To Beat Them, Join Them

Ethical Hacking in an Insecure World

River Valley High School Infocomm 2022

Zhang Zeyu

What is Hacking?

Making computer systems **do things they weren't intended to do**

What is Hacking?

Singtel data breached through hack on third-party file-sharing vendor

Singtel said that its core operations remain "unaffected and sound". ST PHOTO: KUA CHEE SIONG

SINGTEL.COM | CL0P^_- LEAKS

5dzrh6paatvyrzl7ry3zm72zigf4ad.onion/singtel-com

Headquarters 31 Exeter Rd, Singapore 239732

Area served Worldwide

Key people Chua Sock Koong (Group CEO)

Yuen Kuan Moon (Group CEO-designate)

Revenue $16.54 billion SGD (2020)

Owner Temasek Holdings

Number of employees >25,000

Website singtel.com

FILES PART1 EXTRACTED EMAILS+FILES

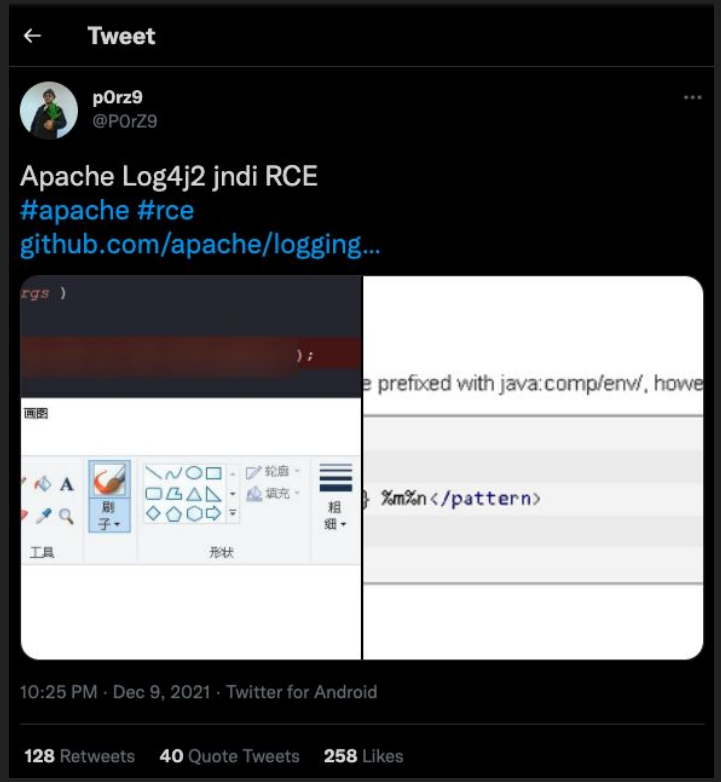
DOWNLOAD1

DOWNLOAD2

DOWNLOAD3

FILES PART2

What is Hacking?



What is it?

On Thursday, December 9th, a 0-day exploit in the popular Java logging library `log4j` (version 2) was discovered that results in Remote Code Execution (RCE), by logging a certain string.

Given how ubiquitous this library is, the impact of the exploit (full server control), and how easy it is to exploit, the impact of this vulnerability is quite severe. We're calling it "Log4Shell" for short.

The 0-day was tweeted along with a POC posted on GitHub. It has now been published as CVE-2021-44228.

This post provides resources to help you understand the vulnerability and how to mitigate it.

This blog post is also available at https://log4shell.com/

- "Zero-day" found by security researcher — new software bug with no available fix
- Depending on ethics, might be first responsibly disclosed to the software vendor/publisher for them to work on a patch
- Public Proof-of-Concept exploit code released

What is Hacking?

Did people already know about this vulnerability but just chose not to disclose it?

Not uncommon for state actors!

Knowledge is power when it comes to cyber warfare

What is Hacking?

Belgian Defense Ministry confirms cyberattack through Log4j exploitation

The Defense Ministry said it first discovered the attack on Thursday.

