COMP644 Week 9)

A NOTE ON ETHICS / LEGALITY

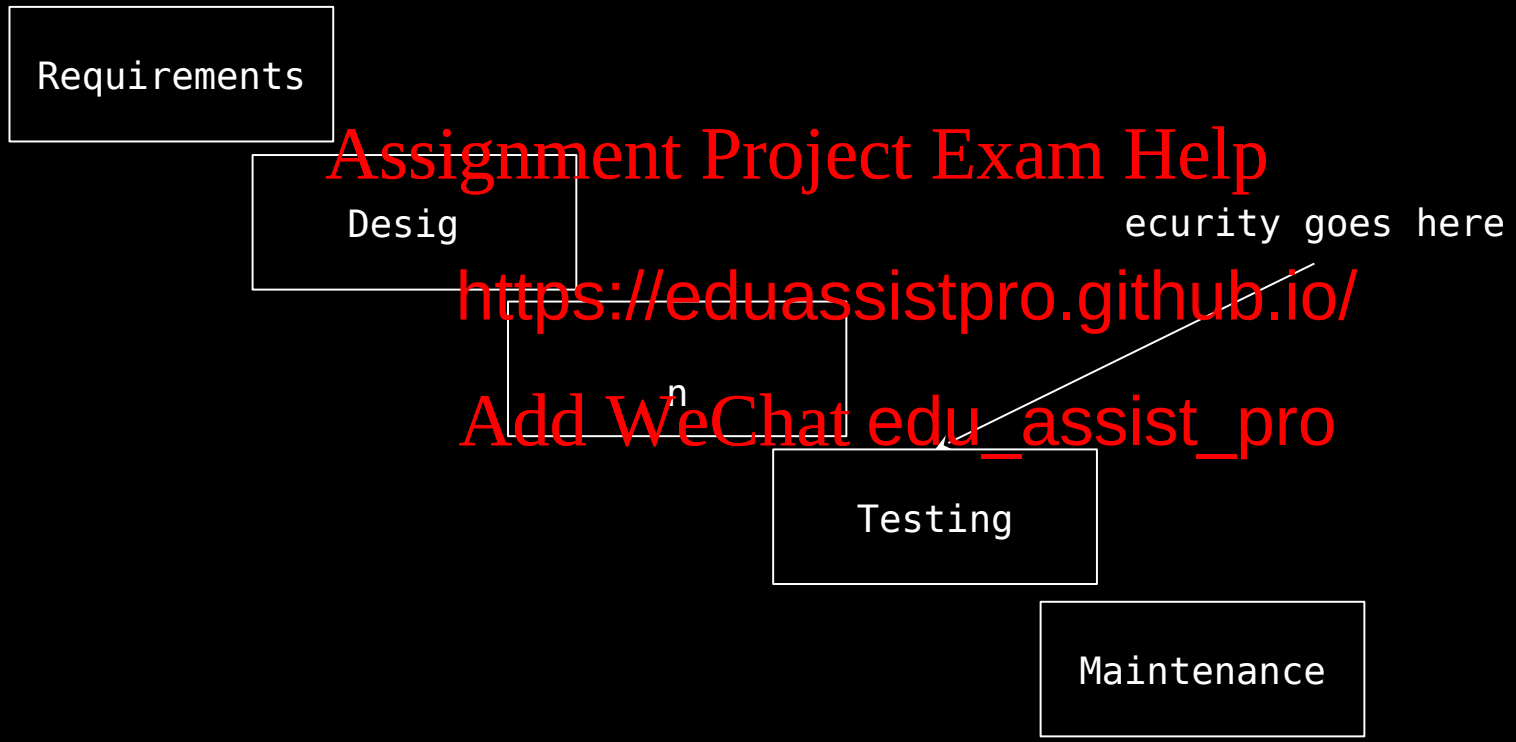
- UNSW hosting this course is an extremely important step forward.
- We expect a hig ism from you, meaning:
  - Respect the property of othe niversity
  - Always abide by the law and egulations
  - Be considerate of others to ensure everyone has an equal learning experience
  - Always check that you have written permission before performing a security test on a system

