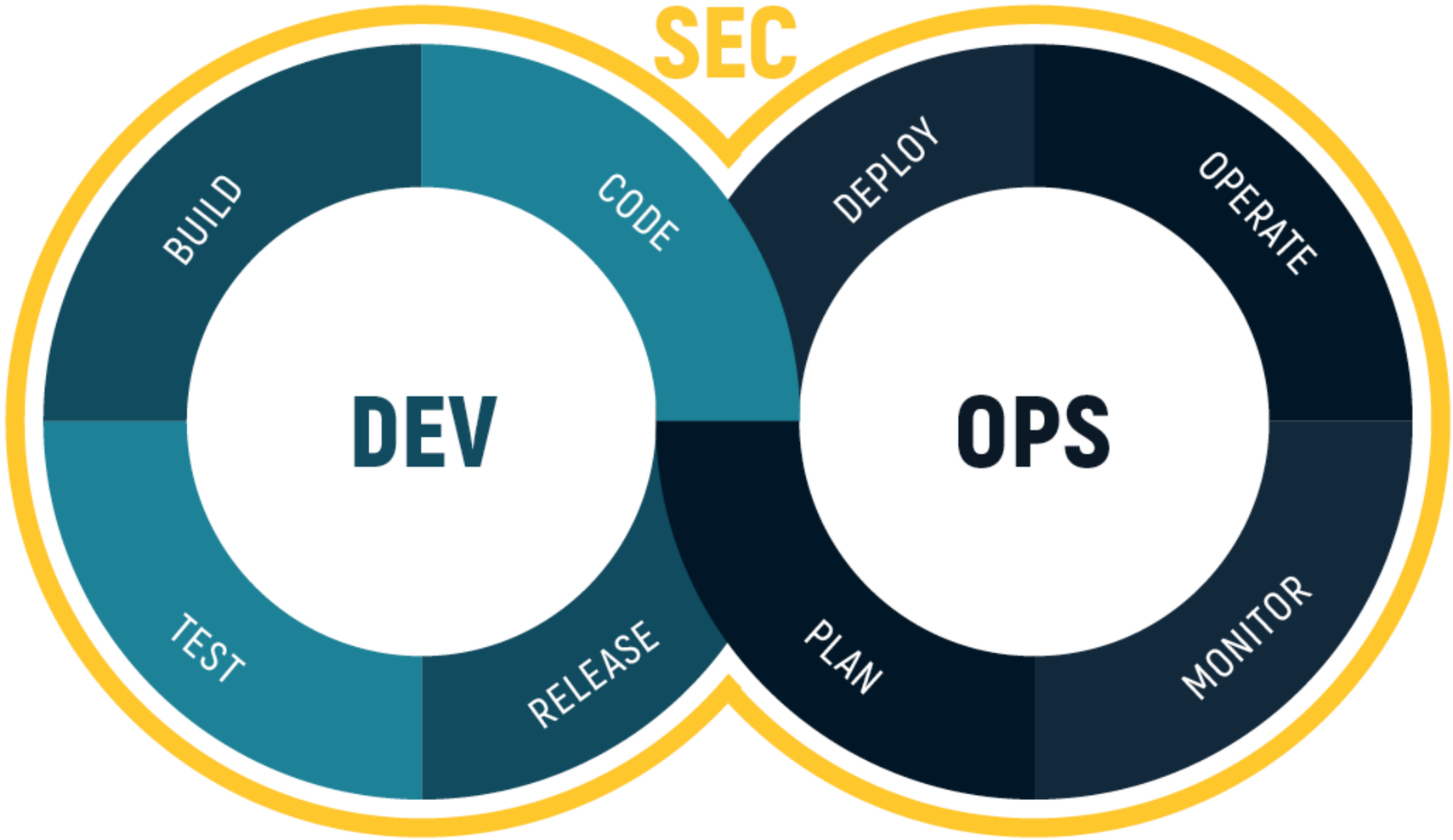
Embedded Security Testing and Automation

Coen De Roover - May 15th, 2025

Need for security testing in every DevOps stage



Application code can be vulnerable …

Common Weakness Enumeration
A Community-Developed List of Software Weakness Types

CWE and SANS Institute TOP 25 MOST DANGEROUS SOFTWARE ERRORS

Home > CWE List > CWE- Individual Dictionary Definition (3.2) ID Lookup: Go

Home | About | CWE List | Scoring | Community | News | Search

CWE-798: Use of Hard-coded Credentials

Weakness ID: 798
Abstraction: Base
Structure: Simple
Status: Incomplete

Presentation Filter: Complete

Description

The software contains hard-coded credentials, such as a password or cryptographic key, which it uses for its own inbound authentication, outbound communication to external components, or encryption of internal data.

Extended Description

Hard-coded credentials typically create a significant hole that allows an attacker to bypass the authentication that has been configured by the software administrator. This hole might be difficult for the system administrator to detect. Even if detected, it can be difficult to fix, so the administrator may be forced into disabling the product entirely. There are two main variations:

Inbound: the software contains an authentication mechanism that checks the input credentials against a hard-coded set of credentials.
Outbound: the software connects to another system or component, and it contains hard-coded credentials for connecting to that component.

In the Inbound variant, a default administration account is created, and a simple password is hard-coded into the product and associated with that account. This hard-coded password is the same for each installation of the product, and it usually cannot be changed or disabled by system administrators without manually modifying the program, or otherwise patching the software. If the password is ever discovered or published (a common occurrence on the Internet), then anybody with knowledge of this password can access the product. Finally, since all installations of the software will have the same password, even across different organizations, this enables massive attacks such as worms to take place.

The Outbound variant applies to front-end systems that authenticate with a back-end service. The back-end service may require a fixed password which can be easily discovered. The programmer may simply hard-code those back-end credentials into the front-end software. Any user of that program may be able to extract the password. Client-side systems with hard-coded passwords pose even more of a threat, since the extraction of a password from a binary is usually very simple.

Relationships

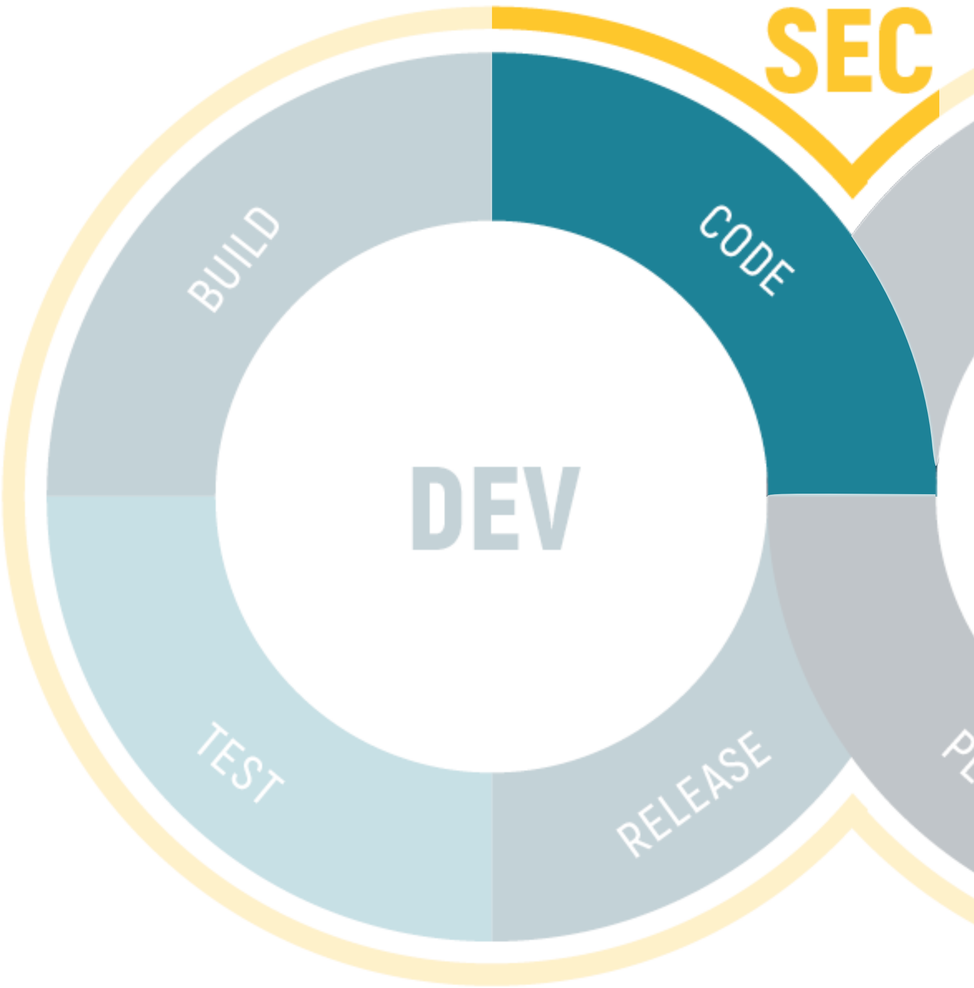
The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOf and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that the user may want to explore.

- ***Relevant to the view "Research Concepts" (CWE-1000)***
- ***Relevant to the view "Architectural Concepts" (CWE-1008)***
- ***Relevant to the view "Development Concepts" (CWE-699)***

Modes Of Introduction

The different Modes of Introduction provide information about how and when this weakness may be introduced. The Phase identifies a point in the software life cycle at which introduction may occur, while the Note provides a typical scenario related to introduction during the given phase.

Phase	Note
Architecture and Design	REALIZATION: This weakness is caused during implementation of an architectural security tactic.



… but so can its build dependencies …

A Year Later, That Brutal Log4j Vulnerability Is Still Lurking

Despite mitigation, one of the worst bugs in internet history is still prevalent—and being exploited.

