trade off for that extra layer of scanning.

squidclamAV is invoked by squid itself, so it does not need to be added to the /etc/rc.local script. Logs need to be created just like squid and ClamAV.

```
#touch /var/log/squidclamAV.log
#chown _squid /var/log/squidclamAV.log
```

P3Scan

P3Scan is limited currently. It has not seen a stable release in almost three years. However, development recently started up again and several development releases have come out. For this system, 2.3.2, the most recent stable release makes the best sense. It will not scan POP3S connections (it crashes out with a kind error), but

there is hope that a new stable release will resolve that. Until then, the system will pass POP3S connections straight out and only deal with POP3 connections. A 3.0 release candidate version is available for testing, but considering the purpose of the Firewall, every piece of software should be a stable release version.

P3Scan 2.3.2 is available in an OpenBSD package at http://rubyurl.com/ZYSL from the ComixWall team, an OpenBSD based firewall distribution. After decompressing and building, the configuration file was changed to the environment.

```
user = _p3scan
notifydir = /var/spool/p3scan/notify
virusdir = /var/spool/p3scan
justdelete
emergcon = wintr@unit2.ca
scannertype = clamd
scanner = 127.0.0.1:3310
virusregexp = .*: (.*) FOUND
```

If ClamAV finds a virus inside of an e-mail, a replacement message is sent, using the template found at /etc/p3scan/p3scan.mail. The default message includes lots of output about the process that found the virus, but for replacement messages to the home network user, a simpler message was used. The trailing period is important and denotes the end of an e-mail when being transmitted, specifically a "<CRLF>.<CRLF>" represents the end of mail data as defined by RFC 2821.

```
MIME-Version: 1.0
Content-Transfer-Encoding: 8bit
Content-Type: text/plain;
   charset="iso-8859-1"

A mail message that was sent to you was found to contain a potentially harmful file.
```

Instead of the infected email this message has been sent to you.

Virus name:
%VIRUSNAME%
(Supposed) Sender of the email:
%MAILFROM%
Sent To:
%MAILTO%
On Date:
%MAILDATE%
Subject:
%SUBJECT%

.

P3Scan needs to be added to the /etc/rc.local file to make sure it starts at boot along with all the other programs loaded at boot. Again, this is shown in Apendix B. Finally, the needed directories should be created and their ownership changed to P3Scan.

```
#mkdir /var/spool/p3scan /var/spool/p3scan/notify
#chown -R _p3scan /var/spool/p3scan
```

Reboot and Troubleshoot

At this point, more than one sanity reboot should be conducted to make sure that everything will come back up properly after a power failure. The BIOS of the device should be checked to make sure that it will reboot after a power failure, and that it will not halt on keyboard or mouse errors (if the machine is to run headless).

Case-Studies -> Breaking The Attack

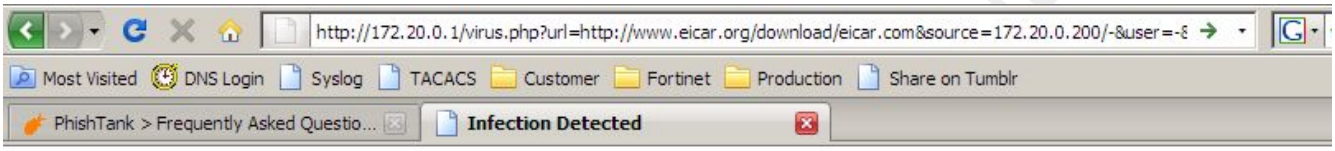
Case A) Eicars test file downloaded through web

When browsing to the EICAR website at http://www.eicar.org while watching the /var/squid/log/access.log file, I tried to download the EICAR virus test file.

```
Fri Jul 18 11:38:21 2008 [22497] DEBUG File size is '.2f
Fri Jul 18 11:38:21 2008 [22497] DEBUG Sending STREAM to clamd.
Fri Jul 18 11:38:21 2008 [22497] DEBUG Received port 1612 from clamd.
Fri Jul 18 11:38:21 2008 [22497] DEBUG Trying to connect to clamd [port: 1612].
Fri Jul 18 11:38:21 2008 [22497] DEBUG Scanning data received against clamd stream
Fri Jul 18 11:38:21 2008 [22497] DEBUG Sending data to clamd
Fri Jul 18 11:38:21 2008 [22497] DEBUG Write 68 bytes on 68 to socket
Fri Jul 18 11:38:21 2008 [22497] DEBUG Connection to clamd on port: 1612 closed.
Fri Jul 18 11:38:21 2008 [22497] DEBUG Reading clamd scan result.
Fri Jul 18 11:38:21 2008 [22497] DEBUG received from Clamd: stream: Eicar-Test-Signature FOUND
Fri Jul 18 11:38:21 2008 [22497] LOG Redirecting URL to:
http://172.20.0.1/virus.php?url=http://www.eicar.org/download/eicar.com&source=172.20.0.200/-&user=-
&virus=stream:+Eicar-Test-Signature+FOUND
Fri Jul 18 11:38:21 2008 [22497] DEBUG End reading clamd scan result.
Fri Jul 18 11:38:21 2008 [22497] STAT Virus Scanning process time  0.348 second(s)
Fri Jul 18 11:38:21 2008 [22497] DEBUG Virus found send redirection to Squid.
Fri Jul 18 11:38:21 2008 [22497] STAT Total process time  1.857 second(s)
```

The EICAR test file is a standardised test file designed to test anti-virus engines without having to resort to passing live viruses around. It was developed by Paul Ducklin and others and is available in several formats from http://www.eicar.org/anti_virus_test_file.htm. Most anti-virus scanners should identify this file with no trouble, it has been around for more than five years and is well documented. It's also a simple 68 byte long DOS program that prints out its name. It is not malicious.

My browser was redirected to the virus.php page installed on the firewall, warning me of the attempted download. Should this be live malware the same should happen. This will be tested in the next case.



.