Written by **Jonathan Greig**, Staff Writer
on December 21, 2021 | Topic: Security

The Belgian Ministry of Defense has confirmed a cyberattack on its networks that involved the Log4j vulnerability.

ZDNet

Chinese regulators suspend Alibaba Cloud over failure to report Log4j vulnerability

Chinese media outlets have reported that Alibaba Cloud is facing backlash from government regulators after they reported the Log4J...

2 weeks ago

Each installation of a software is the exact same — it's like human DNA

Just like how a disease can infect each human the same way, the same exploit can take over each computer the same way

Knowledge of computer exploits therefore carries significant geopolitical ramifications

Ethical Hacking

- Best way to discover your application's vulnerabilities? **Hack it yourself first!**

- Ethical hackers are hired to perform authorized simulated cyberattacks, i.e. penetration tests

- Bug bounty programs are available for the public to report vulnerabilities in exchange for money

Reported August 18, 2021 2:16pm +0800

zeyu2001

Participants

State Triaged (Open)

Reported to [redacted]

Severity High (7 ~ 8.9)

Asset: Dom... [redacted]

Weakness Cross-Site Request Forgery (CSRF)

OK, but...

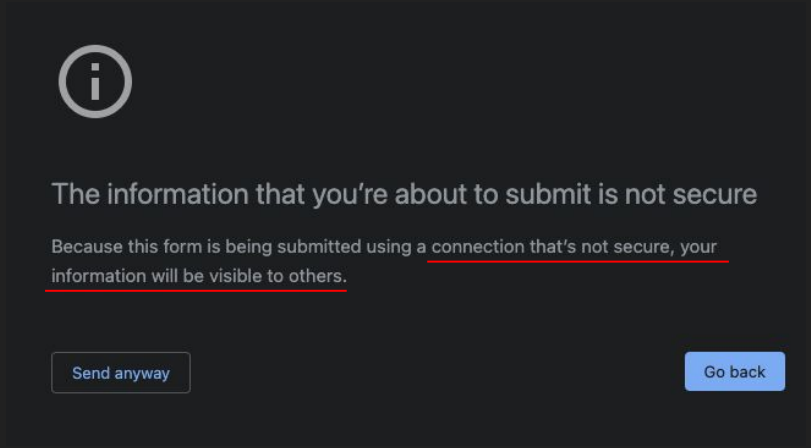
what does hacking *really* look like?

Public WiFi and HTTP — A Case Study

- Ever used free public WiFI before?
- If you’re not careful, anyone can steal your data!

Public WiFi and HTTP — A Case Study

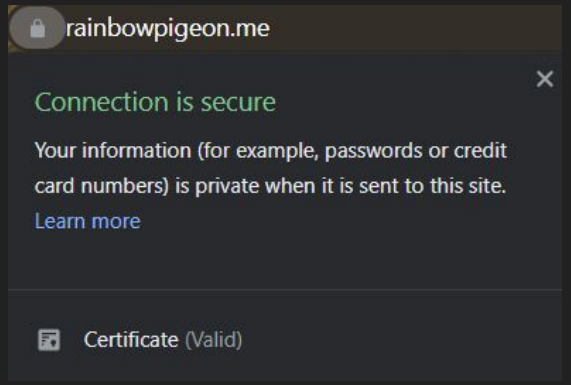
- I was trying to book my driving classes at BBDC and...



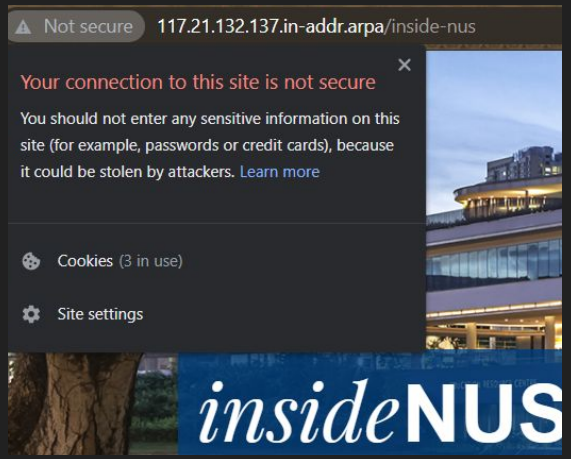
- The booking site is using an **unencrypted** communication protocol, and sending my data in plaintext!

