Application Security

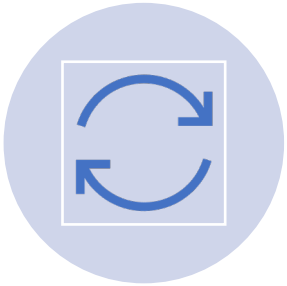
What is it, why do I care & how is it done?

Presented by Derek Hill
@secureITtoday
derek@dh-solutions.com

Agenda

Background

Framework

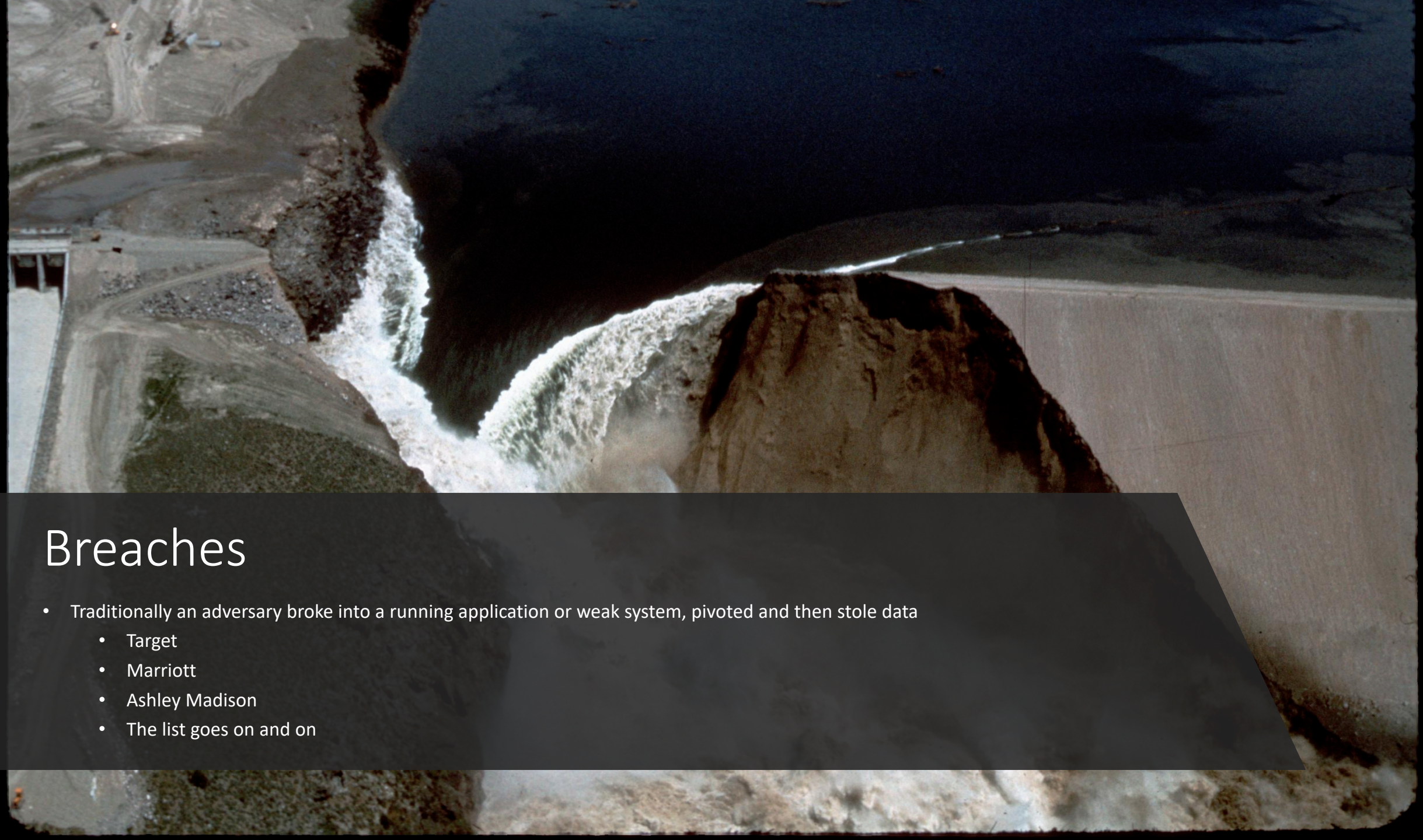


Processes

Tools

Background

Why do I care?



