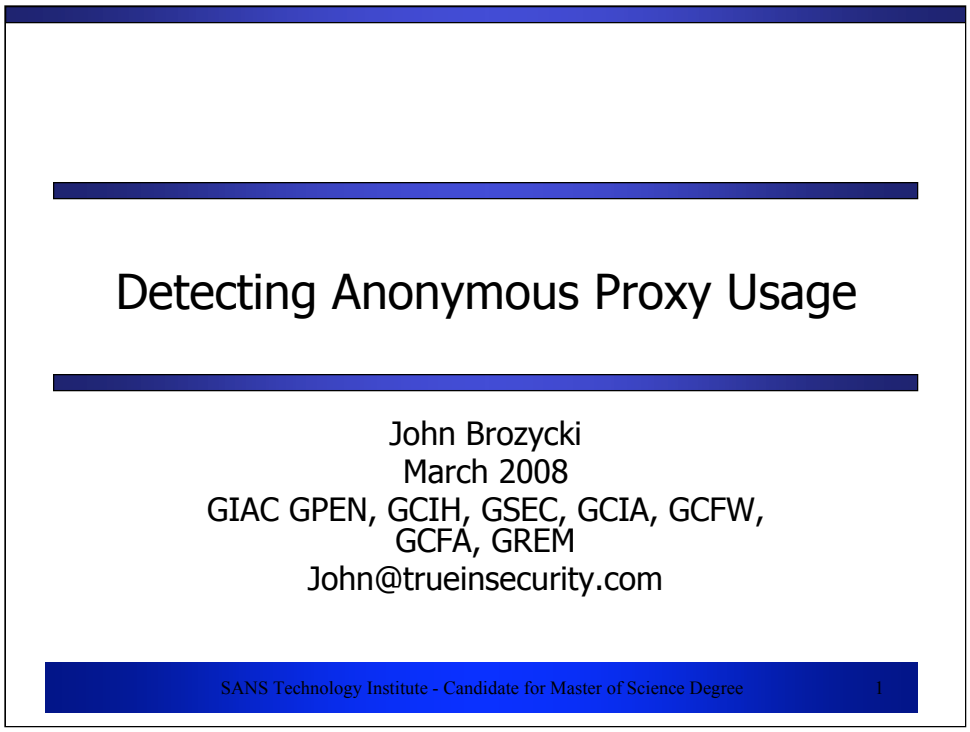
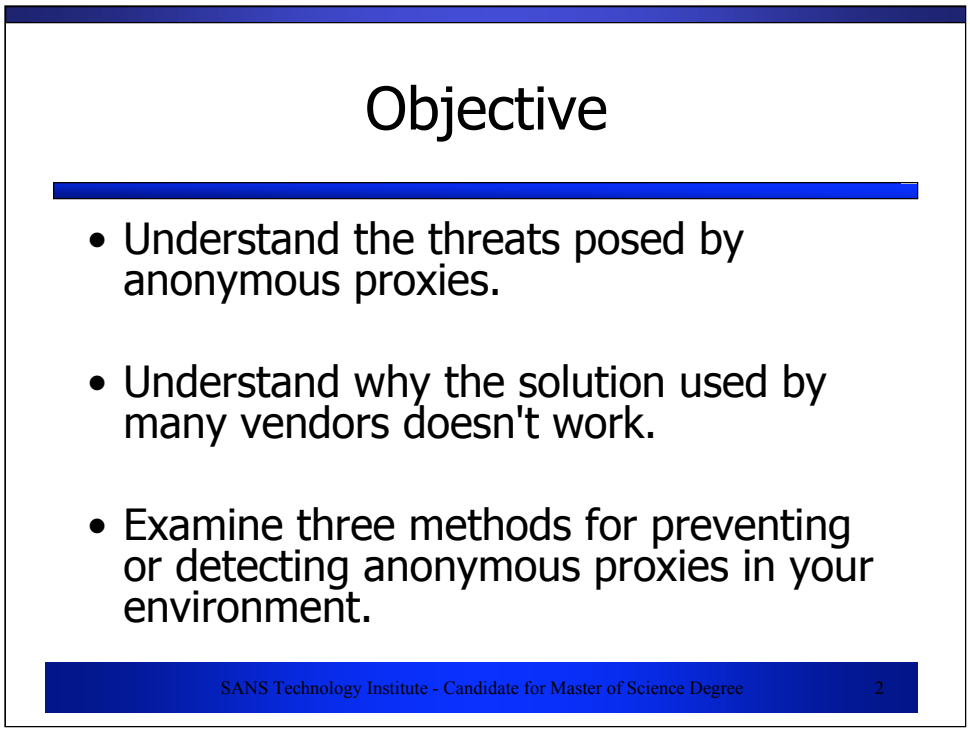