Case B) Downloading of a Zipped virus file

I then attempted to pass a zipped known virus file past ClamAV. Using an old infected .dll file detected as Trojan.BHO-81 by ClamAV, I placed the zipped file on a webhost and tried to download the file. The access.log contained a lot of output commenting how much of the stream had been sent (debug was turned on for testing purposes) and has been cut down to save on space.

```
Fri Jul 18 11:53:08 2008 [22497] DEBUG Request:http://unit2.ca/mal/dll.zip 172.20.0.200/- - GET
Fri Jul 18 11:53:08 2008 [22497] DEBUG regex matched: http://unit2.ca/mal/dll.zip
Fri Jul 18 11:53:08 2008 [22497] DEBUG Getting header for url http://unit2.ca/mal/dll.zip
Fri Jul 18 11:53:08 2008 [22497] DEBUG File size is '.2f
Fri Jul 18 11:53:08 2008 [22497] DEBUG Sending STREAM to clamd.
Fri Jul 18 11:53:08 2008 [22497] DEBUG Received port 1556 from clamd.
Fri Jul 18 11:53:08 2008 [22497] DEBUG Trying to connect to clamd [port: 1556].
Fri Jul 18 11:53:08 2008 [22497] DEBUG Scanning data received against clamd stream
/clipped for brevity/
Fri Jul 18 11:53:24 2008 [22497] DEBUG Connection to clamd on port: 1556 closed.
Fri Jul 18 11:53:24 2008 [22497] DEBUG Reading clamd scan result.
Fri Jul 18 11:53:26 2008 [22497] DEBUG received from Clamd: stream: Trojan.BHO-81 FOUND
Fri Jul 18 11:53:26 2008 [22497] LOG Redirecting URL to:
http://172.20.0.1/virus.php?url=http://unit2.ca/mal/dll.zip&source=172.20.0.200/-&user=-&virus=stream:+Trojan.BHO-81+FOUND
Fri Jul 18 11:53:26 2008 [22497] DEBUG End reading clamd scan result.
Fri Jul 18 11:53:26 2008 [22497] STAT Virus Scanning process time  17.810 second(s)
Fri Jul 18 11:53:26 2008 [22497] DEBUG Virus found send redirection to Squid.
Fri Jul 18 11:53:26 2008 [22497] STAT Total process time  17.851 second(s)
```

Again, the combination of Squid, squidclamAV and ClamAV stopped the file from being downloaded. ClamAV can be configured for different levels of recursion for compressed files inside of compressed files. The resulting landing page looks like the following.

Case C) Phishing e-mail blocked by ClamAV

There are actually two levels of Phishing protection on this Firewall. The first is ClamAV, and the second is the use of OpenDNS. Some e-mails will be blocked by signature by ClamAV, and the ones that get through will not be able to direct the end user to a malicious page as OpenDNS will not resolve the link.

Apparently, Chase Manhattan wanted me to update my Online Banking credentials with them (despite not being a customer of theirs), but for some reason the e-mail's link pointed to a website that was not Chase Manhattan's at all. The Chase-Phishing e-mail contains two worrisome items. The first is a link that, although looks like it points to http://chaseonline.chase.com , actually points to http://chaseonline.chase.com.err.com.es . The Chase-Phishing e-mail can be found in Appendix D, along with a screen shot of the phishing page.

The second is a section of hidden text which is rendered in the html body, but with a white font on a white background. I have not been able to discern the purpose of this text yet.

The hidden text contains several streams which look like the following:

include, FA2X. hex: 0x4938, 0x38, 0x3592, 0x71115789 0x5, 0x713, 0x4, 0x11, 0x1, 0x09, 0x00, 0x0357, 0x30, 0x475, 0x376, 0x3, 0x87, 0x5, 0x76642792

When this e-mail is requested from the POP3 client through P3Scan, the following output is generated.

13:34:50 p3scan[11918]: Forked, pid=23606, numprocs=1
13:34:50 p3scan[23606]: setting the virusdir to /var/spool/p3scan/children/23606/
13:34:50 p3scan[23606]: Initialize Context
13:34:50 p3scan[23606]: starting proxy
13:34:50 p3scan[23606]: POP3 Connection from 172.20.0.200:49419
13:34:50 p3scan[23606]: Real-server address is 139.142.140.126:110
13:34:50 p3scan[23606]: starting mainloop
13:34:50 p3scan[23606]: <-- +OK dovecot ready.
13:34:50 p3scan[23606]: --> CAPA
13:34:50 p3scan[23606]: Client checking server CAPABILITIES...
/clipped for brevity/
13:34:50 p3scan[23606]: --> RETR 8
13:34:50 p3scan[23606]: RETR 8 (1)
13:34:50 p3scan[23606]: <-- +OK 5505 octets
13:34:50 p3scan[23606]: Caught MIME/Subj line, closing header buffer.
13:34:50 p3scan[23606]: Informing email client to wait...
13:34:50 p3scan[23606]: notified=1
13:34:50 p3scan[23606]: got '.\r\n', mail is complete.
13:34:50 p3scan[23606]: Invoking scanner
13:34:50 p3scan[23606]: Clamd TCP scanner says hello
13:34:50 p3scan[23606]: Clamd TCP scanner says goodbye: 2
13:34:50 p3scan[23606]: Scanner returned 2
13:34:50 p3scan[23606]: POP3 from 172.20.0.200:49419 to 139.142.140.126:110 from "Chase" <smrfs@chaseonline.chase.com> to "Wil" <wil@augmentedreality.ca> user: wil virus: Email.Phishing.DblDom-124 file: /p3scan.Om1tu

At this point P3Scan created the virus warning e-mail and sent that back to the POP3 client as opposed to the troublesome e-mail itself.

Case D) Phishing link blocked by OpenDNS

Phishing attacks cost more than $3 Billion in losses last year according to Gartner (McCall, 2007). However, the length of time that most Phising websites remains live still averages three days. (Danchev, 2007). Phishing is an example in motion of economies of scale. The more Phishing attempts sent the better chance at part of that $3 Billion dollars.

Not all phishing attacks are blended with some form of malware, and these attempts will make it right past the Antivirus scanner. Multiple layers of defence again make sense and are the best practice. The three day life span of most Phishing sites makes traditional AntiVirus definition file style indexes somewhat unwieldy, full of links that are no longer harmful or even exist. The second layer of Phish protection, OpenDNS, is uniquely situated to protect from these types of attacks.

Taking another link from a phishing attempt, this attack aimed at PayPal, I created webpage on a host machine behind the firewall with the following code.

