Web Application Security Testing 101: Paros Proxy and Badstore

By Russ McRee

Prerequisites

Firefox 1.5

Paros Proxy 3.2.13

Java 1.4.2 or better

To be certain, Web application security is all the rage these days, and for obvious reasons. As the popularity of PHP-based forums, CMS portals, Wikipedia, and Ajax grow, based on ease of use and multifaceted functionality, the risks increase in parallel. To quote officers from the Web Application Security Consortium (WASC):

> *Understanding the security risks associated with a Web application is of critical importance to improving the security of the Web. [...] Making Web applications safe is in the best interest of all organizations and the general economy. Providing a clearly defined set of web application security best practices will advance security professionals' ability to anticipate and rapidly address potential threats to their enterprise*[1].

Pick your poison: abuse of functionality, denial of service, OS commanding, SQL injection, cross-site scripting... with so many vectors to be concerned with, testing regularly is essential.

Paros Proxy has been available since August 2002, and has been extensively written about by many far brighter than I. We won't blaze any new trails here, but as Web application security vulnerabilities increase, I believe every information security practitioner should have a basic understanding of the benefits of this tool.

There are other excellent freeware applications with which to conduct Web application security testing, including Burp suite, OWASP's WebScarab and Sprajax, as well as ye olde Nikto. Commercial products include services like WhiteHat Sentinel and WebInspect from SPI Dynamics.

Paros Proxy "is a HTTP/HTTPS proxy for assessing web application vulnerability. It supports editing/viewing HTTP messages on-the-fly with client-certificate, proxy-chaining, filtering and intelligent vulnerability scanning"[2].

OWASP's methods for writing up application security findings include understanding the vulnerability and the possible attack scenarios, as well as clarifying a risk level, likelihood (ease of discovery and execution) and business/technical impact[3].

Quite simply, the best way to establish these findings is through the use of a tool like Paros Proxy. But before we dive into some use scenarios, I'd like to offer you some really useful Firefox extensions, nicely described as part of the Web Hacking Toolkit at www.mightyseek.com. Download and install all the extensions described, including Web Developer, SwitchProxy, LiveHTTP Headers, and User Agent Switcher. We won't cover all the extensions here, but trust me, they're useful. We will certainly use SwitchProxy, but keep in mind, at the time of this writing it won't work with Firefox 2.0.

Use scenarios

After you've installed SwitchProxy and followed the very straightforward process to install Paros Proxy, tell SwitchProxy to use Paros Proxy as follows: Click Add, then click Standard, enter "Paros Proxy" in the Proxy Label window, and enter "localhost" and "8080" under HTTP Proxy. Port 8080 is default when you initialize Paros Proxy, but you can set it to any port you prefer. Just remember, if you change it in Paros, also do so in SwitchProxy or your browser. This will keep you from having to manually tweak your browser connection settings every time you want to fire up the proxy or connect normally.

A good first step is typically to see what can be learned by spidering a site. Paros includes the best spidering functionality of any tool I've used for this particular activity. As always, understand the power of the tool at hand. Paros will grab everything you browse, and if you don't limit your spidering or scanning to very specific URLs, it will dig well beyond what may be your desired or even legal bounds.

In the Sites view you can remove URLs from your scope by right-clicking an URL and choosing "Delete from View" or "Purge from DB."

It's always good to have a safe place to play with tools like Paros Proxy. Check out the NTO Hackme Test Site at http://hackme.ntobjectives.com or use Foundstone's Hacme Shipping. But if you just want to learn or practice without hassle, try Badstore, a bootable CD image that can be downloaded at www.badstore.net. This is an offering from a commercial company, NetContinuum, so be aware of followup sales contact.

For practicing and increasing your understanding, Badstore is quite useful for testing the basics of SQL injection and cross-site scripting. There's also a great blurb at http://www.phpsolvent.com/wordpress/?p=1516 regarding a VMware setup for Badstore as well, wherein you can bugger it all up, refresh the VM and start all over in seconds.