John Brozycki. March 2009. SANS 2009



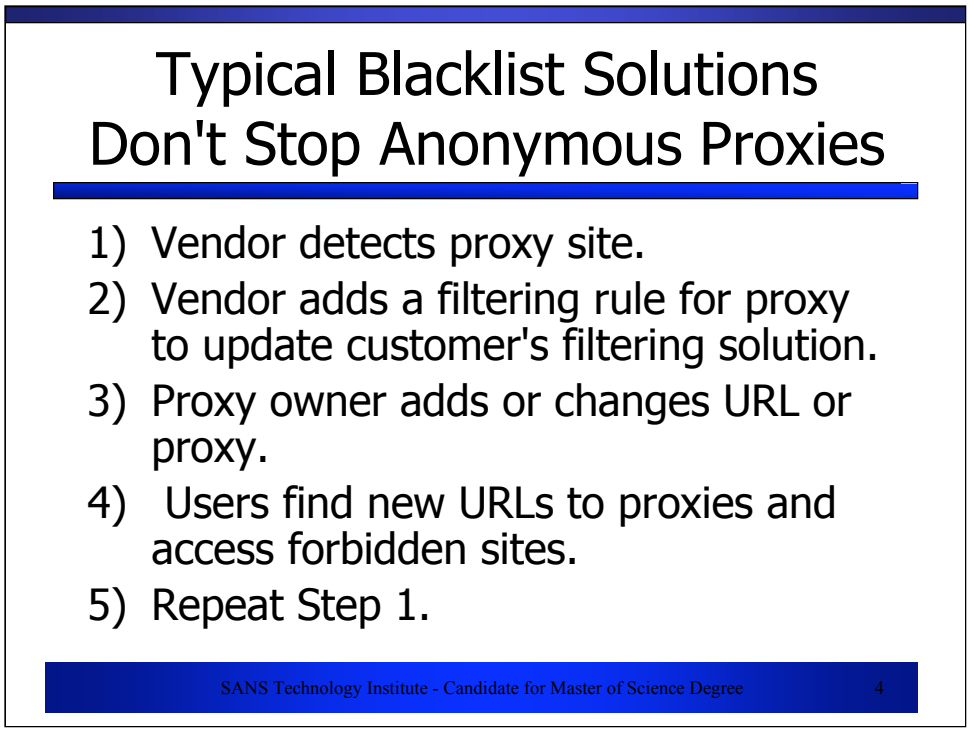
If you manage a user population that accesses the Internet, you've probably faced the problem of restricting the content that users are permitted to view. Some sites violate acceptable use policy while others may be sites known to host malware. You may have implemented filtering solutions, only to find that your users get to the restricted sites anyway thanks to anonymous proxy servers. What are anonymous proxies, why should you be concerned about them, and how can you prevent or at least detect them?

Anonymous Proxies

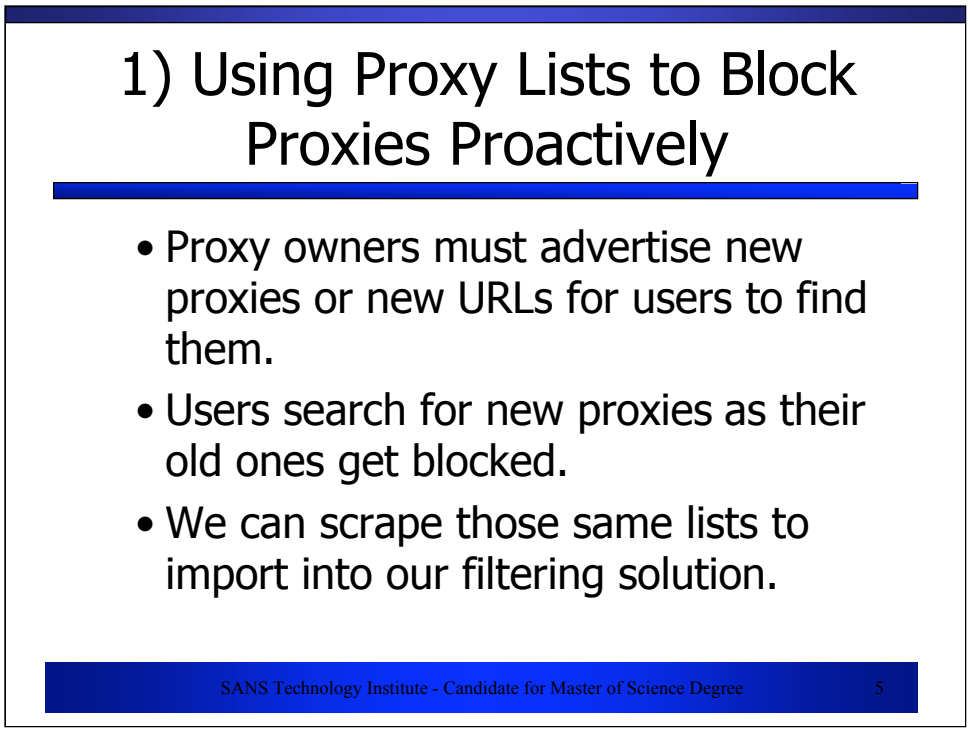
- Allow access to restricted sites, which:
  - violates security policy
  - puts systems at risk to malware
- Change frequently. Sites added daily.
- How can we detect them?