Breaches

- Traditionally an adversary broke into a running application or weak system, pivoted and then stole data
  - Target
  - Marriott
  - Ashley Madison
  - The list goes on and on

Supply Chain Attacks

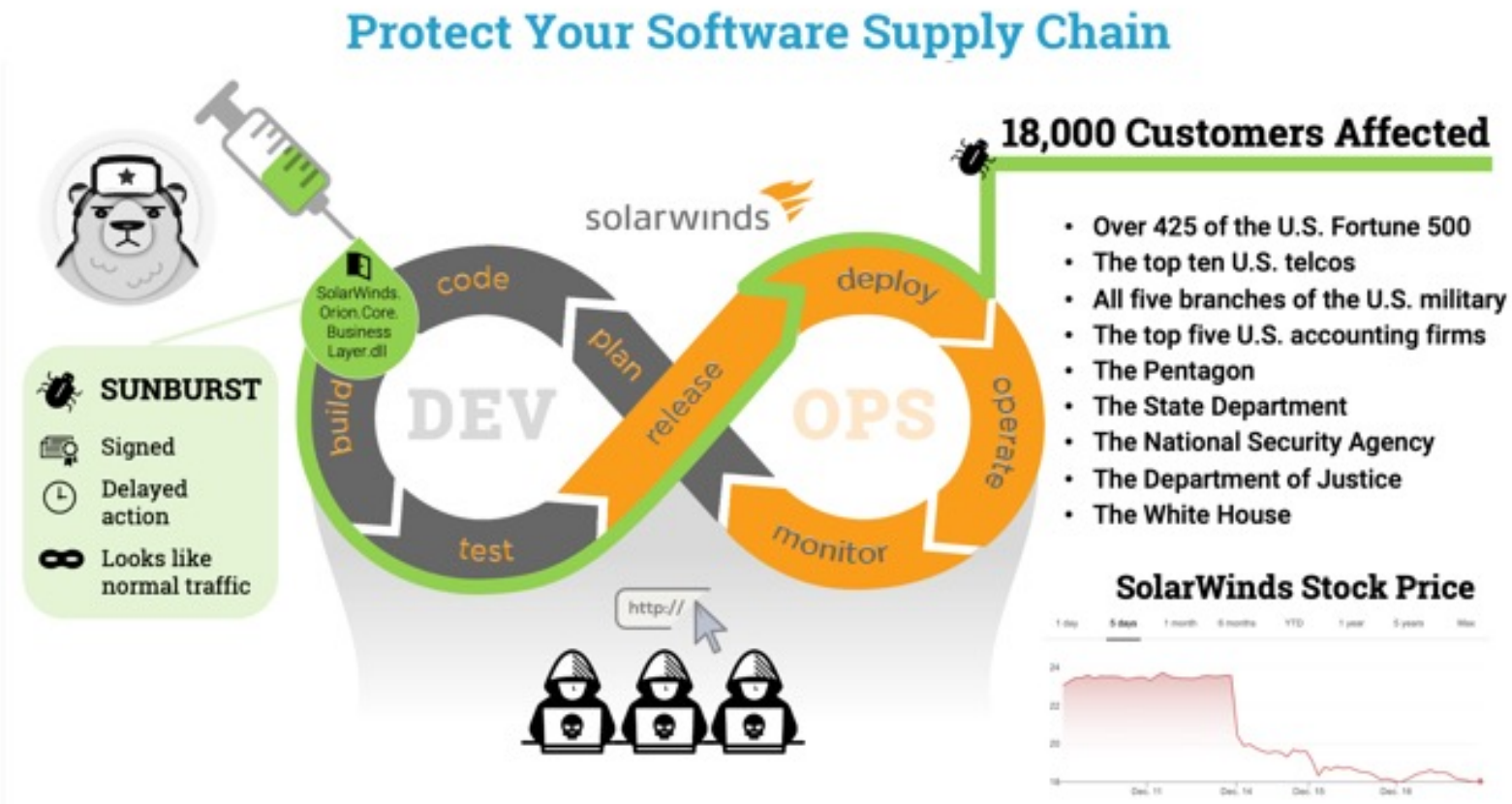
What if you could put your exploit into the application a company sells and distribute it that way

You no longer have to break into individual companies, they are willingly installing the compromised software into their environment

Very public attacks

- NPM
  - bb-builder – password recovery tool
  - Relatively quickly discovered
- Solar Winds (Orion)
  - SW has over 300,000 customers, 33,000 of them use Orion, 18,000 of those had installed the malware, many are unknown, but we know of a large number of customers including Microsoft, Cisco, Intel, Universities, the US Government and so many more
  - Took almost 9 months to discover, found by third party – FireEye (after they were breached)
  - Orion tool was often installed in privileged network segments with high level of access

Why should I care?

