Challenge 1

Overview

The organization allows applications with high severity vulnerabilities to be released, but nothing more severe than high. These three vulnerabilities are considered critical. Resolve the vulnerabilities through implementing changes to the file and once the image is clean, the challenge is complete.

Note: Removal of these dependencies is not considered a solution, while it would work it is not the objective of this challenge.

CVE-2018-18074

Overview

- The Requests package before 2.20.0 for Python sends an HTTP Authorization header to an http URI upon receiving a same-hostname https-to-http redirect, which makes it easier for remote attackers to discover credentials by sniffing the network.
- Impact: Base Score - CVSS v3.0 Score - 9.8 - CRITICAL
- Affected up to 2.20 (excluding)

CVE-2019-8457

Overview

- SQLite3 from 3.6.0 to and including 3.27.2 is vulnerable to heap out-of-bound read in the rtreenode() function when handling invalid rtree tables.

Version and Impact

- Impact: Base Score - CVSS v3.0 Score -9.8 -**CRITICAL**
- Affected up to 3.27.2 (including)

CVE-2018-12699

Overview

- finish_stab in stabs.c in GNU Binutils 2.30 allows attackers to cause a denial of service (heap-based buffer overflow) or possibly have unspecified other impact, as demonstrated by an out-of-bounds write of 8 bytes. This can occur during execution of objdump.

Version and Impact

- Impact: Base Score - CVSS v3.0 Score - 9.8 -**CRITICAL**
- Affected version 2.30

Bryce's Solution

Mitigation(s) for CVE-18074

Upgraded version to 2.22.0 (latest)

Mitigation(s) for CVE-2019-8457

Verified bin-utils is vulnerable in image via build output:

```
Installing sqlite-libs (3.25.3-r1)
```

- Mitigated by upgrading alpine image to 3.10.1 (latest)

Output after upgrade:

```
Installing sqlite-libs (3.28.0-r0)
```

Mitigation(s) for CVE-2018-12699

Verified bin-utils is vulnerable in image via build output:

```
Installing binutils-libs (2.30-r1)
```

- Mitigated by upgrading alpine image to 3.10.1 (latest)

Output after upgrade:

```
(8/35) Installing binutils (2.32-r0)
```

Additional Notes

1. Docker Alpine version 3.7 CVE-2019-5021 which affects 3.7 - https://alpinelinux.org/posts/Docker-image-vulnerability-CVE-2019-5021.html. Side note - if version cant be updated, possible mitigation for disabling root login:

```
RUN sed -i -e 's/^root::/root:!:/' /etc/shadow
```

2. Changed directory application ran from.
3. Could also look at potential running as non-ROOT user and deleting all unncessary files from container.

Challenge 2

Overview

The goal of this challenge is to assess how you resolve vulnerable dependencies within a maven project.

The repo contains a maven manifest file and a file containing the results of a Software Composition Analysis scan.

The organization allows applications with medium severity vulnerabilities to be released however this project contains 31 vulnerabilties considered high severity so in it's current state is not accepted as a release candidate. Resolve the vulnerabilities through implementing changes to the pom file and once the image complies to the oranization's risk apetite, the challenge is complete.

Bryce's Solution

1. Vulnerabilities found with Spring-Bootstarter (spring-boot-starter-web) version 1.1.1.RELEASE. Modified dependency:

```
 <dependency>
     <groupId>org.springframework.boot</groupId>
     <artifactId>spring-boot-starter-web</artifactId>
     <version>2.1.6.RELEASE</version>
</dependency>
```

2. Vulnerability found with Jackson (jackson-databind) verison 2.7.9.4. Modified dependency:

```
 <dependency>
         <groupId>com.fasterxml.jackson.core</groupId>
         <artifactId>jackson-databind</artifactId>
         <version>2.9.9.1</version>
</dependency>
```

Additional Notes

Also added mvn-dependency-check - https://jeremylong.github.io/DependencyCheck/dependency-check-maven/ to pom.xml to analyze dependencies and fail build on any vulnerabilites greater than or equal to 7.

```
<plugin>
              <groupId>org.owasp</groupId>
              <artifactId>dependency-check-maven</artifactId>
              <version>5.2.0</version>
              <configuration>
                  <failBuildOnCVSS>7</failBuildOnCVSS>
              </configuration>
              <executions>
                  <execution>
                      <goals>
                          <goal>check</goal>
                      </goals>
                  </execution>
              </executions>
            </plugin>
```