SANS Technology Institute - Candidate for Master of Science Degree 3

Anonymous Proxies are servers on the Internet that allow users to access sites that are forbidden by a company security policy and blocked by company site filtering solutions. You may block myspace.com, but a user can access "someproxy.com" and enter "myspace.com" in a user entry field, and the proxy server will retrieve the content from myspace.com. Since the user is accessing someproxy.com, not myspace.com, your filtering solution is unaware and effectively bypassed. URLs for anonymous proxies change frequently and new ones are added every day. While there is no single solution that will prevent all anonymous proxy usage, this presentation will discuss some techniques to detect and prevent their usage.



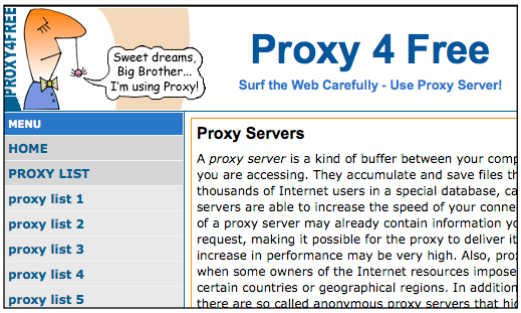
Since proxy URLs may be changed frequently and new proxies are added every day, blacklisting discovered anonymous proxies becomes a game of "catch up" where you are always behind; able to block the proxies that users used today, but not the proxies they'll be using tomorrow. These steps illustrate the process many vendors use to push updates to customers and why this process isn't enough. Much like the game of "Whack a Mole" each time you knock down one mole another one pops up somewhere else.



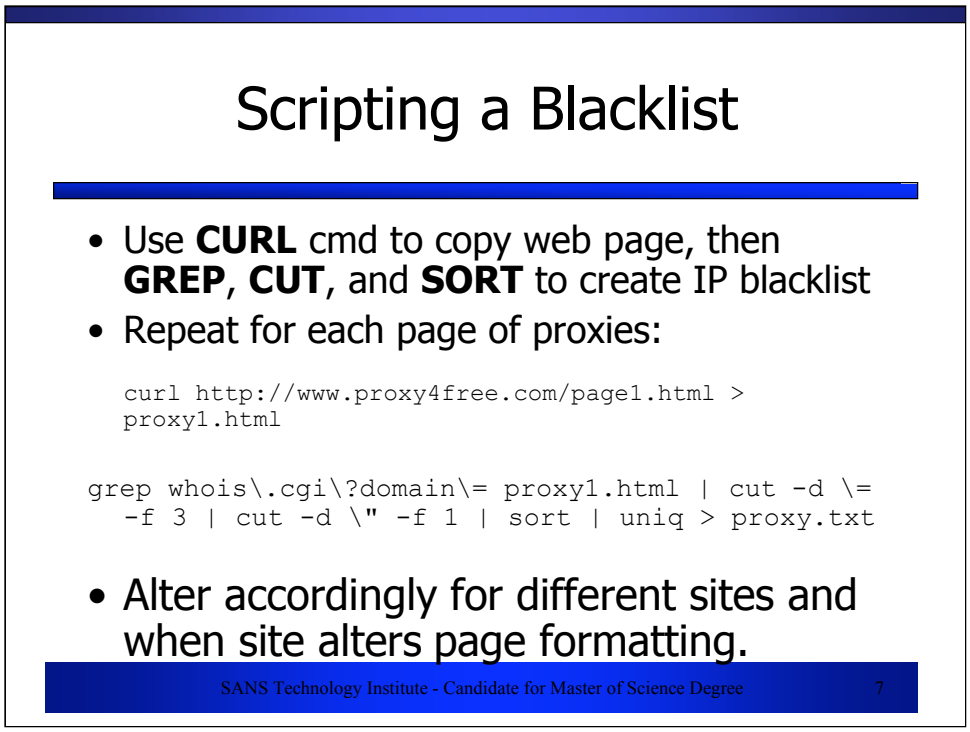
When today's proxies are blocked tomorrow, how do users learn about new ones? Many proxy owners are financially motivated to attract users, often displaying advertisements. Proxy list sites are where the two sides meet up. If you can get a list of fresh proxies the same time your users do, you can create a much better blacklist/blocklist. Method number 1 is to create blocklists using the same proxy list websites that your users are likely using.