Boot Badstore on an available system on your network and take note of the IP address assigned via DHCP by typing ifconfig at the bash prompt. That IP will then be the URL for your study.

1 Aaron C. Newman, Yuval Ben-Itzhak, http://www.webappsec.org/quotes.shtml

2 User Guide for Paros v2.x, http://parosproxy.org/paros_user_guide.pdf

3 http://www.owasp.org/index.php/How_to_write_an_application_security_finding

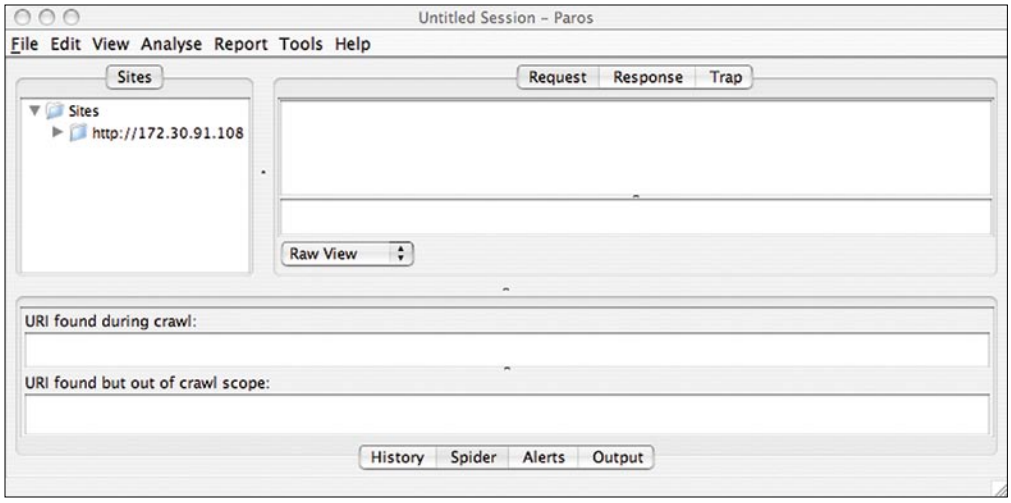

Figure 1 – Default Paros view

Let's begin by selecting Paros Proxy in SwitchProxy, fire up Paros, open a browser, and browse to the URL for your Badstore system. (See Figure 1)

To begin spidering, highlight your Badstore URL under Sites, click Analyse and select Spider, then Start or simply right-click the same selection, choose Spider and Start[4]. (See Figure 2)

You will now note all content found on the site, URI found in scope, as well as URI out of scope. Again, try not to crawl where you ought not. As defined in the Paros User Guide, "Spider is used to crawl the Websites and gather as many URL links as possible. This allows you to have a better understanding of the Website hierarchy tree in a short time before manual navigation."

