Global Information Assurance Certification Paper

Securing Jenkins CI Systems

GIAC (GCIA) Gold Certification

Author: Allen Jeng, ajeng@adobe.com
Advisor: Mohammed Haron
Accepted: March 31, 2016

Abstract

With over 100,000 active installations worldwide, Jenkins became the top choice for continuous integration and automation. A survey conducted by Cloudbees during the 2012 Jenkins Users Conference concluded that 83% of the respondents consider Jenkins to be mission critical. The November 2015 remotely exploitable Java deserialization vulnerability stresses the need to lock down and monitor Jenkins systems. Exploitation of this weakness enables hackers to gain access to critical assets such as source code that Jenkins manages. Enabling password security is the general recommendations for securing Jenkins. Unfortunately, this necessary security measure can easily be defeated with a packet sniffer because passwords are transmitted over the wire as clear text. This paper will look at ways to secure Jenkins system as well as the deployment of intrusion detection systems to monitor critical assets controlled by Jenkins CI systems.

1. Introduction

Continuous integration is a methodology where software developers merge code frequently to a centralized source code repository (Rouse, 2008). The automated build system then checks the code continuously allowing teams to find and fix problems early. The Jenkins Continuous Integration System is an open source tool widely adopted by development teams of all sectors from high-tech companies like Facebook and financial intuitions like Fidelity Investments to government entities such as United Kingdom's Government Digital Service (Gov.uk, n.d.). Knowing quickly that a check-in caused the source repository to fail is one of the biggest advantages of continuous integration. The developers can quickly fix the breakage instead of relying on nightly builds to identify the problem. This alleviates the problem where a quality assurance team is left without a good working build in the morning.

According to a survey conducted by InfoQ in 2014, over 70% of participants choose Jenkins over other Continuous Integration systems (Humble, 2014). 83% of respondents from a survey conducted by Cloudbees at their 2012 Jenkins Users Conference deem Jenkins mission critical to their development process (CloudBees, 2012). With over 100,000 active installations worldwide, Jenkins can become a lucrative target for hackers because it has access to critical company assets such as source code and test data (CloudBees, 2015a). This issue becomes even more critical when using Jenkins to deploy end results to production servers.

Jenkins security is often poorly setup because the developers or release engineers who install them are primarily concerned with just getting a working system. Quick searches on the Internet with regards to Jenkins security or setup results in one thing, turn on global security within the Jenkins' system. The book recommended by Jenkins' official site, "Jenkins The Definitive Guide" from O'Really, makes the same recommendation (Smart, 2011). The Java de-serialize vulnerabilities that build up much attention in November 2015 made it much more crucial to protect Jenkins systems from harm (Breen, 2015). Jenkins is not the only Java deserialization casualty. Michael Stepankin found a critical vulnerability in PayPal's business website that allowed attackers to execute arbitrary commands (Vaas, 2016). The same Java deserialization

Allen Jeng, ajeng@adobe.com

vulnerability affects both PayPal and Jenkins. This paper will present methodologies to lock down Jenkins while minimizing opportunities for hackers to do harm.

2. Jenkins Overview

2.1. Jenkins Client/Server Architecture

Jenkins has client and server mode built into the software, allowing scalability through one centralized place, referred to as the master. Although Jenkins can run in standalone mode, utilizing Jenkins's client-server mode will get the maximum benefit out of Jenkins. Jenkins systems scale with ease by adding clients through the node configuration section. The use of Jenkins master alleviates the duplicated efforts needed in securing additional Jenkins masters and software updates. If an organization enforces a password change every 90 days, plan for extra time and efforts required to update all standalone Jenkins instead of updating one master.

2.2. Jenkins Master and Nodes

Selecting the right operating system to host the Jenkins master depends on personal preference and available skill sets. Do not install Jenkins on an unfamiliar operating system because it is easy to miss important operating system hardening steps. Each flavor of the operating system has their strengths and weaknesses. Install the master on Windows operating system if ones' expertise is the Windows OS.

Linux is great for running Jenkins master because most flavors of Linux do not have steep hardware requirements. Some examples of lightweight Linux OSes are Ubuntu, CentOS, and RedHat server because they do not come with a graphic user interface (GUI). Ubuntu server has a minimum requirement of a 300MHz x86 processor with 192MB of system memory and 1 GB of disk space (Ubuntu, 2014). Microsoft Windows Server Core install is another interesting option because core install removes the GUI, thus, reducing attack surfaces (Microsoft, 2014).

Jenkins is installable on operating systems that support Java. Hence, the operating choices are not limited to Windows or Linux. One can install Jenkins master on Apple's Mac OS X if one is more familiar with that operating system.

2.3. Jenkins Nodes

One of the biggest advantages to Jenkins is the ability to add clients, referred to as nodes or slaves. Two popular methods for adding nodes are ssh and Java web start. Ssh to the node requires username and password, ssh private key, or certificates in the credentials section. As long as the firewall configured correctly, the master will automatically connect to the node. Java web start requires the node to download a jar file from the master and execute it from the command line. The system will attach itself to the master as a node. The minimum software requirement is Java runtime environment (JRE) version 7 or above (Kawaguchi, 2015).

