Web-Application Security: Injection

Week of January 31, 2017

Goals:

- Gain familiarity with basic web application security.
- Implement successful injection attacks on the Mutillidae and DVWA web applications.
- Gain increasing familiarity with Burp Suite and its utilities.

Required:

- Take the relevant injection lessons on hacksplaining.com (command execution, SQL injection).
- Read the 2013 OWASP Top 10 List http://bit.ly/2jOs4ba.
- Install Burp Suite Free Edition from portswigger.net/burp/freedownload (if necessary).
- If you are using Firefox or Chrome, install the FoxyProxy Basic plugin for convenience.
- Download and install python2 (www.python.org/downloads) and SQLmap (sqlmap.org).

Approach:

- We will be attacking the web applications included with the Metasploitable 2 virtual machine, namely Mutillidae and DVWA. If you wish to follow this lab in the future using a fresh copy of Metasploitable 2, there is an error in one of Mutillidae's config files:

```
root@metasploitable:~# cat /var/www/mutillidae/config.inc
<?php
        /* NOTE: On Samurai, the $dbpass password is "samurai" rather than blank
   */

        $dbhost = 'localhost';
        $dbuser = 'root';
        $dbpass = '';
        $dbname = 'metasploit';
?>
```

In /var/www/mutillidae/config.inc, please change the line that reads:

$dbname = 'metasploit';

To:

$dbname = 'owasp10';

Note: The ACM VMs have already had this issue fixed.

1. Every group is assigned an IP address of an instance of Metasploitable. Your group's IP address is:

2. As with the previous lab, fire up Burp Suite with default configurations and navigate to Proxy->Intercept to make sure intercept is turned off again.
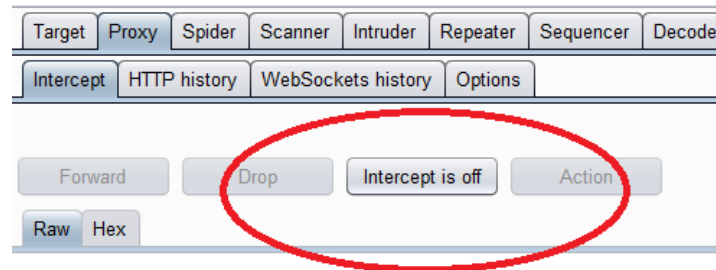
3. This time we will start with DVWA. Start up your browser and make sure FoxyProxy is configured to use Burp Suite as your proxy. Then point your browser at the IP address given to you in #1. Click on DVWA and you'll be presented with this page:

Go ahead and login with the given credentials (admin:password).

4. Read the home page. Once you are done, locate the DVWA Security button on the sidebar. Click on it and set the security settings to low:

5. You might have noticed the CSRF and XSS buttons on the sidebar. Feel free to try to work through them on your own if you didn't get a lot of practice last week. Once you are finished, click on the "Command Execution" button. Go ahead and enter the IP address of one of Google's public DNS servers: 8.8.8.8. Notice the output. Does it look familiar?

Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

[   ] [submit]

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=47 time=34.8 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=47 time=35.0 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=47 time=36.0 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2017ms
rtt min/avg/max/mdev = 34.833/35.316/36.081/0.568 ms
```

This looks like the exact same output as the command-line tool, ping!

```
mbaccia@chopin [06:24 PM] ~
$ ping 8.8.8.8 -c 3
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=46 time=11.2 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=46 time=11.2 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=46 time=11.2 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 11.236/11.249/11.264/0.123 ms
```

6. It appears as if DVWA is using the ping command-line tool behind the scenes to carry out the ping request. This could potentially be dangerous if input is taken directly from the user. Let's try to inject a command and see what happens:

Ping for FREE

Enter an IP address below:

[8.8.8.8; echo testing] [submit]

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=47 time=35.1 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=47 time=34.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=47 time=34.6 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2017ms
rtt min/avg/max/mdev = 34.639/34.807/35.133/0.276 ms
testing
```

Looks like user input isn't being sanitized…

7. Now that we know we can inject commands at will, we can try something a little more malicious. If you're running on Linux (or can ssh into the ACM's servers), you have access to netcat (nc), the swiss-army knife of TCP connections. Let's initiate a remote shell using nc:


8. This causes the server to start listening for incoming connections on port 1234, and execute /bin/bash when there is a connection. You'll notice that after you hit submit, the webpage seems to hang; this is because it's now listening for a connection. Go ahead and connect to using nc from your terminal:
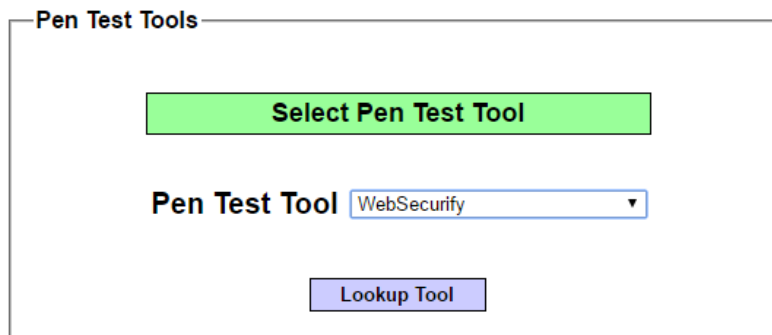

9. We have a shell! You'll notice that the regular prompt doesn't appear, but otherwise it functions like a regular shell.

   *Note: because the -e flag is potentially very dangerous (as we just demonstrated), it is typically not included with default installations of nc. The most popular nc package, the Open-BSD variant, does not include the -e parameter at all. Never include nc on a production machine unless you absolutely need it!

10. DVWA also has SQL injection vulnerabilities, but we will come back to it later. For now, navigate back to Mutillidae. We will be looking at JSON Injection. Navigate to the Pen Test Tool Lookup page:
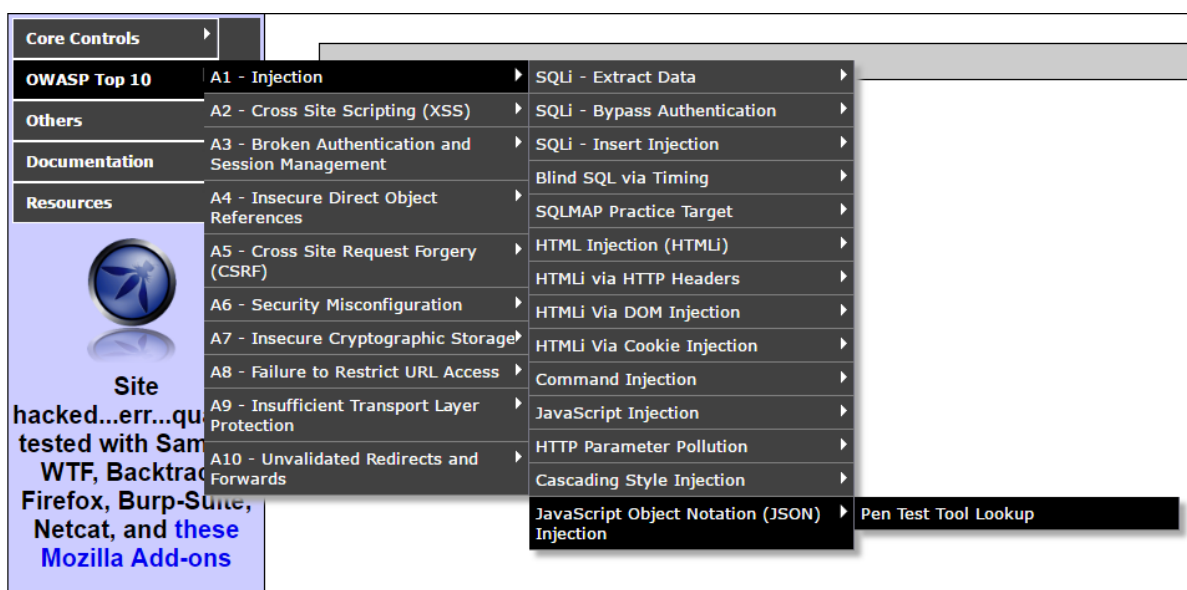
11. This vulnerability is somewhat more difficult to detect and similar to the second XSS attack we carried out last week. We already know that there is a vulnerability somewhere on the page, but it doesn't look like we can enter input anywhere:

Go ahead and post a request and look at what is sent in Burp. Try manipulating some values in the Repeater tab and see if you can find some input that is displayed back to the user without sanitization.