Always err on the side of caution. If you are unsure about

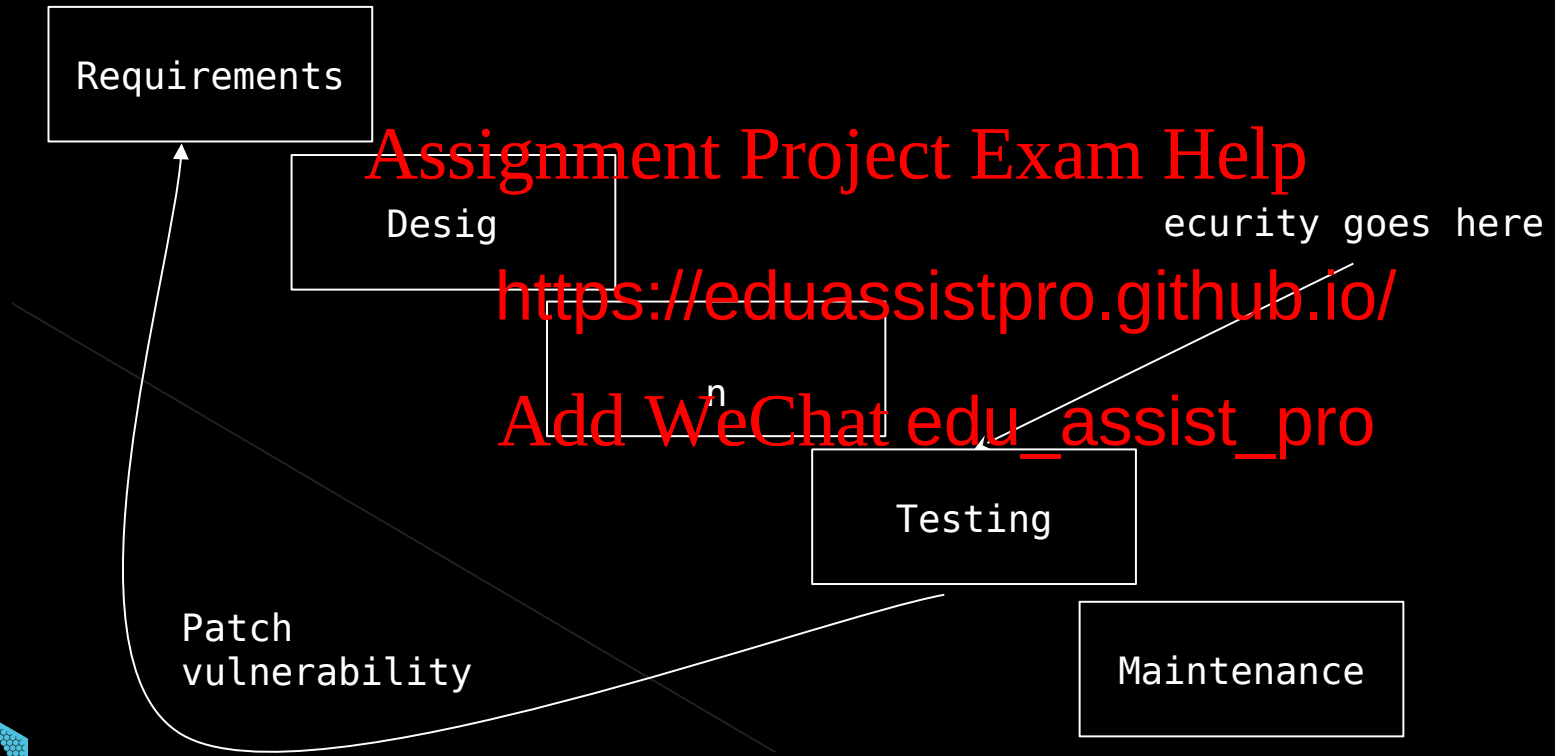
Waterfall development

- Software has been traditionally developed as a sequential project, visualised as a waterfall, with the output of each phase becoming the input to the next.
- Pros:
  - Clear sche
  - Task dependency
  - Accurate planning
- Cons:
  - Inflexibility for changing requirements while a project is being executed
  - Schedule blowout if one phase holds up the subsequent phases
  - Integration occurs at the very end of the process

Security in a waterfall model



Security in a waterfall model



Source: http://ouriken.com/blog/which-one-is-right-for-you-waterfall-or-agile/

Agile manifesto

- We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
  - Individual processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan
- That is, while there is value in the items on the right, we value the items on the left more.