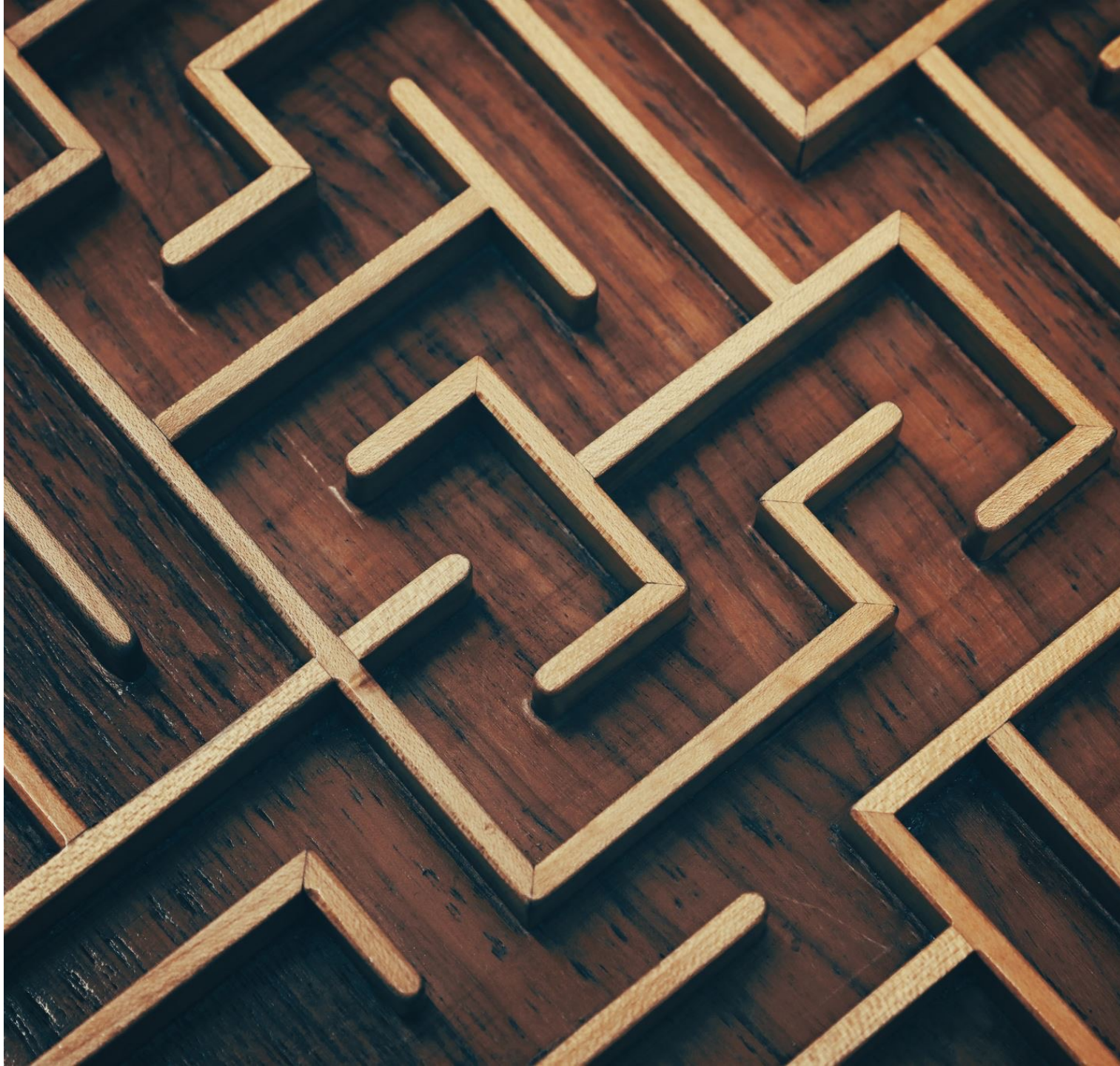


What is the impact if you are in one of the categories below?

