GitHub - nsacyber/Mitigating-Web-Shells: Guidance for mitigation web shells. #nsacyber

By iadgovuser47

Archived: 2026-04-06 01:06:12 UTC

This repository houses a number of tools and signatures to help defend networks against web shell malware. More information about web shells and the analytics used by the tools here is available in NSA and ASD web shell mitigation guidance Detect and Prevent Web Shell Malware.

- NSA press release
- ASD press release

Table of Contents

Background

Web shells are malicious files or code snippets that attackers put on compromised web servers to perform arbitrary, attacker-specified actions on the system or return requested data to which the system has access. Web shells are a well-known attacker technique, but they are often difficult to detect because of their proficiency in blending in with an existing web application.

Detecting/Blocking Web Shells

"Known-Good" file comparison

The most effective method of discovering most web shells is to compare files on a production web server with a known-good version of that application, typically a fresh install of the application where available updates have been applied. Administrators can programmatically compare the production site with the known-good version to identify added or changed files. These files can be manually reviewed to verify authenticity. NSA provides instructions below on how to perform the file comparison in either a Windows or Linux environment.

WinDiff Application

Microsoft developed a tool called WinDiff that allows file or directory comparison for ASCII files. In most cases, this simple but powerful tool is sufficient for comparing known-good and production images. Details about using this utility are provided by Microsoft here.

PowerShell utility for known-good comparison

The provided PowerShell script will compare two directories, a known-good version and a production image. The script will report any new or modified files in the production version. If a web shell is in the web application, then it will appear on this report. Because of the high likelihood that benign file changes occurring, each result will need to be vetted for authenticity.

Requirements

- PowerShell v2 or greater
- Read access to both the known-good image and the production image

Usage

```
PS > .\dirChecker.ps1 -knownGood "<PATH>" -productionImage "<PATH>"
```

Example: Scanning default IIS website: `PS > .\dirChecker.ps1 -knownGood .\knownGoodDir\ -productionImage "C:\inetpub\logs\"`

Linux Diff utility for known-good comparison

Most distributions of Linux include the "diff" utility, which compares the contents of files or directories.

Requirements

- diff utility (typically pre-installed on Linux)
- Read access to both the known-good image and the production image

Usage

```
$ diff -r -q <known-good image> <production image>
```

Example: Scanning default Apache website: `$ diff -r -q /path/to/good/image/ /var/www/html/`

Detecting anomalous requests in web server logs

Because they are often designed to blend in with existing web applications, detecting web shells can be difficult. However, some properties of web shells are difficult to disguise or are commonly overlooked by attackers and can guide defenders to concealed web shells. Of particular interest are the user agent, referrer, and IP address used to access the web shell.

- *User agent HTTP header*: Without an established presence within a network, it is unlikely that an attacker will know which user agent strings are common for a particular web server. Therefore, at a minimum, early web shell access is likely to be performed using a user agent that is uncommon on a target network.
- *Referrer HTTP header*: For most web applications, each user request is appended with a referrer header indicating the URL from which the user request originated. The major exception to this is root level pages, which are often bookmarked or accessed directly. Attackers may overlook the referrer tag when disguising their web shell traffic. If so, these requests should appear anomalous in web server logs.
- *IP Addresses*: Depending on the attacker’s tactics and the victim environment, IP addresses used to conduct the attack may appear anomalous. For instance, a web application may primarily be visited by internal users from a particular subnet. However, the attacker may access the web shell malware from an IP address outside the normal subnet.

This analytic is likely to produce significant false positives in many environments, so it should be employed cautiously. Additionally, attackers who understand this analytic can easily avoid detection. Therefore, this analytic should only be one part of a broader defense in depth approach to mitigating web shells.

Splunk queries for web server logs

The provided Splunk queries can help to identify Uniform Resource Identifiers (URIs) accessed by few User Agents, IP addresses, or using uncommon Referer [sic] headers. Results of these queries are likely to include mostly or entirely benign URIs. However, if a web shell is present on the target server, it is likely to be among the returned results.