2.4. The Power of Jenkins

There are over 1,100 plugins available for Jenkins, according to the index of downloadable plugins (Jenkins-ci.org, n.d.). What makes Jenkins powerful is the sheer number of plugins it has. In addition to plugins written by the original author, Kohsuke Kawaguchi, many plugins come from Jenkins open source community's contribution (Adamson, 2007). There is a plugin written for almost every known source control system out there. If the organization uses Jira, there's a plugin for that. A few Selenium plugins help those that run automated tests with Selenium. Support for FTP and ssh functionality is through plugins. There is even a plugin for notification through IRC or Jabber if one so chooses. Developers can write plugins if there are none that satisfies a particular need.

Scripting support is another powerful feature of Jenkins. Groovy script is the official scripting language embedded into core Jenkins. It allows the user to do anything Groovy script supports such as sending a more detailed email with the use of regular expression after a failed job.

The ability to execute shell and batch command within jobs is the fundamental functions of Jenkins. It allows the user to run any command imaginable through shell and batch, enabling users to write scripts to drive the desired process. The job step then watches for non-zero return code to detect a pass or failed process. It can also fail the process with the help of a plugin called Log Parser (Soto & Goren, 2015). This plugin parses the log with a set of user defined regular expression for pattern matching.

A set of REST API in forms of XML, JSON, and Python provide data access to jobs within Jenkins allowing users to pull data from all Jenkins Master that they wish to monitor. The REST API provides basic information such as job success and failure as well as the developer and the change list that triggered the job request.

Remote access of Jenkins through a Command Line Interface (CLI) allows an administrator to control Jenkins through shell or scripting. A build or job status querying are some of the tasks actionable through the CLI. The ultimate control CLI offers are Groovy Shell access. The Groovy Shell is both powerful and dangerous because it can perform anything a Groovy script is capable. Even though Jenkins CLI is a convenient tool for Jenkins administrators, one should consider disabling CLI due to the power it posses.

2.5. Jenkins Enterprise

CloudBees Inc. offers Jenkins Enterprise and Jenkins Operation Center as part of the paid version of Jenkins. Role-based access control, Single sign-on, enterprise security are some of the benefits of Jenkins enterprise's offerings (CloudBees, n.d.). Jenkins Operation Center is the dashboard and the single sign-on point across all Jenkins masters (CloudBees, 2015b). Once the user authenticates through the Jenkins Operation Center portal, they can gain access to all authorized resources without re-entering their passwords.

3. Vulnerabilities in Jenkins

3.1. Java Deserialization

Jenkins has its own share of vulnerabilities from cross site scripting to the recent zero-day Java deserialization vulnerability. CloudBees and the Jenkins communities are actively providing security fixes and zero-day workarounds discovered by people in the information security.

The vulnerability that gains much attention is the remotely exploitable Java zero-day that allows arbitrary code execution on Jenkins from an unauthenticated remote attacker (Beck, 2015). Serialization is a method for applications to send data over the

network after converting to binary format. On the receiving end, deserialization is when the application turns the transport data back to the application data. The vulnerability occurs during deserialization phase, allowing untrusted data to execute potentially dangerous code (Breen, 2015). Jenkins' CLI command line tool offers convenience for Jenkins administrators to execute Groovy code remotely. Stephen Blewitt was able to find the weakness and successfully send a proof of concept payload by exploiting the Java deserialization vulnerability (Breen, 2015). According to Rapid7, a Metasploit module was created to take advantage this vulnerability (Rapid7, n.d.).

Fortunately, CloudBees quickly issued a workaround immediately after the publicized zero-day vulnerability (Croy, 2015). The real bug was fixed a few days after.

3.2. Cross-Site Scripting

Cross-site scripting is one of the problems web applications face, Jenkins was no exception to this. One such vulnerability allows hijacking legitimate user's session through a crafted URL pointing to Jenkins (Kawaguchi, 2012).

With Jenkins security turned on, job folders which are called workspace can only be accessed with the granted permissions. Workspace is where Jenkins keeps source code, compile results, and test results. Jenkins archive builds logs in the web accessible area so that users can visit the URL tied to the job when a failure occurs. Jenkins artifacts allow extra files such as a zip of the binary or user-generated HTML placed in the same folder as build logs. CVE-2015-7536 allows a lower privilege user to create HTML that could result in cross-site scripting when other users visit this page. The vulnerability fix prevents script execution from the workspace or archived artifacts (Beck, 2015).

Cross-site request forgery (CSRF) is the impersonation of a privileged user while visiting a malicious site (SANS Institute, 2015c, p.217). Jenkins security has a built-in cross-site request forgery protection that's disabled by default. When enabled, this can cause difficulty when using Jenkins features such as remote API and some plugin functionality (Kawaguchi & Beck, 2015). With this protection enabled, Jenkins' CSRF protection still has bugs that allowed attackers to circumvent it with crafted POST requests (Beck, 2015).