… Scrums? Kanban? Sprints? Backlog grooming?

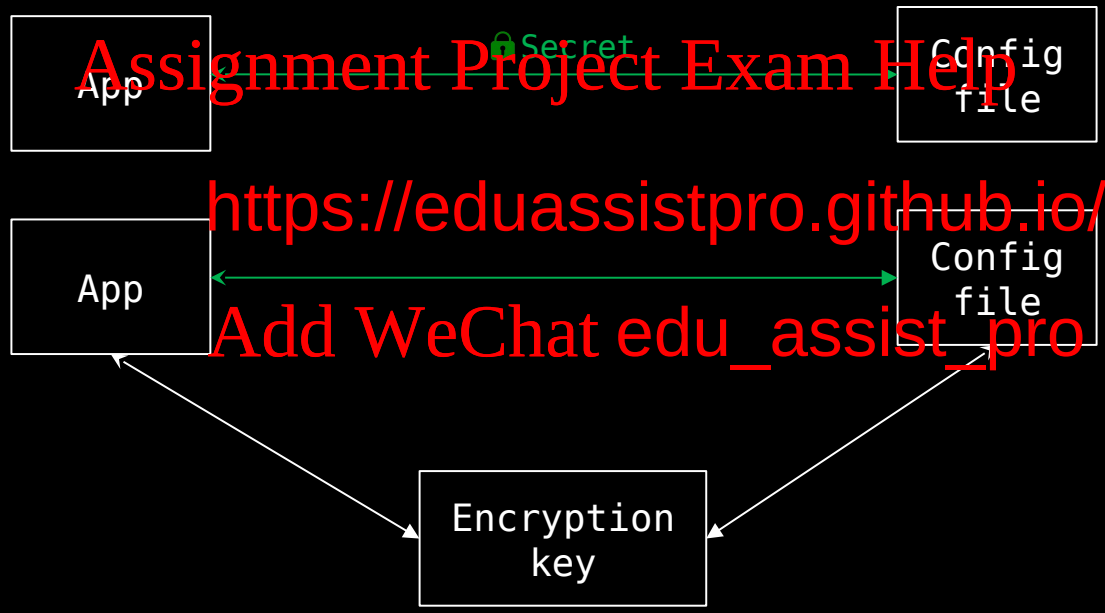
Agile cycle

Phase	Inputs	Outcomes
Backlog	Developer training	Security prioritised
Design	Secr	cure persistency
Development	Soft	cure dependencies
Testing	Static & dynamic analys	bugs
Deployment	Containerisation, hardening	Defence in depth
Review	Root cause analysis	Bug class eradication