Public WiFi and HTTP — A Case Study

- Number 1 rule when handling sensitive data over the internet: **use encryption!**
- HTTPS uses encryption, HTTP does not
- HTTPS? Check for the 🔒 next to the URL in your browser



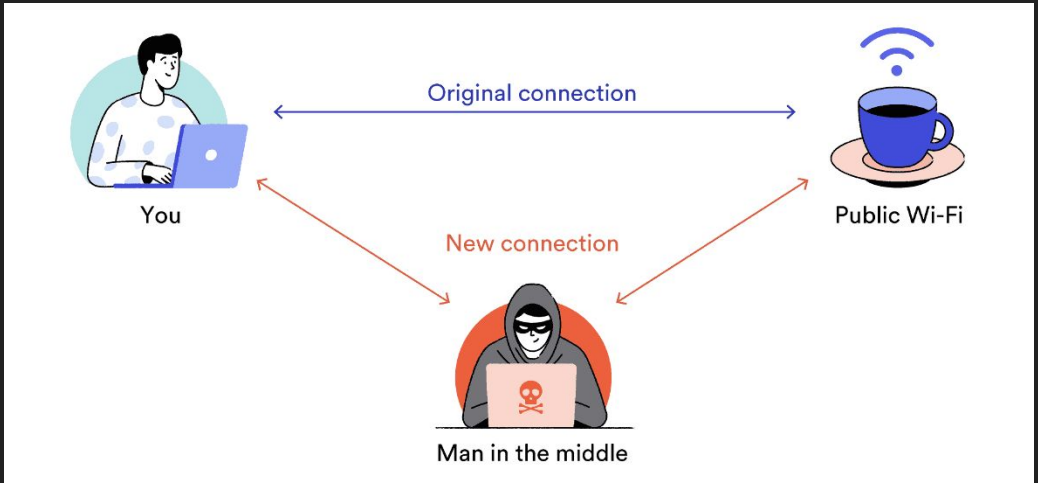
HTTPS



HTTP

Public WiFi and HTTP — A Case Study

- If you're using WiFi, it is **trivial** for anyone on the same WiFi network to execute a man-in-the-middle attack and **sniff all your internet traffic** 😨



Demo!

Hacking someone over WiFi with our custom-made tool

Public WiFi and HTTP — A Case Study

- A hacker would have been able to intercept any data I submitted on the BBDC website such as **passwords** and **personal data**

No.	Time	Source	Destination	Protocol	Length	Info
85	1.969642	192.168.1.32	203.127.7.4	HTTP	1147	GET /bbdc/bbdc_web/header2.asp HTTP/1.1
91	1.999556	203.127.7.4	192.168.1.32	HTTP	88	HTTP/1.1 200 OK (text/html)
1226	26.646618	192.168.1.32	203.127.7.4	HTTP	837	POST /bbdc/bbdc_web/header2.asp HTTP/1.1
1253	29.392711	203.127.7.4	192.168.1.32	HTTP	88	HTTP/1.1 200 OK (text/html)

```
> Frame 1226: 837 bytes on wire (6696 bits), 837 bytes captured (6696 bits) on interface en0, id 0
> Ethernet II, Src: Apple_0e:b3:d6 (88:66:5a:0e:b3:d6), Dst: NokiaSha_b6:3d:a0 (78:17:35:b6:3d:a0)
> Internet Protocol Version 4, Src: 192.168.1.32, Dst: 203.127.7.4
> Transmission Control Protocol, Src Port: 60149, Dst Port: 80, Seq: 2482, Ack: 1023, Len: 771
> Hypertext Transfer Protocol
v HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "txtNRIC" = "FakeUsername123"
  > Form item: "txtpassword" = "FakePassword123"
  > Form item: "ca" = "true"
  > Form item: "btnLogin" = "ACCESS TO BOOKING SYSTEM"
```