Popular Python library, urllib3, subject to a denial of service vulnerability

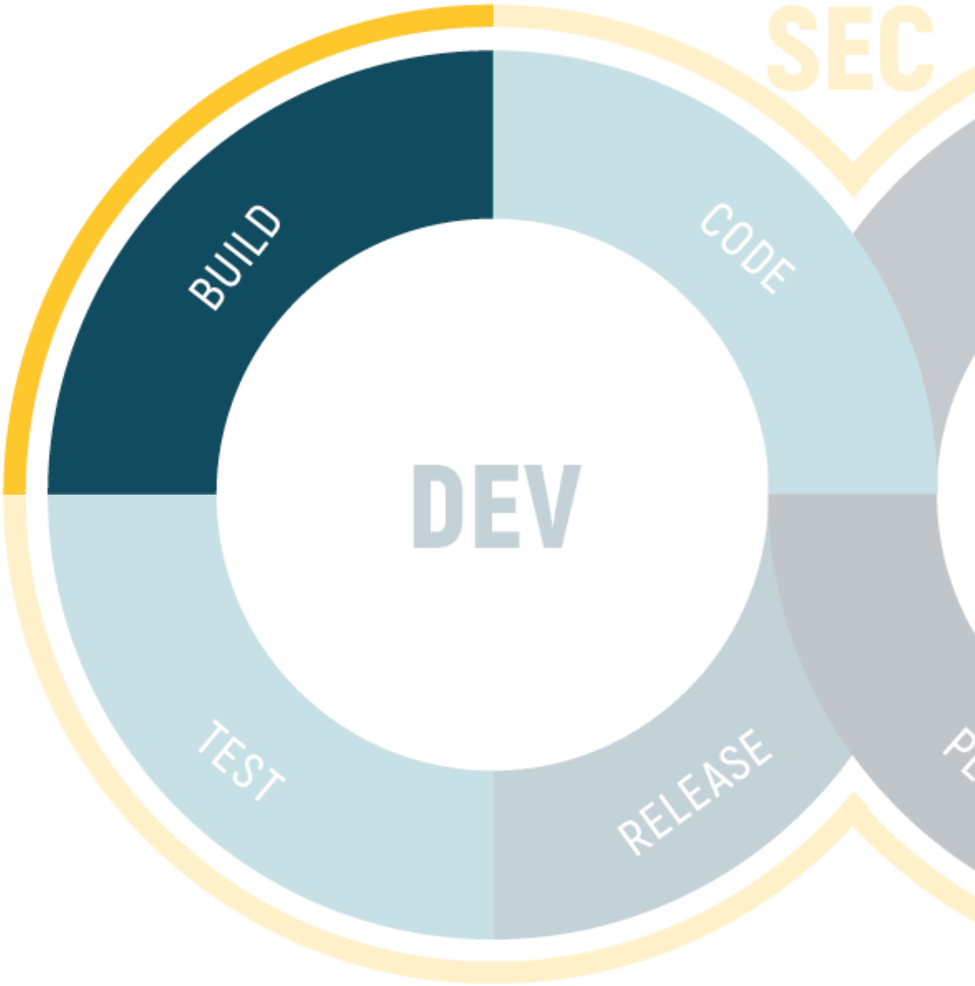
Hayley Denbraver
March 8, 2020

mysticatea/npm-run-all

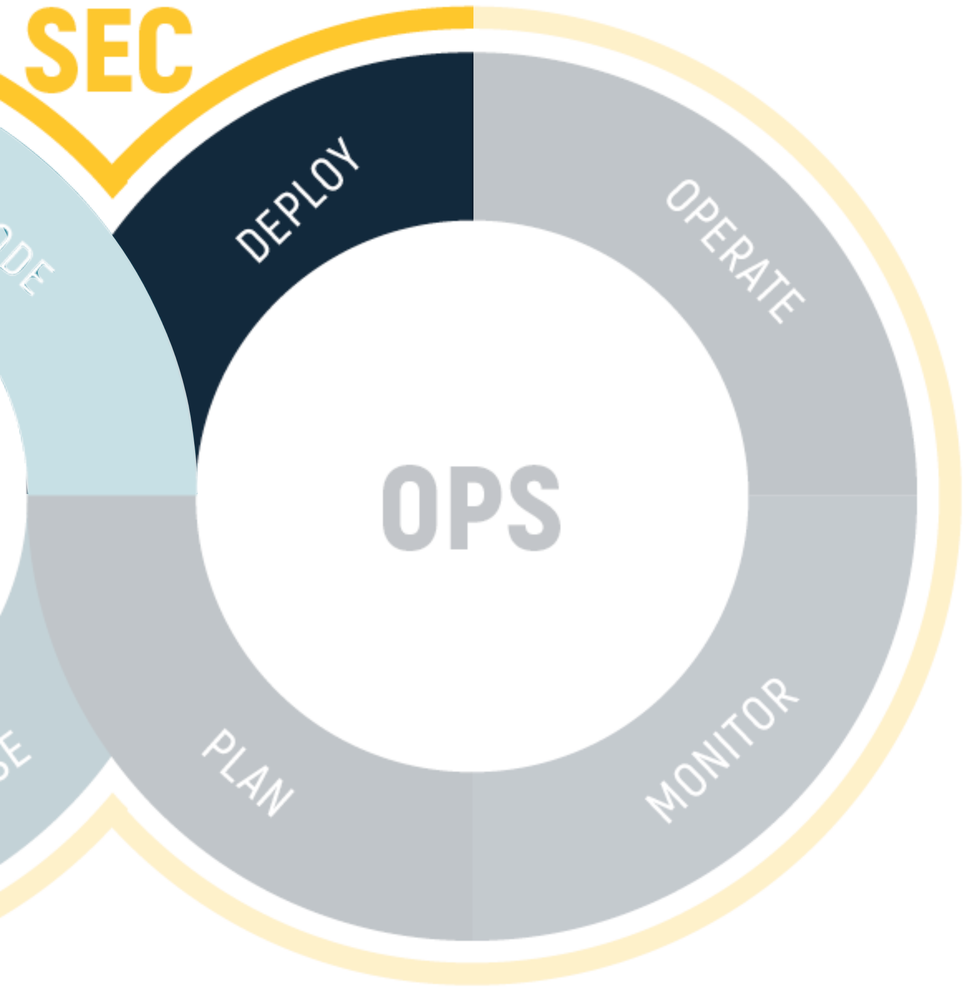
#150 **Vulnerability in event-stream dependency**

3 comments

ChrisBAshton opened on November 26, 2018



… as well as its deployment dependencies …



Heartbleed: Hundreds of thousands of servers at risk from catastrophic bug

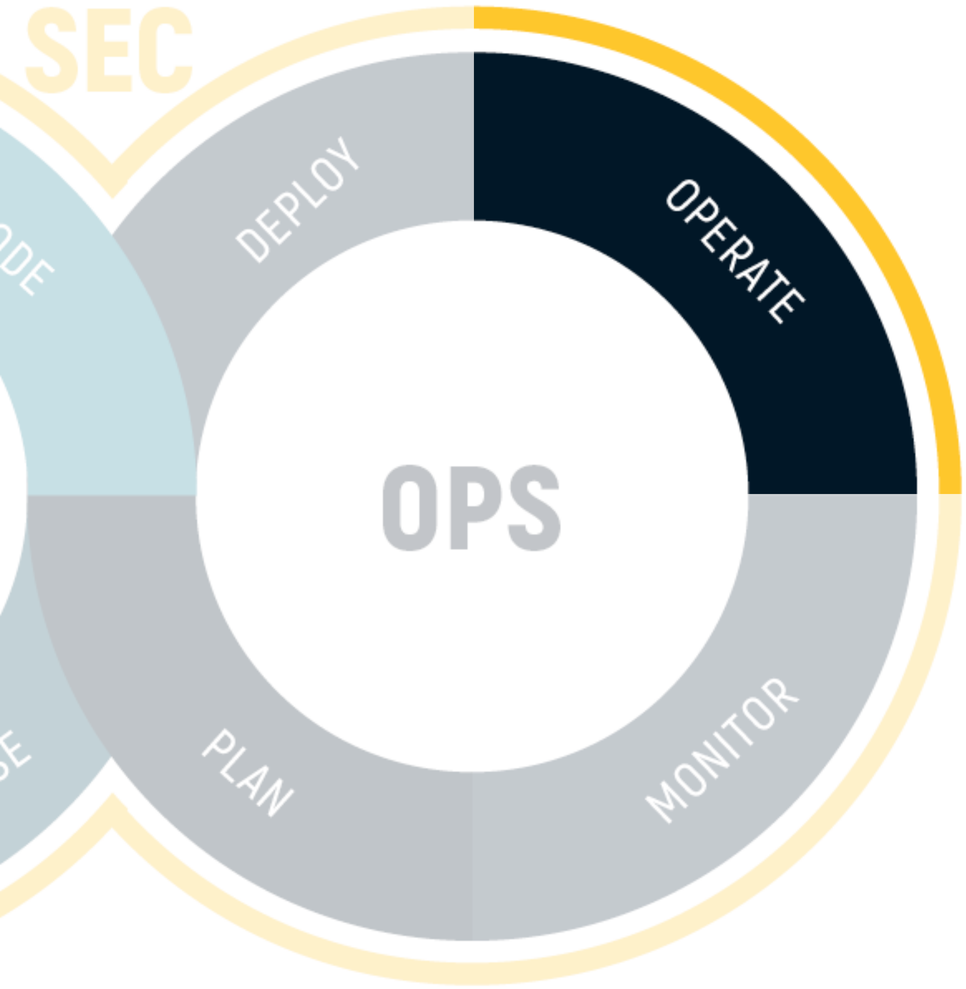
Code error means that websites can leak user details including passwords through 'heartbeat' function used to secure connections

Snyk Blog

High severity vulnerability found in libcurl and curl (CVE-2023-38545)

Written by: Hadas Bloom Tal Dromi Micah Silverman

… and even its deployment configuration!



Default Debian sshd configuration

```
Port 22
ListenAddress 0.0.0.0

LoginGraceTime 2m
MaxAuthTries 6
MaxSessions 10

PermitRootLogin prohibit-password
PasswordAuthentication yes

AllowAgentForwarding yes
AllowTcpForwarding yes
X11Forwarding yes
```