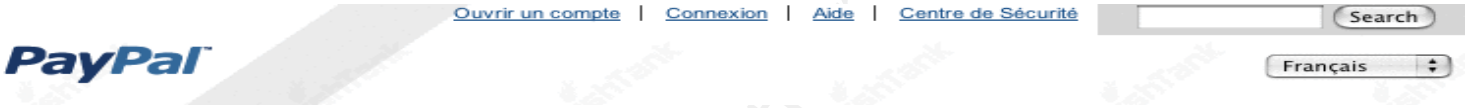
```
<a href="http://hgklapla.ns8-wistee.fr/www.PayPal.Com22/webscrcmd=_login-done&login_access=1190737782.htm">http://paypal.com/secure/OSL.aspx?refer=7923860156l</a>
```

This is a simple hyper link that when viewed in a webbrowser displays the link as,

http://paypal.com/secure/OSL.aspx?refer=79238601561

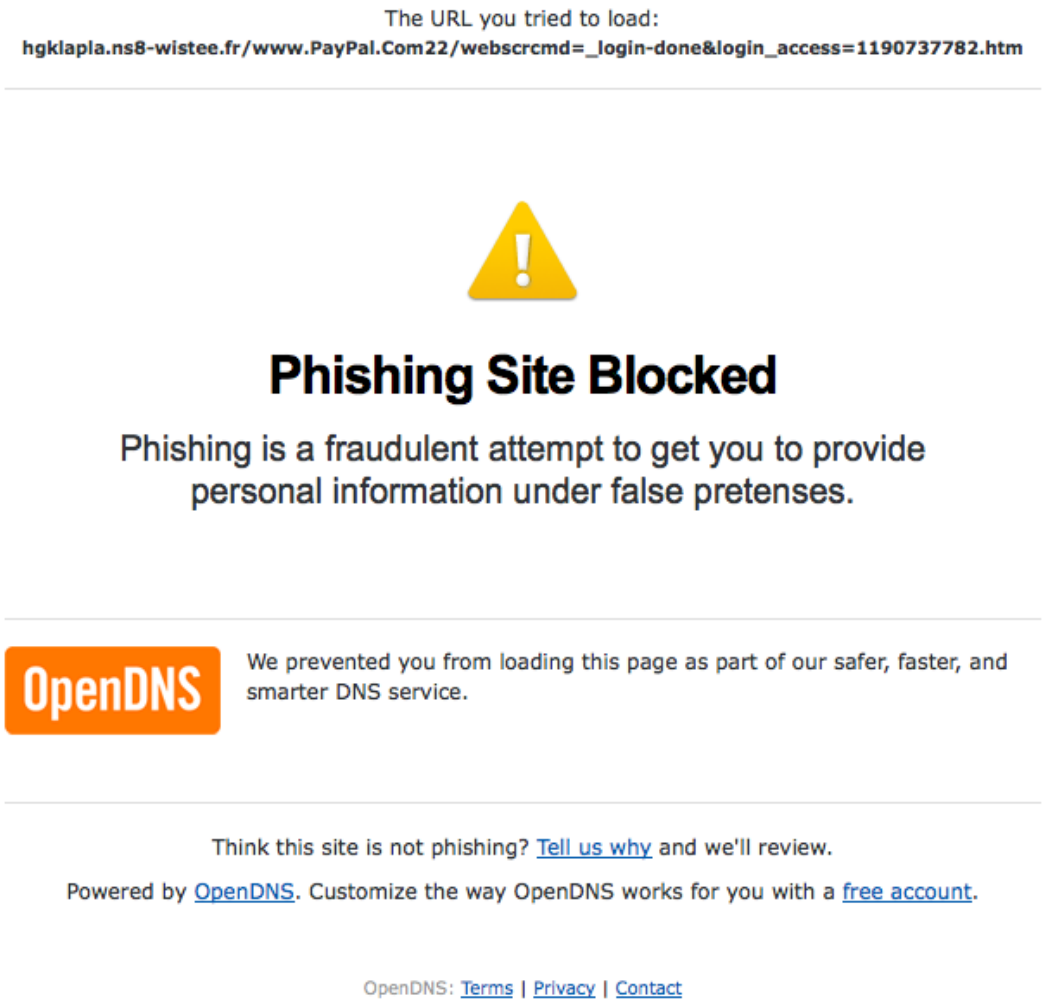
But the link is pointed at a page at http://hgklapla.ns8-wistee.fr, which is not PayPal. This phish was a confirmed phish taken from phishtank, submission #477466 and has since been taken down.

However, when it was live, it was very convincing. Following the link would display



This was followed by a login area for the “customer” to update their information.

However, using the DNSmasq on the firewall to recurse to OpenDNS, and setting the network to block phishing on OpenDNS, that URL cannot be resolved, and is instead pointed at OpenDNS's phishing information page.



Case E) Undetected Malware broken by Egress Rules

After gaining a sample of what the researchers at malwaredb (http://malwaredatabase.net) had described as Zolabc (inside the executable file Gigaticket2018.exe), I passed it through the firewall over http and pop3. In both occurrences an updated ClamAV database, current as of 18 July 2008 (date of testing), did not have a definition for the file yet, and the file was passed.

```
Fri Jul 18 13:18:15 2008 [22497] DEBUG Request:http://unit2.ca/mal/gigaticket2018.exe 172.20.0.201/- - GET
Fri Jul 18 13:18:15 2008 [22497] DEBUG regex matched: http://unit2.ca/mal/gigaticket2018.exe
Fri Jul 18 13:18:15 2008 [22497] DEBUG Getting header for url http://unit2.ca/mal/gigaticket2018.exe
Fri Jul 18 13:18:15 2008 [22497] DEBUG File size is '.2f
Fri Jul 18 13:18:15 2008 [22497] DEBUG Sending STREAM to clamd.
Fri Jul 18 13:18:15 2008 [22497] DEBUG Received port 1533 from clamd.
Fri Jul 18 13:18:15 2008 [22497] DEBUG Trying to connect to clamd [port: 1533].
Fri Jul 18 13:18:15 2008 [22497] DEBUG Scanning data received against clamd stream
Fri Jul 18 13:18:15 2008 [22497] DEBUG Sending data to clamd
Fri Jul 18 13:18:15 2008 [22497] DEBUG Write 1057 bytes on 1057 to socket
Fri Jul 18 13:18:15 2008 [22497] DEBUG Scanning data received against clamd stream
Fri Jul 18 13:18:19 2008 [22497] DEBUG Connection to clamd on port: 1533 closed.
Fri Jul 18 13:18:19 2008 [22497] DEBUG Reading clamd scan result.
Fri Jul 18 13:18:20 2008 [22497] DEBUG received from Clamd: stream: OK
Fri Jul 18 13:18:20 2008 [22497] DEBUG End reading clamd scan result.
Fri Jul 18 13:18:20 2008 [22497] STAT Virus Scanning process time  5.048 second(s)
Fri Jul 18 13:18:20 2008 [22497] DEBUG No virus detected for URL: http://unit2.ca/mal/gigaticket2018.exe.
Fri Jul 18 13:18:20 2008 [22497] STAT Total process time  5.292 second(s)
```