- Investor
- Employee
- Customer
- Vendor

Now What

- What are the challenges
- How do we prevent this
- Show me the way



Challenges

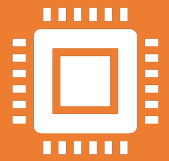
- We write applications, however, only 5-15% of the code in an application is something we write.
  - We include a lot of 3rd party software
    - Think NPM modules, i.e. bb-builder
  - Where did that software come from
  - Can we trust it
- We love to reuse stuff to make our life easier
  - VM Images
  - Code others have written
  - Movies & Music
- We are often pressured to deliver something quickly and developing it ourselves takes a lot of time

The Building Blocks

- Framework
- Processes
- Tools

Framework –
SDLC, SDL, SSDLC, etc.

Many terms for the same thing



Security is now integrated into the software development lifecycle

A lot of this work was pioneered by Microsoft with their Trustworthy Computing initiative in late 2004

It creates standards for your software

What is acceptable

What does a company value

It outlines the processes your team needs to follow

Developers

Security People

QA

Program Managers

Microsoft SDL (Secure Development Lifecycle)
https://www.microsoft.com/en-us/securityengineering/sdl/practices

- Practice #1: Provide Secure Coding Training
- Practice #2: Define Security Requirements
- Practice #3: Define Metrics and Compliance Reporting
  - Security requirements
  - Security reporting
- Practice #4: Perform Threat Modeling
- Practice #5: Establish Design Requirements
- Practice #6: Define and Use Cryptography Standards
- Practice #7: Manage the Security Risk of Using Third-Party Components
- Practice #8: Use Approved Tools
  - Development Tools
  - Security Tools
  - Deployment Tools
- Practice #9: Perform Static Analysis Security Testing (SAST)
- Practice #10: Perform Dynamic Analysis Security Testing (DAST)
- Practice #11: Perform Penetration Testing
- Practice #12: Establish a Standard Incident Response Process

Adapting the SDL to your organization

Part 1:

- Make the SDL reflect your current state
  - Inventory your procedures and processes
  - GAP analysis perhaps

Part 2:

- Adopt what you need/want
  - Work with stakeholders, this needs to be collaborative, otherwise it will fail
- Create a version for a desired future state
- Strive toward that state, it will take time, it will take multiple iterations & money
- Review for effectiveness along the way and adjust

Policy Examples

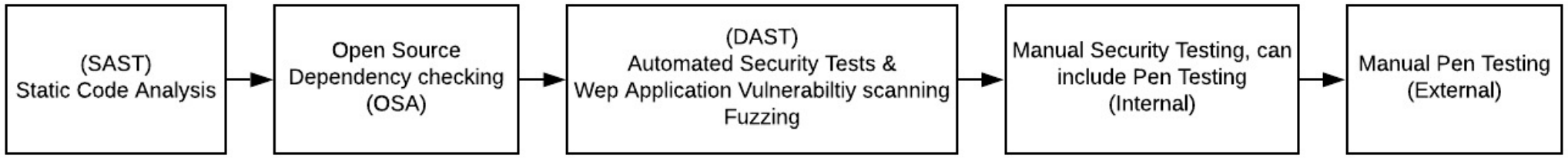
Code needs to be reviewed by at least one other developer before it can be merged into master

Deliver applications “secure by default”

Process Examples

- Prevent direct merges to master, only the CI tool can merge to master, after it meets the following requirements:
  - PR has to be approved by at least one other developer
  - PR has to pass all automated security tests

- Ensure that all communication is over secure channels, ie. TLS or SSH
  - Block insecure ports to prevent clear text communication (i.e. 20,21,23,80, etc)
- Require complex passwords
  - Enforce a password policy that has a minimum length, types of characters, etc.
- Enforce secure settings
  - HSTS (HTTP Strict Transport Security)



Security Pipeline (an example)

Tools

for Application Security

Static Application Security Testing (SAST)