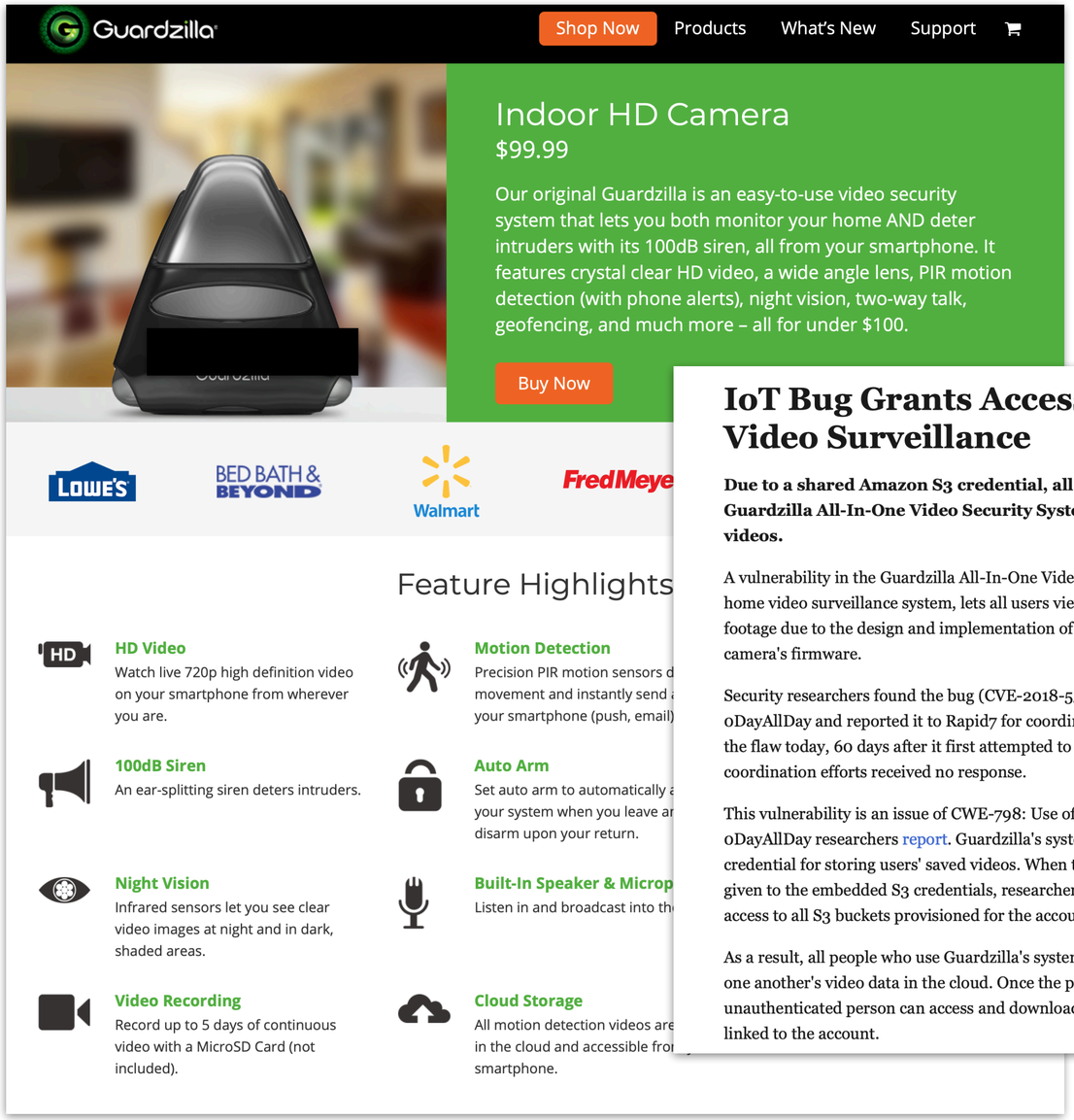
Unrestricted address

Prone to DoS attacks

Weak authentication

Unnecessary features

Embedded software is no exception!



IoT Bug Grants Access to Home Video Surveillance

Due to a shared Amazon S3 credential, all users of a certain model of the Guardzilla All-In-One Video Security System can view each other's videos.

A vulnerability in the Guardzilla All-In-One Video Security System, an IoT-enabled home video surveillance system, lets all users view one another's saved surveillance footage due to the design and implementation of Amazon S3 credentials inside the camera's firmware.

Security researchers found the bug (CVE-2018-5560) during an event held by 0DayAllDay and reported it to Rapid7 for coordinated disclosure. Rapid7 published the flaw today, 60 days after it first attempted to contact the vendor. Multiple coordination efforts received no response.

This vulnerability is an issue of CWE-798: Use of Hard-coded Credentials, 0DayAllDay researchers report. Guardzilla's system uses a shared Amazon S3 credential for storing users' saved videos. When they investigated the access rights given to the embedded S3 credentials, researchers found they provide unlimited access to all S3 buckets provisioned for the account.

As a result, all people who use Guardzilla's system for home surveillance can view one another's video data in the cloud. Once the password is known, any unauthenticated person can access and download stored files and videos in buckets linked to the account.

Embedded S3 Credentials Unlimited Access Policy

Once the binaries were extracted from the firmware they were analyzed in IDA Pro to determine if any vulnerabilities could be identified. Once the main.exe had been disassembled and analyzed it was noted that a set of strings resembled AWS credentials:

```
.rodata:002FF5...  00000015  C  AKIAJQDP34RKL7GGV7OQ
.rodata:002FF6...  00000029  C  igH8yFmmpMbnkcUaCqXJIRIozKVaXaRhE7PWHAYa
.rodata:002FF6...  00000011  C  s3.amazonaws.com
.rodata:002FF6...  00000011  C  motion-detection
```

Following the references, we can see that they are exports from the binary that are labeled: accessKey, secretAccessKey, hostname, and bucket. This format lines up with how AWS bucket keys are designed:

```
.data:00390230                  EXPORT accessKeyIdG
.data:00390230 accessKeyIdG     DCD aAkiajqdp34rkl7     ; DATA XREF: .got:accessKeyIdG_ptr↑o
.data:00390230                                          ; "AKIAJQDP34RKL7GGV7OQ"
.data:00390234                  EXPORT secretAccessKeyG
.data:00390234 secretAccessKeyG DCD aIgh8yFmmpMbnkc     ; DATA XREF: .got:secretAccessKeyG_ptr↑o
.data:00390234                                          ; "igH8yFmmpMbnkcUaCqXJIRIozKVaXaRhE7PWHAY"...
.data:00390238                  EXPORT hostName
.data:00390238 hostName         DCD aS3_amazonaws_c     ; DATA XREF: .got:hostName_ptr↑o
.data:00390238                                          ; "s3.amazonaws.com"
.data:0039023C                  EXPORT bucket
.data:0039023C bucket           DCD aMotionDetectio     ; DATA XREF: aws_video_upload1_thread+188↑o
.data:0039023C                                          ; aws_video_upload1_thread+190↑r ...
```

AccessKeyIdG	AKIAJQDP34RKL7GGV7OQ
secretAccessKeyG	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
hostName	s3.amazonaws.com
bucket	motion-detection

The following script was developed to test the S3 credentials to determine if they were valid as well as determine what access writes the credentials had:

```
import boto3
# Create an S3 client
s3 = boto3.client('s3',aws_access_key_id='AKIAJQDP34RKL7GGV7OQ',aws_secret_access_key
='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',region_name='us-west-1')

try:
    result = s3.get_bucket_policy(Bucket='motion-detection')
    print(result)
except Exception as e:
    print(e)
```

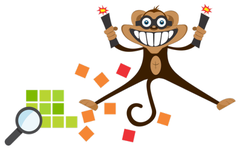
On 29th of September 2018, it was discovered that the **credentials for the cloud storage** of an indoor **camera** were **hardcoded** in its software.

With those credentials, anyone could access the uploaded videos from any customer.

Security testing to the rescue

fuzzing and **concolic** testing of event-driven applications

automated **resilience testing** through fault injection



ask me during the break!

SAST

Static Application Security Testing

SCA

Software Composition Analysis

DAST

Dynamic Application Security Testing

Static application security testing

```
const onClickHandler = () => {
  const $ = document.querySelector;
  let pass = $("#pass").value;
  console.log(pass);
}
```

sink to be avoided

source of sensitive information

- Where is this class instantiated?
- Which code will never be executed?
- Can this acces raise a NullPointerException?
- Can this integer arithmetic overflow?
- **May sensitive information leak outside?**
- …

answer questions about **any execution** of the **program**, **without executing** it