This points out the need for multiple layers of security. ClamAV would later identify this file as Trojan.SdBot-9400, but at one point while testing Microsoft, AVG, e-Trust, A-Squared, F-Prot, Normand, and VirusBuster found nothing. CA WebScan could only identify it as VMalum.DJPF, which is a catch all definition for code that displays malware like properties, but no signature was available for at the time. Please see Appendix E for more results.

If the virus scanner on the host machine that had downloaded that file had also found nothing, then there is a chance that the malware would have been run by a user. At time of writing there still was not a solid write up on what the Malware would attempt to do.

I was also not able to get the malware to display its behaviour to me through SysInternals' Process Explorer and tcpdump when executed on a virtual machine, as most malware has been smart enough to detect the virtual environment to help hide itself from prying eyes.

Loaded on a sacrificial laptop running WinXP Pro SP2, the malware ran. It created a single process that was not obscured from the Task Manager and did not seem to do anything else beyond try to connect out. Perhaps it needed to connect to Command and Control to be told where to download further payload and receive orders. Here is the tcpdump output of its activity.

```
12:40:51.023021 172.20.0.201.1025 > 172.20.0.1.53: 45193+ A? mm.esskil99.info. (34)
        4500 003e 03bf 0000 8011 ddfd ac14 00c9
        ac14 0001 0401 0035 002a 241b b089 0100
        0001 0000 0000 0000 026d 6d08 6573 736b
        696c 3939 0469 6e66 6f00 0001 0001
12:40:51.183407 172.20.0.1.53 > 172.20.0.201.1025: 45193 1/0/0 A 72.8.143.164 (50)
        4500 004e de6d 0000 4011 433f ac14 0001
        ac14 00c9 0035 0401 003a fda9 b089 8180
        0001 0001 0000 0000 026d 6d08 6573 736b
        696c 3939 0469 6e66 6f00 0001 0001 c00c
        0001 0001 0000 0e10 0004 4808 8fa4
```

The first thing it attempted to do was a domain lookup against mm.esskil99.info, which is not a phishing URL, but a Command and Control channel. At this time, OpenDNS does not maintain a database of possible command and control channels.

12:40:51.186248 172.20.0.201.1061 > 72.8.143.164.17766: S 4104519829:4104519829(0) win 65535 <mss 1460,nop,nop,sackOK> (DF)
4500 0030 03c0 4000 8006 727e ac14 00c9
4808 8fa4 0425 4566 f4a6 0095 0000 0000
7002 ffff bfce 0000 0204 05b4 0101 0402
12:40:54.131015 172.20.0.201.1061 > 72.8.143.164.17766: S 4104519829:4104519829(0) win 65535 <mss 1460,nop,nop,sackOK> (DF)
4500 0030 03c1 4000 8006 727d ac14 00c9
4808 8fa4 0425 4566 f4a6 0095 0000 0000
7002 ffff bfce 0000 0204 05b4 0101 0402
12:41:00.140286 172.20.0.201.1061 > 72.8.143.164.17766: S 4104519829:4104519829(0) win 65535 <mss 1460,nop,nop,sackOK> (DF)
4500 0030 03c2 4000 8006 727c ac14 00c9
4808 8fa4 0425 4566 f4a6 0095 0000 0000
7002 ffff bfce 0000 0204 05b4 0101 0402

Next, Kolabc tries to initiate a 3 way hand shake against that server at some high port. However, the egress rules on the firewall drop that packet on sight. Kolabc waits around fifteen seconds and repeats the process again. After a certain amount of time, Kolabc tries to lookup another IP address, this time for jerusalem.cjb.net, but OpenDNS either does not have a record for it or has blocked it due to phishing or malware, and returns with it's own IP. This is the normal behaviour for OpenDNS, as the next request for a web client would be that IP address with a url, to which the OpenDNS landing page would provide information on why it was blocked. Kolabc however, does not try to web browse, but tries something else.

12:59:02.214701 172.20.0.201.1025 > 172.20.0.1.53: 57986+ A? jerusalem.cjb.net. (35)
12:59:02.391063 172.20.0.1.53 > 172.20.0.201.1025: 57986 1/0/0 A 208.67.216.132 (51)
12:59:02.520484 172.20.0.201.1108 > 208.67.216.132.6667: S 919331184:919331184(0) win 65535 <mss 1460,nop,nop,sackOK> (DF)
12:59:05.513882 172.20.0.201.1108 > 208.67.216.132.6667: S 919331184:919331184(0) win 65535 <mss 1460,nop,nop,sackOK> (DF)
12:59:11.523143 172.20.0.201.1108 > 208.67.216.132.6667: S 919331184:919331184(0) win 65535 <mss 1460,nop,nop,sackOK> (DF)

Port 6667 is traditionally an IRC connection. IRC is not approved by the administrator of this firewall, so the SYN packet is dropped. Beyond that, it was not heading to the command and control channel the malware thought it was going to in the first place.

If the network administrator was to allow IRC, it would be enough to create a rule on the inside interface allowing TCP 6667 out to the IRC server of choice, and no other, much like the current rule for SMTP. In that situation, the packets would be dropped again.

Not all Malware uses high ports to connect to Command and Control, some malware has begun to use TCP 80 outbound as it is normally allowed a direct connection out if there is no webproxy. The Win32/Pushdo family of malware uses it quite extensively (Stewart, 2007). In that situation, the malware is not aware of the web proxy, so it breaks the attack. If we were to run the proxy in transparent mode, the malware is not speaking in standard HTTP, so the proxy would still break the attack.