3.3. Compromised administrator or privileged account

Jenkins uses a matrix-based security system that can grant access from a very granular level. Any user with job creation and configuration privileges has shell access to the nodes because job configuration allows an arbitrary command to run on the nodes. A compromised Jenkins administrator or privileged account is a recipe for disaster because the attacker now has control over all nodes attached to the master. The attacker can do anything the operating system user's privilege allows when using a compromised Jenkins account. Jenkins CLI has a feature called Groovy Shell. It allows remote access once authenticated with proper privileges. An account is not needed to download the Jenkins CLI app at http://YourJenkinsURL:8080/jnlpJars/jenkins-cli.jar. Once obtained, the attacker can execute arbitrary shell command desired through groovy. The command *"cat /etc/passwd".execute().text* will cause Groovy to run the shell command cat and dump the contents of /etc/passwd to the console. These vulnerabilities have dire consequences when exploited.

4. Securing the Jenkins CI System

4.1. Enable Jenkins security

Jenkins global security is the first line of defense in protecting the asset it controls. Core Jenkins supports four security realms: delegate to servlet container; Jenkins's own user database; LDAP; and Unix user/group database.

Web servers such as Apache Tomcat that wish to handle their own user authentication should choose the delegate to servlet container option. Knowledgeable web server administrators need to configure the realm for this option to work (Damay, 2009). Jenkins comes with built-in user database for basic authentication if there are no external systems available (Smart, 2011, p. 173). The "Unix user/group database" option uses Unix's PAM database to authenticate Jenkins users. This is useful for extending Jenkins users with Unix servers preconfigured with LDAP (Nemeth, Hein, & Snyder, 2006). The best authentication method is LDAP because most organizations that use it enforces a 90-day password change. Some organization's IT department will actively

monitor accounts for abuse. If the LDAP account gets compromised, an account lockout or password reset also protects the Jenkins system that benefits from it.

There are many types of security realms extended through Jenkins plugins. Active Directory, Central Authentication Service for single sign-on, Google login, and Kerberos SSO are some of the alternatives to the core Jenkins security realm.

The second part of selecting a security realm is authorization method. Matrix-based security allows user permissions configuration at a global level. Project-based matrix authorization strategy extends matrix-based security by allowing security on a per-job basis. This option is beneficial for restricting access to jobs on a per group or user basis.

Enable the slave to master access control prevents the node from asking the master to do harmful things. For example, the master needs to temporarily take control of a user's machine to do a specific job. Turn off this option if all nodes are under full control of the Jenkins master (Kawaguchi & Nord, 2015).

4.2. Enable SSL encryption

No matter how secure Jenkins security is, passwords get passed around as clear text without Secure Sockets Layer (SSL). Wireshark successfully captures the clear text password of "poorpassword" for user "pwned-admin" as shown in Figure 1.

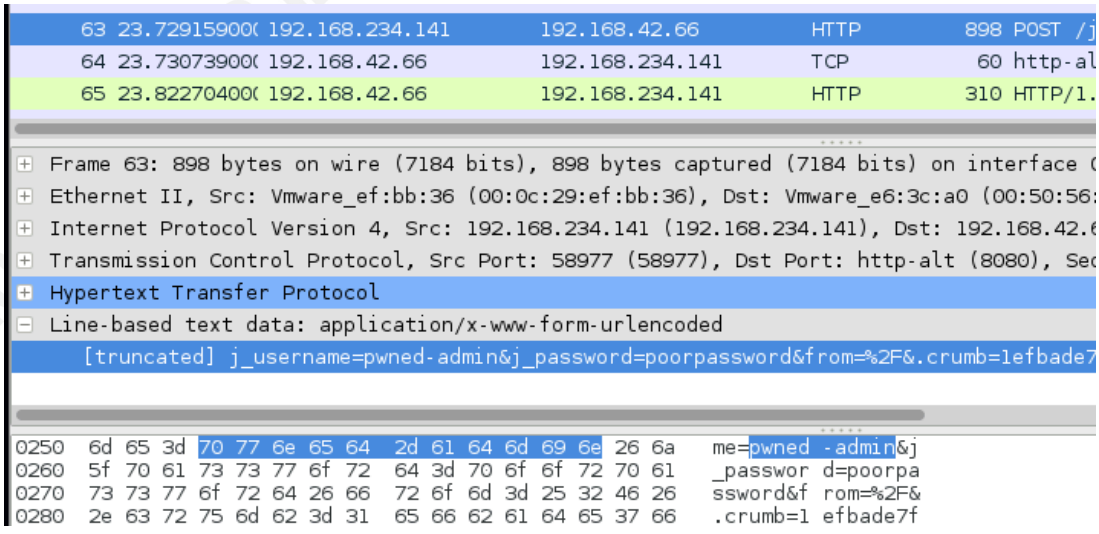

Figure 1: Wireshark packet capture of the clear text password

The remedy for passing around password in clear text is to use SSL certificate. It is easy to generate a self-signed SSL key with Java's keytool with the following command:

Keytool -genkey -alias jenkins -keyalg RSA -keystore jenkins.key -keysize 2048

Re-launching Jenkins with a self-signed SSL certificate in https mode now encrypts all traffic between the browser and the Jenkins server.