Making Your Own Blacklist

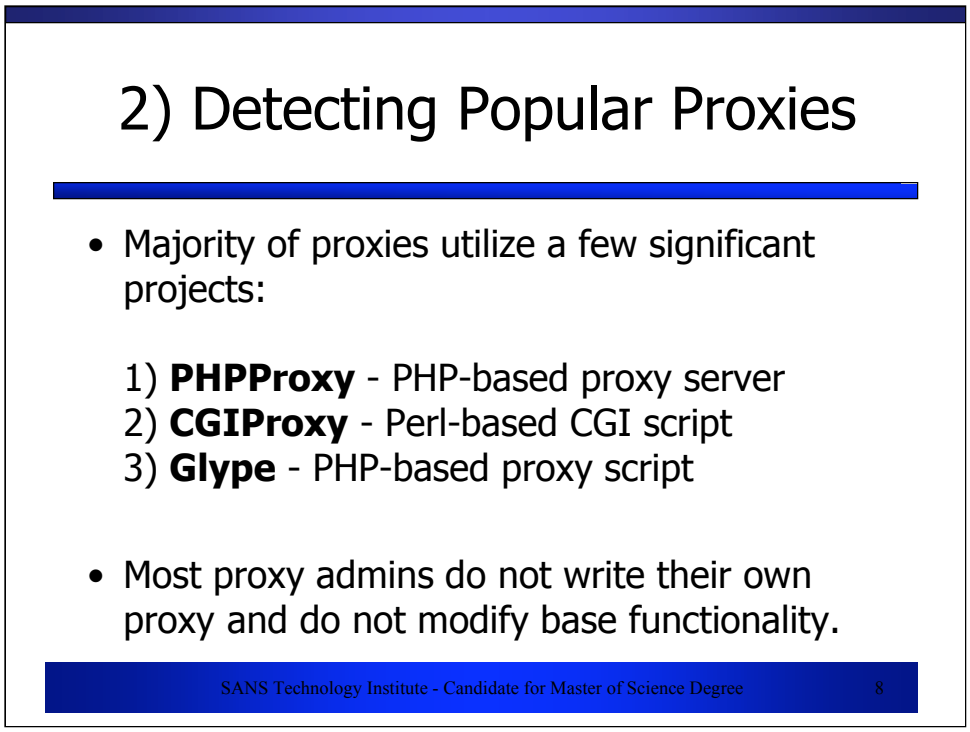
- Example: www.proxy4free.com
- Contains five pages of IP addresses of proxies.
- We can make our own blacklist.



Here is an example of one such proxy list. Some will provide IP addresses, others names. Some update more frequently and more reliably. Once you find one or more good ones, you can utilize them for your blacklist. Integrate your list with your filtering solution, and you've blocked access to these proxies.



Using a few Linux/Unix commands (ports also available for Windows) the proxy lists can be downloaded, parsed, and formatted into something that you should be able to integrate into your Internet filtering solution. Remember that sites can change their formatting and proxy list sites can come and go. The key point here is not that you can use the list from proxy4free.com in your blocking (although you certainly can) but that you can make use of these resources with a minimal amount of effort to provide much better site blocking.



Not all proxies will be caught by blacklisting, so let's look at method number two. A trend that I found is that most proxies are installed from only a few different open source projects. These key projects provide themes and templates and make it easier for the proxy owner to integrate advertising. Why reinvent the wheel when these projects already exist? Another trend that I found was that the proxy owners don't customize the functionality or formatting from the base project. This gives the potential to detect proxies by the class of proxy server that they are running.

