

# #NoHobbyists

What's really needed to shift Cyber Security?

**Trac Bannon**  
Senior Principal  
Advanced Software Innovation Center

**MITRE** | SOLVING PROBLEMS  
FOR A SAFER WORLD



Red Cross servers  
'were hacked via  
unpatched flaw'

18 February 2022

California public office admits Covid-19  
healthcare data breach

Network cavity  
blamed for data  
breach at Japanese  
candy maker

29 March 2022

SQL injection  
vulnerability found in  
Moodle e-learning  
platform

08 March 2022

Zero-day XSS  
vulnerability found in  
Horde webmail client

24 February 2022

Attackers getting  
faster at latching  
onto unpatched  
vulnerabilities

28 March 2022

Nvidia cyber-attack  
linked to Lapsus\$  
ransomware gang

28 February 2022

GitLab addresses  
critical account  
hijack bug

01 April 2022

Japanese retailer  
traces breach to  
third-party hack

04 March 2022

Prison data breaches

UK Ministry of Justice recorded more  
than 2,000 incidents over 12 months

14 March 2022

Okta investigation

Authentication and identity management giant probes LAPSUS\$ gang's  
compromise claims

22 March 2022

Source: <https://portswigger.net/daily-swig>

**Quick!  
Shift Left!**

**Shifting Security “left” means...**

Security activities start at design

Extend through out the SDLC

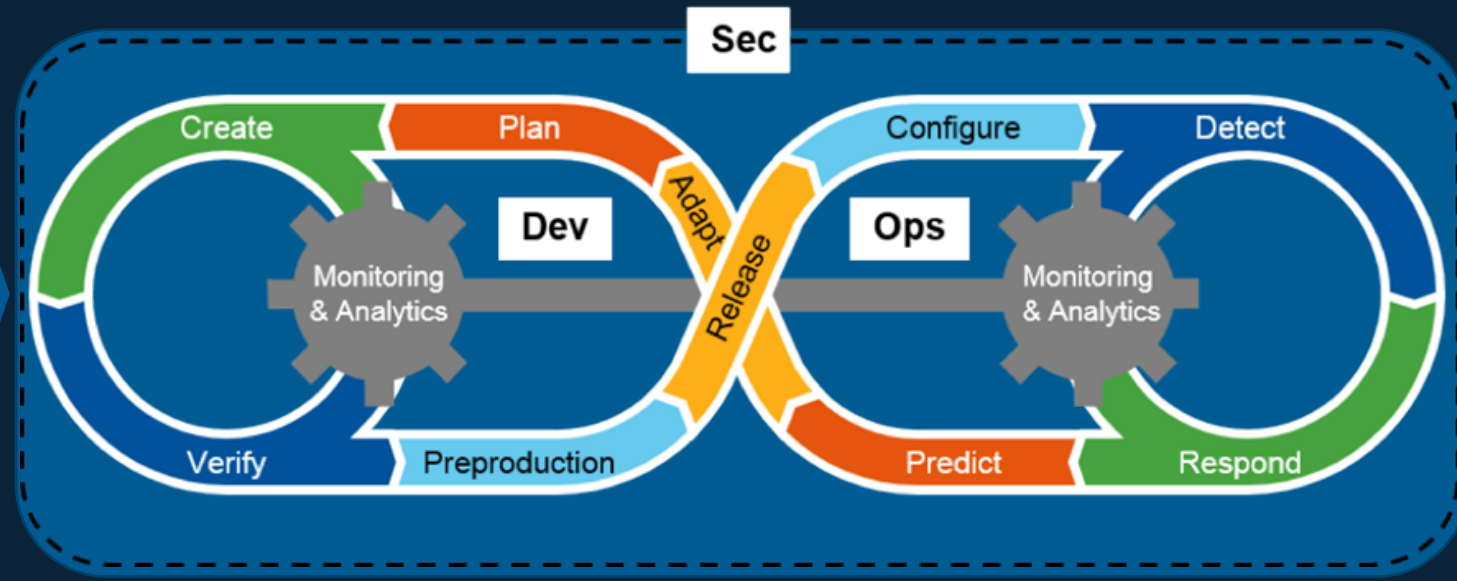
From vision through operate

Continuous feedback at every step



# Make Cyber Security Ubiquitous

Don't Shift Left!  
Shift Everywhere!