Self-signed SSL certificate is fine if Jenkins is running in standalone mode. Unfortunately, Java Web Start will fail to start on a node because it cannot validate server's self-signed SSL certificate. One can surely bypass this check with the use of *noCertificateCheck* parameter, but this is dangerous. All nodes that connect to the master will get assigned a unique 64-character hash key with security enabled. Disabling certificate check will cause the node to blindly connect to the master. An attacker can intercept the initiation process and pass itself as the master so that the node would exchange its 64-character secret key with the attacker. The attacker can impersonate itself as the new node with stolen secret key. If the "slave to master access control" is not enabled, the attacker can now send requests for the master to do, hence, causing harm. In addition to the security problem that the node faces, users who visit the Jenkins master URL are being reminded by the browser that the site is not trusted. For these reasons, one should get an SSL certificate from a trusted certificate authority. Trusted SSL certificate ensures the Jenkins master URL is a trusted site, and it also establishes trust between the nodes and the master.

4.3. Use a web server or a Winstone configuration file

There are many options to host Jenkins such as the embedded lightweight web server Winstone or standalone web servers such as Apache Tomcat, JBoss, or nGenx. The easiest way to run Jenkins is by executing *java -jar* from the command line (Basch & Griffiths, 2016).

java -jar jenkins.jar --httpPort=8080

Starting Jenkins with the above command is not the best way to run because any user who runs the command "ps aux | grep java" can see command line options used. On Windows, use task manager's details tab with command line column selected. Specifying

all parameters on the command line becomes a security risk because enabling SSL certificate requires exposing the keystore password in the parameter.

java -jar jenkins.jar --httpPort=-1 --httpsPort=8443 --httpsKeyStore=jenkins.key --httpsKeyStorePassword=exposedpassword

Winstone servlet container offers configuration file in place of the complicated command line parameters (Knowles, n.d.). For security purposes, storing KeyStore password in a configuration file means not exposing passwords on the command line. Remove all permissions to the configuration file for group members' permission and other users' permission so that only the user who runs Jenkins master has access. This is a non-issue for web servers because they already utilize configuration files.

4.4. Disable CLI

Java deserialization vulnerability is exploitable through the Jenkins CLI. If this API is not used, disabling it removes an avenue for attackers to use. According to Jenkins's official site, the best way to disable CLI is by placing a CLI shutdown script with the contents in figure 2 at $JENKINS_HOME/init.groovy.d (Cory, 2015).

```
import jenkins.*;
import jenkins.model.*;
import hudson.model.*;

// disabled CLI access over TCP listener (separate port)
def p = AgentProtocol.all()
p.each { x ->
  if (x.name.contains("CLI")) p.remove(x)
}

// disable CLI access over /cli URL
def removal = { lst ->
  lst.each { x -> if (x.getClass().name.contains("CLIAction"))
lst.remove(x) }
}
def j = Jenkins.instance;
removal(j.getExtensionList(RootAction.class))
removal(j.actions)
```

Figure 2: Code for shutting down Jenkins CLI (Cory, 2015)

Next, disable Jenkins's sshd daemon because every Jenkins have this feature on by default (Kawaguchi, 2011b). Kohsuke Kawaguchi claims that sshd daemon is a secure way to use Jenkins CLI without the extra jar. Jenkins will open random ephemeral ports to obfuscate the location of sshd daemon. Kawaguchi designed Jenkins's authentication by using the SSH public keys coupled to user's configuration (Kawaguchi, 2011a). Disabling features that are not in used are good security practices because the goal is to make it hard if not impossible for attackers to penetrate the Jenkins systems. If there is a need for these features, the recommendation is to configure the sshd daemon to a known port so that firewall rules can whitelist Jenkins administrator's IP (Kawaguchi, 2011b).

5. OS Hardening

Hardening the OS is an essential step because it reduces the attack surface and opportunities for hackers to get into the system. The Linux Security Checklist from SANS has a step by step guidance one can follow in securing an OS (Homsher & Evans, n.d.).

5.1. Never run Jenkins with root/administrator privileges

One should never run with administrator or root privileges regardless of the operating system platform. If a hacker manages to get in, he or she will still need to find ways to perform privilege escalation to cause more harm. Hopefully, this buys enough time for security professionals within the company to notice abnormal behavior and take action.

In addition to not running with administrator or root privileges, implement least privileges by removing sudoer access to the account that Jenkins uses (Mutch & Anderson, 2011, p.90). Jenkins master installed on Linux never need sudo access. On Windows, make sure Jenkins user only belongs to "Users" group. Mac OS X should run Jenkins as "Standard User" because they are not allowed sudo access by default. The same rules apply to Jenkins nodes because applications such as a compiler or automated

testing run fine with non-administrator privileges. The goal here is to minimize attack surfaces by taking away administrator privileges when access is not required in the first place. Implementing the principle of least privileges can reduce the damage caused by a compromised account by as much as 86% according to research data from Dr. Eric Cole (Cole, 2014).

5.2. Apply OS Patches

Regularly applying system patches and OS updates are crucial to security because OS vendors routinely fix vulnerabilities and bugs. Applying the latest OS updates will include security patches such as the ones Apple fixed for Mac OS X (SecurityWeek News, 2016). Both Windows and Mac OS X recommend enabling the automatic patching policy. If the only time critical Jenkins systems can receive maintenance is during off-peak hours, all operating systems can run system updates from the command line. On Ubuntu, *sudo apt-get update* and *sudo apt-get upgrade* will update the OS with the latest recommended patches. Mac OS X performs the update from running *sudo softwareupdate --install --all* from Terminal (Kessler, 2012). In addition to OS updates, updating Java is equally important because vulnerabilities in Java will put Jenkins at risk.