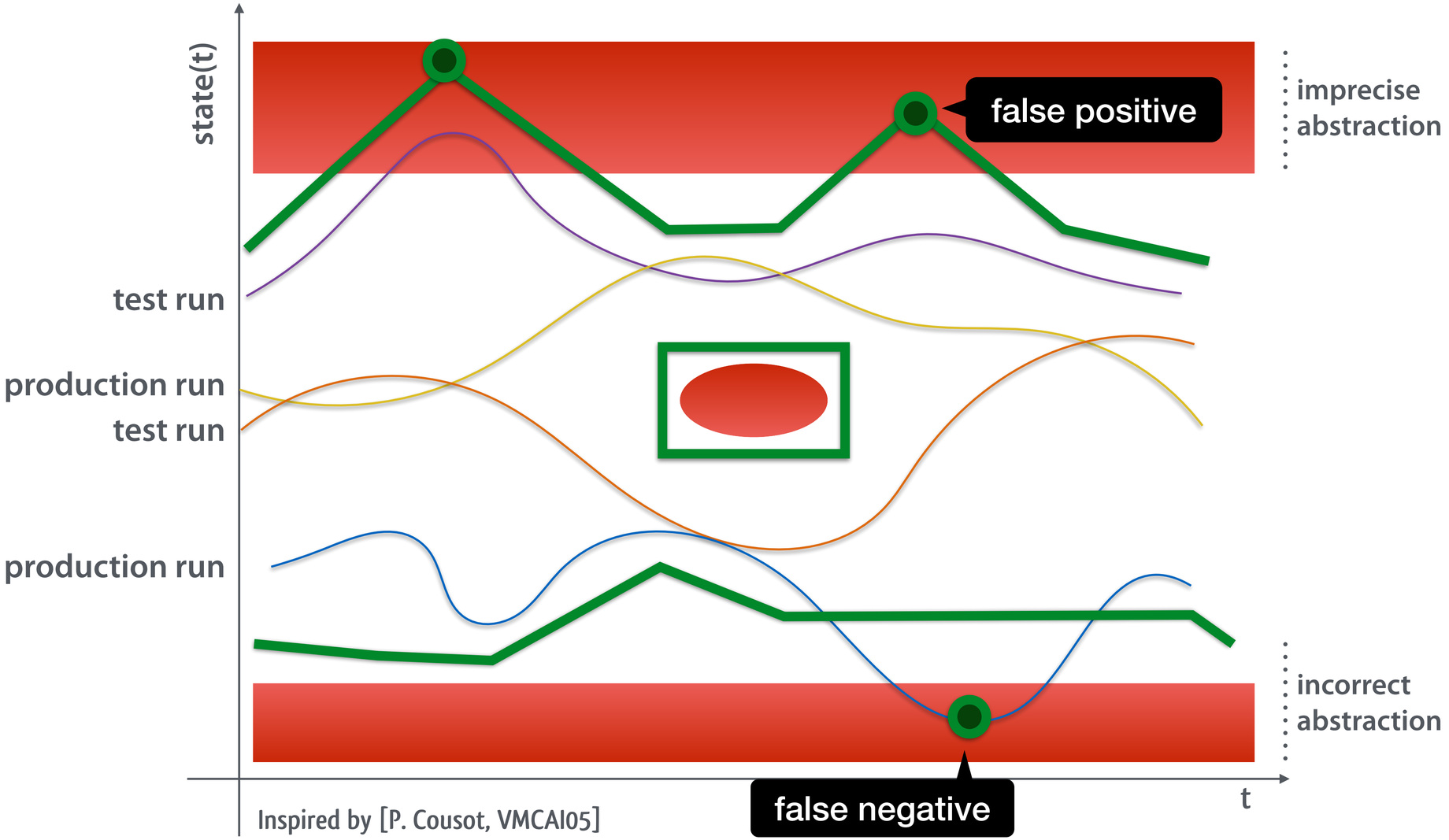
Desiderata

termination

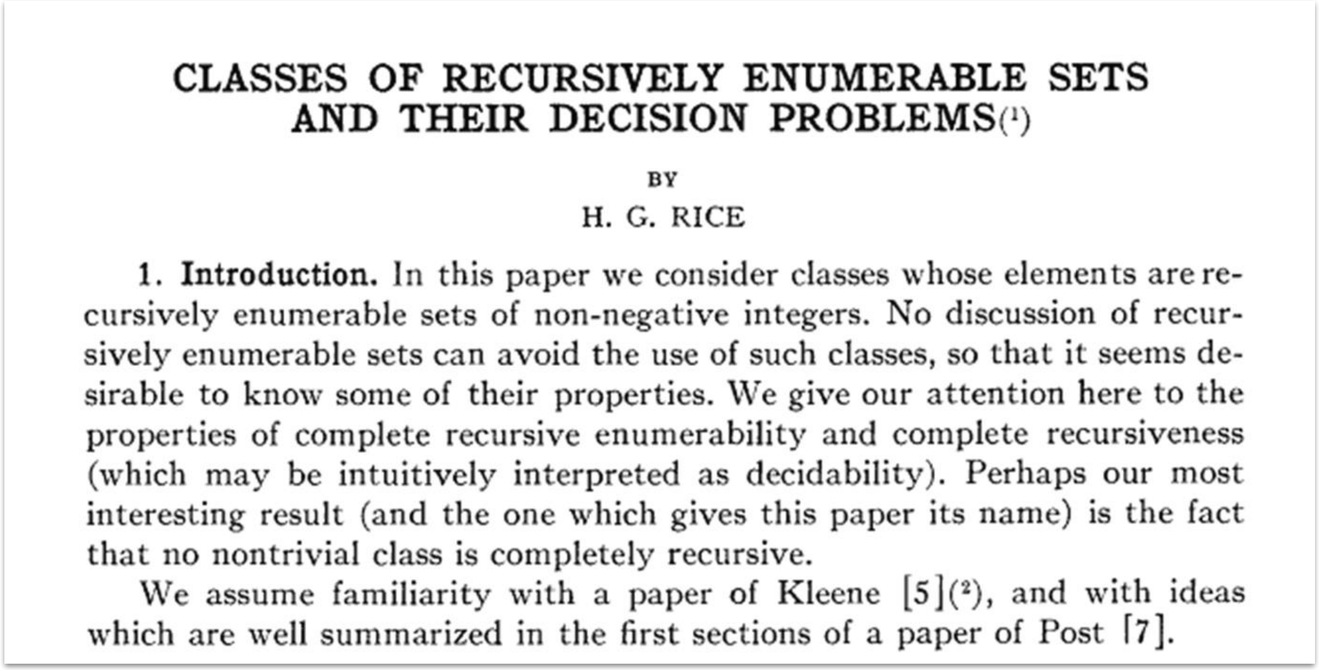
automation

fully precise: no false positives

complete: no false negatives



Just one minor problem



CLASSES OF RECURSIVELY ENUMERABLE SETS AND THEIR DECISION PROBLEMS(1)

BY

H. G. RICE

1. **Introduction.** In this paper we consider classes whose elements are recursively enumerable sets of non-negative integers. No discussion of recursively enumerable sets can avoid the use of such classes, so that it seems desirable to know some of their properties. We give our attention here to the properties of complete recursive enumerability and complete recursiveness (which may be intuitively interpreted as decidability). Perhaps our most interesting result (and the one which gives this paper its name) is the fact that no nontrivial class is completely recursive.

We assume familiarity with a paper of Kleene [5](2), and with ideas which are well summarized in the first sections of a paper of Post [7].

~ “Any non-trivial semantic property about the behaviour of a program in a Turing-complete language is undecidable”

Rice’s theorem, 1953

Solution: compute a
(mostly sound and reasonably precise)
under/over **approximation** of the behaviour
(within a reasonable time budget).

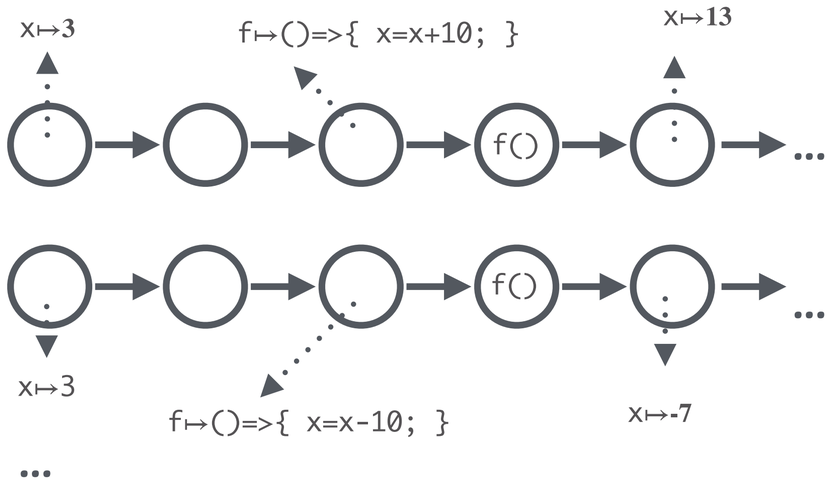
reasonable for the use case at hand

Computing an over-approximation

concrete runs

$$
\begin{aligned}
\varsigma_{\text{CESK}} \in \Sigma_{\text{CESK}} &= \mathit{Control} \times \mathit{Env} \times \mathit{Store} \times \mathit{Addr} \\
\mathit{Control} &= \mathit{Exp} + \mathit{Val} \\
\rho \in \mathit{Env} &= \mathit{Var} \rightharpoonup \mathit{Addr} \\
\sigma \in \mathit{Store} &= \mathit{Addr} \rightharpoonup \mathit{Val} \\
\mathit{val} \in \mathit{Val} &= \mathit{Clo} + \mathit{Kont} \\
\kappa \in \mathit{Kont} &::= \mathbf{halt} \mid \mathbf{ar}(e, \rho, a) \mid \mathbf{fn}(\mathit{clo}, a) \\
\mathit{clo} \in \mathit{Clo} &::= (\lambda v.e) \times \mathit{Env} \\
a, b \in \mathit{Addr} &\quad \text{an infinite set of addresses}
\end{aligned}
$$

$$
\begin{aligned}
\langle v, \rho, \sigma, a, t\rangle &\to \langle \sigma(\rho(v)), \rho, \sigma, a, u\rangle. \\
\langle (\lambda v.e), \rho, \sigma, a, t\rangle &\to \langle ((\lambda v.e), \rho), \rho, \sigma, a, u\rangle. \\
\langle (e_0\ e_1), \rho, \sigma, a, t\rangle &\to \langle e_0, \rho, \sigma[b \mapsto \mathbf{ar}(e_1, \rho, a)], b, u\rangle. \\
\langle \mathit{clo}, \rho, \sigma, a, t\rangle &\to \langle e, \rho', \sigma[b \mapsto \mathbf{fn}(\mathit{clo}, a')], b, u\rangle \ \text{ if } \kappa = \mathbf{ar}(e, \rho', a'), \\
\langle \mathit{val}, \rho, \sigma, a, t\rangle &\to \langle e, \rho'[v \mapsto b], \sigma[b \mapsto \mathit{val}], a', u\rangle \ \text{ if } \kappa = \mathbf{fn}((\lambda v.e), \rho'), a').
\end{aligned}
$$



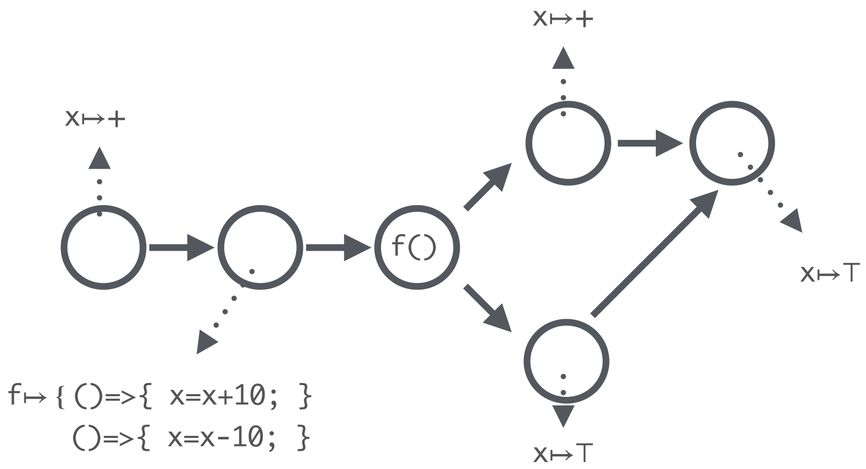
our preferred approach

systematic, powerful, and applicable to modern languages

abstract model

$$
\begin{aligned}
\hat{\varsigma}_{\text{CESK}} \in \hat{\Sigma}_{\text{CESK}} &= \widehat{\mathit{Control}} \times \widehat{\mathit{Env}} \times \widehat{\mathit{Store}} \times \widehat{\mathit{Addr}} \\
\hat{c} \in \widehat{\mathit{Control}} &= \mathit{Exp} + \widehat{\mathit{Val}} \\
\hat{\rho} \in \widehat{\mathit{Env}} &= \mathit{Var} \rightharpoonup \widehat{\mathit{Addr}} \\
\hat{\sigma} \in \widehat{\mathit{Store}} &= \widehat{\mathit{Addr}} \rightharpoonup \mathcal{P}(\widehat{\mathit{Val}}) \\
\widehat{\mathit{val}} \in \widehat{\mathit{Val}} &= \widehat{\mathit{Clo}} + \widehat{\mathit{Kont}} \\
\hat{\kappa} \in \widehat{\mathit{Kont}} &::= \mathbf{halt} \mid \mathbf{ar}(e, \hat{\rho}, \hat{a}) \mid \mathbf{fn}(\widehat{\mathit{clo}}, \hat{a}) \\
\widehat{\mathit{val}} \in \widehat{\mathit{Val}} &::= (\lambda v.e) \times \widehat{\mathit{Env}} \\
\hat{a}, \hat{b} \in \widehat{\mathit{Addr}} &\quad \text{a finite set of addresses}
\end{aligned}
$$