Public WiFi and HTTP — A Case Study

No.	Time	Source	Destination	Protocol	Length	Info
376	28.432704	203.127.7.4	192.168.1.32	HTTP	194	HTTP/1.1 200 OK (text/html)
389	28.451247	203.127.7.4	192.168.1.32	HTTP	87	HTTP/1.1 200 OK (text/html)
400	28.466134	192.168.1.32	203.127.7.4	HTTP	1159	GET /bbdc/inc-webpage/image/invis.gif HTTP/1.1
402	28.468655	203.127.7.4	192.168.1.32	HTTP	462	HTTP/1.1 200 OK (text/html)
410	28.483052	203.127.7.4	192.168.1.32	HTTP	289	HTTP/1.1 404 Not Found (text/html)
424	28.568748	192.168.1.32	203.127.7.4	HTTP	1136	GET /bbdc/image/await.gif HTTP/1.1
427	28.643763	192.168.1.32	203.127.7.4	HTTP	1101	GET /bbdc/admin/inc-script/chromeless.js HTTP/1.1
429	28.643940	192.168.1.32	203.127.7.4	HTTP	1159	GET /bbdc/image/site/logo-booking.gif HTTP/1.1
437	28.684983	203.127.7.4	192.168.1.32	HTTP	405	HTTP/1.1 200 OK (GIF89a)
453	28.692560	203.127.7.4	192.168.1.32	HTTP	292	HTTP/1.1 200 OK (GIF89a)
473	28.704333	203.127.7.4	192.168.1.32	HTTP	296	HTTP/1.1 200 OK (application/x-javascript)

```
  <td colspan="2" class="bluetxtbold" align="center">ZHANG ZEYU</td>\r\n
</tr>\r\n
<tr> \r\n
  <td colspan="2" class="txtbold" align="center"> </td>\r\n
</tr>    \r\n
<tr> \r\n
  <td width="50%" align="right" class="bluetxtbold">Account Id:</td>\r\n
  <td width="50%" class="bluetxt">[redacted]</td>\r\n
</tr>          \r\n
<tr> \r\n
  <td width="50%" align="right" class="bluetxtbold">NRIC
  <td width="50%" class="bluetxt">[redacted]</td>\r\n
</tr>\r\n
<tr> \r\n
  <td width="50%" align="right" class="bluetxtbold">Course:</td>\r\n
  <td width="50%" class="bluetxt">3A</td>\r\n
</tr>\r\n
<tr> \r\n
  <td width="50%" align="right" class="bluetxtbold">Last Login:</td>\r\n
```

Account ID

NRIC Number

Public WiFi and HTTP — A Case Study

What more can the attacker do? A lot, but here's a few examples:

- Log into my account — **no username and password required**
- Perform actions on my behalf
- Serve me a **malicious** version of the website, with malware or tracking code surreptitiously included

HTTPS = safe?

It depends!

How does the web work, anyway?

- You type in *www.google.com*, and you see Google's homepage
- Is that it?

A story of two builders

- Two main browsers - **Netscape** and **Internet Explorer**
- Browsers convert **Hypertext Markup Language** (**HTML**) documents to visible content
- Early on, there was no standard **HTML** structure
- A document that rendered fine in Netscape could turn into complete garbage on Internet Explorer

A story of two builders

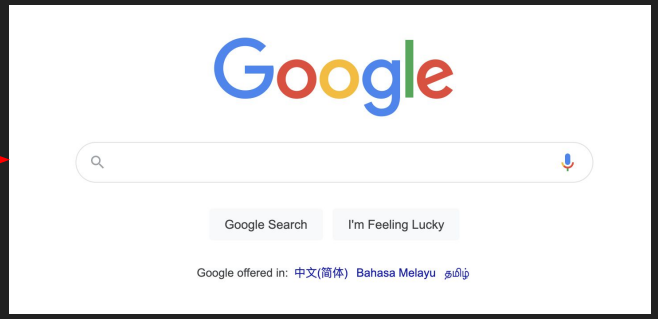
- To differentiate itself, Netscape added the <blink> element - content inside it would blink
- Internet Explorer never implemented support for this
- Over time, browser support for elements would have diverged
- Each browser would have implemented custom elements that the other would not support
- Similar to the imperial vs metric debate, but worse

A story of two builders

- A World Wide Web Consortium (W3C) was set up and created standards for web browsers to follow
- Now, a webpage renders the same, no matter which browser you choose
- Most browsers actually run on the same Chromium engine
- You can trust that your webpage will look the same anywhere

All within a few milliseconds...