So, if a malware of this style was to get inside the network to some degree, the outbound rules would neuter it. It is not able to download further malware, nor is it able to get commands from the bot-herders. This is not a perfect system, but it does illustrate that network security should not focus solely on blocking unsolicited traffic in but also on the types of traffic trying to leave. The fact that the malware was partially neutered also gives the AntiVirus definitions time to catch up.

Future Work and Conclusion.

With well spent time, it is possible to build a very capable NAT Firewall that also offers some of the features that are offered by Unified Threat Management devices. Not only does it have more than one means to protect from phishing attacks, it also adds a second line of defence against viruses, and could possibly stop malware from functioning properly if it were to get on the network.

I feel slightly better about the security posture of Mom and Dad's home network. It is possible to mitigate most attacks that they would face on a day to day. Host based Antivirus, upto date patching, host based firewalls and a little education are still completely required. But it is best to be over prepared than found lacking.

There are several features that would make a solid addition to this setup. Snort IDS could be installed on the box and set up with a few of the bleeding-edge rules which looks for personal identifying information (credit card numbers, social insurance numbers, ect). The rules, found at http://www.bleedingthreats.net/rules/bleeding-policy.rules (2001328, and 2001375 through 2001384) could be set up to alert or drop such detections, to make sure that some new breed of malware or some unlucky phished user does not leak credit card numbers out to the Internet.

The OpenBSD ports contain a copy of squidguard, which is a URL redirector that works off of blacklists. With squidguard installed, it is possible to block websites by categories to provide further

protection. However, OpenDNS has done a wonderful job for now, and does not rely on another URL redirector plugged into squid.

OpenVPN could be built in to create a network of devices from one family residence to another. While file-sharing is one idea, a larger rsync backup scheme could be implemented to provide off site backups for all members of the network, should the worst case scenario happen. With a family VPN, a central management system could be built to which logs could be centrally stored, trended, and alerted on. A spike of denied packets at Mom's place could indicate that some malware has gotten through and a cleaning might be in order, or that a port needs to be opened up for her to run video chat or some other application with an old friend.

It is a platform that can grow, and even if she can't understand it, Mom would appreciate it. I'm sure she would cook a meal on the day of deployment.

References

Higgins, Kelly J (2007, Febuary 15). New 'Drive-By' Attack Is Remote. Retrieved July 30, 2008, from Dark Reading Web site: http://www.darkreading.com/document.asp?doc_id=117497

Hacquebord, Feike (2008, August 7). ZLOB Enters The Search Engine Market. Retrieved August 26, 2008, from Trendlabs Malware Blog Web site: http://blog.trendmicro.com/zlob-enters-the-search-engine-market/

Nussbaum, Lucas (2006, June 1). IP over DNS. Retrieved August 18, 2008, from Lucas Nussbaum's Blog Web site: http://www.lucas-nussbaum.net/blog/?p=168

Gil, Thomer M. (2005, December 20). ICMPTX (IP-over-ICMP) HOWTO. Thomer M. Gil, Retrieved August 18, 2008, from http://thomer.com/icmptx/

Xue, Feng (2008, Febuary 25). Attacking the Antivirus. Retrieved August 26, 2008, from Blackhat Web site: http://www.blackhat.com/presentations/bh-europe-08/Feng-Xue/Whitepaper/bh-eu-08-xue-WP.pdf

Hatch, Brian (2003, Febuary 13). Egress filtering for a healthier Internet. Retrieved September 2, 2008, from Hacking Linux Exposed Web site: http://www.hackinglinuxexposed.com/articles/20030213.html

Danchev, Dancho (2008, July 25). How OpenDNS, PowerDNS and MaraDNS remained unaffected by the DNS cache poisoning vulnerability. Retrieved July 30, 2008, from Zero Day Blog Web site: http://blogs.zdnet.com/security/?p=1562

Kaminsky, Dan (2008, July 24). Details. Retrieved July 30, 2008, from DoxPara Research Web site: http://www.doxpara.com/?p=1185

Cross, Tom (2008, July 10). DNS Cache Poisoning and Network Address Translation. Retrieved September 3, 2008, from Frequency X Blog Web site: http://blogs.iss.net/archive/dnsnat.html

Hart, Jon (2008, July 24). Mitigating DNS cache poisoning with pf. Retrieved July 30, 2008, from Jon hart's Blog Web site: http://blog.spoofed.org/2008/07/mitigating-dns-cache-poisoning-with-pf.html

McCall, Tom (2007. December 17). Gartner Survey Shows Phishing Attacks Escalated in 2007; More than $3 Billion Lost to These Attacks . Retrieved September 29, 2008, from Gartner.com Web site: http://www.gartner.com/it/page.jsp?id=565125

Danchev, Dancho (2007, December 12). Phishing Metamorphosis in 2007 - Trend and Developments. Retrieved September 29, 2008, from WindowSecurity.com Web site: http://www.windowsecurity.com/articles/Phishing-Metamorphosis-2007-Trend-Developments.html

Stewart, Joe (2007, December 17). Pushdo - Analysis of a Modern Malware Distribution System. Retrieved September 29, 2008, from SecureWorks.com Web site: http://www.secureworks.com/research/threats/pushdo/?threat=pushdo