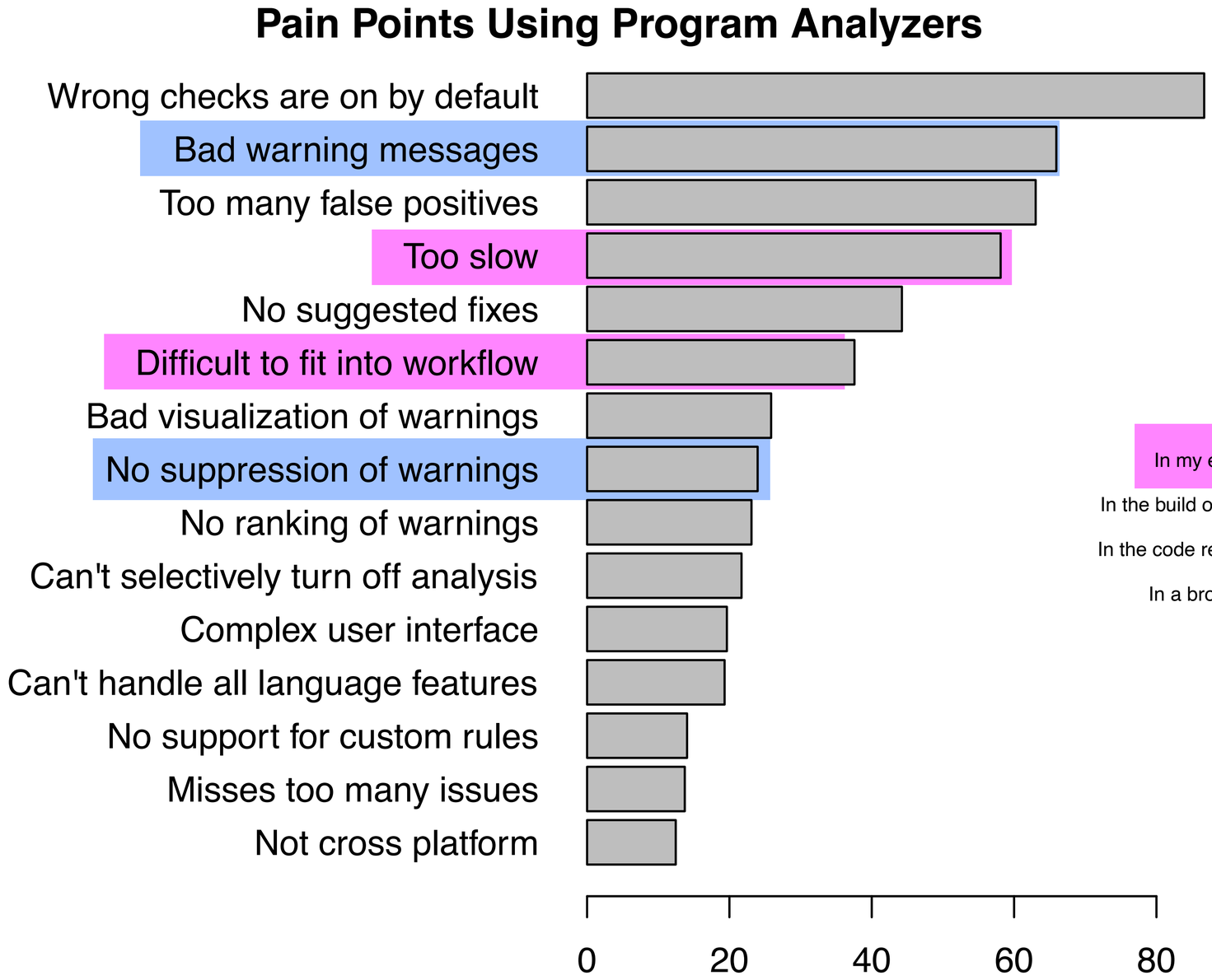
$$
\begin{aligned}
\langle v, \hat{\rho}, \hat{\sigma}, \hat{a}, \hat{t}\rangle &\mathrel{\widehat{\to}} \langle \widehat{\mathit{clo}}, \hat{\rho}', \hat{\sigma}, \hat{a}, \hat{u}\rangle \\
&\quad \text{where } \widehat{\mathit{clo}} \in \hat{\sigma}(\hat{\rho}(v)). \\
\langle (\lambda v.e), \hat{\rho}, \hat{\sigma}, \hat{a}, \hat{t}\rangle &\to \langle ((\lambda v.e), \hat{\rho}), \hat{\rho}, \hat{\sigma}, \hat{a}, \hat{u}\rangle \\
\langle (e_0\ e_1), \hat{\rho}, \hat{\sigma}, \hat{a}, \hat{t}\rangle &\mathrel{\widehat{\to}} \langle e_0, \hat{\rho}, \hat{\sigma} \sqcup [b \mapsto \mathbf{ar}(e_1, \hat{\rho}, \hat{a})], \hat{b}, \hat{u}\rangle. \\
\langle \widehat{\mathit{clo}}, \hat{\rho}, \hat{\sigma}, \hat{a}, \hat{t}\rangle &\mathrel{\widehat{\to}} \langle e, \hat{\rho}', \hat{\sigma} \sqcup [b \mapsto \mathbf{fn}(\widehat{\mathit{clo}}, \hat{a}')], \hat{b}, \hat{u}\rangle \ \text{ if } \hat{\kappa} = \mathbf{ar}(e, \hat{\rho}', \hat{a}'), \\
\langle \mathit{val}, \hat{\rho}, \hat{\sigma}, \hat{a}, \hat{t}\rangle &\mathrel{\widehat{\to}} \langle e, \hat{\rho}'[v \mapsto \hat{b}], \hat{\sigma} \sqcup [\hat{b} \mapsto \mathit{val}], \hat{a}', \hat{u}\rangle \ \text{ if } \hat{\kappa} = \mathbf{fn}((\lambda v.e), \hat{\rho}'), \hat{a}').
\end{aligned}
$$

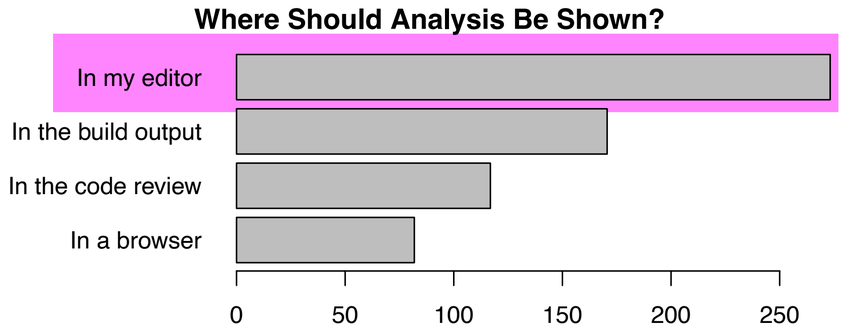


$\widehat{-}$	$\bot$	$\emptyset$	-	+	$\top$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$\emptyset$	$\bot$	$\emptyset$	+	-	$\top$
-	$\bot$	-	$\top$	-	$\top$
+	$\bot$	+	+	$\top$	$\top$
$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$

What (375 Microsoft) developers want: shift left

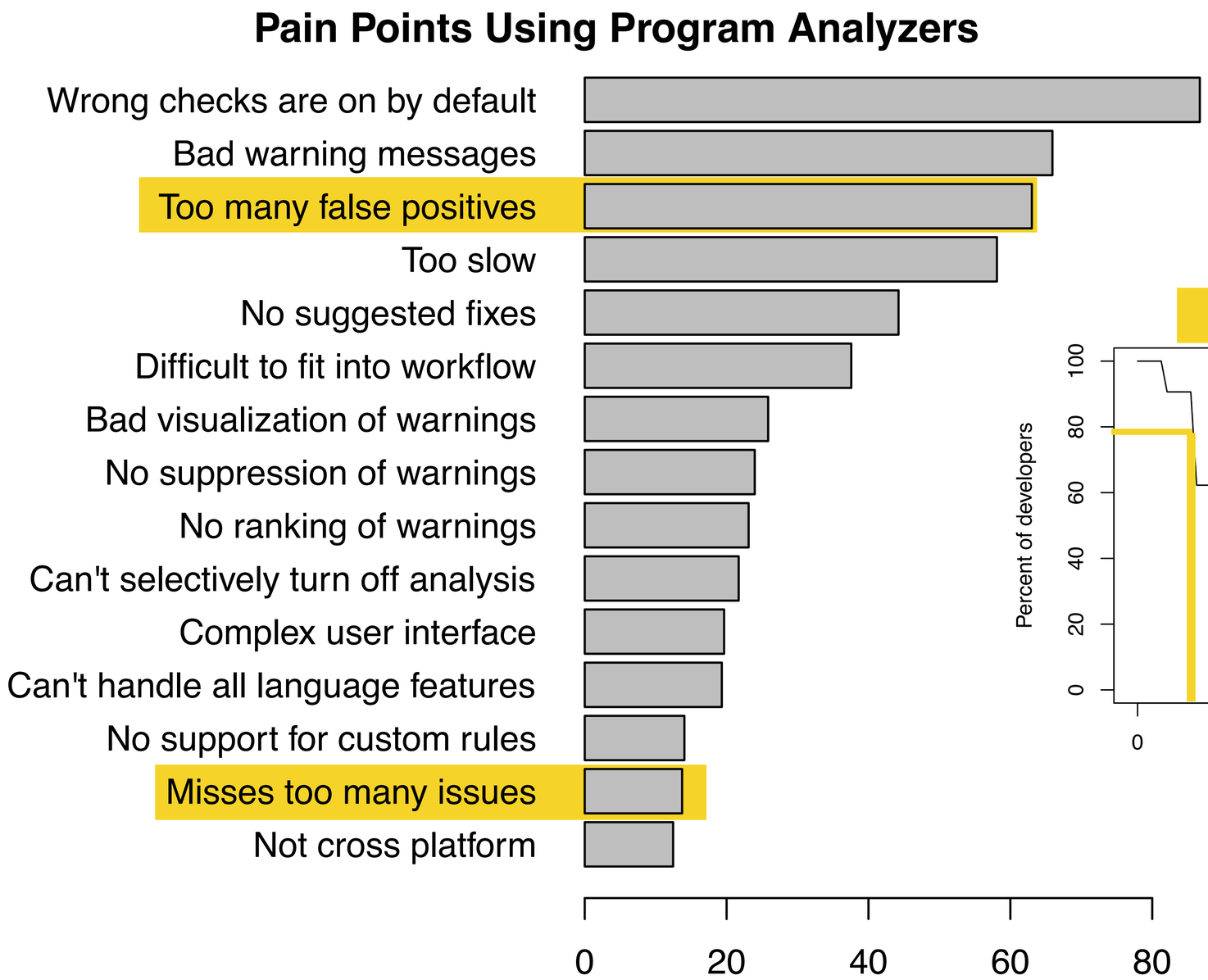
What developers want and need from program analysis: an empirical study, Christakis et al. [ASE16]