5.3. Disable Unnecessary Services

Disabling unnecessary services are next on the OS hardening list (Homsher & Evans, n.d.). Look for all running processes with *ps -ax* on Linux and Mac OS X or Task Manager on Windows. If the service is not essential to Jenkins or the OS, disable it. For example, Jenkins master and its nodes do not need a file sharing service to function properly.

5.4. Protect Sensitive Files

There are a few files on Jenkins master that should only be accessible by the account running Jenkins. If Jenkins uses Winstone, make sure winstone.properties file is accessible only by the account running Jenkins because it contains KeyStore password. Make sure the configuration files are properly locked down by granting access only to the owner if running under web servers such as Tomcat or JBoss. The other file that requires the same permission lockdown is the SSL certificate file. The last set of files resides in

the folder pointing to JENKINS_HOME because no users other than the one running Jenkins masters should have access to these. Jenkins's master key and salt they use to encrypt necessary information live within that folder.

5.5. Application whitelisting and Antivirus

The machine that hosts the Jenkins master and nodes are running on are usually forgotten after setup. Application whitelisting is a necessity for these systems because it accepts processes and applications from an approved list and deny everything else (Shackleford, 2009). Due to the nature of Jenkins master and their nodes, the application that runs on them are finite. This makes application whitelisting an excellent tool for protecting Jenkins.

Antivirus software is still essential in doing fundamental protection against known virus and malware. Having them installing and monitoring AV on Jenkins master and nodes are better than nothing.

5.6. Turn on host-based firewall

Turning on the host-based firewall on Jenkins servers and nodes is another layer of protection against attacks. The only system nodes should talk to are Jenkins master, source code servers, operating system's patch update servers, antivirus update server, and the IP addresses of administrators who does system maintenance. In addition to the set that nodes talk to, Jenkins masters have a wider audience because it needs to include the group of users that need access to the front end web application as well as the attached nodes. Jenkins web users are still a finite set of acceptable IP addresses to quarantine. By configuring the firewall to reject everything else will make it tougher for the adversary to penetrate Jenkins systems.

6. Vulnerability Analysis

6.1. Run vulnerability assessment

After performing the necessary steps to secure Jenkins and OS hardening, one needs to verify that Jenkins is properly locked down. A quick nmap scan will check for open ports on the master and nodes (Wieldman, 2014, p.125). The only open ports on the

master are SSH, SSL/HTTP, and a pair of random ephemeral port if Jenkins' sshd daemon is enabled. Nmap was able to identify Jetty Winstone-2.9 running on port 8443. Running nmap scans allows one to find unexpected listening ports missed during OS hardening.

Nikto was then used to make a quick vulnerability evaluation of Jenkins. By running the command *nikto -h your-jenkins-ip -p 8443*, Nikto was able to uncover CLI port and sshd port number (Wieldman, 2014, p.149).

```
+ Uncommon header 'x-jenkins-cli-port' found, with contents: 56509
+ Uncommon header 'x-jenkins-cli2-port' found, with contents: 56509
+ Uncommon header 'x-hudson-theme' found, with contents: default
+ Uncommon header 'x-ssh-endpoint' found, with contents: 192.168.42.79:58880
+ Uncommon header 'x-hudson' found, with contents: 1.395
+ Uncommon header 'x-jenkins-session' found, with contents: a4989993
+ The site uses SSL and the Strict-Transport-Security HTTP header is not defined.
+ No CGI Directories found (use '-C all' to force check all possible dirs)
```

Figure 3: Nikto identifying CLI and SSHD ports

By running nmap on the SSH port, it was able to identify that as Apache Mina sshd 0.8.0. If Jenkins CLI and sshd is not required, leaving them in the default running state allows an attacker to find them with Nikto.

OpenVAS is an excellent free vulnerability scanner for finding known vulnerabilities on systems running Jenkins (Pritchett & Smet, 2013, p.120-139). By scanning Jenkins master and its nodes, one will be able to check if known vulnerabilities exist. By identifying exposed open ports and unpatched vulnerabilities, one can assess if further OS hardening or Jenkins securing is necessary.

6.2. Analyze traffic to know what is normal

It is impossible to know what normal traffic looks like without analysis. One can theorize and make assumptions, but will never know until actual traffic is collected for further analysis. Following the operational lifecycle process is strongly recommended before deploying any traffic inspection tool because one does not want to blindly set up Network Intrusion Detection System (NIDS) before knowing what needs protecting (SANS Institute, 2015d, p.5). The goal of the planning stage is to determine where to deploy Jenkins, where to put the sensors, type of traffic inspection tool to deploy, and types of hardware needed to handle the speed of the network (SANS Institute, 2015d,

p.6-7). The use of switch's span port or taps requires the assistance of trained network IT engineer.