Appendix A – pf.conf

```
#     $OpenBSD: pf.conf,v 1.35 2008/02/29 17:04:55 reyk Exp $
#
# See pf.conf(5) and /usr/share/pf for syntax and examples.
# Remember to set net.inet.ip.forwarding=1 and/or net.inet6.ip6.forwarding=1
# in /etc/sysctl.conf if packets are to be forwarded between interfaces.

####################################
# Interfaces and other macros
####################################

# Interfaces
ext_if="fxp0"
int_if="xl0"
loop_if="lo0"

#hosts
isp_smtp=”24.71.223.43”

# Networks
lan1="172.20.0.0/24"
not_private="{ !192.168.0.0/16, !172.16.0.0/12, !127.0.0.0/8, !10.0.0.0/8,
!0.0.0.0/8, !255.255.255.255/32 }"

# Ports
fw_tcp="{ 22, 80, 3128 }"
fw_udp="{ 53, 123 }"
internet_tcp="{ 22, 10443 }"

# Drop grinders
grind="(max-src-conn 3, max-src-conn-rate 2/5, overload <abusers> flush global)"

# tcp flags
tcp_flags="flags S/SA keep state"

# Firewall options
table <abusers> persist
scrub in all

####################################
# NATs and Redirects
####################################

# FTP proxy setup
nat-anchor "ftp-proxy/*"
rdr-anchor "ftp-proxy/*"

# RDR
rdr on $int_if proto tcp from $lan1 to $not_private port 110 -> $loop_if port 8110
rdr on $ext_if proto tcp from $not_private to $ext_if port 40443 -> $ext_if port 22
```

```
# NAT
nat on $ext_if from $lan1 to any -> $ext_if

# FTP Proxy anchor
anchor "ftp-proxy/*"

# Antispoof
antispoof quick for { $loop_if, $int_if }

####################################
# Internal Interface Rules
####################################

# Block in and out everything
block log quick from <abusers>
block in log on $ext_if all
block out log on $ext_if all

# Pass out our traffic. Pass out from the box allows us to let the firewall connect
however it needs to. Handy for things like sshing over non-standard ports. Admin
can ssh into this box from $lan1 and then leap frog from there. All filtering out
to the internet from $lan1 is done on ingress on $int_if
pass out on $ext_if inet proto { udp, icmp } all keep state
pass out on $ext_if inet proto tcp $tcp_flags

####################################
# Internal Interface Rules
####################################

# Block in and out everything
block in log on $int_if all
block out log on $int_if all

# Pass in traffic destined for the internet
pass in on $int_if inet proto tcp from $lan1 to $not_private port $internet_tcp
$tcp_flags
#pass in on $int_if inet proto udp from $lan1 to $not_private port $internet_udp
keep state

# Pass in pings to opendns. Helpful for a quick conectivity check from behind the
firewall
pass in on $int_if inet proto icmp from $lan1 to 208.67.220.220 icmp-type echoreq
keep state

# Pass in traffic destined for the firewall itself only on services offered
pass in quick on $int_if inet proto tcp from $lan1 to $int_if port $fw_tcp
$tcp_flags
pass in quick on $int_if inet proto udp from $lan1 to $int_if port $fw_udp keep
state
pass in quick on $int_if inet proto udp from any to 255.255.255.255 port 67 keep
state
pass in quick on $int_if inet proto icmp from $lan1 to $int_if icmp-type echoreq
keep state
```

```
pass in quick log on $int_if inet proto tcp from $lan1 to lo0 port 8110 $tcp_flags
pass in quick on $int_if inet proto tcp from $lan1 to $isp_smtp port 25 $tcp_flags
```

Appendix B – rc.local start up script

```
#       $OpenBSD: rc.local,v 1.39 2006/07/28 20:19:46 sturm Exp $

# Site-specific startup actions, daemons, and other things which
# can be done AFTER your system goes into securemode.  For actions
# which should be done BEFORE your system has gone into securemode
# please see /etc/rc.securelevel.

echo -n 'starting local daemons:'

# Add your local startup actions here.

echo '.'

if [ -x /usr/local/bin/dnsmasq ]; then
    echo -n ' DNSmasq'; /usr/local/bin/dnsmasq &
fi

if [ -x /usr/local/bin/clamd]; then
    echo -n ' ClamAV'; /usr/local/bin/clamd
fi

if [ -x /usr/local/bin/p3scan]; then
    echo -n ' P3Scan'; /usr/local/bin/p3scan
fi
```

