Security Compass

Artifacts from processes and tools to maintain trusted source code and verifying and mitigating software vulnerabilities

My Background

Rohit Sethi

CEO,

Security Compass

- 17 years experience focused on secure SDLC with Fortune 1000 customers
- Featured on Bloomberg, CNBC, FoxNews, CNN.com, Huffington Post and many others
- “Balancing Act” podcast host interviewing product security leaders from Cisco, Adobe, Honeywell, JCI, SAP, Dell, Carrier, Goldman Sachs, Yahoo, LinkedIn, Xylem & others

USCYBERCOM Cybersecurity Alert
@CNMF_CyberAlert

Mass exploitation of Atlassian Confluence CVE-2021-26084 is ongoing and expected to accelerate. Please patch immediately if you haven't already— this cannot wait until after the weekend.

9:43 AM · Sep 3, 2021 · Twitter for iPhone

Organizations that have not patched this Confluence Server and Confluence Data Center vulnerability should do so on an emergency basis.

-- Rapid7

Update your Confluence server now

Malefactors are looking for vulnerable Confluence servers and exploiting CVE-2021-26084, an RCE vulnerability.

-- Kasperskey

CVE-2021-26084: Atlassian Confluence OGNL Injection Vulnerability Exploited in the Wild

-- Tenable

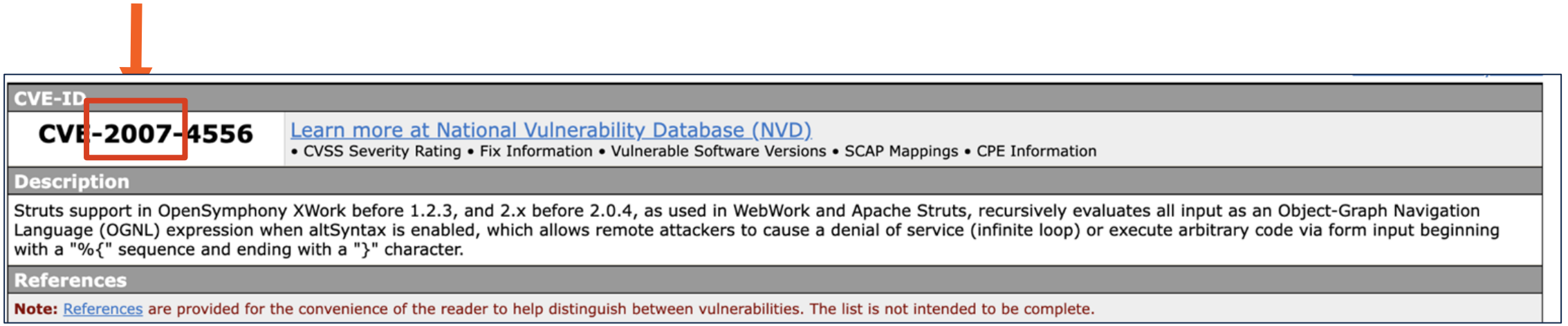
This isn’t about Atlassian,
this an industry-wide problem

After an incident like this, all attention is on:

- Patching
- Detecting & blocking attacks in the wild
- Incident response

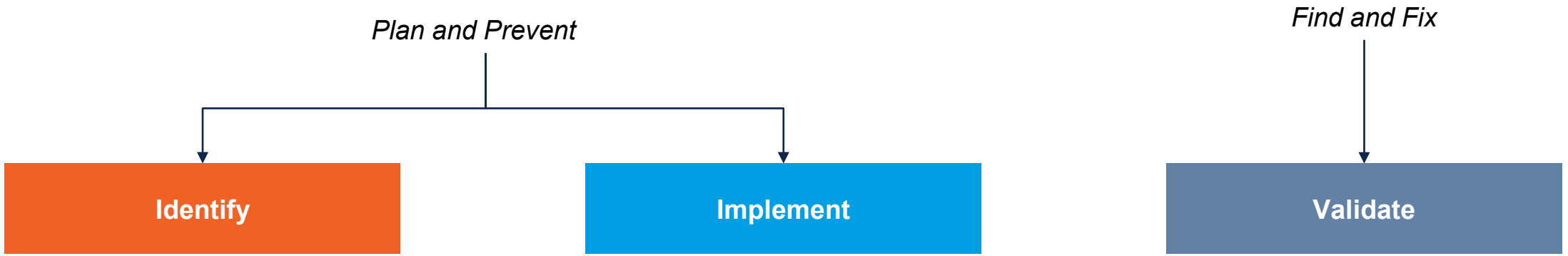
Reviewing the Root Cause

“The vulnerability is an **Object-Graph Navigation Language (OGNL) injection**…”



CVE-ID	
CVE-2007-4556	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information
Description	
Struts support in OpenSymphony XWork before 1.2.3, and 2.x before 2.0.4, as used in WebWork and Apache Struts, recursively evaluates all input as an Object-Graph Navigation Language (OGNL) expression when altSyntax is enabled, which allows remote attackers to cause a denial of service (infinite loop) or execute arbitrary code via form input beginning with a "%{" sequence and ending with a "}" character.	
References	
Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.	