- Browser requests for the *google.com* document from Google's server
- HTML document is received by your browser
- Document is interpreted and rendered by browser
- You see the webpage!

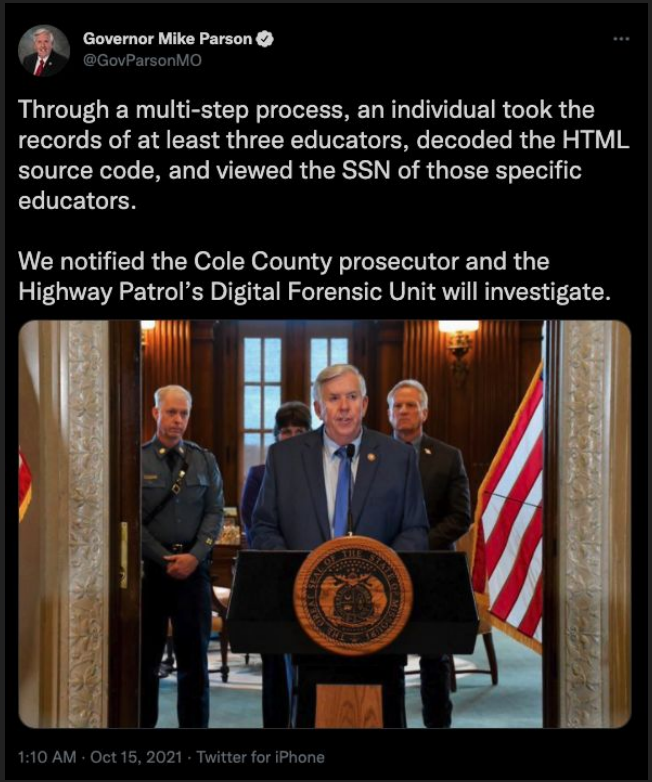


What information can the HTML tell us?

Do hackers look at HTML? **Yes.**

Is looking at the HTML illegal? **No.**

Do people often leave sensitive information in the HTML? **Evidently so**.



Do It Yourself: Viewing HTML

- https://learn.zeyu2001.com/
- Try viewing the original HTML of the page
- See if you can gain some interesting information!

The Workings of Web Servers

- Webpages are served to your computer by other computers elsewhere.
- These computers, like any other, have filesystems.
- Web routes often correspond to file structure on disk, i.e. *http://example.com/about/index.html* often corresponds to the *index.html* file in a folder called *about* on disk.

```
[tecmint@centos8 opt]$ sudo tree
.
├── apache-tomcat-9.0.26
│   ├── bin
│   │   ├── bootstrap.jar
│   │   ├── catalina.bat
│   │   ├── catalina.sh
│   │   ├── catalina-tasks.xml
│   │   ├── ciphers.bat
│   │   ├── ciphers.sh
│   │   ├── commons-daemon.jar
│   │   ├── commons-daemon-native.tar.gz
│   │   ├── configtest.bat
│   │   ├── configtest.sh
│   │   ├── daemon.sh
│   │   ├── digest.bat
│   │   ├── digest.sh
│   │   ├── makebase.bat
│   │   ├── makebase.sh
│   │   ├── setclasspath.bat
│   │   ├── setclasspath.sh
│   │   ├── shutdown.bat
│   │   ├── shutdown.sh
│   │   ├── startup.bat
│   │   ├── startup.sh
│   │   ├── tomcat-juli.jar
│   │   ├── tomcat-native.tar.gz
│   │   ├── tool-wrapper.bat
│   │   ├── tool-wrapper.sh
│   │   ├── version.bat
│   │   └── version.sh
│   ├── BUILDING.txt
│   ├── conf
│   │   ├── catalina.policy
│   │   ├── catalina.properties
│   │   ├── context.xml
│   │   ├── jaspic-providers.xml
│   │   ├── jaspic-providers.xsd
│   │   ├── logging.properties
│   │   ├── server.xml
│   │   ├── tomcat-users.xml
│   │   ├── tomcat-users.xsd
│   │   └── web.xml
│   ├── CONTRIBUTING.md
│   ├── lib
│   │   ├── annotations-api.jar
│   │   ├── catalina-ant.jar
│   │   ├── catalina-ha.jar
```