Detecting a Glype Server

Example Glype URL:

http://www.reverseproxy.us/browse.php?u=Oi8vd3d3Lm15c3BhY2UuY29t&b=143

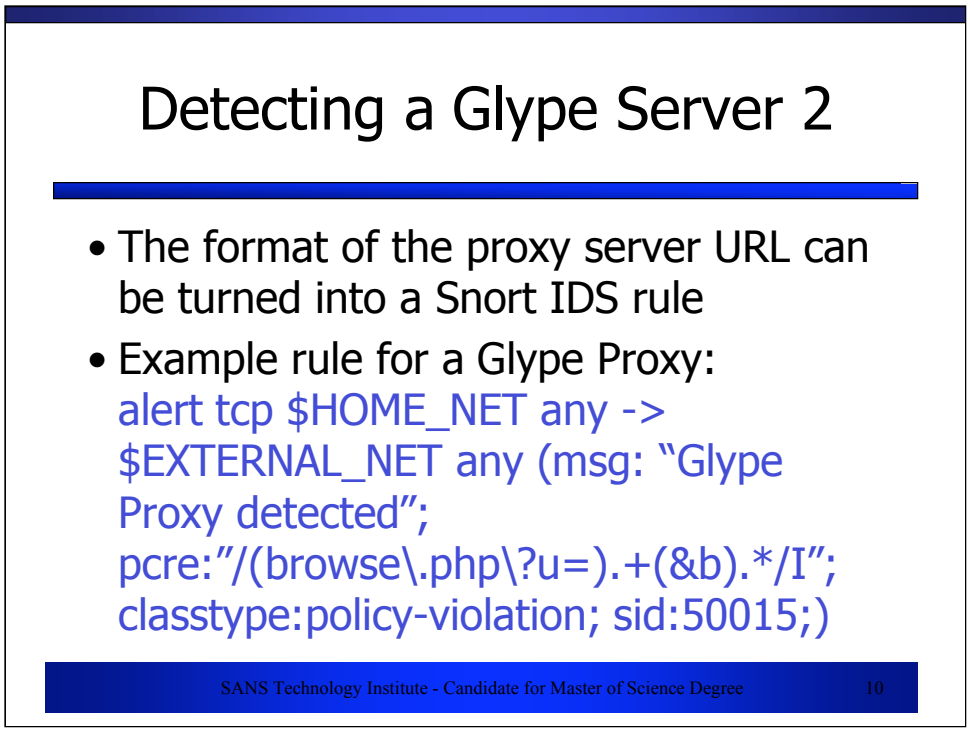
Format:

{hostname}/browse.php?u={obfuscatedURL}&b={identifier}

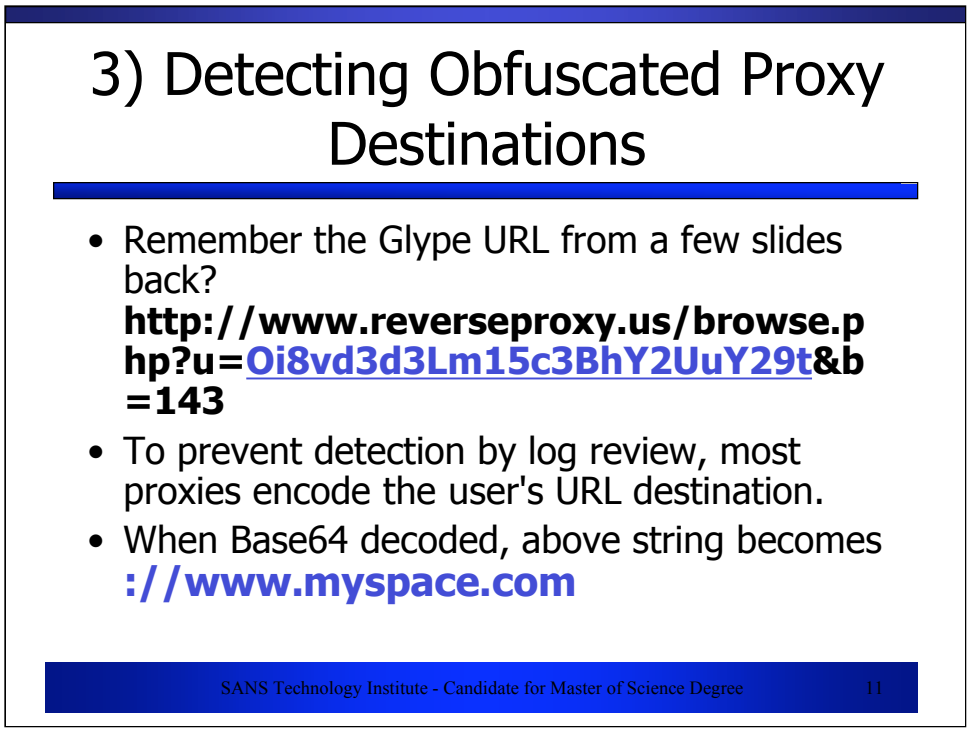
Regular Expression to Match:

(browse\.php\?u=).+(&b).*

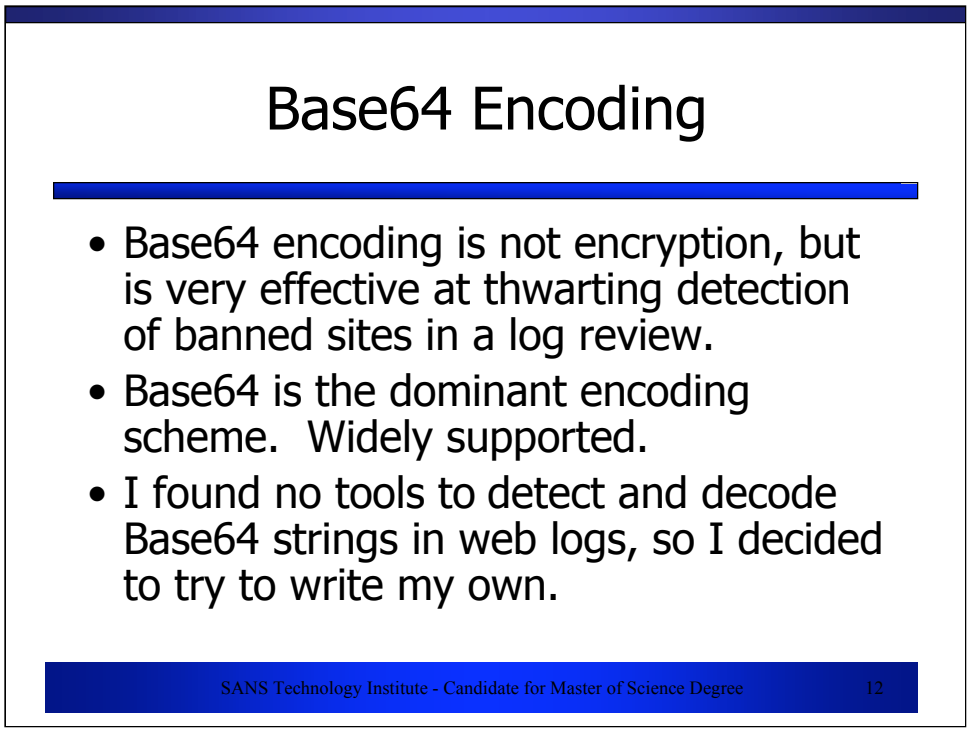
In this example we see the format of the URL sent from the user's browser back to the proxy server. Note the obfuscated URL, which we'll discuss in a few slides. We're able to find some text that is common to all default installs of Glype proxies that is relatively unique and probably won't match on any legitimate site. We can turn this text into a regular expression. We can use grep to find this in our Internet access logs, but there's something even better we can do with it.



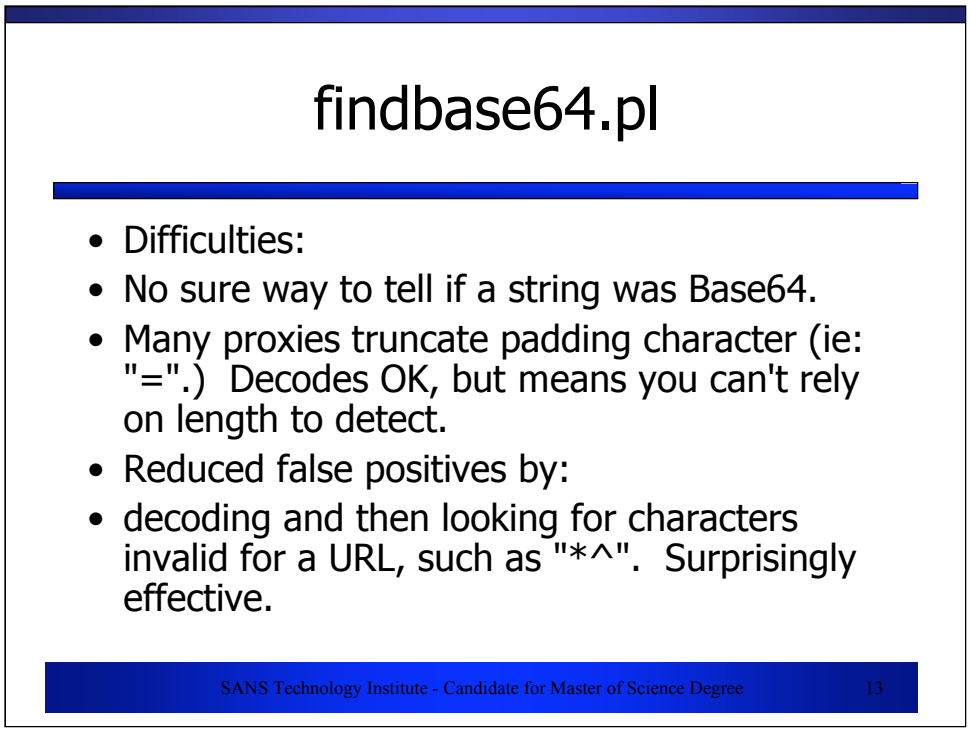
Snort is a very popular, open source IDS. Other IDS systems may also support Snort signatures or regular expressions. Here, the Perl Compatible Regular Expression (PCRE) functionality is used to make a rule that can alert you in real time of Glype proxy usage. Rules can also be made for the other proxy projects, allowing you to detect whole classes of proxies without relying on proxy site names and IP addresses.