PowerShell script for Microsoft IIS logs

The provided PowerShell script will attempt to identify anomalous entries in IIS web server logs that could indicate the presence of a web shell. The script calculates the URIs successfully handled by the server (status code 200-299) which have been requested by the least number of user agents or IP addresses. This analytic will *always* produce results regardless of whether a web shell is present or not. The URIs in the results should be verified benign.

Requirements

- PowerShell v2 or greater
- Full Language Mode of PowerShell enabled (see here)
    - This is the default mode for most systems
- Read access to IIS logs

Usage

```
PS > .\LogCheck.ps1 -logDir "<path to IIS log directory>"
```

Example: Scanning logs stored at the default IIS location: `PS > .\LogCheck.ps1 -logDir "C:\inetpub\logs\"`

Additionally, the size of the percentile returned can be modified with the '-percentile N' option. The default is to show URIs in the bottom 5 percentile for unique user agent requests and client IP addresses.

Python script for Apache httpd logs

The provided Python script will attempt to identify anomalous entries in Apache web server logs that could indicate the presence of a web shell. The script calculates the URIs successfully handled by the server (status code 200-299) which have been requested by the least number of user agents or IP addresses. This analytic will *always* produce results regardless of whether a web shell is present or not. The URIs in the results should be verified benign.

Requirements

- Python 3
- Read access to Apache logs

Usage

```
$ LogCheck.py "<path to Apache log file>"
```

Detecting host artifacts of common web shells

Web shells are easy to modify without losing functionality and can thus be tailored to avoid host base signatures such as file artifacts. In rare cases, web shells may even run entirely in memory (i.e., fileless execution) making file based detection impossible. However, attackers may make little to no modifications to web shells for a variety of reasons. In these cases, it may be possible to detect common web shells using pattern-matching techniques, such as YARA rules. YARA rules can be imported by a variety of security products or can be run using a

standalone YARA scanning tool. The instructions below assume use of the standalone YARA scanning tool. For other security products, consult documentation or talk to the vendor to determine if YARA is supported.

YARA rules for detecting common web shells

The YARA rules contain a variety of signatures for common web shells and some heuristic approaches to identifying artifacts typically found in web shells. Two sets of signatures are provided, *core* (compiled core rules) and *extended* (compiled extended rules). It is recommended to use the compiled versions of the rules, as some host-based security systems will falsely flag the uncompiled rules as malicious.

- *core* rules include only indicators for common web shells that are unlikely to occur in benign web application files. These rules should return minimal false positive results but are also less likely to detect variations of common shells
- *extended* rules include both indicators for common web shells and indicators of techniques frequently used in web shells, such as obfuscation, dynamic execution, and encoding. Because these techniques are also sometimes used in benign web applications, these rules are likely to produce a significant number of false positive results but are also detect a broader range of web shells.

Administrators should prioritize vetting results from the core rules before investigating results from the extended rules. For some applications, the extended rules will result in too many false positives to be useful.

Requirements

- a YARA compatible security application or the standalone YARA tool (v3+)
- file-level, read access to the web directories for the target web application

Usage

```
> .\yara64.exe -r -C <yara file> <path to web directory>
```

Example: Scanning default IIS web directories using the base ruleset `> .\yara64.exe -r -C base.yara.bin C:\inetpub\wwwroot\`

Example: Scanning default Apache web directories using the extended ruleset `$ yara -r extended.yara.bin /var/www/html/`

Detecting network artifacts of common web shell malware

Web shells frequently use encryption and encoding to evade network-based detection. Additionally, "hard-coded" parameters can easily be modified to further evade signatures. However, if network architectures allow for TLS inspection for web servers (e.g., through a reverse proxy or web application firewall), then administrators may be able to detect unmodified or slightly modified common web shell malware at the network level. Network signatures are, at best, only partially effective at detecting web shells and should be used as one part of a defense-in-depth strategy.