4 User Guide for Paros v2.x, http://parosproxy.org/paros_user_guide.pdf, p. 8

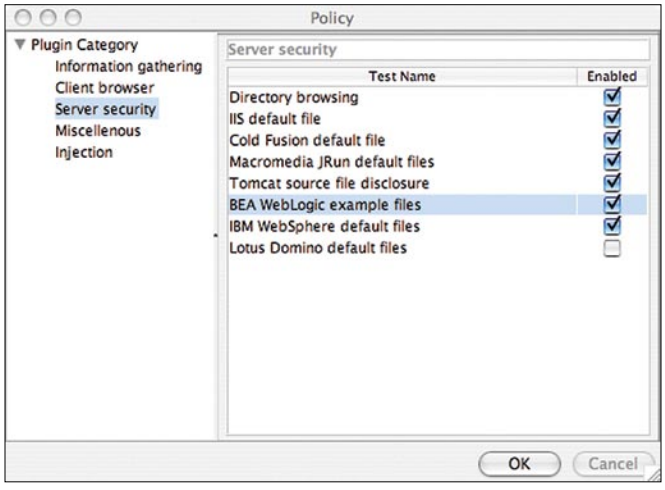

Figure 3 – Paros Scan Policy

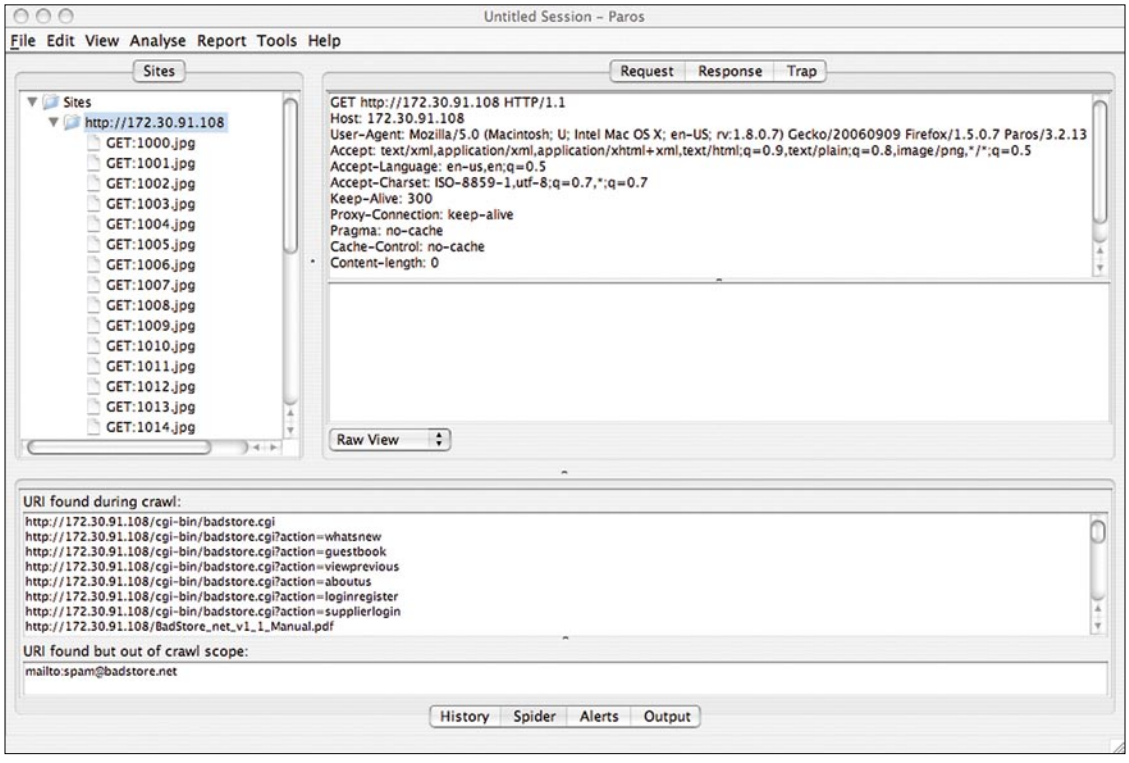

Figure 2 – Paros, post-Spider

Before moving to a scan as your next step, be sure to tune your Scan Policy to avoid scanning for irrelevant information.

Choose Analyse, then Scan Policy. Here you'll note plugin options, including "Server security." Decide what is truly relevant in your scan. Quite possibly you'll determine that Lotus Domino default files aren't part of your scope, and decide to disable it. (See Figure 3)

Scanning with Paros will be seen in logs as quite aggressive. Be sure, as always, that you have permission to run Paros against your chosen target, if and when you decide to use it as part of your toolkit.

Under Analyse you'll see options for Scan or Scan All. Choose Scan All for complete discovery. (See Figure 4)

The scan will show High alerts for SQL injection, Medium alerts for directory browsing and cross-site scripting, and Low alerts for Obselete files.