To run, enter command:

```
mvn verify
```

Output from tool on unmodified pom.xml:

```

[INFO] Analysis Started
[INFO] Finished Archive Analyzer (0 seconds)
[INFO] Finished File Name Analyzer (0 seconds)
[INFO] Finished Jar Analyzer (0 seconds)
[INFO] Finished Dependency Merging Analyzer (0 seconds)
[INFO] Finished Version Filter Analyzer (0 seconds)
[INFO] Finished Hint Analyzer (0 seconds)
[INFO] Created CPE Index (1 seconds)
[INFO] Finished CPE Analyzer (2 seconds)
[INFO] Finished False Positive Analyzer (0 seconds)
[INFO] Finished NVD CVE Analyzer (0 seconds)
[INFO] Finished Sonatype OSS Index Analyzer (1 seconds)
[INFO] Finished Vulnerability Suppression Analyzer (0 seconds)
[INFO] Finished Dependency Bundling Analyzer (0 seconds)
[INFO] Analysis Complete (4 seconds)
[WARNING]

One or more dependencies were identified with known vulnerabilities in siwilkins:

jackson-databind-2.7.9.4.jar (pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.7.
spring-boot-starter-1.1.1.RELEASE.jar (pkg:maven/org.springframework.boot/spring-boot-st
spring-boot-1.1.1.RELEASE.jar (pkg:maven/org.springframework.boot/spring-boot@1.1.1.RELE
logback-core-1.1.2.jar (pkg:maven/ch.qos.logback/logback-core@1.1.2, cpe:2.3:a:logback:l
tomcat-embed-core-7.0.54.jar (pkg:maven/org.apache.tomcat.embed/tomcat-embed-core@7.0.54
tomcat-embed-el-7.0.54.jar (pkg:maven/org.apache.tomcat.embed/tomcat-embed-el@7.0.54) :
```

```
tomcat-embed-el-7.0.54.jar (pkg:maven/org.apache.tomcat.embed/tomcat-embed-el@7.0.54) :
hibernate-validator-5.0.3.Final.jar (pkg:maven/org.hibernate/hibernate-validator@5.0.3.F
spring-core-4.0.5.RELEASE.jar (pkg:maven/org.springframework/spring-core@4.0.5.RELEASE,
spring-web-4.0.5.RELEASE.jar (pkg:maven/org.springframework/spring-web@4.0.5.RELEASE, cp
spring-aop-4.0.5.RELEASE.jar (pkg:maven/org.springframework/spring-aop@4.0.5.RELEASE, cp
spring-webmvc-4.0.5.RELEASE.jar (pkg:maven/org.springframework/spring-webmvc@4.0.5.RELEA


See the dependency-check report for more details.


[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  8.183 s
[INFO] Finished at: 2019-07-31T22:09:26-06:00
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.owasp:dependency-check-maven:5.2.0:check (default) or
[ERROR]
[ERROR] One or more dependencies were identified with vulnerabilities that have a CVSS s
[ERROR]
[ERROR] jackson-databind-2.7.9.4.jar: CVE-2017-17485, CVE-2018-5968, CVE-2017-15095, CVE
[ERROR] spring-boot-starter-1.1.1.RELEASE.jar: CVE-2017-8046
[ERROR] spring-boot-1.1.1.RELEASE.jar: CVE-2017-8046
[ERROR] logback-core-1.1.2.jar: CVE-2017-5929
[ERROR] tomcat-embed-core-7.0.54.jar: CVE-2016-3092, CVE-2016-5018, CVE-2016-8745, CVE-2
[ERROR] spring-core-4.0.5.RELEASE.jar: CVE-2015-5211, CVE-2018-1272, CVE-2016-5007, CVE-
[ERROR] spring-web-4.0.5.RELEASE.jar: CVE-2015-5211, CVE-2018-1272, CVE-2018-1270
[ERROR] spring-aop-4.0.5.RELEASE.jar: CVE-2018-1272, CVE-2018-1270
[ERROR] spring-webmvc-4.0.5.RELEASE.jar: CVE-2015-5211, CVE-2018-1272, CVE-2018-1270
[ERROR]
[ERROR] See the dependency-check report for more details.
[ERROR]
[ERROR]
[ERROR] -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the fo
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
```

Output from modified pom.xml:

```
[INFO] Analysis Started
[INFO] Finished Archive Analyzer (0 seconds)
[INFO] Finished File Name Analyzer (0 seconds)
[INFO] Finished Jar Analyzer (0 seconds)
[INFO] Finished Dependency Merging Analyzer (0 seconds)
[INFO] Finished Version Filter Analyzer (0 seconds)
[INFO] Finished Hint Analyzer (0 seconds)
[INFO] Created CPE Index (1 seconds)
[INFO] Finished CPE Analyzer (2 seconds)
[INFO] Finished False Positive Analyzer (0 seconds)
[INFO] Finished NVD CVE Analyzer (0 seconds)
[INFO] Finished Sonatype OSS Index Analyzer (0 seconds)
[INFO] Finished Vulnerability Suppression Analyzer (0 seconds)
[INFO] Finished Dependency Bundling Analyzer (0 seconds)
[INFO] Analysis Complete (3 seconds)
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  6.804 s
[INFO] Finished at: 2019-07-31T22:07:11-06:00
[INFO] ------------------------------------------------------------------------
```

Challenge 3

Bryce's Solution

There are a couple issues with the provided statement in my opinion, but the most glaring / obvious would be the omission of a prepared statement for the SELECT query. Instead, the author has opted to use a string which could be easily escaped via:

```
' OR 1=1
```

Effectively escaping the userName entry and appending to the query statement an OR operand which could evaluate to true. A better solution would be to use a prepared statement where input is compiled beforehand and do not use the same formatting / protocol.