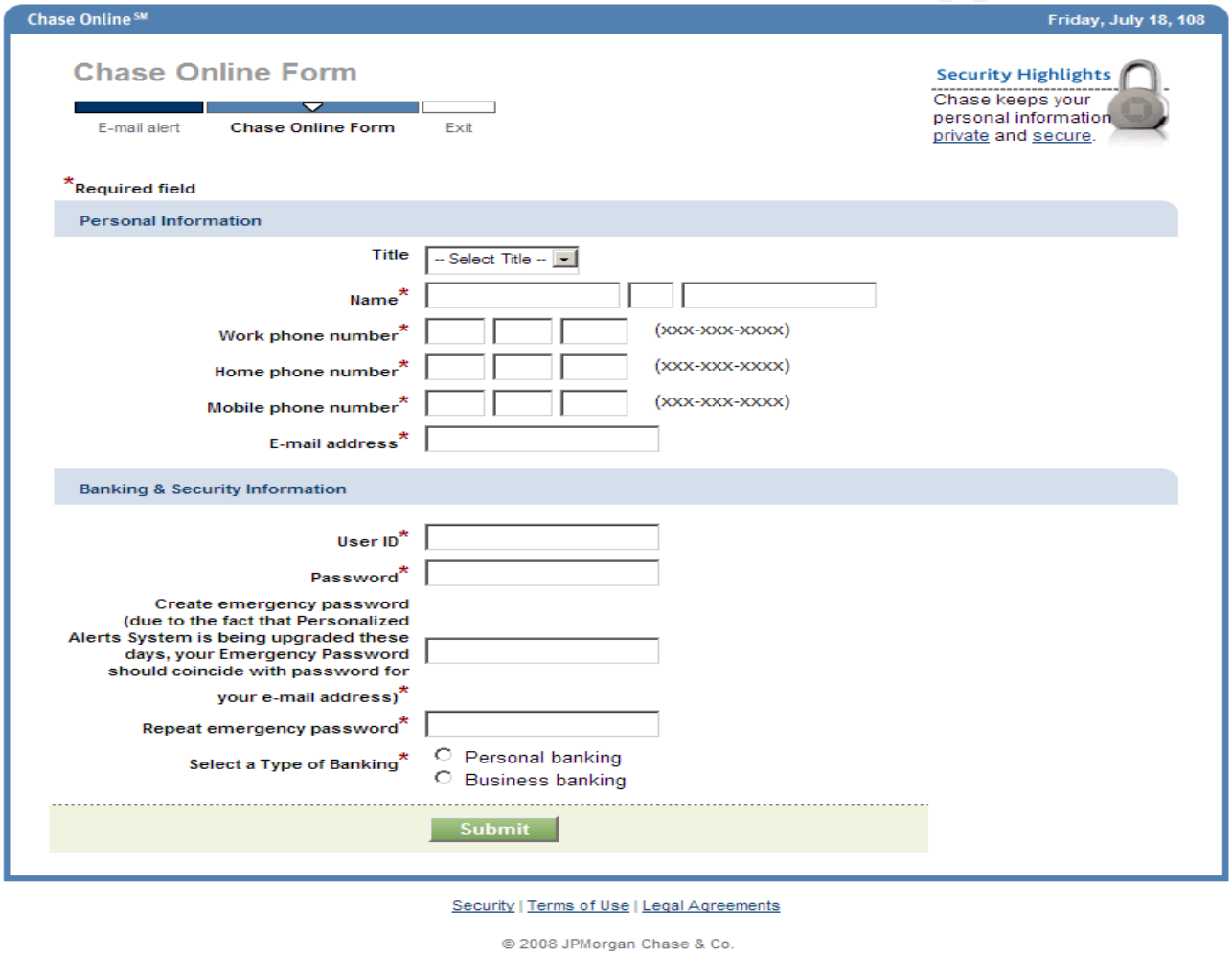
Appendix C – squid.conf

```
acl all     src   0.0.0.0/0.0.0.0
acl localhost     src   127.0.0.1/255.255.255.255
acl lan     src   172.20.0.0/24
acl manager       proto       cache_object
acl to_localhost  dst   127.0.0.1/255.255.255.255
acl SSL_ports     port  443
acl Safe_ports    port  80 21 443
acl CONNECT method      CONNECT
http_access deny to_localhost
http_access allow localhost
url_rewrite_access deny localhost
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow lan
http_access deny all
icp_access allow all
http_port 3128
hierarchy_stoplist cgi-bin ?
cache_dir ufs /var/squid/cache 100 16 256
logformat squid  %ts.%03tu %6tr %>a %Ss/%03Hs %<st %rm %ru %un %Sh/%<A %mt
access_log /var/squid/logs/access.log squid
cache_log /var/squid/logs/cache.log
cache_store_log /var/squid/logs/store.log
ftp_user wintr@unit2.ca
url_rewrite_program /usr/local/bin/squidclamav
url_rewrite_children 30
acl QUERY urlpath_regex cgi-bin \?
cache deny QUERY
refresh_pattern ^ftp:         1440  20%   10080
refresh_pattern ^gopher:      1440  0%    1440
refresh_pattern .       0     20%   4320
acl apache rep_header Server ^Apache
broken_vary_encoding allow apache
cache_mgr wintr@unit2.ca
cache_effective_user _squid
cache_effective_group _squid
dns_nameservers 208.67.220.220 208.67.222.222

coredump_dir /var/squid/cache
```

Appendix D – The Chase Phishing Attempt

The webpage that is linked in the e-mail:



The E-mail itself:

```
From smrfs@chaseonline.chase.com  Fri Jul 18 13:33:24 2008
Return-Path: <smrfs@chaseonline.chase.com>
X-Original-To: wil@augmentedreality.ca
Delivered-To: wil@augmentedreality.ca
Received: from cm-85-152-220-194.telecable.es (cm-85-152-220-194.telecable.es
[85.152.220.194])
```

```
        by white.augmentedreality.ca (Postfix) with SMTP id DABC53D8005
        for <wil@augmentedreality.ca>; Fri, 18 Jul 2008 13:33:23 -0600 (MDT)
Received: from doctrinaire.astrons.com (damhost.com.pornomir.com [28.0.44.38])
     by priest.com with SMTP id LF3I4B1LF7
     for <wil@augmentedreality.ca>; Fri, 18 Jul 2008 15:33:27 -0500
Received: from altern.org (unknown [61.66.40.234])
     by dreamhost.com with SMTP id X79407CC1C
     for <wil@augmentedreality.ca>; Fri, 18 Jul 2008 18:25:27 -0200
From: "Chase" <smrfs@chaseonline.chase.com>
To: "Wil" <wil@augmentedreality.ca>
Subject: Please Update Your Details
X-Authenticated: #85288774
User-Agent: Calypso Version 3.30.00.00
X-Mailer: Calypso Version 3.30.00.00
X-Priority: 3 (Normal)
MIME-Version: 1.0
Content-Type: multipart/alternative;
        boundary="--2Qwzsj34hjj93yuxx41df9GN"
Message-Id: <20080718193323.DABC53D8005@white.augmentedreality.ca>
Date: Fri, 18 Jul 2008 13:33:23 -0600 (MDT)
Status: O
X-UID: 397591
Content-Length: 4137
X-Keywords:

----2Qwzsj34hjj93yuxx41df9GN

Content-Type: text/html;
Content-Transfer-Encoding: 7Bit

<head>
<meta     http-equiv="Content-Type"     content="text/html;  charset=iso-8859-1">
</head>
<body>
<p><font      face="Courier New, Courier, mono">Note:       This   is       a
service   message      regarding       the     Chase      Online
Form.</font></p>
<p><font     face="Courier New, Courier, mono">Dear       customer:</font></p>
<p><font    face="Courier New, Courier, mono">As     part   of   the    new
security   measures,       all   Chase      bank      customers are   required   to
complete Chase    Online      Form. Please   complete    the       form     as soon
as     possible.</font></p>
<p><font    face="Courier New, Courier, mono">To  access the    form       please
click    on  the     following     link:</font></p>
<p><font    face="Courier New, Courier, mono"><a
href="http://chaseonline.chase.com.err.com.es/Secure/webform/OSL.aspx?LOB=
93263883029001427299543640047084629349812346321757 49&amp;refer=79238601561">http://
chaseonline.chase.com/Secure/webform/OSL.aspx?LOB=
93263883029001427299543640047084629349812346321757 49&amp;refer=79238601561</a></fon
t></p>
<p><font    face="Courier New, Courier, mono">Thank  you      for being   a
valued   customer.</font></p>
<p> </p>
<p><font    face="Courier New, Courier, mono">Sincerely,</font></p>
```

```
<p><font       face="Courier New, Courier, mono">Online  Banking
Team</font></p>
<p><font   color="#FFFFF4"  face="Courier New, Courier, mono">L1RX: 0x36586918,
0x238, 0x4373, 0x5, 0x0870, 0x2149, 0x1392, 0x0, 0x1, 0x00, 0x7794, 0x35060371,
0x81       rcs, 6H3, 6TPU, common, F5Y.    0x36, 0x08, 0x2650, 0x36, 0x7, 0x390,
0x6, 0x51759720, 0x5, 0x3, 0x588, 0x6667, 0x87, 0x76679496   5224    HR5F: 0x72,
0x807 ZKQ: 0x12, 0x13756426, 0x1, 0x80674208, 0x0830, 0x28, 0x3, 0x0184, 0x2768,
0x06, 0x8402      end: 0x4913, 0x890, 0x2351, 0x50924674     0x77, 0x63924809,
0x3262, 0x575, 0x287 S79: 0x45464113, 0x4, 0x46604878, 0x8, 0x791, 0x03494786,
0x9564, 0x4956      hex: 0x36, 0x1463</font></p>
<p><font  color="#FFFFF1" face="Courier New, Courier, mono">0x8 N98: 0x4, 0x69,
0x4074, 0x57, 0x00, 0x44, 0x825, 0x0382, 0x2143, 0x90126048, 0x64, 0x6577 R42Z:
0x15, 0x36237263, 0x647, 0x79, 0x0213, 0x498, 0x942, 0x2288, 0x856, 0x602, 0x185,
0x16604934, 0x207, 0x665       LEC8, type, EVNL, 79VK, K1GU, function.  0x23,
0x9408, 0x998, 0x9, 0x8187, 0x785, 0x3308, 0x08389288, 0x504, 0x41533904,
0x37698417  92794934298415947353559     85S: 0x63427562, 0x01400903, 0x030, 0x977,
0x31466291, 0x681, 0x94096519, 0x5876, 0x397, 0x5471, 0x700, 0x
6032, 0x2822 include, FA2X. hex: 0x4938, 0x38, 0x3592, 0x71115789     0x5, 0x713,
0x4, 0x11, 0x1, 0x09, 0x00, 0x0357, 0x30, 0x475, 0x376, 0x3, 0x87, 0x5, 0x76642792
QFOL: 0x03, 0x8, 0x15089148, 0x00717911, 0x94867224, 0x05268588, 0x6, 0x16,
0x20785804, 0x7, 0x28889233</font></p>
<p><font  color="#FFFFFA"  face="Courier New, Courier, mono">create: 0x86, 0x6
BSF: 0x3, 0x80, 0x33581351, 0x2545, 0x2665, 0x61, 0x74588974, 0x2197, 0x7,
0x91863398, 0x7683, 0x3542, 0x3198, 0x3654   VJ7R: 0x88, 0x48790600, 0x84354712,
0x138, 0x6, 0x758     0x7915, 0x56784301, 0x17410703     rcs.    common: 0x93,
0x893, 0x8595, 0x5, 0x9344, 0x8, 0x33, 0x8826
, 0x2067, 0x0572, 0x2472, 0x17, 0x44502540      0x268, 0x9752, 0x62, 0x81743727,
0x0463, 0x426, 0x183, 0x42, 0x7, 0x23115033, 0x7786, 0x9    api: 0x02, 0x9375
8LC, 700, 1AP, hex, dec.   0x5315, 0x9, 0x154, 0x21431968, 0x1, 0x20, 0x2804, 0x34,
0x3, 0x66, 0x238, 0x2</font>    <font  color="#FFFFF0" face="Courier New, Courier,
mono">0198296465876180191566824917296899</font></p>
<p><font  color="#FFFFF9"    face="Courier New, Courier,
mono">21162504830523329646</font>      <font   color="#FFFFF0"     face="Courier
New, Courier, mono">0x307, 0x2, 0x6, 0x40, 0x7, 0x841, 0x4, 0x1267, 0x72, 0x52,
0x9, 0x83   0x03026272, 0x5715, 0x863, 0x277, 0x426, 0x4, 0x62701033, 0x3, 0x0056,
0x8    N4S: 0x33, 0x85979221, 0x88, 0x3, 0x4, 0x522, 0x322, 0x6994, 0x72, 0x2276,
0x3, 0x24748022</font></p>
</body>
</html>
```

Appendix E – Kolabc information

From http://www.cyber-ta.org

```
Antivirus Detection Summary: file 16d7c031c57e093eb2576ed780e90ffc
       1: AhnLab-V3                found nothing
       2: AntiVir                  found [TR/Crypt.XPACK.Gen]
       3: Authentium               found nothing
       4: Avast                    found nothing
       5: AVG                      found nothing
       6: BitDefender              found nothing
       7: CAT-QuickHeal            found [(Suspicious) - DNAScan]
       8: ClamAV                   found nothing
       9: DrWeb                    found nothing
      10: eSafe                    found [Suspicious File]
      11: eTrust-Vet               found nothing
      12: Ewido                    found nothing
      13: F-Prot                   found nothing
      14: F-Secure                 found [Suspicious:W32/Malware!Gemini]
      15: Fortinet                 found [W32/Dorf.CMJ!worm.im]
      16: GData                    found [Net-Worm.Win32.Kolabc.cmj]
      17: Ikarus                   found [Trojan.Crypt.XPACK]
      18: Kaspersky                found [Net-Worm.Win32.Kolabc.cmj]
      19: McAfee                   found nothing
      20: Microsoft                found nothing
      21: NOD32v2                  found [Win32/Kolab.CHD]
      22: Norman                   found nothing
      23: Panda                    found [Suspicious file]
      24: Prevx1                   found [Malicious Software]
      25: Rising                   found nothing
      26: Sophos                   found [Mal/TibsPak]
      27: Sunbelt                  found nothing
      28: Symantec                 found nothing
      29: TheHacker                found nothing
      30: TrendMicro               found [PAK_Generic.001]
      31: VBA32                    found nothing
      32: VirusBuster              found [Packed/PCMM]
      33: Webwasher-Gateway        found [Trojan.Crypt.XPACK.Gen]
CREDITS:  Antivirus malware test results are from submissions to
www.virustotal.com.
http://www.cyber-ta.org/releases/malware-
analysis/public/SOURCES/16d7c031c57e093eb2576ed780e90ffc/16d7c031c57e093eb2576ed780
e90ffc.virus-labels
```