Network signatures for common web shell malware

The provided Snort signatures can be used to detect some common web shells that have not been modified to evade detection. Some intrusion detection/preventions systems and web application firewalls may already implement these or other web shell signatures. Organizations are encouraged to understand their existing network signature posture in this regard.

Detecting unexpected network flows

In some cases, web shell traffic will cause unexpected network flows. For example, when a web shell infected web server is being used proxy requests into a network, the web server would make web requests to internal network nodes. This behavior is abnormal and could be easily recognized in network activity logs. Another example would be for a non-web server node (e.g., a network device) to suddenly be replying to responding to a web requests from outside the network. Administrators should have an intimate network knowledge and understand which protocols that nodes should be using. Contradictions to these norms should be investigated.

Snort signatures to detect unexpected network flows

The provided Snort signature can aid administrators in identifying unexpected network flows. This signature simply alerts when a node in the targeted subnet responds to an HTTP(s) request. This signature should be tailored for a specific network by targeting only non-web server nodes.

```
alert tcp 192.168.1.0/24 [443,80] -> any any (msg: "potential unexpected web server"; sid 4000921)
```

Note: ensure that the subnet above (192.l68.1.0) is changed to something appropriate for the target subnet

Endpoint Detection and Response (EDR) capabilities

Some EDR and enhanced logging solutions may be able to detect web shells based on system call or process lineage abnormalities. These security products monitor each process on the endpoint including invoked system calls. Web shells usually cause the web server process to exhibit unusual behavior. For instance, it is uncommon for most benign web servers to launch the ipconfig utility, but this is a common reconnaissance technique enabled by web shells. EDRs and enhanced logging solutions will have different capabilities, so administrators are encouraged to understand the solutions available for their environment.

Detecting Web Shells in Windows with Sysmon

Microsoft Sysmon is a logging tool that enhances logging performed on a Windows system. Among other things, Sysmon logs information about how each process is created. This information is valuable for identifying anomalous behavior, such as in the case of malicious web shells. Sysmon can be obtained from Microsoft and must be installed on a system to begin enhanced logging. Ideally, Sysmon and other Windows logging should be mirrored to a central Security Information and Event Management (SIEM) server where it can be aggregated and queried.

The query below will report executables launched by an IIS web server process. In many cases, the web application will cause IIS to launch a process for entirely benign functionality. However, there are several

executables commonly used by attackers for reconnaissance purposes, which are unlikely to be used by a normal web application. Some of these executables are listed below.

PowerShell script to identify Sysmon entries for IIS

```
Get-WinEvent -FilterHashtable @{logname="Micorosft-Windows-Sysmon/Operational";id=1;} | Where {$_.message -like "*ParentImage: C:\Windows\System32\inetsrv\w3wp.exe*"} | %{$_.properties[4]} | Sort-Object -Property value -Unique
```

Windows executables commonly used by attackers and rarely launched by benign IIS applications

- arp.exe
- at.exe
- bitsadmin.exe
- certutil.exe
- cmd.exe
- dsget.exe
- dsquery.exe
- find.exe
- findstr.exe
- fsutil.exe
- hostname.exe
- ipconfig.exe
- nbstat.exe
- net.exe
- net1.exe
- netdom.exe
- netsh.exe
- netstat.exe
- nltest.exe
- nslookup.exe
- ntdsutil.exe
- pathping.exe
- ping.exe
- powershell.exe
- qprocess.exe
- query.exe
- qwinsta.exe
- reg.exe
- rundll32.exe
- sc.exe
- schtasks.exe
- systeminfo.exe

- tasklist.exe
- tracert.exe
- ver.exe
- vssadmin.exe
- wevtutil.exe
- whoami.exe
- wmic.exe
- wusa.exe

Detecting Web Shells in Linux with Auditd

Auditd is the userspace component of the Linux Auditing System. Auditd can provide users with insight into process creation logs. The information is valuable for identifying anomalous behavior, such as in the case of malicious web shells. Auditd is available in default repositories for many Linux distributions and must be installed and configured before it can log relevant web server process data. Ideally, auditd and other Linux logging should be mirrored to a central Security Information and Event Management (SIEM) server where it can be aggregated and queried.