For purposes of reporting after a scan that can be used for remediation, click Report, then Last Scan Report. This step will create a .htm report and advise you of the file location. (See Figure 5)

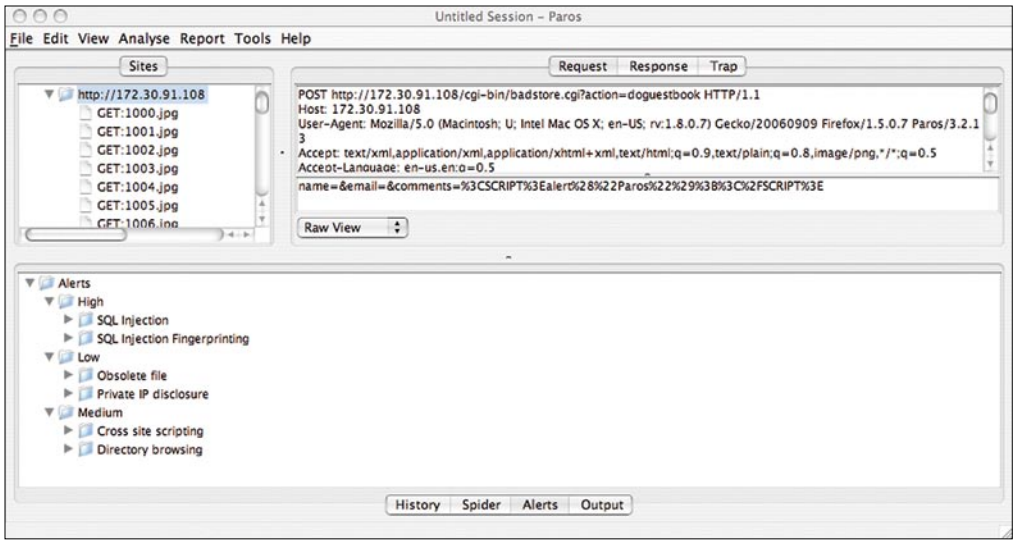

Figure 4 – Paros Scan results

You'll note alert details as well as parameters passed, and a detailed description of the vulnerability.

Finally, one of my favorite elements in this tool is the built-in Encode/Hash option.

I'll give you one little nugget that you can easily find when practicing on Badstore, but I won't reveal any further secrets, as Kurt Roemer, Badstore's creator, obviously intends for you to seek these out yourself.

I am by no means an expert Web assessor, but I know enough to always look for a `robots.txt` file. While Paros won't necessarily discover it for you while spidering, you can always conduct a manual attempt. Doing so with Badstore will reveal this:

```
# /robots.txt file for http://www.badstore.net/
# mail webmaster@badstore.net for constructive criticism

    User-agent: badstore_webcrawler

    Disallow:

    User-agent: *

    Disallow: /backup

    Disallow: /supplier

    Disallow: /upload
```

If you explore these disallowed entries, you'll discover that `backup` is readable, but empty; `upload` gets you nothing; but `supplier` yields gold. Hiding there is an account file. I won't reprint it here (dig out for yourself), but I'll use one of the hashes for our experiment:

```
amFuZXVzZXIvd2FpdGluZzRGcmlkYXkvMTcyLjIyLjEyLjE5
```

Click Tools, then Encode/Hash. Paste the hash where its says "Enter text below to be decoded," and select Base64 Decode. (See Figure 6)

The results are quick and productive: janeuser/waiting4Friday/172.22.12.19

We've yielded a username and a password, more than enough information for further exploration.

Conclusion

From OWASP we learn that principles, with regard to application security, are essential:

> *Principles must be evaluated, interpreted, and applied to address a specific problem. Although principles can serve as general guidelines, simply telling a software developer that their software must "fail safely" or that they should do "defense in depth" won't mean that much*[5].

A tool like Paros Proxy will aid you and your organization in providing "information in depth" to ensure a more secure Web-facing posture.

5 http://www.owasp.org/index.php/Category:Principle

Paros Scanning Report

Report generated at Wed, 8 Nov 2006 13:31:38.