Prepared statements are resilient against SQL injection because values which are transmitted later using a different protocol are not compiled like the statement template. Example using parameter / prepared statement:

```
db->prepare("SELECT * FROM users where userid=?");
```

The second issue is purely poor programming practice with using base64 encoding for obfuscation /

"securing" of password information. If this state is executed without encryption over the network, it is simple to reverse the base64 encoding and identify the user's password.

Bryce Notes

1. **getBytes()** encodes a string and returns an array of bytes.
2. **encodeToString()** - Encodes the specified byte array into a String using the Base64 encoding scheme.

Challenge 4

Build a tool that can be used to automate identification of expired and soon-to-be expired HTTPS certificates associated with Overstock.com.

This challenge has a few components.

1. Identify which public subnets belong to Overstock.com.

2. Build a tool that can scan those subnets to identify hosts listening on port 443

3. Once those servers are identified, extract the HTTPS certificate and parse the validity information to see when it will expire.

4. Generate output that can be used to determine which certificates are (1) already expired or (2) will expire within the next year. This can be an email or a report, but it should be understandable and efficiently denote which certificates should be prioritized for update.

Overview

I will build a tool that generates a report enumerating expired or soon-to-be expired HTTPS certifcates for subdomains within Overstock. The enumerated host list will be generated via 3rd party too named Dr_robot - https://github.com/sandialabs/dr_robot. Dr. Robot aggregates host information from numerous sources such as ARIN, Shodan, Sublist3r, etc. Sources like Sublist3r are able to enumerate domains from data provided from major search engines (Google, Bing, etc.). It also enumerates subdomains from other sources such as Virustotal, ThreatCrowd, DNSdumpster, and ReverseDNS.

Once I've genearated a list of subdomains, I will validate TLS certificate expiration, version and type via OpenSSL for each domain's certificate (if presented) and create an easily readible output to be consumed by other clients.

Alternatively, instead of using a tool for generation of a domain and subdomain list, one could grab the publically available IPv4 range for a domain found at: https://whois.arin.net, and use nmap to programatically request :443 information from each IP within a range.

For instance, a quick search of *overstock.com returns the following IP ranges from ARIN:

- 173.241.144.0 - 173.241.159.255 - /20 ~4094 hosts
- 63.239.22.56 - 63.239.22.63 - /29 - ~7 hosts

- 65.116.112.0 - 65.116.119.255 - /21 ~2046 hosts
- 67.110.104.0 - 67.110.111.255 - /21 ~2046

nmaping ~10k hosts on port 443 only, running in concurrence, I would expect could be done within a couple hours (max). The only thing I am concerned with is overstock.com could choose to not publish certain IP ranges to ARIN. (This is what makes Dr Robot so powerful). Furthermore,

Once I have the NMAP results, I would use the same code as below to grab certificate information and parse it into a consumable format (format_host_data). Perhaps the only tricky part about this would be for subdomains located on the same IP. But I believe this could be mitigated with resources like Shodan and ARIN.

I can go into detail with this more on the call.

Automating identificaiton of expired and soon-to-be-expired HTTPS certificates

1. Downloaded dr_robot and configured tool to my enviornment.
2. Dr Robot both a list of ips, but also a list of domains.
3. With the list of domains in hand, my**format_host_data.py** script can request TLS certficiate expiration information via openssl. To speed the process up a bit more, I implemented threading.
4. Once TLS information is returned, I create an object until the program returns and I am left with an array of objects which I can export as JSON for consumption (final.json)

Example outputs

Example output of Dr.Robot IP list:

```
 air.travel.overstock.com
traveltrack.overstock.com
trk.overstock.com
tt11.overstock.com
um.overstock.com
univision.overstock.com
usedcars.overstock.com
vacations.overstock.com
vcsgw.overstock.com
```

Example output after parsing certificate informaiton:

```
[
  {
    "host": "www.overstock.com",
    "version": 2,
    "sig_algorithm": "sha256WithRSAEncryption",
    "expiration": 1620842400.0
  },
  {
    "host": "3dview.overstock.com",
    "version": 2,
    "sig_algorithm": "sha256WithRSAEncryption",
    "expiration": 1591552800.0
  }
```

Bonus

If I have spare time:

1. I will create a webapp for viewing all 4 challenge results in MD format -**DONE**
2. Dockerize dr_robot and report generation for easy deployment anywhere / testability by Overstock.com management and technical teams - **Updated** INCOMPLETE - Ran out of time. See Dockerfile-unfinished for progress. I probably need about another two hours but seems like pipenv was not playign nicely with my Docker-In-Docker container.

To Run

1. Install docker
2. Install docker-compose
3. From within challenge 4 directory, run:

```
sudo docker-compose up --build
```

4. Navigate to localhost:8081 in your browser to view MD and Table at end of Challenge 4 MD doc with certificate expiration information.