A Framework for Prevention



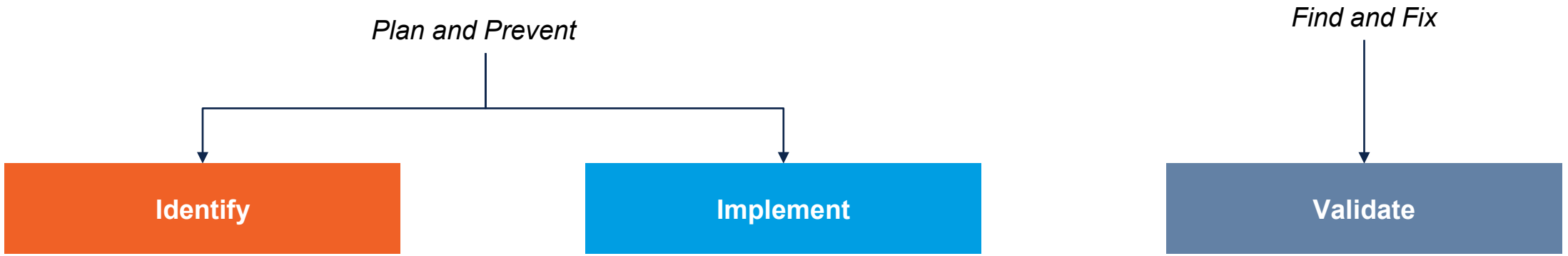
Potential software weaknesses for a given product

Controls to mitigate those weaknesses

Existence of controls and absence of vulnerabilities

Nothing will prevent 100% of vulnerabilities, we are focused on significant reduction of *known, preventable* vulnerabilities

Current State of Best Practice: Software Security

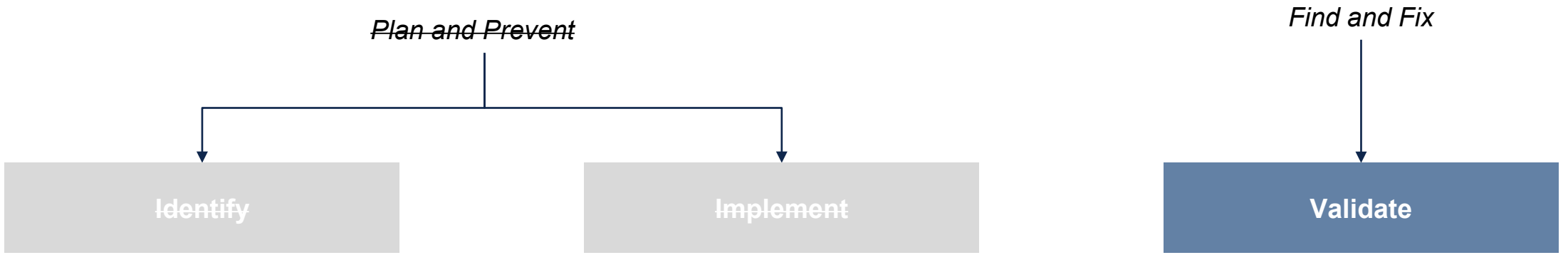


Potential software weaknesses for a given product

Controls to mitigate those weaknesses

Existence of controls and absence of vulnerabilities

Current State of Best Practice: Software Security



~~Potential software weaknesses for a given product~~

~~Controls to mitigate those weaknesses~~

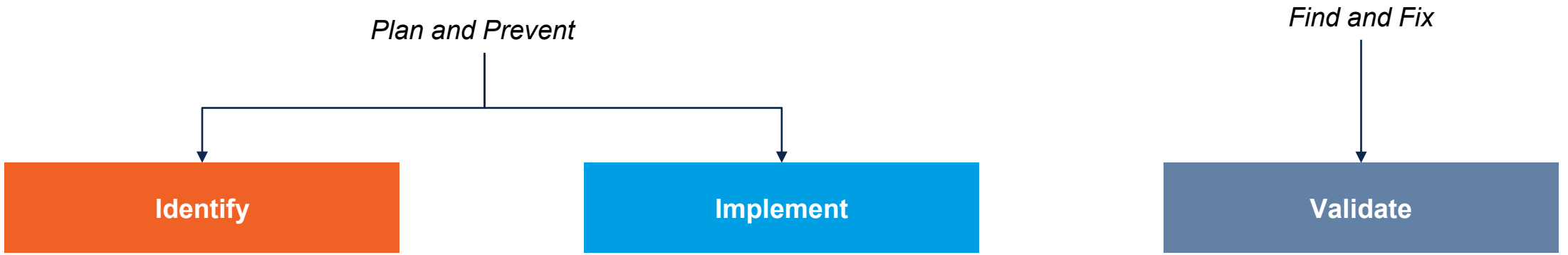
~~Existence of controls and~~ absence of vulnerabilities