Summary of Alerts

Risk Level	Number of Alerts
High	2
Medium	1
Low	2
Informational	0

Alert Detail

High (Warning)	SQL Injection
Description	SQL injection is possible. User parameters submitted will be formulated into a SQL query for database processing. If the query is built by simple 'string concatenation', it is possible to modify the meaning of the query by carefully crafting the parameters. Depending on the access right and type of database used, tampered query can be used to retrieve sensitive information from the database or execute arbitrary code. MS SQL and PostGreSQL, which supports multiple statements, may be exploited if the database access right is more powerful. This can occur in URL query strings, POST paramters or even cookies. Currently check on cookie is not supported by Paros. You should check SQL injection manually as well as some blind SQL injection areas cannot be discovered by this check.
URL	http://172.30.91.108/cgi-bin/badstore.cgi?action=register
Parameter	fullname=1&email=1&role=U%27INJECTED_PARAM&Register=Register
Other information	SQL
URL	http://172.30.91.108/cgi-bin/badstore.cgi?action=register
Parameter	fullname=1&email=1%27INJECTED_PARAM&role=U&Register=Register
Other information	SQL
URL	http://172.30.91.108/cgi-bin/badstore.cgi?action=register
Parameter	fullname=1%27INJECTED_PARAM&email=1&role=U&Register=Register
Other information	SQL
URL	http://172.30.91.108/cgi-bin/badstore.cgi?action=supplierportal

Figure 5 – Paros report

References

User Guide for Paros v2.x:

http://parosproxy.org/paros_user_guide.pdf

About the Author

Russ McRee, GCIH, is a security analyst working in the Seattle area. He is a member of ISSA, Pacciso, InfraGard, and CCSA (Cyber Conflict Studies Association). Russ maintains holisticinfosec.org. Contact him at russ@holisticinfosec.org.

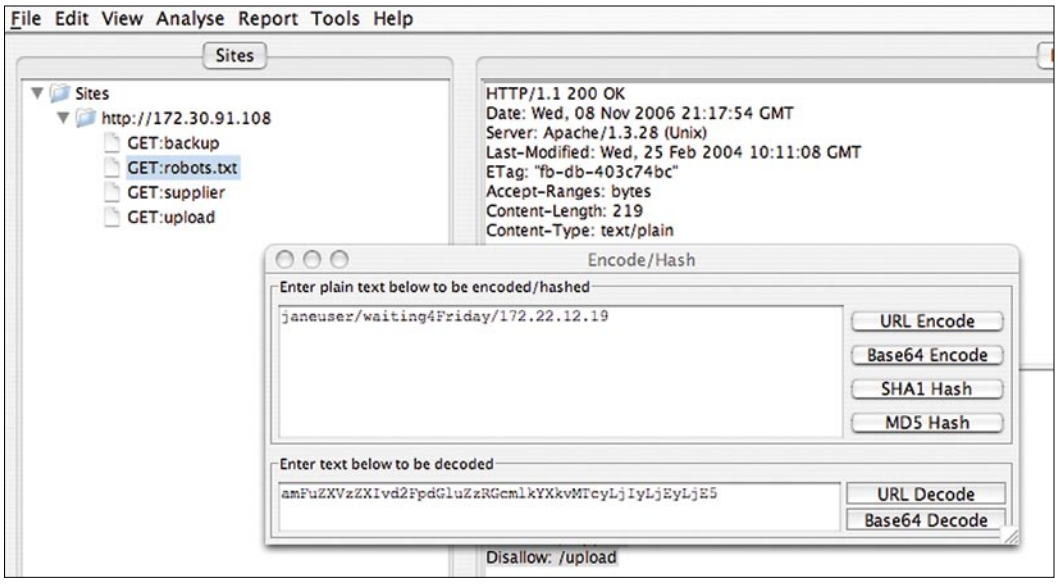

Figure 6 – Encode/Hash