To collect data on routine traffic, attach a computer with Wireshark or tcpdump to span port and perform Jenkins operations that cause network traffic. Some examples of these actions are source code pulling, node connection via Java web start and SSH, nodes exdcuting shell calls such as dir/ls or script calls, and CLI if enabled. To get an idea of what is normal, use Wireshark's statistics function for further analysis (SANS Institute, 2015a, p.52). Protocol hierarchy will show statistics on the type of protocol passing through Jenkins and nodes. Be sure to check if all traffic flowing between the Jenkins clusters is SSL encrypted. Normal traffic coming from ARP and DNS traffic needs spot checking to make sure the DNS servers captured belong to the organization. Wireshark's conversation statistics should show traffic flow between the master, node, source control server and the administrator's system. If there is out-of-place traffic captured, one should analyze them further to ensure traffic seen is normal. If Jenkins master and nodes are the only systems attached to the switch, traffic coming into the switch should only be source code servers, administrator's machine, Jenkins users, and the organization's DNS servers. If the analyzed traffic is normal, backup the captured traffic to a write once media so that a baseline is available for comparison.

6.3. Setup and Deploy NIDS

With good baseline data, NIDS is ready for deployment. There are many resources on installing free NIDS such as *Snort 3.0 Beta 3 for Analysts* by Doug Burks. Customizing Snort signatures become an easier task thanks to the good baseline data and host-based firewall configuration (SANS Institute, 2015d, p.13). NIDS needs to pay close attention to traffic that is not from Jenkins master, nodes, source code server, and DNS. Nothing should communicate with the nodes other than a source code server, the master, and the administrator's system. NIDS should inspect traffic between Jenkins users and Jenkins master. NIDS should seriously consider decrypting Jenkins traffic because if one of the Jenkins user's account gets compromised, NIDS would want to know if there are abnormal traffic such as an attempt to create a new Jenkins job for reconnaissance or source code theft.

Allen Jeng, ajeng@adobe.com

There are two solutions to the encryption problem. The first is to run Jenkins on both non-encrypted channels as well as encrypted. The drawback to this solution is when users forget and enter their password through the non-encrypted port. The second solution is to use the Jenkins Operation Center because it offers single sign-on for all Jenkins masters. Jenkins Operation Center with SSL enabled will encrypt authentication traffic. After the user authenticates, Jenkins Operation Center re-directs the user to the non-encrypted master running their job. NIDS can easily monitor unencrypted traffic between the master and the nodes.

Auditing and refinement are equally important because NIDS is useless if it cannot detect malicious traffic. For effective auditing, the team of penetration testers should not have prior knowledge of the infrastructure (SANS Institute, 2015d, p.15-16). Knowledge of infrastructure can often steer a pen tester toward particular test and missed something a real attacker might do. After obtaining results from the pen testers, refine the rules accordingly based on the findings. Some NIDS such as Snort and Bro have built-in routines for measuring performance as part of the refinement step (SANS Institute, 2015d, p.17).

There are new software and rules updates for the NIDS that can cause changes. After the update is complete, repeat the cycle for customization, auditing and refinement (SANS Institute, 2015d, p.19). Keep in mind that NIDS is not an install and forget technology. It requires time and effort from security professionals to evaluate alerts generated by NIDS. Having a security information and event management (SIEM) will help correlation of indicators from NIDS, firewall, OS logs, routers, etc. to help the security professional to determine if a threat is real or if NIDS sensor needs adjusting (SANS Institute, 2015e, p.158).

7. Conclusion

Jenkins CI has gained popularity throughout different segments of the industry from software and hardware companies to government and healthcare industries (Orr & Prokop, 2016). With Jenkins in control of critical assets, it is essential to protect and secure Jenkins by enabling Jenkins security and SSL. It is vital to reduce attack surfaces

Allen Jeng, ajeng@adobe.com

by disabling Jenkins CLI and sshd when not used. Never forget to protect the OS that Jenkins is running on because attackers will find unpatched vulnerabilities in the operating system during their reconnaissance. Firewalls, application whitelisting and antivirus will make it more challenging for attackers to take control of Jenkins systems. Finally, potential threats against Jenkins require constant monitoring of alerts and logs from all systems that generate information. Most hackers today seek profit instead of fame (Carbon Black, 2016). If an organization have something valuable to steal, they will come. It is impossible to prevent hackers from penetrating companies, but actively protecting critical assets by doing due diligence with security and reacting to abnormal traffic through active monitoring will minimize the damage attackers can cause.