- Considered whitebox testing as you have access to the source code
- Test for security flaws in your code
- Tool is only as good as the tuning / maintenance
  - You don’t want a lot of false positives (FP), your users will not trust the tool
  - You also don’t want false negatives (FN) as those are real issues
  - Tuning has to be a balance, when in doubt error on the side of a small number of FP’s
- Run it early and often in the pipeline
  - Scan the developer branches
  - Don’t let the security flaws to be merged into the master branch
- The Security tools often lag new languages or versions of language, be careful if your team wants to use the latest and greatest

```
  FileUtils.copyFile(new File(identity_governance_installer_openidm_path + '/workflow/entitlement-remediation.bpmn20.xml'), new File(proj
}
else {
  FileUtils.copyFile(new File(identity_governance_installer_openidm_path + '/workflow/entitlement-remediation7.bpmn20.xml'), new File(pro
}
println "Done copying workflow files into IDM"

//Update integrations
GroovyShell gs = new GroovyShell()

if ( IS_INITIAL_INSTALL ){
  //Exclude for Cluster
  println "Backing up managed.json"
  try {
    FileUtils.copyFile(new File(project_path + '/conf/managed.json'), new File(project_path + '/backup/conf/managed.json.pre_AR_install-'
    println "Done backing up managed.json file"
  } catch (IOException) {
    println("Copy failed")
  }

  copyFileNoClobberFromIdmToProject('/conf/managed.json')

  def managed = gs.parse(new File(identity_governance_installer_integrations_path + '/conf/managed.json.groovy'))
  managed.execute(project_path, identity_governance_installer_integrations_path)

} else {
  println "Skipping managed.json -  non-primary installation"
}

if ( IS_INITIAL_INSTALL ){
  //Exclude for Cluster
  println "Backing up policy.json"
  try {
    FileUtils.copyFile(new File(project_path + '/conf/policy.json'), new File(project_path + '/backup/conf/policy.json.pre_AR_install-'+d
    println "Done backing up policy.json file"
  } catch (IOException) {
    println("Copy failed")
  }

  copyFileNoClobberFromIdmToProject('/conf/policy.json')

  def policy = gs.parse(new File(identity_governance_installer_integrations_path + '/conf/policy.json.groovy'))
  policy.execute(project_path)
} else {
  println "Skipping policy.json - non-primary installation"
```

args

project_path

project_path

execute

Scan Results | Severity

Groovy

- High
- Medium
- Low

The application's /* method calls an OS (shell) command with execute, at line 152 [redacted] using an untrusted string with the command to execute.
This could allow an attacker to inject an arbitrary command, and enable a Command Injection attack.
The attacker may be able to inject the executed command via user input, args, which is retrieved by the application in the /* method, at line 35 of [redacted]

Results | Graph | Codebashing

Result State | Result Severity | Assign to User | Comments | Save Scan Subset | Open Ticket | Filters | Group By

Id	Detection	Direct	Query Nar	Status	Source Fo	Source Fil	Source Lir	Source Ob	Destinatio	Destinatio	Destinatio	Destinatio	Result Sta	Result Sev	BFL Node	BFL Group	Priority	Assigned	Ticket ID	Comments

Open Source Analysis (OSA)

- Checks the state of 3rd party components / code you are importing
- Most tools don't actually check the code, but rather look at various DB's such as NIST CVE's, etc.
- Can also check licensing (important from an intellectual property (IP) perspective, but not relevant to security)
- Compile a list of all included libraries, explicit AND implicit
- DO YOU KNOW WHAT IS IN YOUR APPLICATION?

Alerts 4 All Time [redacted] [redacted] Apply Ignore Selected Export

Filter

Library	Type	Description	Library Type	Creation Date	Modified Date	Occurrences	
jmh-core-1.21.jar	Policy Violation	Black list (viral licenses)	Java	11-09-2020	15-03-2021	1 project details	ignore
commons-codec-1.10.jar	Security Vulnerability	Medium: 1 details	Java	11-09-2020	15-03-2021	1 project details	ignore
commons-codec-1.11.jar	Security Vulnerability	Medium: 1 details	Java	11-09-2020	15-03-2021	1 project details	ignore
httpclient-4.5.6.jar	Security Vulnerability	Medium: 1 details	Java	24-12-2020	15-03-2021	1 project details	ignore

