



Oh What a Wicked Web We Weave

By: *Bill Mathews*

For some time now the prevailing trend in IT and business development has been towards building web applications to interface with business systems. They all have different names (intranet, extranet, partnerweb, e-commerce, etc.), but they're all the same basic idea. Build a web application that talks to a back-end database and let customers do their business. Simple enough - make your business stuff accessible and easy to use for all of your customers, then profit! Great plan.

In the old days (mid to late 90s), much of this code was written with CGI (Common Gateway Interface)¹ and the programmers really had to know what they were doing. They had to at least know a programming language to really do it. Shortly afterwards quite a few scripting languages came along (PHP,² ASP,³ etc.), and more and more web programmers became web developers and believe me there is a difference. Then, and no one consulted me about this, the introduction of "frameworks" happened. Frameworks basically abstract the "hard" part of web programming and make it easier to concentrate on design and function.⁴ Admirable goals indeed.

Now I'm not saying frameworks make all the work go away, they don't. You have to understand how to code to the framework, but in a lot of cases you don't really need to understand what's going on under the covers. You

don't need to worry about how the frameworks interact with the database, nor how the framework generates the pages your users see. That stuff is just magic and makes your life a little bit easier and more productive. Again, all wonderful intentions.

There is absolutely nothing wrong with any of this until you realize what's really going on here. What we have is a whole lot of people building a whole lot of applications without really understanding what's going on. We're seeing a lot of people who don't even understand the frameworks they're using. They just point and click their way to an enterprise web application. This is dangerous stuff! I do many talks about input validation and a lot of developers just give me a blank stare, as if to say, "my framework takes care of this" or "Oracle does this for me". The truth is most of these developers don't understand what input validation is because they've never built anything outside of a framework. They don't understand data profiling at even a basic level.

This isn't really a knock on current web development trends, instead it is a knock on people not taking an interest in their craft. You need to be building security into your applications from the ground up and divorce yourself from the idea that the "framework will protect me from all evil", it won't, nothing will - not web application frameworks, not web application fire-

walls, nothing. The only way to a safer web application is through knowing how your applications really work and understanding what your data is and how to process it correctly. I've always contended that building in good security will help your application perform better and generally be more robust. Understanding how your application works will make you a better developer and who knows, maybe some day we'll see the return of web programming.

Upcoming Events

April 10: **Check Point R70 Preview**, Hurricane Labs Classroom, Independence, OH

April 21-23: **Check Point Essentials Class**, Hurricane Labs Classroom, Independence, OH

May 1: **Security on a Shoestring Budget**, Hurricane Labs Classroom, Independence, OH

May 22: **Check Point R70 Preview**, Hurricane Labs Classroom, Independence, OH

June 12: **Check Point R70 Preview**, Hurricane Labs Classroom, Independence, OH

Register for any of these events or classes at www.hurricanelabs.com.

1: <http://hoohoo.ncsa.uiuc.edu/cgi/>

2: <http://www.php.net>

3: http://en.wikipedia.org/wiki/Active_Server_Pages

4: http://en.wikipedia.org/wiki/List_of_web_application_frameworks

Common SSL Service Configuration Errors

By: Steve Benson

Understanding the Problem

One very common vulnerability pointed out on vulnerability assessment and security audit reports are configuration errors made in SSL servers. These configuration errors can cause an insecure network connection to be established where a secure one was expected and expose sensitive data. Furthermore, these configuration errors will often result in the failure of a security audit.

Normally an SSL client will choose a secure SSL version and ciphers when given a choice. However if both the client and the server support it, an attacker can coerce them into establishing a weak connection which the attacker can then break.⁵

The configuration errors made generally fall in to two different categories:

- Permitting an obsolete version of the SSL protocol itself to be spoken with a client (generally only SSL version 2).
- Allowing a connection to be encrypted with weak encryption methods (such as known weak or “export grade” ciphers).

The consequences of both of these errors are similar: an attacker who can

tamper with a weak connection between the SSL client and server can read and possibly alter the sensitive data within it, negating the security of SSL.

Note that in order for a weak connection to be established, both the client and server must agree on a weak method. Usually it is the case that misconfigured servers will permit weak connections in addition to strong ones; it is left up to the client to choose the connection parameters.

Practically speaking, modern SSL clients will almost always choose a strong method. Therefore, the presence of a misconfigured SSL server on your network does not necessarily mean all data is being transmitted insecurely to and from that server.