Allen Jeng, ajeng@adobe.com

References

Adamson, C. (2007, June 11). *Top 50: Interview with Kohsuke Kawaguchi of the Hudson Project blog*. Retrieved February 16, 2016, from https://community.oracle.com/docs/DOC-983718

Basch, D., & Griffiths, T. (2016, January 27). *Starting and accessing Jenkins*. Retrieved February 20, 2016, from Jenkins Wiki: https://wiki.jenkins-ci.org/display/JENKINS/Starting+and+Accessing+Jenkins

Beck, D. (2015, November 19). Jenkins Best Practices. Retrieved February 17, 2016, from Jenkins Wiki: https://wiki.jenkins-ci.org/display/JENKINS/Jenkins+Best+Practices

Beck, D. (2015, November 18). *Jenkins security advisory 2015-11-11*. Retrieved February 16, 2016, from Jenkins Wiki: https://wiki.jenkins-ci.org/display/SECURITY/Jenkins+Security+Advisory+2015-11-11

Beck, D. (2015, December 9). *Jenkins security adversary 2015-12-09*. Retrieved February 16, 2016, from Jenkins Wiki: https://wiki.jenkins-ci.org/display/SECURITY/Jenkins+Security+Advisory+2015-12-09

Blewitt, A. (2015, November 7). *Remotely exploitable Java zero day exploits through deserialization*. Retrieved from http://www.infoq.com/news/2015/11/commons-exploit

Breen, S. (2015, November 6). *What do WebLogic, WebSphere, JBoss, Jenkins, OpenNMS, and your application have in common? This vulnerability.* Retrieved from http://foxglovesecurity.com/2015/11/06/what-do-weblogic-websphere-jboss-jenkins-opennms-and-your-application-have-in-common-this-vulnerability/

Burks, D. (2009, April 17). *Snort 3.0 Beta 3 for Analysts* [White paper]. Retrieved from SANS reading room: https://www.sans.org/reading-room/whitepapers/detection/snort-3-0-beta-3-for-analysts-33068

Carbon Black. (2016). *Breach detection: what you need to know* [PDF document]. Retrieved from https://www.carbonblack.com/wp-content/uploads/2015/12/2016_Cb_ebook_breach_detection_v2.pdf

CloudBees, Inc. (n.d.). *CloudBees Jenkins Platform*. Retrieved from https://www.cloudbees.com/products/cloudbees-jenkins-platform#compare

Allen Jeng, ajeng@adobe.com

CloudBees, Inc. (2012, October). *Jenkins Community Survey Results* [PDF document]. Retrieved from https://www.cloudbees.com/blog/2012-jenkins-survey-results-are

CloudBees, Inc. (2013, January 8). *Survey shows Jenkins is mission-critical platform for application development.* Retrieved from https://www.cloudbees.com/press/survey-shows-jenkins-mission-critical-platform-application-development

CloudBees, Inc. (2015a, February 26). *Jenkins Community Reaches Milestone: More Than 100,000 Active Installations Worldwide*. Retrieved from https://www.cloudbees.com/press/jenkins-community-reaches-milestone-more-100000-active-installations-worldwide

CloudBees, Inc. (2015b). *CloudBees Jenkins Operation Center* [PDF document]. Retrieved from http://pages.cloudbees.com/rs/cloudbees/images/Jenkins-Operations-Center-by-CloudBees.pdf

Cole, E. (2014, August). *Insider threats in law enforcement* [White paper]. Retrieved from SANS reading room: http://www.sans.org/readingroom/whitepapers/threats/insider-threats-law-enforcement-35402

Croy, T. (2015, November 6). *Mitigating unauthenticated remote code execution 0-day in Jenkins CLI*. Retrieved from https://jenkins-ci.org/blog/2015/11/06/mitigating-unauthenticated-remote-code-execution-0-day-in-jenkins-cli/

Damay, J. (2009, November 27). Delegate security to servlet container. Retrieved February 18, 2016, from Jenkins Wiki: https://wiki.jenkins-ci.org/display/JENKINS/Delegate+security+to+servlet+container

Gov.uk. (n.d.). Colophon for GOV.UK at launch. Retrieved February 14, 2016, from https://gds.blog.gov.uk/govuk-launch-colophon/

Homsher, L. & Evans, T. (n.d.). *Linux Security Checklist* [White paper]. Retrieved from SANS: https://www.sans.org/media/score/checklists/linuxchecklist.pdf

Humble, C. (2014, March 4). *What CI server do you use?* Retrieved from http://www.infoq.com/research/ci-server

Allen Jeng, ajeng@adobe.com

Jenkins-ci.org. (n.d.). Index of /download/plugins. Retrieved February 20, 2016, from https://updates.jenkins-ci.org/download/plugins/

Kawaguchi, K. (2011a, June 23). *Public key authentication in Jenkins CLI*. Retrieved from https://www.cloudbees.com/blog/public-key-authentication-jenkins-cli

Kawaguchi, K. (2011b, December 27). *Jenkins now acts as an SSH daemon*. Retrieved from http://kohsuke.org/2011/12/27/jenkins-now-acts-as-an-ssh-daemon/

Kawaguchi, K. (2012, September 17). *Jenkins security advisory 2012-09-17*. Retrieved February 16, 2016, from Jenkins Wiki: https://wiki.jenkins-ci.org/display/SECURITY/Jenkins+Security+Advisory+2012-09-17

Kawaguchi, K. (2015, April 6). Good bye Java6. Retrieved from https://jenkins-ci.org/blog/2015/04/06/good-bye-java6/

Kawaguchi, K. & Nord, J. (2015, July 6). Slave to master access control. Retrieved February 17, 2016, from Jenkins Wiki: https://wiki.jenkins-ci.org/display/JENKINS/Slave+To+Master+Access+Control/

Kawaguchi, K. & Beck, D. (2015, December 7). *Securing Jenkins*. Retrieved February 16, 2016, from Jenkins Wiki: https://wiki.jenkins-ci.org/display/JENKINS/Securing+Jenkins