Dynamic Application Security Testing (DAST)

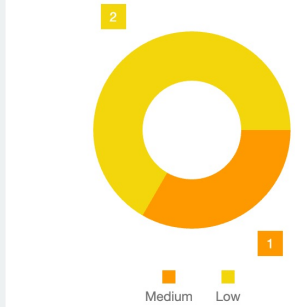
- Considered black box testing, can be run authenticated or not
  - This is generally what a hacker is exposed to, unless they get access to your source code
- Tests not only the code, but the entire stack (running application)
- Application has to run before you can test, later in the pipeline
- Tool can have FP's, might require flagging of issues

Acunetix

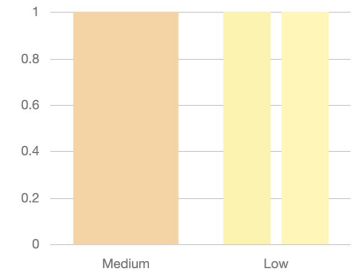
Launch Scan

- Discovery
- Websites
  - Dashboard
  - Websites
  - New Website
  - Import Websites
  - Manage Groups
  - New Group
- Scans
- Scheduled Scans
- Reporting
- Issues
- Technologies
- Policies
- Notifications
- Integrations
- Team
- Activity
- Agents
- Settings

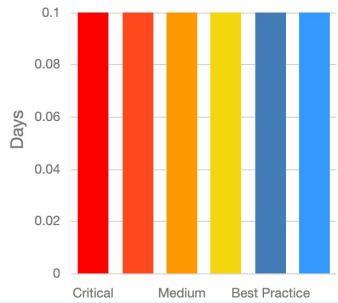
Severities



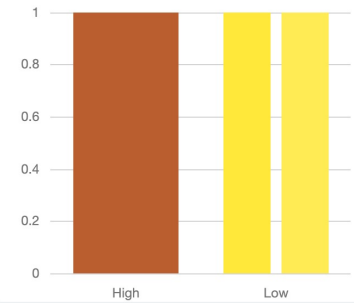
Issues



Average Time to Fix



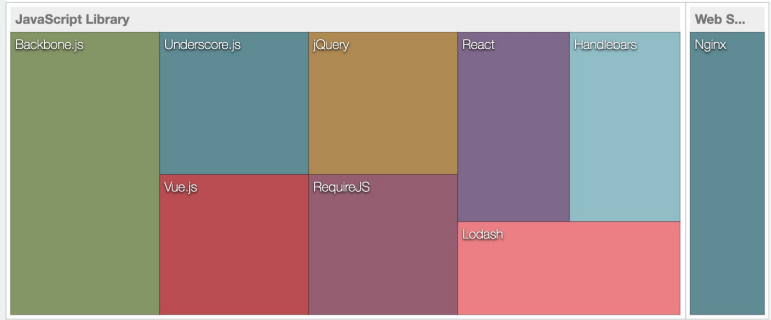
Fixed Issues



Issues

Out-of-date Version (jQuery)	Present
Missing X-Frame-Options Header	Present
Internal Server Error	Present

Most Identified Technologies



IDM

Delete

Recent Scans

12:35 PM	0 0 1 2	Scan	
11:00 AM	0 0 1 2	Scan	
10:27 PM	0 0 1 2	Scan	
11:05 PM	0 0 0 1	Scan	
10:18 PM	0 0 1 2	Scan	
02:24 AM	0 0 0 1	Scan	
09:59 AM	0 0 1 2	Scan	
10:25 PM	0 0 0 2	Scan	

Scan History

License Status

Subscription Website

Your subscription expires on [redacted] Once it expires, you can launch no further scans. Please contact us at support@acunetix.com to extend or renew your subscription.

Manual/Automated Security Testing (internal)

- Can consist of commercial, open source and even custom written applications
- Fuzzers
- Pen testing
- Scanners
- Manual testing
- Threat Hunting exercises

External Pen Testing

- Penetration testing conducted by independent third party
- Limited scope
- Slow and expensive
- A passing test does not mean your application is secure, it means they couldn’t find anything
- Often required by customers / compliance
- Annual exercise, perhaps even multiple times a year

Other types of testing

Not going to cover – but it is available

Interactive Application Security Testing (IAST)

Runtime Application Self Protection (RASP)

Bug Bounty Programs

Image/Container scanning

How to select the tools?