For that same reason, any concern that you should continue to permit old SSL versions or weak ciphers in case you have older clients using them can probably be set aside. You probably have few if any such clients. Export grade ciphers have not been necessary since 1996 when the relevant laws changed.⁶ The first standard draft for SSLv3 was published in 1996 as well. In any case, it is would be a better idea to break a few clients and let the cards fall where they may than to risk the exposure of sensitive data.

Finding Insecure SSL Services on Your Network

There are many ways to find an insecure SSL service on your network. Nearly all general-purpose security tools and services will identify such services to you in their reports.

Hurricane Labs also introduced a security tool specifically for finding, reporting and monitoring for the presence of weak SSL services on your network: an open source tool called *sslsweep*. It can produce reports in several different formats that make it convenient to check the security of many SSL services at once. (See below.) It can also be run as a Nagios plugin for ongoing monitoring. *Ssweep* can be downloaded for free from our web site at <https://www.hurricanelabs.com/software/sslsweep/>.

No matter how you identify insecure SSL services on your network, the fix is usually a simple server reconfiguration. Thanks to the prevalence of this problem, there is quite a bit of documentation on the web for nearly any kind of server software so fixing these problems should be a quick and easy process.

Host	Port	State	Cert CN	Expires	SSL Versions	Ciphers	
192.168.55.84	443	Good	*.example.com	Aug 29 21:56:10 2011 GMT	SSLv3 TLSv1	HIGH MEDIUM	
192.168.55.26	443	Weak	www.example.com	Sep 11 23:59:59 2009 GMT	SSLv3 TLSv1	HIGH MEDIUM LOW	
192.168.55.12	443	Error	_ssl.c:485: EOF occurred in violation of protocol				
192.168.55.15	443	Weak	extranet.example.com	Oct 27 23:59:59 2009 GMT	SSLv3 TLSv1	HIGH MEDIUM LOW	
192.168.55.201	443	Error	_ssl.c:485: EOF occurred in violation of protocol				
192.168.55.202	443	Error	_ssl.c:485: EOF occurred in violation of protocol				
192.168.55.75	443	Error	_ssl.c:485: EOF occurred in violation of protocol				
192.168.55.39	443	Weak	order.example.com	Apr 27 23:59:59 2009 GMT	SSLv3 TLSv1	HIGH MEDIUM LOW	
192.168.55.52	9640	Weak	tstparts.example.com	Jul 16 23:59:59 2009 GMT	SSLv3 TLSv1	HIGH MEDIUM LOW	

5: <http://www.scanit.be/uploads/ssl%20security%20in%20be%20-%202003-2008.pdf>

6: http://www.fas.org/irp/offdocs/eo_crypt_9611_memo.htm

Regular Expressions Should Be More Regular

By: Brian Glenn

Pattern matching is a common action performed by administrators every day. Almost all activities in IT involve patterns in some way such as searching through logs, defining policies on a proxy, even setting email filters. The standard pattern matching format with which most people are familiar is called a glob pattern. Glob patterns are used on the Windows and UNIX command lines as arguments to pretty much any filesystem command, such as "dir *.*". Glob patterns are quite limited for describing information that has a little more nuance than just wildcard characters. A more advanced form of pattern matching called regular expressions (regex for short) is available in both programming languages as well as many applications. Many administrators balk at the idea of learning this tool as most examples of it look like someone dropped an anvil on the keyboard and used what came out. While the topic of regular expressions is quite expansive and could (and has) filled many books, the basics are quite easy to grasp. If you have familiarity with glob patterns, then regex is an easy jump to make.

Here are the basic patterns:

- . (period) - Matches any character (like ? in glob patterns)
- * (asterisk) - Modifies the previous pattern to mean, matches ZERO or more of these
- + (plus) - Modifies the previous pattern to mean, matches ONE or more of these
- \ (backslash) - Escapes the following character (used to mean any of the above literally)

Another common wildcard used in glob patterns is the asterisk by itself to mean "all"; or more specifically, zero or more of any character. To emulate this in regex, you would use the combination of the period and asterisk together. To show

a few examples of the above symbols, take the following pattern: `mi*s.+t\.`

It would match the following strings:

- `msoft.` - (remember * means zero or more)
- `miiiiiiiiisabct.` - (remember . means any character)

But not:

- `microsoft.`
- `mist.`
- `msoft`

Now that we have some basics down, we can put it to practical use. One of the most common uses of regex in the Check Point environment is in the HTTP Worm Catcher IPS Protection built into every version of Firewall-1 since NG AI came out. The idea behind the Worm Catcher is that all URLs passing through the firewall in an HTTP stream are matched against a list of known bad patterns for different sorts of worms and other nasties. At the top of the configuration section for the protection, there is an Edit Patterns button that allows the administrator to define (or redefine) protections of her own. To illustrate the point, one of the classic examples of poor coding in a web application is a program called `formmail.cgi`.⁷ This is likely not an application that should be running anywhere, so we can use the Worm Catcher to stop it. Remember that a period by itself means any character, so it needs to be escaped with a backslash as follows:

```
formmail\.cgi
```

Unescaped periods are the most common error in writing regular expressions for network administrators. Remember that IP addresses will be written as `192\ .168\ .1\ .1` not as they normally look.

Another example of regex is in the ubiquitous command line tool in a UNIX environment called "grep". The "re" in grep stands for "regular expression," and this is the only thing that it does. Given a stream of text, search for a regular expression pattern and only print the lines that match. For example, if you wanted to see how many times a particular PDF had been downloaded from your webserver, the information can be easily extracted from the webserver logs:

```
webserver:/var/log/apache2# grep
-c project\.pdf access.log
42
webserver:/var/log/apache2#
```

The above searches the file named `access.log` for lines containing "project.pdf" and counts them, outputting the result. If the "-c" argument was left out, the actual matching lines from the log would be printed out.

There are plenty of resources available on the Internet if you would like to delve deeper into the subject of regular expressions. They are used across almost all disciplines of IT, so finding assistance online is not difficult. As always, if you have questions about using regular expressions in your own environment, please do not hesitate to contact Hurricane Labs for assistance.

Read more at our blog:
<http://blog.hurricanelabs.com/>

7: <http://www.scriparchive.com/formmail.html>

Extending Open Source Software

By: Steve McMaster

When talking about Open Source Software, everyone relates it to being “free as in beer” or “free as in freedom”. Because it is “free as in freedom”, you are able to tweak, mold and extend the software so that it fits your purposes. This past week I had experience with a particular open source project, trying to tweak and extend it to fit my purpose. In this case, my purpose was pretty simple: I did not want my two-factor client to rely on Java. I’ve not quite achieved my goal yet, but I’m getting there. Between the time I started and finished writing this, I did in fact achieve the goal.⁸

The software we use for our two-factor authentication is called WiKID.⁹ The server runs on Linux (it is distributed as RPM’s, but they have instructions for installing it on Ubuntu), and the token client is available for Windows, Mac OS X, and Linux,

as well as Windows Mobile, BlackBerry, all platforms supporting J2ME, and soon the iPhone. For the average user, this works quite well. It provides you with a two-factor token client on your desktop, and on your mobile phone. However, there are many people out there, including people here at Hurricane Labs, who have a distaste (of varying levels of extremity) for Java. So I took it into my own hands (though it took a lot of help from them, too) to port the token client to Python.

Open Source software means that you have the option of going through the source, figuring out exactly what is happening, and rewriting that in another language. This is exactly what I did (however, I did this only because I am impatient - the WiKID team was more than willing to provide protocol documentation). I dissected their

Java source, determined what they were doing, and reimplemented it in another language (Python). I can now take this code and do other things with it. We are considering integrating it into some home-brewed software we already have.

What does this mean for the rest of the world? It means that when you find a piece of Open Source software that fits your needs (well, mostly fits your needs), you are welcome to make changes to it. Or, if your project has specific needs, but needs to interoperate with your open source application, the same applies. It’s just like getting a recipe for cheesecake off of a Philadelphia Cream Cheese™ box – you’re free to use the recipe as is, or change it as you see fit.

8: <http://code.google.com/p/pywikid/>

9: <http://www.wikidsystems.net>



Hurricane Labs

4401 Rockside Road, Suite 310

Independence, Ohio 44131

216-923-1330

www.hurricanelabs.com



What's New? Hurricane Labs has MOVED! Due to the need for more space and future expansion, Hurricane Labs has moved to the 3rd Floor of their current building. Their classroom has also moved to the third floor. Feel free to visit our new space the next time you're in the area. The new suite number is 310!