Developer training

Developer training

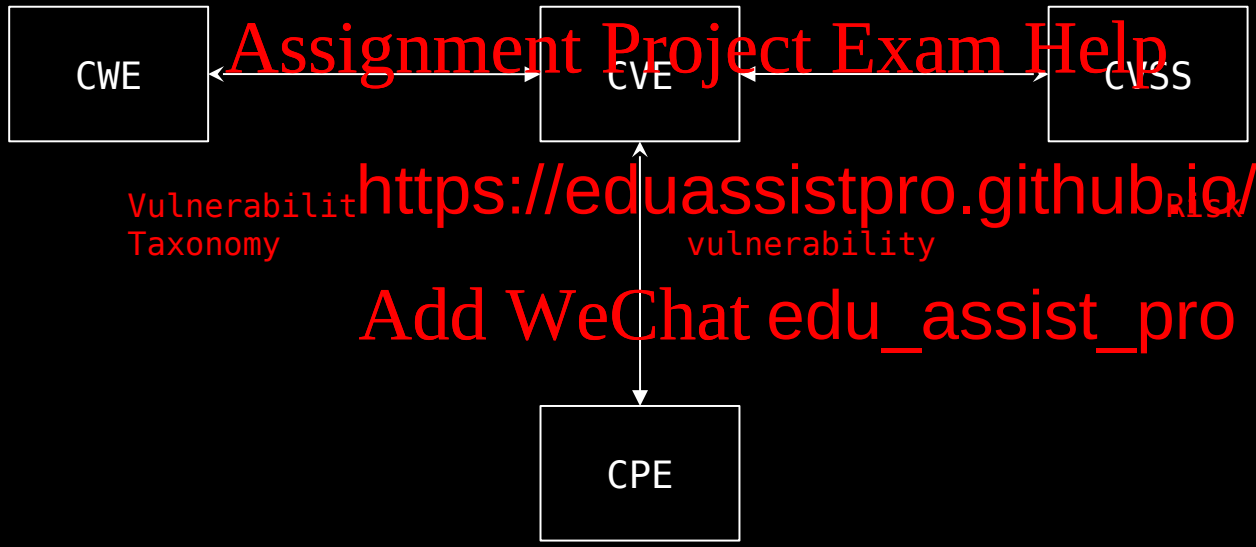
Secrets management



- Password vaults are the current best solution

Common Vulnerability Enumeration CVE

OVAL & NIST NVD



NVD Example

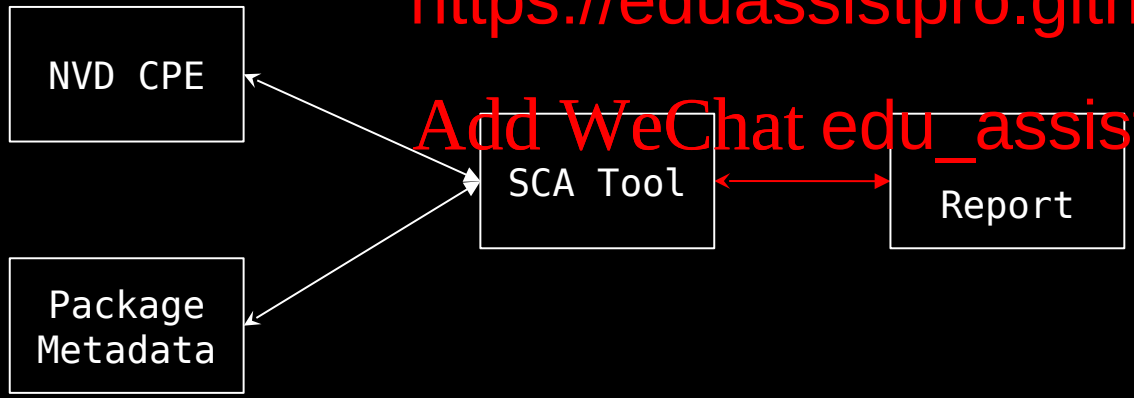
- https:// CVE-2014-0003

NVD Problems

NVD Problems

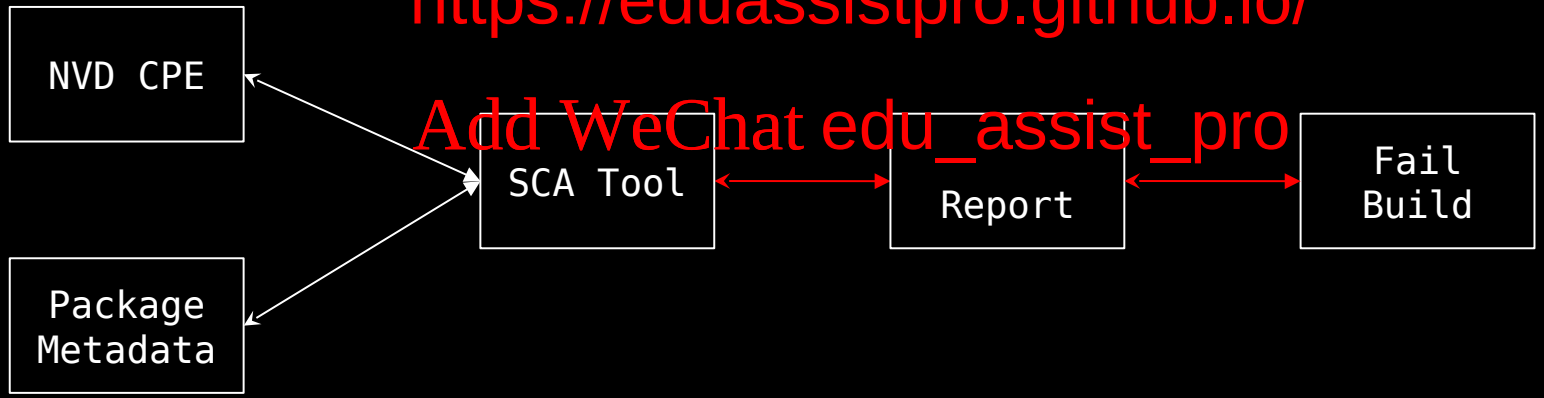
Dependency identification

- NVD CPE identifies known vulnerable versions
- Package metadata identifies version used
- SCA tool attempts to match the two and identify known vulns



Dependency identification in Java

- NVD:
  - cpe:/a:springsource:spring_framework:3.2.0
    cpe:/a:pivotal:spring_framework:3.2.0
    cpe:/a:pivotal_software:spring_framework:3.2.0
- GAV:
  - org.springfra E