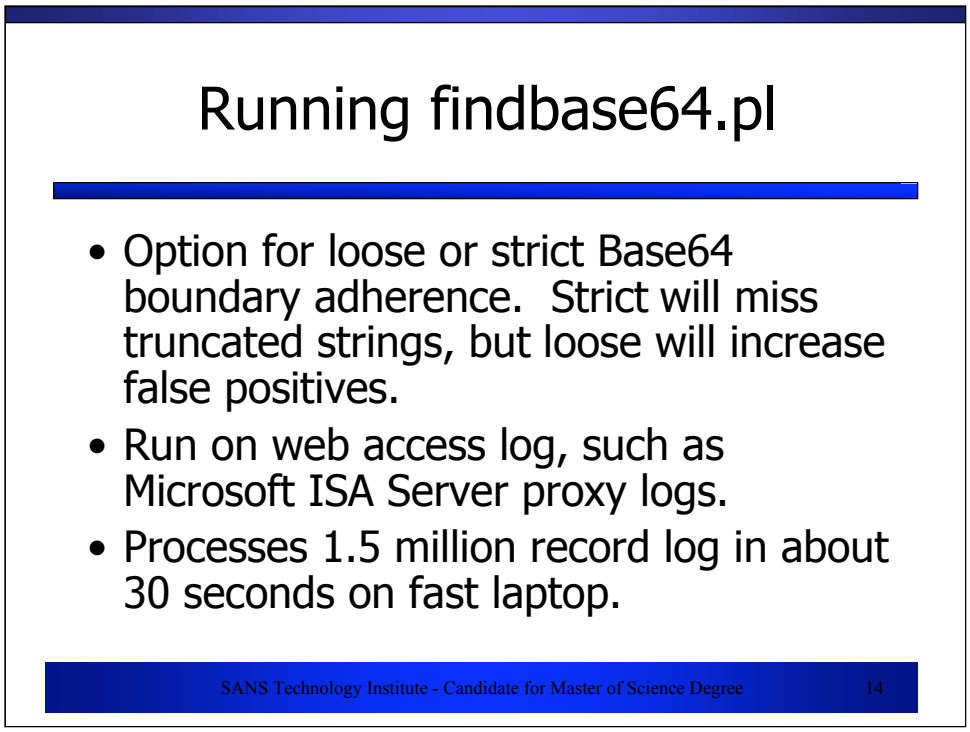
The third method is to detect obfuscated proxy destinations in your web access logs. To prevent detection, most proxies utilize obfuscation of the destination web site. Somehow, the user has to tell the proxy what site to go and retrieve. I've observed that Base64 is the prominent method of obfuscation, as it's well understood, well supported in programming environments, and efficient. The encoding happens within the browser and is sent to the proxy server, which decodes it to the real URL.



To determine if users have still been able to access prohibited sites, it would be very convenient if we could look for Base64 entries in Internet access logs and automatically decode them. I was unable to find any existing tool that could do this. I decided to try to write my own in Perl.



