PRISMO SYSTEMS

Detection of WebShells and Command Injection Attacks.

Abhishek Singh , Ramesh Mani, Anjan Venkatramani, Chih-Wei Chao

Introduction

Web shells are one of the critical malware which has extensively been used by threat actors. Web applications exploits or configuration weakness including Cross-Site Scripting, SQL injection, Vulnerability in applications/services, File processing Vulnerabilities, Remote File Inclusion (RFI) and Local File Include (LFI) vulnerabilities [2] can be used to deliver web shells. Once a web shell has been installed on a web server, it will act as a remote access tool for a threat actor. It will allow him to launch the next stage of attack such as execute commands, upload malware. APT32, APT34, Deep Panda, Dragonfly, OilRigs are some of the threat actors who have made use of Web shells for persistence and privilege escalations [1]. The web shell can be PHP, ASP, JSP, Perl, Ruby, Python applications. In this document, we have taken the PHP based Web shell, explained the execution flow of code and then discuss the solution to detect it. The same methodology can be applied to the non-PHP based web shells as well.

Technical Details

Figure 1.0 shows the code of pws.php web shell. As shown in figure 1.0, the code accepts the input from the threat actor via the data field of the POST method. The data field then acts as an input to the passthru() PHP function which executes the input from a threat actor

```
<?php
if ($_POST['cmd']){
$cmd = $_POST['cmd'];
passthru($cmd);
}
?>
```

Figure 1.0 showing code of PHP WebShell.

Similar to the POST method, web shells have also made use of the GET method. <?php eval($_GET[‘c’]);> is the code of another web shell, which will take the input from threat actor via the data field of the GET method and executes the input by making a call to eval(). Eval (), passthu(), shell_exec(), system() are some of the PHP API’s which have been used by web shells to execute malicious commands send via the data field of GET and POST methods. Besides the execution of the malicious commands, Web Shells are also used for uploading files to the web server, sending SQL queries to the Database servers or used as a relay to communicate with the host in the internal networks.

PRISMO SYSTEMS

Detection of Webshells

The algorithm to detect web shell leverages application level hooking to trace the data passed to the GET, POST methods. If the data passed to the GET, POST methods are direct parameters to the PHP program execution functions such as eval(), passthru(), shell_exec(), system(), proc_open(), then the process is marked as malicious.

The solution is independent of the applications which are hosted on the Web Server. If there is a known or a 0-day command injection vulnerability in a legitimate hosted application, it will get detected as well. In case of command injection vulnerability in a valid application, the exact input to the data field of GET and POST methods will act as a parameter to the program execution functions such as eval(), passthru(),shell_exe(), system() allowing a threat actor to execute malicious commands. The flow will get marked by the algorithm as malicious and alert of compromise will be raised.

Conclusion

The algorithm to detect the web shells involves application level hooking, tracing the data passed to the GET and POST methods and raising an alert if the data is a direct parameter to any program execution function. The solution provided the following inherent advantages.

- It is independent of the first stage of exploitation or delivery vector which leads to the installation of the web shells.
- It is independent of the later stages of exploitation such as defensive evasion, credential access, discovery, lateral movement, collection, exfiltration, command, and control phases.
- It is independent of the signatures of the past web shells [3].

These factors make it a recommended solution to prevent web shells.

Reference:

[1] Web Shell, https://attack.mitre.org/wiki/Technique/T1100
[2] Compromise Web Servers and Web Shells - Threat Awareness and Guidance, https://www.us-cert.gov/ncas/alerts/TA15-314A
[3] PHP Shell Detector, https://github.com/emposha/PHP-Shell-Detector/blob/master/README.md