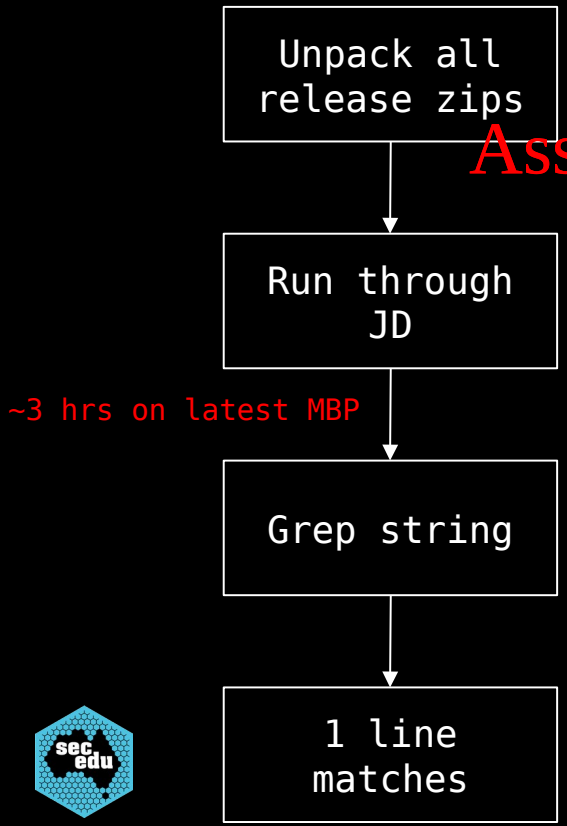


Source code analysis

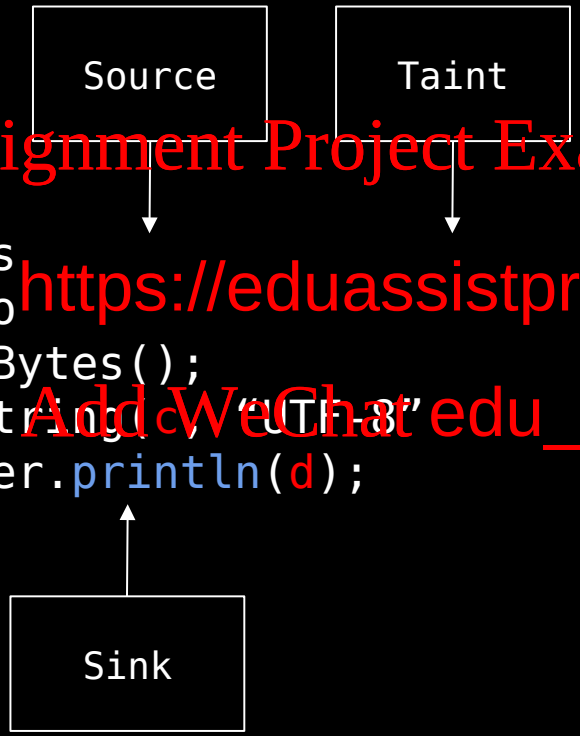
```
$ grep -L "parameter-entities" $(grep -l -R "general-
entities" *)
resteasy-jaxrs-2.3.2.Final/providers/jaxb/src/main/java/
org/jboss/resteas                                    /
ExternalEntityUnm
```

https://www.openwall.com/lists/oss-security/2014/06/03/5

Source code analysis



Sources, sinks & taints



Static application security testing

Pros	Cons
Find & fix vulns early	Massive false positives
Identify vulns configuration	ge & n
Open source tools available	C plo ments = $ $
Potential for bug class eradication	Complexity of tweaking rules