12. Once you've identified your injection point, look at the surrounding context. How can we break out of the JSON context and into the encapsulating script tags?

```
<!-- BEGIN HTML OUTPUT -->
<script type="text/javascript">

        var gUseJavaScriptValidation = "FALSE";
var gDisplayError = "FALSE";
try{
                                var gPenTestToolsJSON = ( {"query": {"toolIDRequested": "CANARY",
"penTestTools": []}} );
                }catch(e){
                                alert("Error trying to evaluate JSON: " + e.message);
                };
```

Injection Point

13. This can be a little tricky, depending on your experience with web development. Essentially, what we want to do is insert closing parentheses, brackets and semicolons to escape the JSON context and get back into the JS context. Go ahead try to bring up a simple alert box. You can view the response in your browser by right-clicking the response in the Repeater tab and selecting "Show request in browser". If all goes well, you should be greeted with a familiar face:
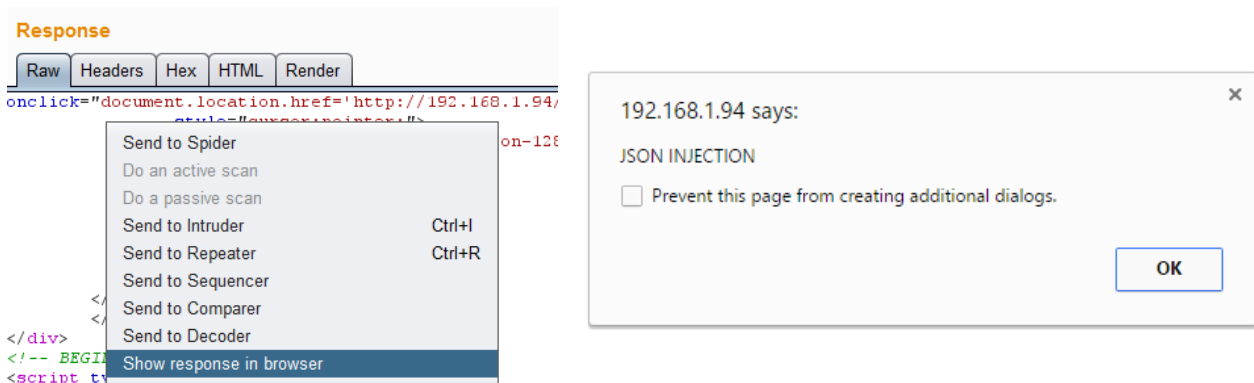
If you're having trouble, go ahead and turn on hints. The hints will spoil the answer, so be careful!

14. Mutillidae also has a command injection vulnerability as noted in the menu. It's similar to the vulnerability in DVWA so feel free to try it if you are so inclined.

15. Our final attack will explore SQL injection. If you are not familiar with SQL or databases in general, a lot of this might fly over your head. If that is the case, I would highly recommend looking into them as a lot of applications and services are built on top of a database backend. They're fairly ubiquitous. Navigate to the login page in Mutillidae:
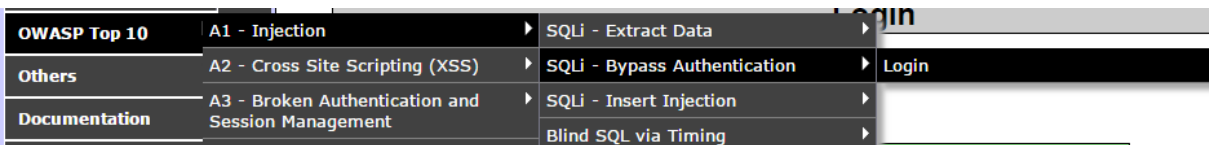
16. As far as SQL injection attacks are concerned, this example is extremely basic but demonstrates how damaging they can be. We are given two fields to work with. Since we already know that this page is vulnerable to SQL injection, go ahead and try entering potentially troublesome characters (i.e. single-quote, double-quote, ending parentheses, two hyphens followed by a space):

17. How can you tell if you were successful? Luckily, Mutillidae is misconfigured and tries to be helpful by displaying an error message:

That diagnostic information looks pretty handy. We don't have a list of usernames, but I'll give you a hint and say that there's an "admin" account. Try to get the password check to pass by injecting 'OR 1=1' along with whatever else is necessary to generate valid SQL. If you need help, see the next page.

18. Were you able to authenticate as the administrator? Try entering the following:

19. SQL injection attacks tend to be very complicated in a real world setting, and many people (myself included) are not able to manually craft some of the more intricate injections. While it is ideal to be able to craft specifically what you need, sometimes we value speed and efficiency. Tools like SQLmap help in this regard. SQLmap is a python script that will automatically search for injection points, finger-print databases, exfiltrate data and even open up command shells automatically. We will return to DVWA for the next attack.

20. While I was in the process of creating a tutorial, I stumbled upon the following blogpost that serves as a great introduction: https://pentestlab.blog/2012/11/24/owning-the-database-with-sqlmap/. Once you complete that guide and feel comfortable, try to run SQLmap on Mutillidae to see if you can dump the passwords for DVWA again through Mutillidae's SQL injection vulnerabilities.