Roll the dice

Trial and Error – coin flip

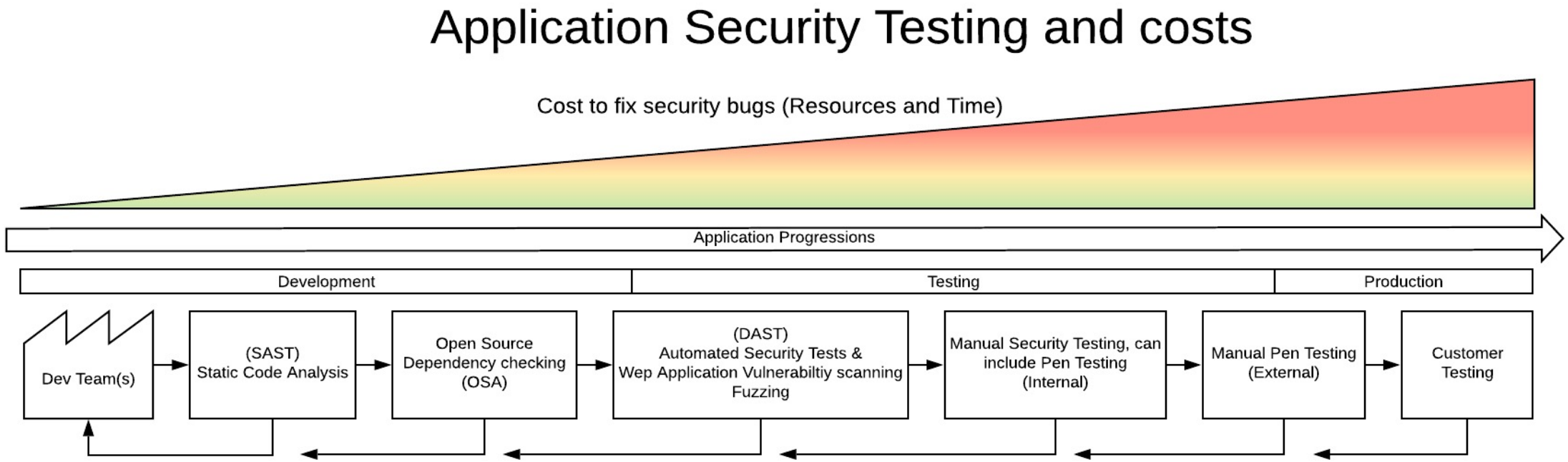
Talk to others

Personal experience

Managed or not

Cost of security defects

- The longer you wait to fix an issue, the more expensive it is
  - More people have to touch the code
  - More support calls
  - Customers lose faith
  - Breach
  - Bad Press
  - etc.