The query below will report applications launched by an Apache web server process. In many cases, the web application will cause Apache to launch a process for entirely benign functionality. However, there are several applications commonly used by attackers for reconnaissance purposes, which are unlikely to be used by a normal web application. Some of these applications are listed below.

Configuring Auditd

1. *Determine the web server uid*: After installing auditd (i.e., using "apt -y install auditd"), determine the uid of the web server using: `apachectl -S` This will return apache details including the user id in a line such as: `User: name="www-data" id=33` Here the uid is "33"
2. *Add the following auditd rules (/etc/audit/rules.d/audit.rules) replacing "XX" with the uid identified above*: `-a always,exit -F arch=b32 -F uid=XX -S execve -k apacheexecve` `-a always,exit -F arch=b64 -F uid=XX -S execve -k apacheexecve`
3. *Restart auditd*: `service auditd restart`

Reviewing Auditd Log

1. *Identify applications launched by Apache with*: `cat /var/log/auditd/audit.* | grep "apacheexecve"` This will return the path to the launched application (see bolded path in the example output below): type=SYSCALL msg=audit(1581519503.841:47): arch=c000003e syscall=59 success=yes exit=0 a0=563e412cbbd8 a1=563e412cbb60 a2=563e412cbb78 a3=7f065d5e5810 items=2 ppid=15483 pid=15484 auid=4294967295 uid=33 gid=33 euid=33 suid=33 fsuid=33 egid=33 sgid=33 fsgid=33 tty=(none) ses=4294967295 comm="cat" **exe="/bin/cat"** key="apacheexecve"
2. *Analyze the results to determine if unusual applications are launched (see list below)*

3. *Detailed information, including call arguments, can be obtained using*: `/var/log/auditd/audit.* | grep "msg=audit(1581519503.841:47)"` Replace the value of msg=audit with the value returned in step #1

Linux applications commonly used by attackers and rarely launched by benign Apache applications

- cat
- crontab
- hostname
- ifconfig
- ip
- iptables
- ls
- netstat
- pwd
- route
- uname
- whoami

Preventing Web Shells

McAfee Host Intrusion Prevention System (HIPS) rules to lock down web directories

Web shells almost always rely on modifying existing web application files or creating new files in a web accessible directory. Using a file integrity monitoring system can block file changes to a specific directory or alert when changes occur. The provided McAfee Host Intrusion Prevention System rules can be tailored for a specific web application to block file system changes that web shell exploitation usually relies on. Other file integrity monitoring systems are likely able to accomplish similar blocking as well; consult documentation or seek vendor guidance for other such products.

Requirements

- McAfee Host Based Security System (HBSS)

Usage

HIPS File integrity rules can be added to HBSS through the ePolicy Orchestrator portal. Before adding the rules, administrators should tailor them the target environment. This is initially done by modifying the "Include" path to target one or more web applications. Then exceptions to the rule can be added on the signature page under the exception rule tab. Exceptions should include things like allowed file extensions of benign file types that the target web application is required to handle (e.g., ".pdf", ".jpg", ".png"). HIPS rules should be added as an Informational alert (Level 1) for logging purposes to help administrators identify possible exceptions to the rule. Once administrators are confident that exceptions have been added, the rule(s) should be changed to Level 4 to block file changes.

Related Content

Date	Press Release	Report
2021-07-01	NSA, Partners Release Cybersecurity Advisory on Brute Force Global Cyber Campaign	CSA: Russian GRU Conducting Global Brute Force Campaign to Compromise Enterprise and Cloud Environments
2021-05-07	NSA CISA, FBI, and the UK NCSC further expose Russian Intelligence Cyber Tactics	CSA: Further TTPs associated with SVR cyber actors
2021-04-15	n/a	CSA: Russian SVR Targets U.S. and Allied Networks

License

See LICENSE.

Contributing

See CONTRIBUTING.

Disclaimer

See DISCLAIMER.