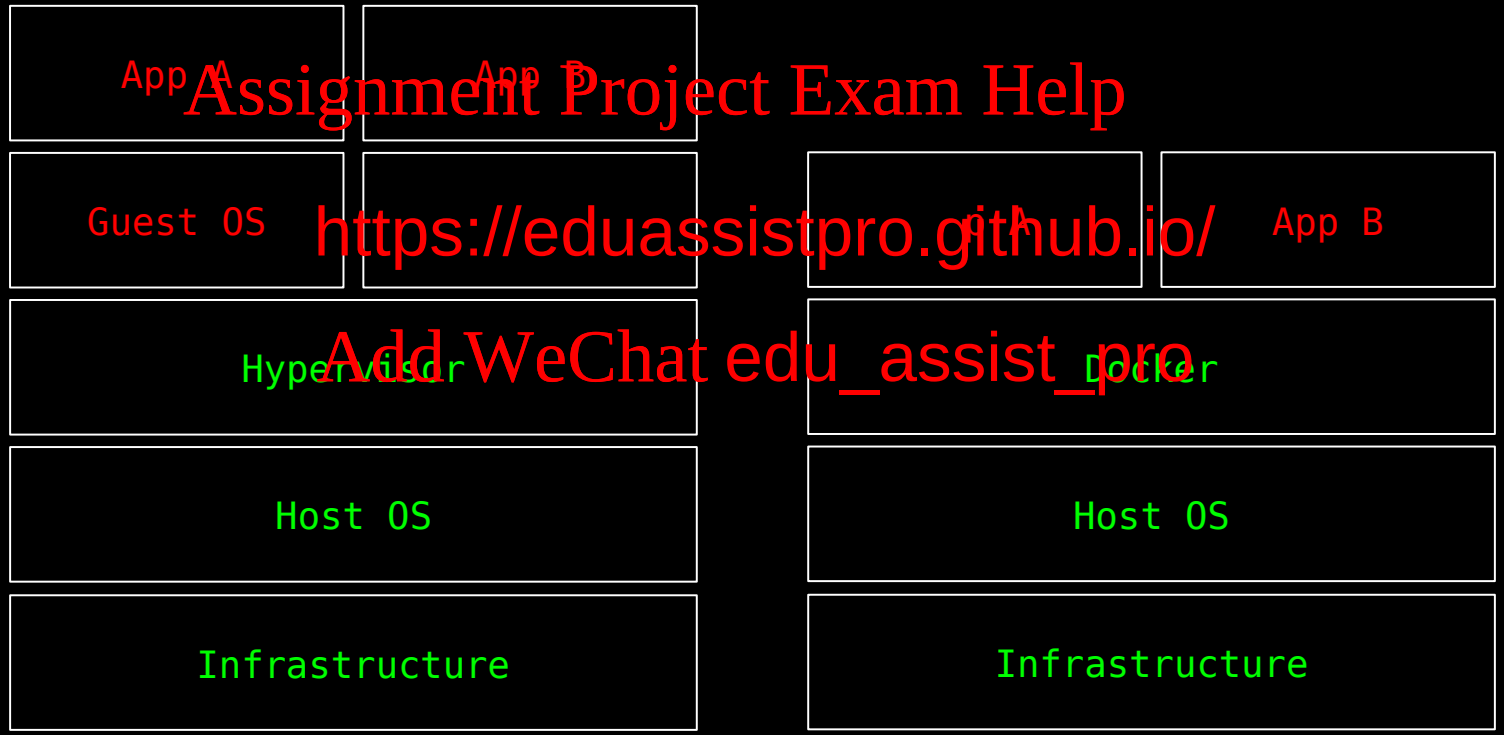
Dynamic application security testing

AKA DAST. Many tools, big commercial ones include Netsparker, Tenable, CheckMarx and Veracode.

Dynamic application security testing

Pros	Cons
Scanning of live targets	Data corruption
Language indep	config files
Cloud based deployment	C tand complex d t/server
Less false positives than SAST	Relies on configuration to map attack surface

Virtualisation vs containerisation



Container breakout CVE-2019-5736

RunC is a container runtime originally developed as part of Docker and later extracted out as a separate open source tool and library. As a "low level" container runtime, runC is mainly used by runtimes (e.g. Docker) to spawn though it can be used as a stand-a container runtimes like Docker will normally implem nalities such as image creation and management an runC to handle tasks related to running containers – creating a container, attaching a process to an existing container (docker exec) and so on.

Credit: https://unit42.paloaltonetworks.com/breaking-docker-via-runc-explaining-cve-2019-5736/

Container breakout CVE-2019-5736

procfs is a virtual fs in Linux that presents information about processes, mounted to /proc. It can be thought of as an interface to system data that the kernel exposes as a filesystem. Each ctory in procfs, at /proc/[pid] /proc/self points Each process's directory contains information o ss. For the vulnerability, the relevant item

- /proc/self/exe – a symbolic link to the executable file the process is running
- /proc/self/fd – a directory containing the file descriptors open by the process.

For example, using ls /proc/self one can see that /proc/self/exe points to the 'ls' executable.

Container breakout CVE-2019-5736