Shift Left

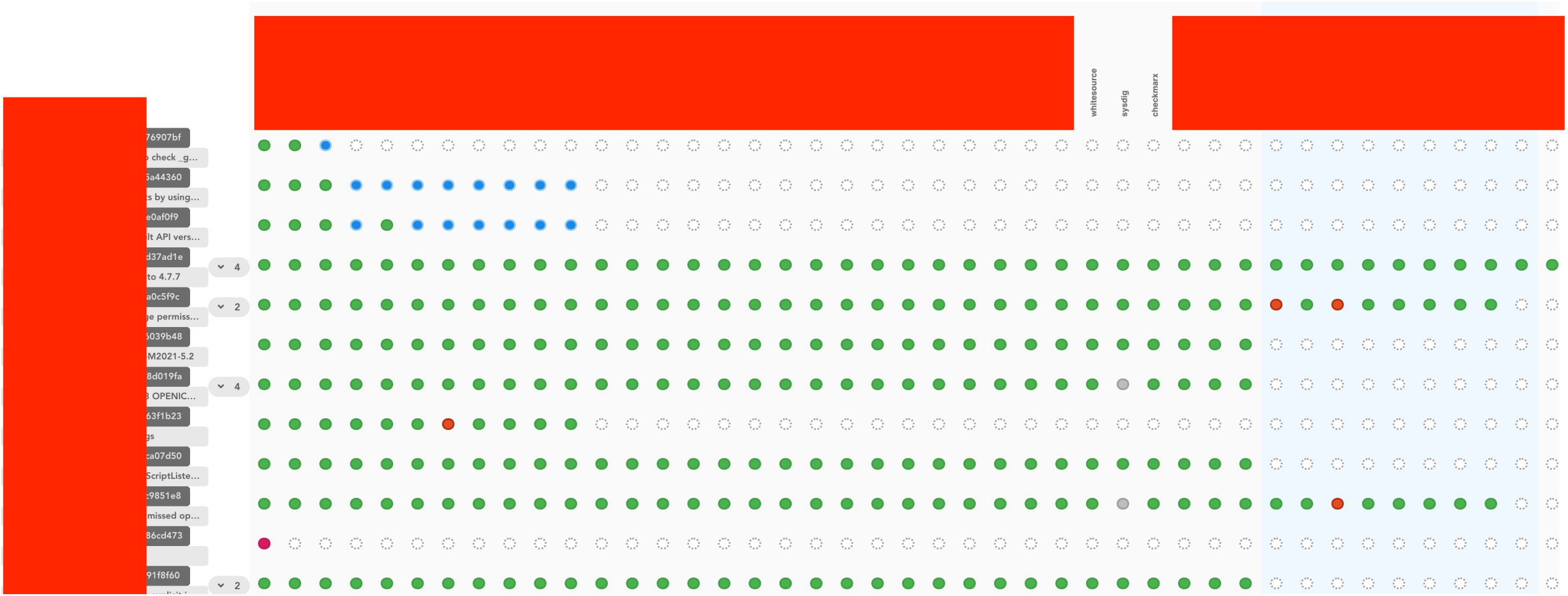
RUN THE TOOLS EARLY IN THE DEVELOPMENT PROCESS

QUICKER FEEDBACK

REDUCE CODING BY LIMITING REWRITES & REVIEWS

AUTOMATE WHERE POSSIBLE

REDUCE THOSE COSTS


whitesource
sysdig
checkmarx
76907bf
check _g...
5a44360
s by using...
e0af0f9
lt API vers...
d37ad1e
4
to 4.7.7
a0c5f9c
2
ge permiss...
5039b48
M2021-5.2
8d019fa
4
3 OPENIC...
63f1b23
gs
ca07d50
ScriptListe...
c9851e8
missed op...
86cd473
91f8f60
2

Developer Secure Coding Training

- Bugs not created don't have to be remediated
- Start this process early
- Benefits:
  - Developers learn new skills
  - Improve the code they produce
  - Get them involved, even have code hunting parties

1 Locate Vulnerability — 2 Identify Solution — Challenge Complete

Locate Vulnerability

Identify and select the code blocks that cause the vulnerability listed below by clicking the ⚠ next to the line numbers in the code viewer.
Files containing selectable code blocks have been marked with ⚠.

Vulnerability Category

Memory Corruption - Null Dereference

Submit Your Answer

There is **1** vulnerable block in the source code that you need to locate.
You have selected **0** code blocks

Skip | Hint | Next

Quickswitch | Jump

- .gitignore
- ⚠ **dir_traverse.cpp**
- dir_traverse.sln
- dir_traverse.vcxproj
- dir_traverse.vcxproj.filters

```
1  // dir_traverse.cpp : Defines the entry point for the console ap
2  //
3
4  #include <filesystem>
5  #include <iostream>
6
7  // Alias the filesystem namespace so changing this later just re
8  namespace std { namespace filesystem = std::experimental::filesy
9
10
11 void list_all_sub_directories(std::filesystem::path const& root)
12 {
13   std::vector<std::string> file_paths;
14
15   for (auto const& next : std::filesystem::recursive_directory_i
16   {
17     file_paths.push_back(next.path().u8string());
18   }
19
20   for (auto const& f : file_paths)
21   {
22     std::cout << f << std::endl;
23   }
24 }
25
26
27 int main(int argc, char** argv)
28 {
⚠ 29   std::string const arg(argc > 0 ? argv[1] : "");
⚠ 30   list_all_sub_directories(arg.empty() ?
31     std::filesystem::current_path() :
⚠ 32     std::filesystem::path(arg));
33   return 0;
34 }
35
36
```

Summary

FRAMEWORK

PROCESSES

TOOLS

Questions