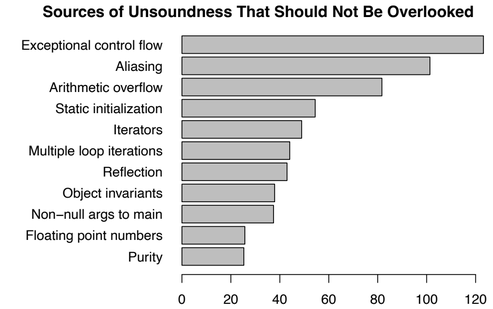


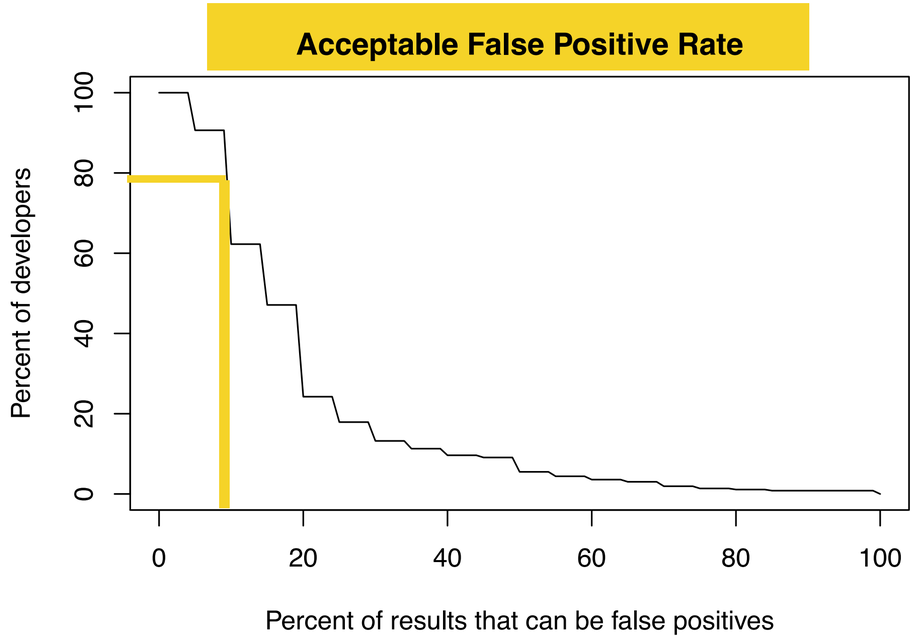


What (375 Microsoft) developers want: precision

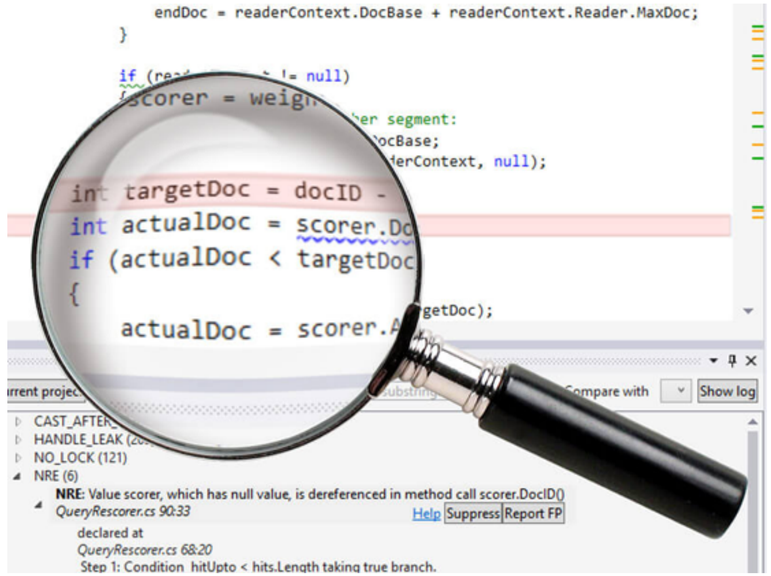
What developers want and need from program analysis: an empirical study, Christakis et al. [ASE16]

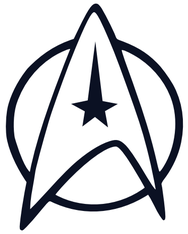






Our own research into SAST

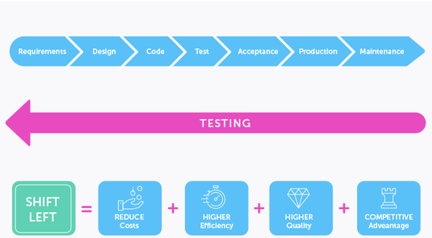




“to boldy go where no SAST tool has gone before”

binaries

deployment automation



analysis designs for “shift left”

- **Static Stack-Preserving Intra-Procedural Slicing of WebAssembly Binaries**
  Quentin Stiévenart, Dave Binkley and Coen De Roover – ICSE 2022
- **Compositional Information Flow Analysis for WebAssembly Programs**
  Quentin Stiévenart and Coen De Roover – SCAM 2020
- **Control and Data Flow in Security Smell Detection for Infrastructure as Code: Is It Worth the Effort?**
  Ruben Opdebeeck, Ahmed Zerouali and Coen De Roover - MSR 2023

- **Result Invalidation for Incremental Modular Analyses**
  Jens Van der Plas, Quentin Stiévenart and Coen De Roover – VMCAI 2023
- **Change Pattern Detection for Optimising Incremental Static Analysis**
  Cindy Wauters, Jens Van der Plas, Quentin Stiévenart and Coen De Roover – SCAM 2023
- **A Parallel Worklist Algorithm and Its Exploration Heuristics for Static Modular Analyses**
  Quentin Stiévenart, Noah Van Es, Jens Van der Plas, and Coen De Roover - In JSS 2021

Deployment automation

```yaml
- name: Ensure default user account is deleted
  user:
    name: "{{ default_user_name }}"
    state: absent
  when: default_user_name is defined
  become: true

- name: Ensure default user group is deleted
  group:
    name: "{{ default_user_group }}"
    state: absent
  when: default_user_group is defined

- name: Ensure root password is changed
  user:
    name: root
    password: >-
      {{
        root_password
        | password_hash(
          'sha512',
          65534 | random(seed=inventory_hostname) | string)
      }}
    update_password: always
    state: present
  become: true

- name: Ensure sshd configuration is correct
  template:
    src: sshd_config.j2
    dest: /etc/ssh/sshd_config
    owner: root
    group: root
    mode: 0600
  become: true
  notify: restart sshd

- include_role:
    name: Oefenweb.ufw
    apply:
      become: true
  vars:
    ufw_logging: "on"
    ufw_rules:
      - rule: allow
        direction: in
        to_port: "{{ ansible_port }}"
        protocol: tcp
        interface: "{{ ansible_default_ipv4.interface }}"
        comment: Allow incoming SSH traffic

- include_role:
    name: fail2ban
```

puppet

ANSIBLE

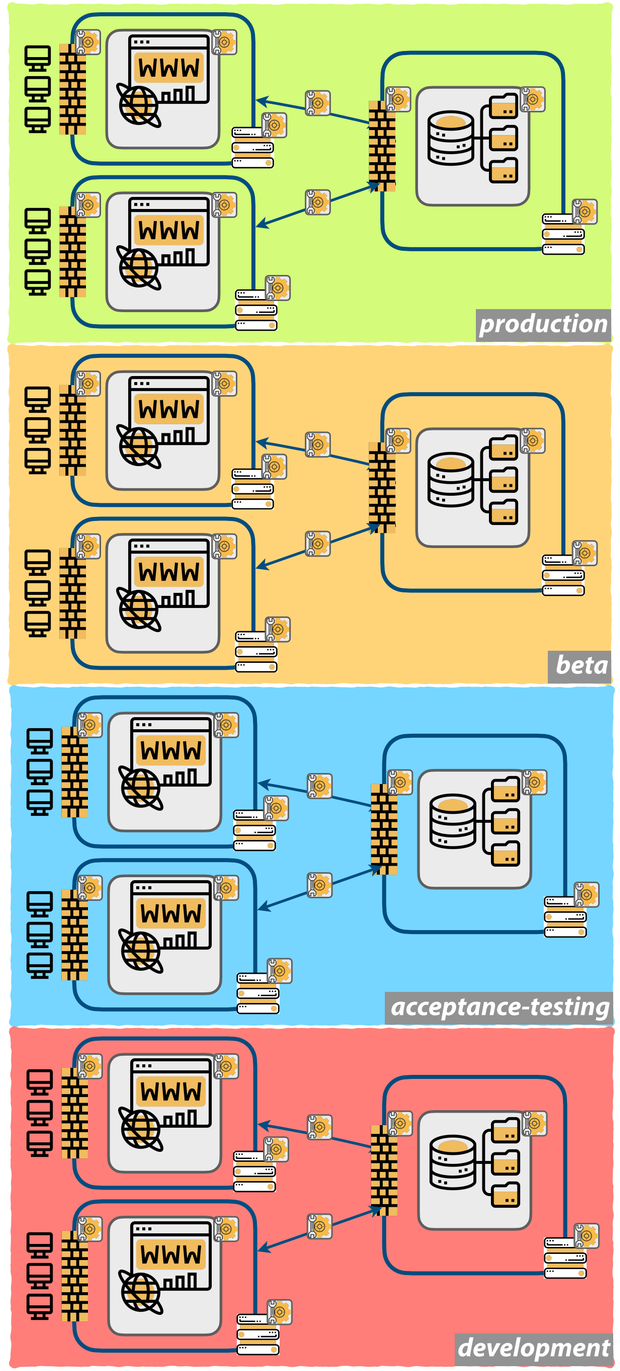
HashiCorp Terraform

pulumi

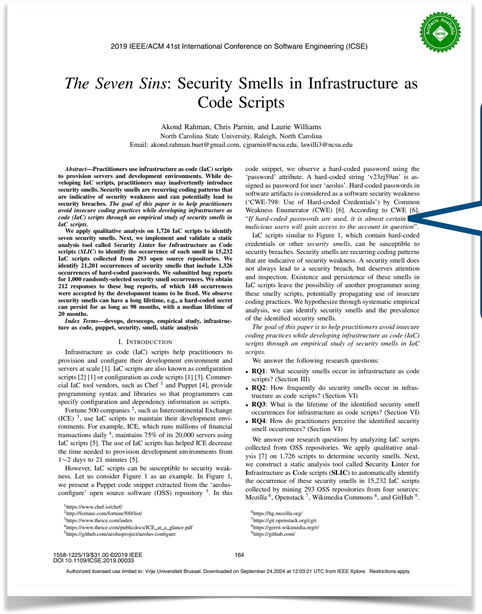
Provision

Configure

Manage



Security Problems in IaC



2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)

The Seven Sins: Security Smells in Infrastructure as Code Scripts

Akond Rahman, Chris Parnin, and Laurie Williams
North Carolina State University, Raleigh, North Carolina
Email: akond.rahman.buet@gmail.com, cjparnin@ncsu.edu, lawilli3@ncsu.edu

Abstract*—Practitioners use infrastructure as code (IaC) scripts to provision servers and development environments. While developing IaC scripts, practitioners may inadvertently introduce security smells. Security smells are recurring coding patterns that are indicative of security weakness and can potentially lead to security breaches. *The goal of this paper is to help practitioners avoid insecure coding practices while developing infrastructure as code (IaC) scripts through an empirical study of security smells in IaC scripts.

We apply qualitative analysis on 1,726 IaC scripts to identify seven security smells. Next, we implement and validate a static analysis tool called Security *Linter* for *I*nfrastructure as Code scripts (*SLIC*) to identify the occurrence of each smell in 15,232 IaC scripts collected from 293 open source repositories. We identify 21,201 occurrences of security smells that include 1,326 occurrences of hard-coded passwords. We submitted bug reports for 1,000 randomly-selected security smell occurrences. We obtain 212 responses to these bug reports, of which 148 occurrences were accepted by the development teams to be fixed. We observe security smells can have a long lifetime, e.g., a hard-coded secret can persist for as long as 98 months, with a median lifetime of 20 months.

***Index Terms*—devops, devsecops, empirical study, infrastructure as code, puppet, security, smell, static analysis**

I. Introduction

Infrastructure as code (IaC) scripts help practitioners to provision and configure their development environment and servers at scale [1]. IaC scripts are also known as configuration scripts [2] [1] or configuration as code scripts [1] [3]. Commercial IaC tool vendors, such as Chef [1] and Puppet [4], provide programming syntax and libraries so that programmers can specify configuration and dependency information as scripts.

Fortune 500 companies [2], such as Intercontinental Exchange (ICE) [3], use IaC scripts to maintain their development environments. For example, ICE, which runs millions of financial transactions daily [4], maintains 75% of its 20,000 servers using IaC scripts [5]. The use of IaC scripts has helped ICE decrease the time needed to provision development environments from 1~2 days to 21 minutes [5].