Source: [https://tech.gsa.gov/guides/understanding\\_differences\\_agile\\_devsecops/](https://tech.gsa.gov/guides/understanding_differences_agile_devsecops/)

Security and Cyber Resilience is Everyone's Responsibility

# Address the Four Facets!

People – roles, autonomy, upskilling

Process – workflows, ceremonies, domain specific

Technology – new tech insertion, digital platforms, production

Culture – Trustworthiness, psychological safety, collaboration

Any “shifts”  
will fail unless  
you are  
holistic.

**Sounds logical, but how?**

Start by  
beginning to  
build a new  
culture

## Together ...

Define unifying principles

Agree on roles and responsibilities

Define your Security Software Development  
Framework (**SSDF**)

Work elbow-to-elbow

Demonstrate model behaviors



# Secure Software Development Framework

## Don't start from scratch

Use industry standards like NIST Special Publication 800-218.

This tells you what to do; you define how.

- Clearly defined roles and responsibilities
- Provide adequate software security training
- Agree on secure software development lifecycle
- Establish secure coding standards
- Build and leverage reusable objects
- Verify security control

# Four Question Framework<sup>1</sup>

Make threat  
modeling a  
team sport

What are we working on?

What can go wrong?

What are we going to do about it?

Did we do a good job?

Is it time to code **yet**?

# What is a Security Hobbyist?

- ! More responsibility is being placed on developers
- ! Constant addition of new tools
- ! Training lacks depth
- ! Continued pressure to “go fast”
- ! Reading and experimenting on their own time, if at all



*Developers are **not** experts “waving magic wands to cast spells that can defend against **evil hackers in black hoodies.**”<sup>2</sup>*

# #NoHobbyists

# From Hobbyist to Pro

How can we protect and enable the developers?

- ✓ Training and Education
- ✓ Make sure you are secure by design
- ✓ Use secure coding standards
- ✓ Leverage design patterns
- ✓ Encourage experimentation
- ✓ Double down on tools to help reduce noise and cognitive overload