Activities like:

- Static, dynamic & interactive application security testing
- Software composition analysis
- Penetration testing and/or bug bounties

The “Find and Fix Addiction”

Ideal State of Best Practice: Software Security



Potential software weaknesses for a given product

Controls to mitigate those weaknesses

Existence of controls and absence of vulnerabilities

Activities like:

- Developer education
- Security requirements
- Threat modeling

Activities like:

- Static, dynamic & interactive application security testing
- Software composition analysis
- Penetration testing and/or bug bounties

A04:2021 – Insecure Design

Factors

CWEs Mapped	Max Incidence Rate	Avg Incidence Rate	Avg Weighted Exploit	Avg Weighted Impact	Max Coverage	Avg Coverage
40	24.19%	3.00%	6.46	6.78	77.25%	42.51%

Overview

A new category for 2021 focuses on risks related to design and architectural flaws, with a call for more use of threat modeling, secure design patterns, and reference architectures. As a community we need to move beyond "shift-left" in the coding space to pre-code activities that are critical for the principles of Secure by Design. Notable Common Weakness Enumerations (CWEs) include *CWE-209: Generation of Error Message Containing Sensitive Information*, *CWE-256: Unprotected Storage of Credentials*, *CWE-501: Trust Boundary Violation*, and *CWE-522: Insufficiently Protected Credentials*.

Description

What Artifacts Should we Aim For?

- Should not perpetuate the “Find and Fix Addiction”, need to incorporate “Plan and Prevent”
- Needs to recognize the current state of software development with DevOps -> certifying a “release” is antiquated
  - Focus on process over focus on releases
- To be useful to the broader public, must be easily understood by a non expert
- Needs to protect vendor IP
- Should be practical to implement (e.g. open source & commercial tool supported)
- Software Bill of Materials (SBOM) – separate discussion

Current State of Best Practice: Software Vendor Security

Enterprise security certifications are not product/software security certifications

Existing Standards Already Implement Holistic Software Security

Industrial Society of Automation 62443 set of Standards

Payment Card Industry: Software Security Framework

NIST: Secure Software Development Framework

ISASecure®

IEC 62443 CONFORMANCE CERTIFICATION

Certifying Industrial Control System Devices and Systems

HOME ABOUT US CONTACT Search...

JOIN NOW SIGN IN

CERTIFICATION | CURRENT MEMBERS | END USERS | SUPPLIERS | CERTIFICATION BODIES | TEST TOOLS | NEWS / EVENTS | LEARNING CENTER

Home » End Users

IEC 62443-4-1 Certified Development Organizations

IEC 62443-4-2 CERTIFIED COMPONENTS | IEC 62443-3-3 CERTIFIED SYSTEMS | IEC 62443-4-1 CERTIFIED DEVELOPMENT ORGANIZATIONS

PCI Security Standards Council®

Contact Change Your Language

Get Started | Assessors & Solutions | Document Library | Training & Qualification | About Us | Get Involved | Newsroom | FAQs

SECURE SLC-QUALIFIED SOFTWARE VENDORS

The PCI Secure Software Lifecycle (SLC) Standard is part of the PCI Software Security Framework and helps software vendors to ensure that security is designed and integrated at each stage of the software lifecycle. Software vendors can engage a Secure SLC Assessor to have their SLC assessed and validated for compliance with the Secure SLC Standard. The assessment and validation are documented by the Secure SLC Assessor in a Report on Compliance (ROC). Software vendors that have undergone this validation process are listed on PCI SSC's Secure SLC-Qualified Software Vendors list.

Although the PCI Council reviews these reports for quality

Certification / Labelling Options

- Self attestation to NIST SSDF and/or equivalent software security framework (with penalties for non-compliance)
- ISA & PCI Approach: 3rd party life-cycle assessment by accredited auditors

We have an opportunity to measurably improve cybersecurity posture **forever**

Thank You

For more information, contact us at www.securitycompass.com

SecurityCompass