However, IaC scripts can be susceptible to security weakness. Let us consider Figure 1 as an example. In Figure 1, we present a Puppet code snippet extracted from the 'aeolus-configure' open source software (OSS) repository [5]. In this code snippet, we observe a hard-coded password using the 'password' attribute. A hard-coded string '=23q59ur' is assigned as password for user 'aeolus'. Hard-coded passwords in software artifacts is considered as a software security weakness ('CWE-798: Use of Hard-coded Credentials') by Common Weakness Enumerator (CWE) [6]. According to CWE [6], *"If hard-coded passwords are used, it is almost certain that malicious users will gain access to the account in question"*.

IaC scripts similar to Figure 1, which contain hard-coded credentials or other *security smells*, can be susceptible to security breaches. Security smells are recurring coding patterns that are indicative of security weakness. A security smell does not always lead to a security breach, but deserves attention and inspection. Existence and persistence of these smells in IaC scripts leave the possibility of another programmer using these smelly scripts, potentially propagating use of insecure coding practices. We hypothesize through systematic empirical analysis, we can identify security smells and the prevalence of the identified security smells.

The goal of this paper is to help practitioners avoid insecure coding practices while developing infrastructure as code (IaC) scripts through an empirical study of security smells in IaC scripts.

We answer the following research questions:

- **RQ1**: What security smells occur in infrastructure as code scripts? (Section III)
- **RQ2**: How frequently do security smells occur in infrastructure as code scripts? (Section VI)
- **RQ3**: What is the lifetime of the identified security smell occurrences for infrastructure as code scripts? (Section VI)
- **RQ4**: How do practitioners perceive the identified security smell occurrences? (Section VI)

We answer our research questions by analyzing IaC scripts collected from OSS repositories. We apply qualitative analysis [7] on 1,726 scripts to determine security smells. Next, we construct a static analysis tool called Security Linter for Infrastructure as Code scripts (**SLIC**) to automatically identify the occurrence of these security smells in 15,232 IaC scripts collected by mining 293 OSS repositories from four sources: Mozilla [8], Openstack [9], Wikimedia Commons [10], and GitHub [11].

[1]https://www.chef.io/chef/
[2]http://fortune.com/fortune500/ice/
[3]https://www.theice.com/index
[4]https://www.theice.com/publicdocs/ICE_at_a_glance.pdf
[5]https://github.com/aeolusproject/aeolus-configure
[8]https://hg.mozilla.org/
[9]https://git.openstack.org/cgit
[10]https://gerrit.wikimedia.org/r/
[11]https://github.com/

1558-1225/19/$31.00 ©2019 IEEE
DOI 10.1109/ICSE.2019.00033

21,201 security weaknesses *across 293 IaC repositories*

Twilio Security Incident Shows Danger of Misconfigured S3 Buckets

Twilio says attackers accessed its misconfigured cloud storage system and altered a copy of the JavaScriptSDK it shares with customers.

Software Supply Chain Attacks To Cost The World $60 Billion By 2025

POLICY AND LEGISLATION | Publication 15 September 2022

European Commission

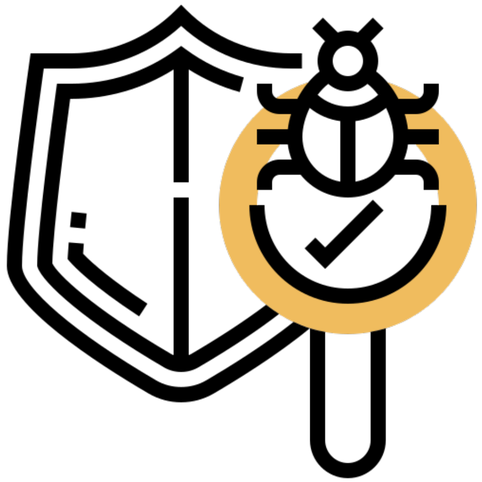
Cyber Resilience Act

The proposal for a regulation on cybersecurity requirements for products with digital elements, known as the Cyber Resilience Act, bolsters cybersecurity rules to ensure more secure hardware and software products.

Our Solutions

GASEL

***G**raph-based **A**nsible **Se**curity **L**inter*



Deep static application security testing (SAST)

GASEL: Security Smell Detection

```
basicauth_enabled: False
basicauth_username: wunder
basicauth_password: wunder123
basicauth_ip:
  - address: 127.0.0.1
  - address: 192.168.0.1
```

Hardcoded secret!

```
- name: Yum install Zabbix release
  yum:
    name: >-
      http://repo.zabbix.com/zabbix/3.0/rhel/
      {{ ansible_distribution_major_version }}
      /x86_64/zabbix-release-3.0-1.el
      {{ ansible_distribution_major_version }}
      .noarch.rpm
  when: ansible_os_family == "RedHat"
```

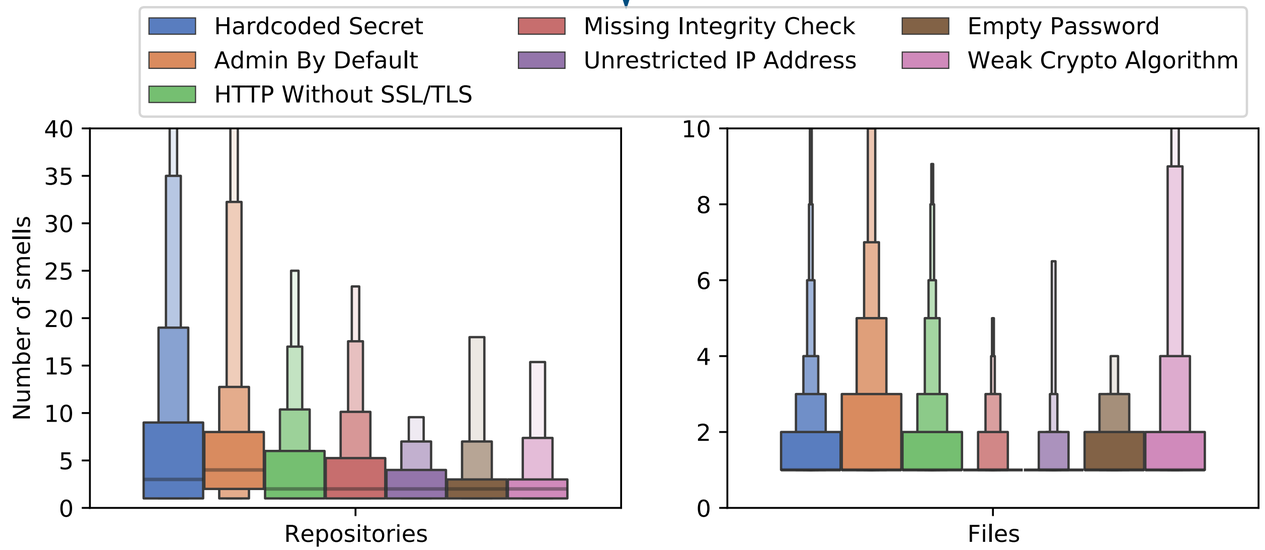
HTTP Without TLS/SSL!

No integrity check!

GASEL: Empirical Evaluation

	GASEL	
Smell type	**% P**	**% R**
ADMIN	98.11	**81.25**
EMPTY	**44.44**	**80.00**
HTTPS	**100.00**	**88.57**
SECRET	45.45	**90.91**
INTEGRITY	**96.15**	**92.59**
IP	76.60	76.60
CRYPTO	**97.67**	**95.45**

Robust: Applied on 25K open-source repositories



GASEL is highly effective and detected security smells are pervasive

GASEL report

MissingIntegrityCheck

The integrity of source code needs to be checked with cryptographic hashes after downloading
Source location: /Users/ruben/Desktop/AnsibleRepos/drupal-vm/provisioning/roles/geerlingguy.php-tideways/defaults/main.yml:4:24

```
---
workspace: /root

4 tideways_download_url: https://github.com/tideways/php-xhprof-extension/archive/master.zip
tideways_download_folder_name: php-xhprof-extension-master
tideways_extension_name: tideways_xhprof.so

# If you use the Tideways UI, set this variable to your API key. Otherwise the
# extension can be used along with the XHProf UI to view profiles.
```

source

Sink location: /Users/ruben/Desktop/AnsibleRepos/drupal-vm/provisioning/roles/geerlingguy.php-tideways/tasks/main.yml:23:3

```
  with_items:
    - make
    - gcc
    - unzip

23 - name: Download and untar Tideways.
  unarchive:
    src: "{{ tideways_download_url }}"
    dest: "{{ workspace }}"
    copy: false
    creates: "{{ workspace }}/{{ tideways_download_folder_name }}"
```