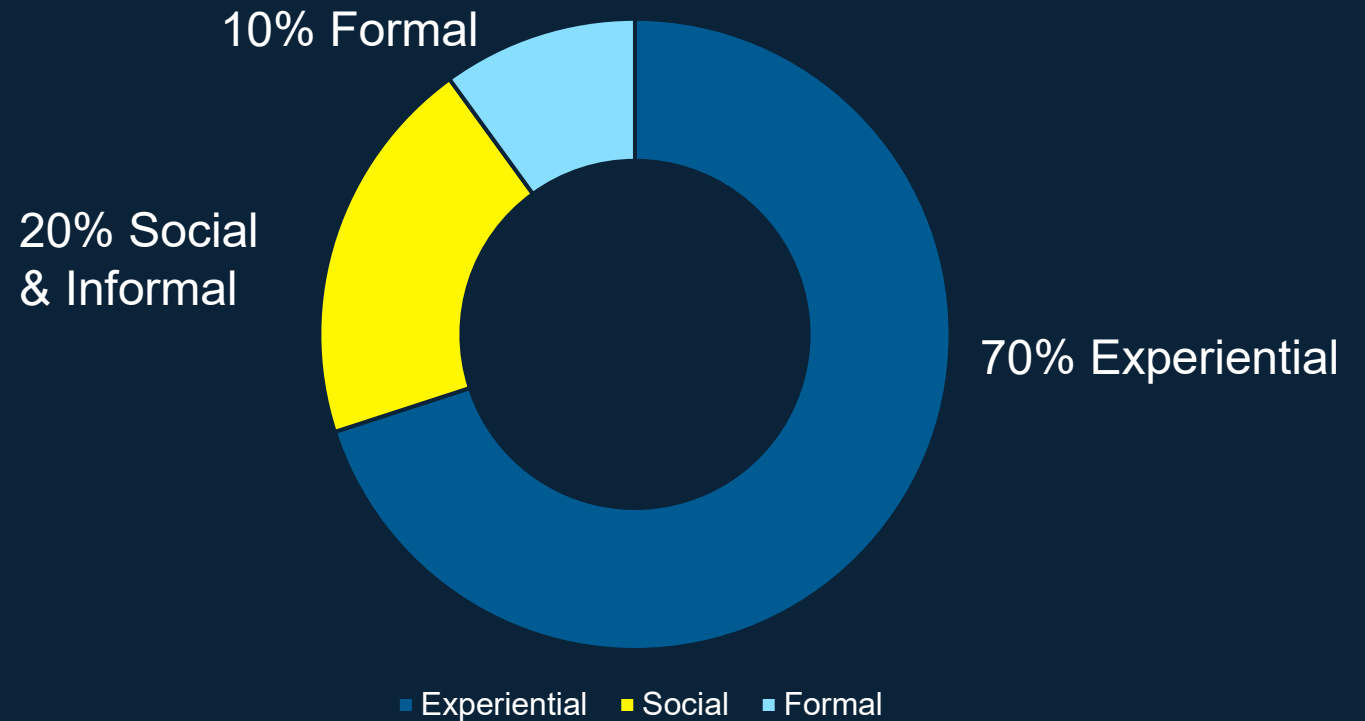


# Upskill and Train Developers

## Use a 70:20:10 Model

Current cybersecurity training for developers lacks depth

Start with core concepts then move to experiences





# Non-classroom Training Examples

Run a team **hack-a-thon** using an OWASP project like Juice Shop or Web Goat (<https://owasp.org/projects/>)

Join a competition at the MITRE Cyber Academy (<https://mitrecyberacademy.org/>)

Pair program with **SAST** tools (Static Application Security Testing)

**Storm-the-castle** lunch events using OWASP ZAP for DAST (Dynamic Application security testing)

Is it time to code **yet**?

“Design is the **guiding principle** for how a system is built and is applicable on all levels, from code to architecture. It includes any activity that involves **active decision-making**.”<sup>3</sup>

Before you code,  
be secure by  
design

Security is a **concern**, not a feature

Explicitly **think** about security

**Context** and Archetype matter

Focus on **domain**; many security bugs are caught implicitly

# Secure Coding Practices

Estimated **82% of software vulnerabilities** are from coding errors.<sup>2</sup>

Identify **secure coding standards** as a team

Make sure everyone **knows the standard**

Run Static Application Security Tools (**SAST**)

Developers can run SAST **before code complete**



# Sample Secure Coding Standards<sup>5</sup>



# Sample Secure Coding Practices

- CWE and CWE Top 25
- CERT – From Carnegie Mellon’s SEI - C, C++, Java, Perl, and Android
- OWASP and OWASP Top 10 – Web apps and APIs
- CVE - Cybersecurity vulnerabilities and exposures
- NVD -
- DISA STIG – DOD’s Secure Technical Implementation Guide
- PA-DSS – Payment application systems
- IEC 62443 - Industrial networks

- Input validation
- Output encoding
- Authentication and password management
- Session management
- Access control
- Error handling and logging
- Data protection
- Communication safety
- System configuration
- Database security
- File management memory management
- . . .

Is it time to code **yet**?

# Dedicate time to dependencies

Document package & program dependencies

Align to stream your **SSDF**

Run Software Composition Analysis (**SCA**)

Automate **SBOM** creation

Learn to **break** things!

# Learn to love Application Security Testing (AST)

Static App Sec Testing (SAST) – **Earliest Detection**

Dynamic App Sec Testing (DAST) – outside  
perspective **before production**

Interactive App Sec Testing (IAST) – **agent based**  
runtime grey-box

SCA - Software Composition Analysis

“There is simply **too much code** being produced for humans to handle alone from a security perspective.”<sup>3</sup>



# Tools to Help Developers

(just a TINY sample)



OWASP Threat Dragon



OWASP Threat  
Model Cookbook



WhiteSource  
<cure>

Automated Code  
Security Remediation



Code Quality & SAST



GitGuardian  
Secrets  
Detection

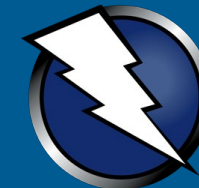


HashiCorp Vault  
Secrets  
Management



snyk

Dependency  
Scanning



OWASP ZAP  
DAST "black box"  
testing

Security is **everyone**'s responsibility.  
**Everyone** needs to be equipped.



Tracy L. Bannon

[TBannon@MITRE.org](mailto:TBannon@MITRE.org)

[TracyBannon@gmail.com](mailto:TracyBannon@gmail.com)

 <https://www.linkedin.com/in/tracylbannon>

 @TracyBannon

Disclaimer: The views, opinions and/or findings contained in this report are those of The MITRE Corporation and should not be construed as an official government position, policy, or decision, unless designated by other documentation.

**MITRE** | **SOLVING PROBLEMS  
FOR A SAFER WORLD™**

## References:

<sup>1</sup> [https://owasp.org/www-community/Threat\\_Modeling](https://owasp.org/www-community/Threat_Modeling)

<sup>2</sup> <https://www.ptsecurity.com/ww-en/analytics/web-vulnerabilities-2020/>

<sup>3</sup> <https://www.securecodewarrior.com/blog/certified-security-awareness-an-executive-order-to-elevate-developers>