procfs is a virtual fs in Linux that presents information about processes, mounted to /proc. It can be thought of as an interface to system data that the kernel exposes as a filesystem. Each ctory in procfs, at /proc/[pid] /proc/self points Each process's directory contains information o ss. For the vulnerability, the relevant item

- /proc/self/exe – a symbolic link to the executable file the process is running
- /proc/self/fd – a directory containing the file descriptors open by the process.

For example, using ls /proc/self one can see that /proc/self/exe points to the 'ls' executable.

Container breakout CVE-2019-5736

- Anattacker can trick runC into executing itself by asking it to run /proc/self/exe, which is a symbolic link to the runC binary on the host.
- An attacker w ntainer can then use /proc/[ru e to the runC binary on the
- Root access in the container to perform this attack as the runC binary is ct.
- The next time runC is executed, the attacker will achieve code execution on the host.
- Since runC is normally run as root (e.g. by the Docker daemon), the attacker will gain root access on the host.

docker-bench-security

Continuous integration|deployment

Source: atlassian.com

Continuous integration|deployment

READING MATERIAL (REFERENCE)

Find-sec-bugs
https://find-sec-bugs.github.io/
Tracking vulnerable JARs
https://www.slide king-vulnerable-ja
rs

OWASP dependency check
https://owasp.org/www-project- check/
OWASP ZAP
https://owasp.org/www-project-zap/
Jenkins
https://www.jenkins.io/
Docker-bench-security
https://github.com/docker/docker-bench-security

WEEK 9 ASSESSMENT

- Exam question based on provided scenario
- Similar in structure to a job interview question
- Answer will b t

Please call out if you get stuck.
Support one another, your tutors are here to help!

THAN TO US

questions? slack / email talk to us