sink

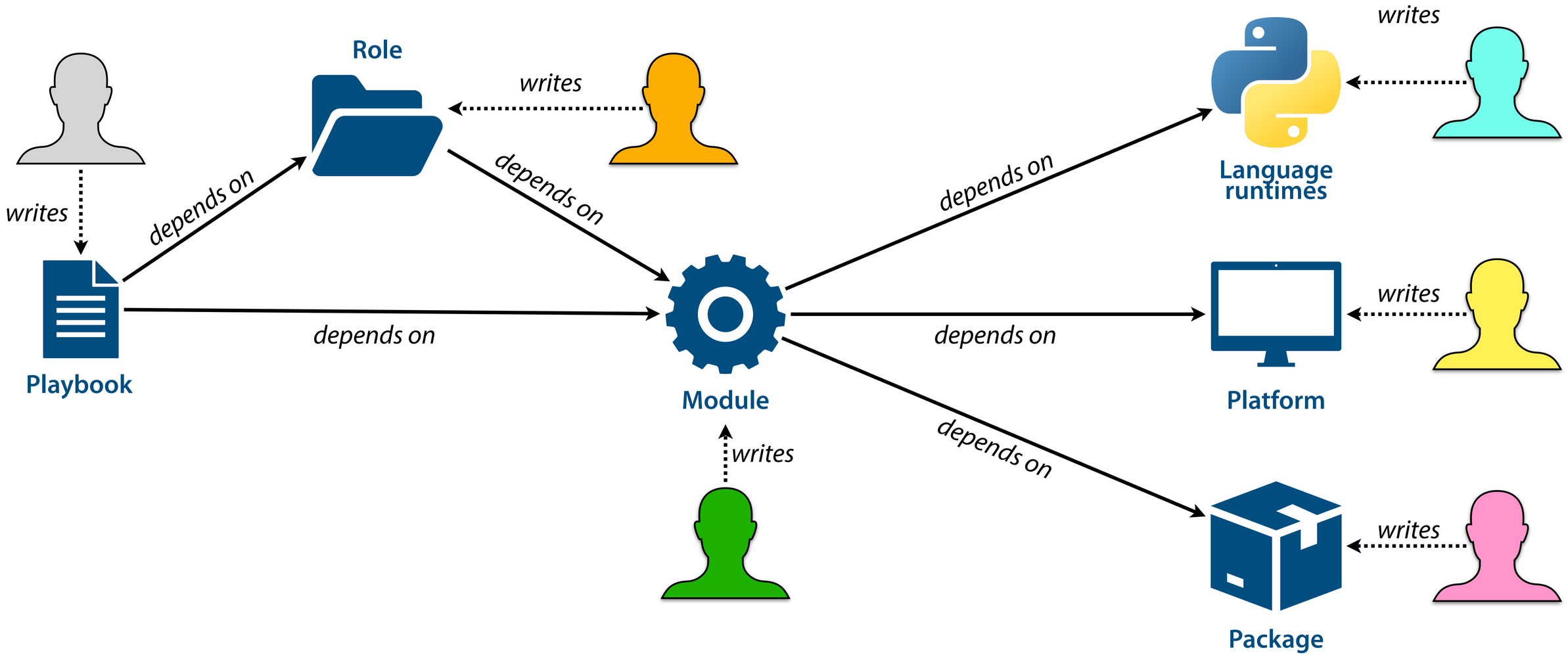
Our Solutions

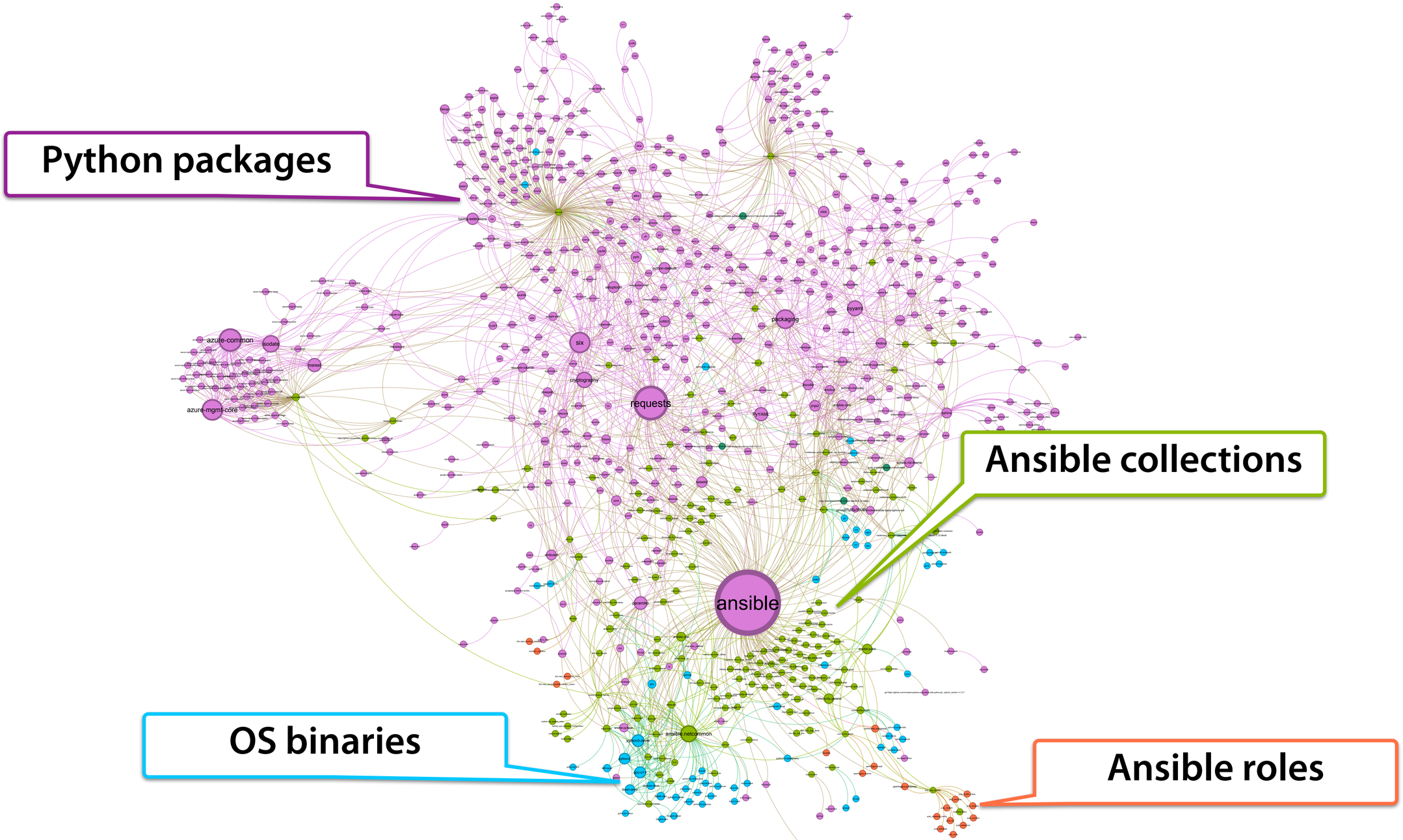
SCAnsible

***S**oftware **C**omposition **An**alysis for **An**sible*

Deep software composition analysis (SCA)

Ansible Software Supply Chain

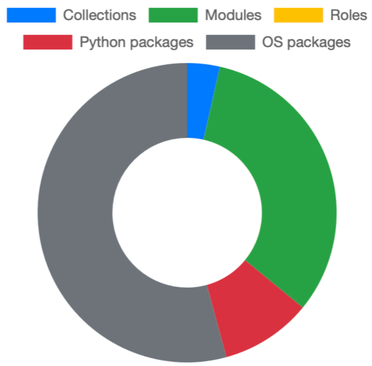



Python packages
Ansible collections
OS binaries
Ansible roles
azure-common
isodate
msrest
azure-mgmt-core
six
cryptography
requests
packaging
pyyaml
flyYAML
ansible
ansible_netcommon

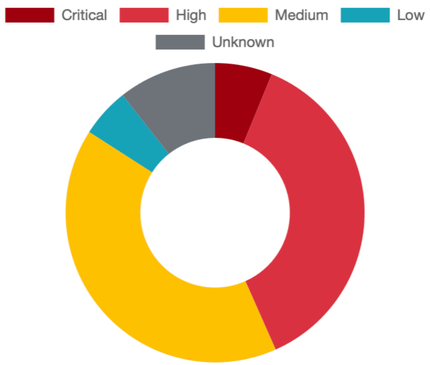
Scansible report

Dependency report for Ansible project "drupal-vm"

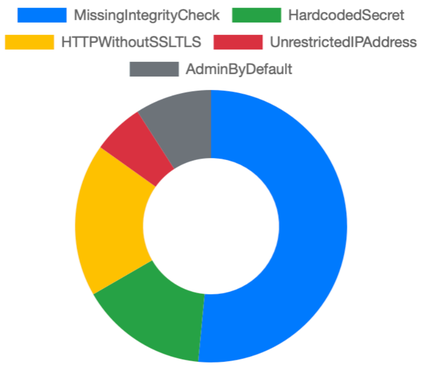
Dependency types



Vulnerabilities



Weaknesses



Top collections

Collection	#usages	#modules
ansible.builtin	487	30
community.general	19	10
community.mysql	10	3
community.postgresql	3	2
ansible.posix	2	1

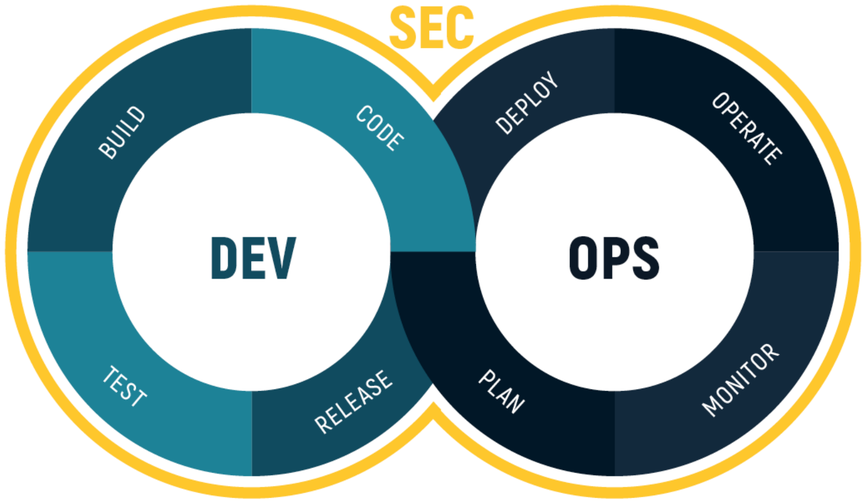
Top modules

Module	#usages
ansible.builtin.file	78
ansible.builtin.command	62
ansible.builtin.template	58
ansible.builtin.apt	50
ansible.builtin.service	47

Top dependencies

Dependency	#usages
lsattr OS	10
chattr OS	9
md5 Python	5
sha Python	5
rpm Python	3

Need for security testing in every DevOps stage



Security testing to the rescue

SAST

Static Application Security Testing

SCA

Software Composition Analysis

DAST

Dynamic Application Security Testing

fuzzing and **concolic** testing of event-driven applications

automated **resilience testing** through fault injection

ask me during the break!

Deployment automation

```
- name: Ensure default user account is deleted
  user:
    name: "{{ default_user_name }}"
    state: absent
  when: default_user_name is defined
  become: true

- name: Ensure default user group is deleted
  group:
    name: "{{ default_user_group }}"
    state: absent
  when: default_user_group is defined

- name: Ensure root password is changed
  user:
    name: root
    password: >-
      {{
        root_password
        | password_hash(
          'sha512',
          65534 | random(seed=inventory_hostname) | string)
      }}
    update_password: always
    state: present
  become: true

- name: Ensure sshd configuration is correct
  template:
    src: sshd_config.j2
    dest: /etc/ssh/sshd_config
    owner: root
    group: root
    mode: 0600
  become: true
  notify: restart sshd

- include_role:
    name: Oefenweb.ufw
  apply:
    become: true
  vars:
    ufw_logging: "on"
    ufw_rules:
      - rule: allow
        direction: in
        to_port: "{{ ansible_port }}"
        protocol: tcp
        interface: "{{ ansible_default_ipv4.interface }}"
        comment: Allow incoming SSH traffic

- include_role:
    name: fail2ban
```

...

puppet

ANSIBLE

Terraform

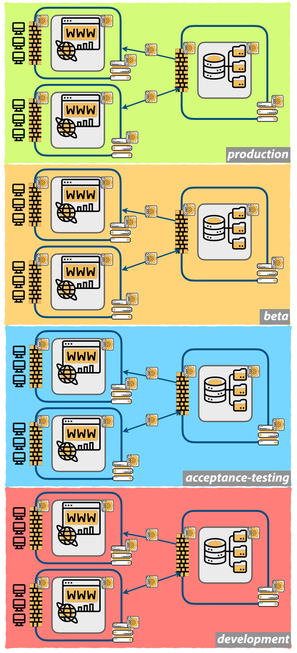
pulumi

...

Provision

Configure

Manage



Our Solutions

GASEL

***G**raph-based **A**nsible **Se**curity **L**inter*

Deep static application security testing (SAST)

SCAnsible

***S**oftware **C**omposition **An**alysis for **An**sible*

Deep software composition analysis (SCA)