Kessler, T. (2012, September 19). *How to apply OS X software updates from the command line*. Retrieved from CNET: http://www.cnet.com/news/how-to-apply-os-x-software-updates-from-the-command-line/

Knowles, R. (n.d.). Winstone Servlet Container. Retrieved from http://winstone.sourceforge.net/

Krill, P. (2012, September 17). *Four vulnerabilities, including two affecting the Jenkins core and one deemed critical, have been identified*. Retrieved from http://www.infoworld.com/article/2614952/security/jenkins-integration-server-suffers-security-vulnerabilities.html

Microsoft. (n.d.). *Searching, downloading, and installing updates*. Retrieved from Microsoft MSDN: https://msdn.microsoft.com/en-us/library/aa387102%28VS.85%29.aspx

Microsoft. (2014, November 12). *Windows Server installation options*. Retrieved from Microsoft TechNet: https://technet.microsoft.com/en-us/library/hh831786.aspx

Allen Jeng, ajeng@adobe.com

Mutch, J., & Anderson, B. (2011). *Preventing good people from doing bad things: Implementing least privilege*. Berkeley, Calif.: Apress. Available from http://techbus.safaribooksonline.com/book/operations/9781430239215

Nemeth, E., Hein, T. R., & Snyder, G. (2006). *Linux Administration Handbook, Second Edition*. Upper Saddle River, NJ: Prentice Hall. Available from http://techbus.safaribooksonline.com/book/operating-systems-and-server-administration/linux/9780137002900/sharing-system-files/525?query=%28%28Pluggable+Authentication+Modules%3aThe+Definitive+Guide+to+PAM+for+Linux+SysAdmins+and+C+Developers%29%29#X2ludGVybmFsX0J2ZGVwRmxhc2hSZWFkZXI/eG1saWQ9OTc4MDEzNzAwMjkwMC81MjU=

Oracle. (n.d.). keytool-Key and Certificate Management Tool. Retrieved from http://docs.oracle.com/javase/6/docs/technotes/tools/solaris/keytool.html

Orr, C., & Prokop, M. (2016, January 13). Who is using Jenkins? Retrieved February 14, 2016, from Jenkins Wiki: https://wiki.jenkins-ci.org/pages/viewpage.action?pageId=58001258

Perforce Software. (2015). *Continuous Delivery: The New Normal for Software Development* [PDF document]. Retrieved from https://www.perforce.com/pdf/continuous-delivery-report.pdf

Pritchett, W. & Smet, D. (2013). *Kali Linux Cookbook*. Birmingham, UK: Packt Publishing Ltd.

Rouse, M. (2008, July). *Continuous Integration (CI)*. Retrieved from http://searchsoftwarequality.techtarget.com/definition/continuous-integration

SANS Institute. (2015a). *503.1 Fundamentals of Traffic Analysis: Part I*. Bethesda, MD: SANS Institute.

SANS Institute. (2015b). *503.2 Fundamentals of Traffic Analysis: Part II*. Bethesda, MD: SANS Institute.

SANS Institute. (2015c). *503.3 Application Protocol and Traffic Analysis*. Bethesda, MD: SANS Institute.

Allen Jeng, ajeng@adobe.com

SANS Institute. (2015d). *503.4 Open Source IDS: Snort and Bro*. Bethesda, MD: SANS Institute.

SANS Institute. (2015e). *503.5 Network Traffic Forensics and Monitoring*. Bethesda, MD: SANS Institute.

Shackleford, D. (2009 October). *Application whitelisting: enhancing host security* [White paper]. Retrieved from SANS reading room: https://www.sans.org/reading-room/whitepapers/analyst/application-whitelisting-enhancing-host-security-34820

Smart, J. F. (2011). *Jenkins: The definitive guide*. Sebastopol, CA: O'Reilly. Available from http://www.wakaleo.com/books/jenkins-the-definitive-guide

Rapid7 (n.d.). *CVE-2015-8103 Jenkins CLI RMI Java deserialization vulnerability*. Retrieved from https://www.rapid7.com/db/modules/exploit/linux/misc/jenkins_java_deserialize

SecurityWeek News. (2016, January 20). *Apple patches multiple vulnerabilities in iOS, OS X*. Retrieved from http://www.securityweek.com/apple-patches-multiple-vulnerabilities-ios-os-x

Soto, M., Goren, R. (2015, October 20). Log Parser Plugin. Retrieved February 14, 2016, from Jenkins Wiki: https://wiki.jenkins-ci.org/display/JENKINS/Log+Parser+Plugin

Ubuntu (2014, October 30). Ubuntu Installation/System requirements. Retrieved February 25, 2016, from Ubuntu wiki: https://help.ubuntu.com/community/Installation/SystemRequirements

Vaas, L. (2016, January 27). *Critical Java bug found in PayPal servers*. Retrieved from https://nakedsecurity.sophos.com/2016/01/27/critical-java-bug-found-in-paypal-servers/

Wieldman, G. (2014). *Penetration Testing*. San Francisco, CA: No Starch Press.

Allen Jeng, ajeng@adobe.com