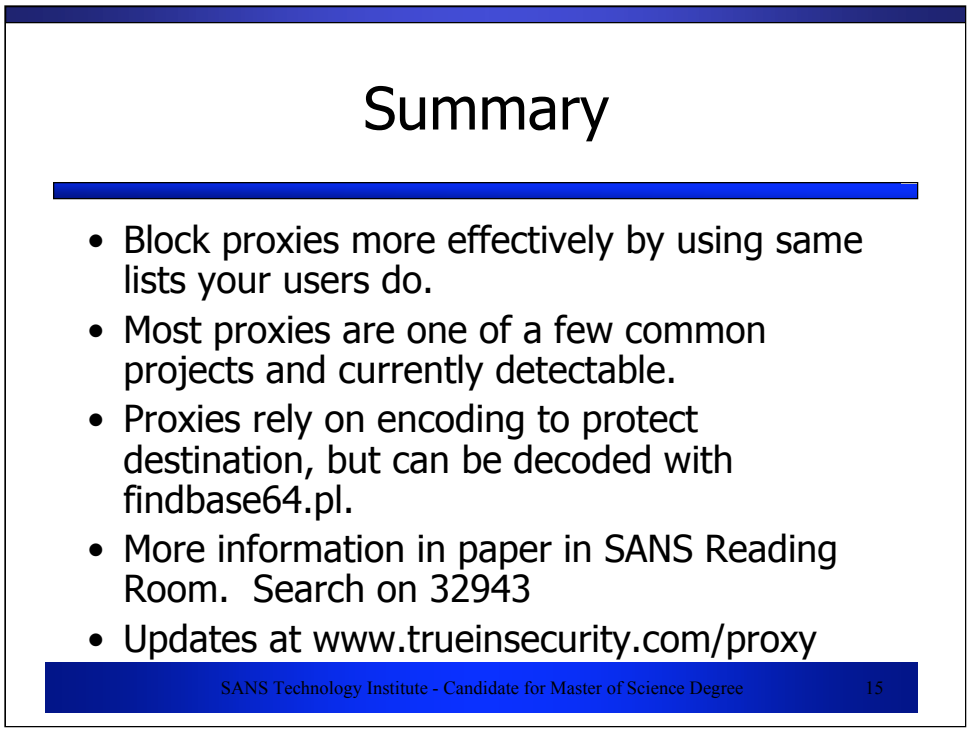
Perl has built in functionality for Base64 encoding and decoding, so that wasn't the hard part. After getting through the parsing of the URL string, the biggest problem was determining if the string was Base64 or not. Base64 characters are a-z, A-Z, 0-9, and a couple others that depend upon the implementation. Often, a "=" is used for padding, as Base64 values are always a multiple of 4 bytes. Some implementations drop the use of padding characters, meaning the encoded string may not be divisible by 4. This doesn't prevent decoding, but it makes it more difficult to programmatically determine whether a set of characters has been Base64 encoded. Initially, I had a lot of false positives, and the script decoded strings that were not Base64 encoded. I noticed a lot of character combinations that weren't valid for URLs. I realized that if I checked for some of these after decoding and they were found ,then the original string wasn't Base64 encoded. This false positive elimination worked out even better than I thought.



Findbase64.pl takes a file of URLs as its input and looks for a file named "weblog.txt". Depending upon how you log your users' Internet access, you may have to filter out additional fields that may be provided. For example, if you use Microsoft's ISA Server to control Internet access, its default log has many additional fields that you don't need and that need to be removed before running findbase64.pl. To filter out just the URLs from the ISA log, you could run the following command:

```
awk -F "\t" '{print $16}' nameofloffile.log > weblog.txt.
```

Awk parses each line of the input log file, using the TAB character as a delineator (as specified with the -F parameter) and then sends the 16th field, which is where the full URLs are, to the output. Note that while you can accomplish this with the CUT command, I have experienced issues doing this where CUT quits during processing of the data, especially in the case of ISA logs. A search on the Internet revealed that others have had problems with the CUT command. Unless and until this is fixed, I recommend using awk. If you do use CUT, make sure that your output file has the same number of records as your input file. After creating the weblog.txt file, type "perl findbase64.pl". The resulting file, base64check.txt, can be loaded into a spreadsheet program for review.



While there is no way to prevent or detect anonymous proxy usage 100% of the time, this presentation has presented three techniques, which may be used in unison, to greatly decrease the possibility that anonymous proxies are used without your knowledge. More techniques are discussed in my research paper, which can be found by going to the SANS site (www.sans.org/reading_room) and searching on the paper number, 32943.

Like all things on the Internet, anonymous proxies, proxy list sites, and detection methods are subject to rapid change. I've set up a web page where I will try to provide updated information as time allows. The URL is www.trueinsecurity.com/proxy