Web Request Types

- When you open a webpage in your browser, you are making a request
- Two main request types - *GET* and *POST*
- *GET* requests are the default - browser URLs will make *GET* requests
- *POST* requests are used for submitting data
- Both requests support passing data, often as key-value pairs, like a dictionary
- *GET* parameters are exposed in the address bar

https://www.google.com/search?client=firefox-b-d&q=how+to+improve+gpa

Can you tell what query I typed into Google?

localhost/remote_storage?file=../../../../../../../../users/jaredsong/Desktop/secret.txt

What about this?

We have LFI!

Local file inclusion, that is

Explanation and Demo

- You can navigate to any folder on your computer using a filepath!
- Try it!
  - Mac users, open Finder and press Cmd + Shift + G
  - Windows users, press Winkey + R
- Using "../" means going up one directory
  - "cd ../" when you're in C:/sample_directory will take you to C:
- If the remote server has LFI, you can view any file that the webserver user has access to.
  - View source code, hidden files, developer notes
  - Badly implemented web apps may even have databases or credential files in the same directory

Do It Yourself:

- What else did we tell the robots?
- Check out that URL path - it seems the developers are really lazy

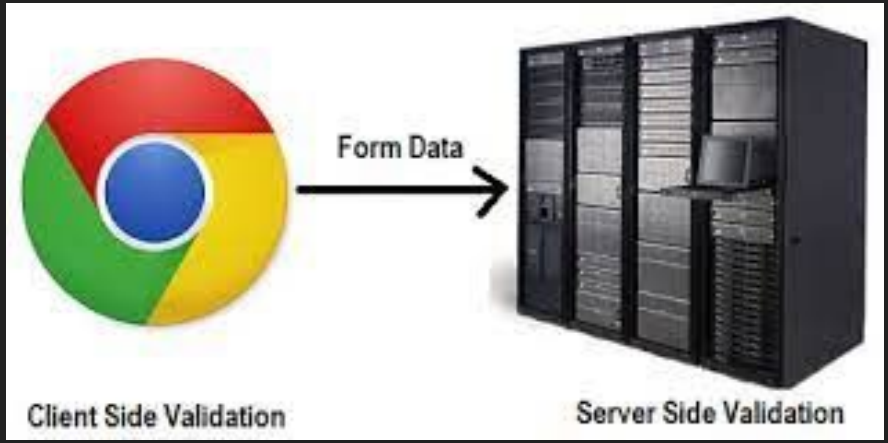
What Else Can You Find?

- With LFI, we can view anything on the web server
  - Developer notes
  - Application source code
  - Personal files
- This can reveal hidden information, like routes under development
- What can you find?

Client-Side Validation

- HTML attributes — maxlength="10", type="email", etc.
- JavaScript:

```
function validateForm() {
  let x = document.forms["myForm"]["fname"].value;
  if (x == "") {
      alert("Name must be filled out");
    return false;
  }
}
```



Client-Side Validation

- Great for user experience (faster than sending a HTTP request)
- Reduces server load
- But **must** be complemented by server-side validation as well!
- Always remember that HTTP requests are always attacker-controlled — the browser must **not** be trusted to secure user input

Client-side validation is *nice to have*, but server-side validation is a **must**.

Do It Yourself:

- Can you hack the GPA shop?