

# Infra Security Automation in CI/CD: Impossible is Nothing

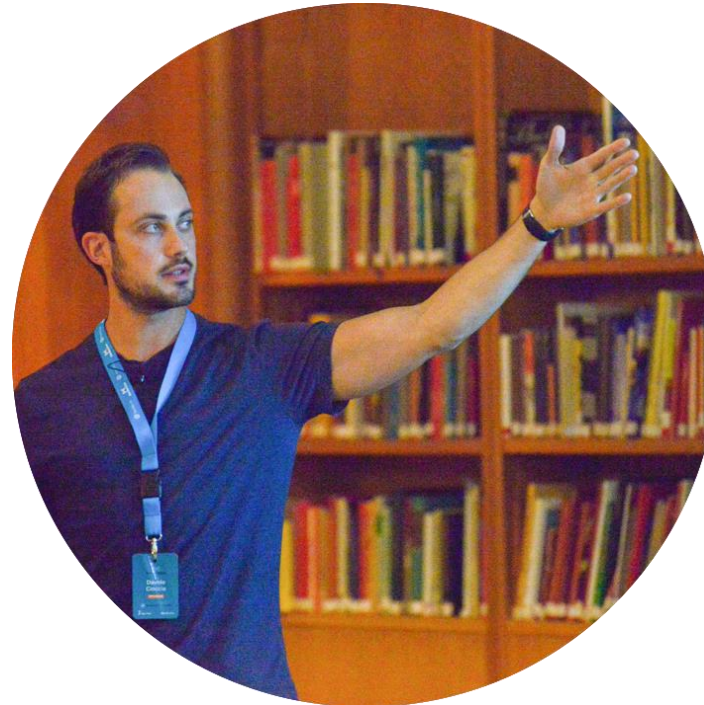
Spyros Manglis – Davide Cioccia



#whoarewe



Spyros Manglis  
Security Engineer @ ING

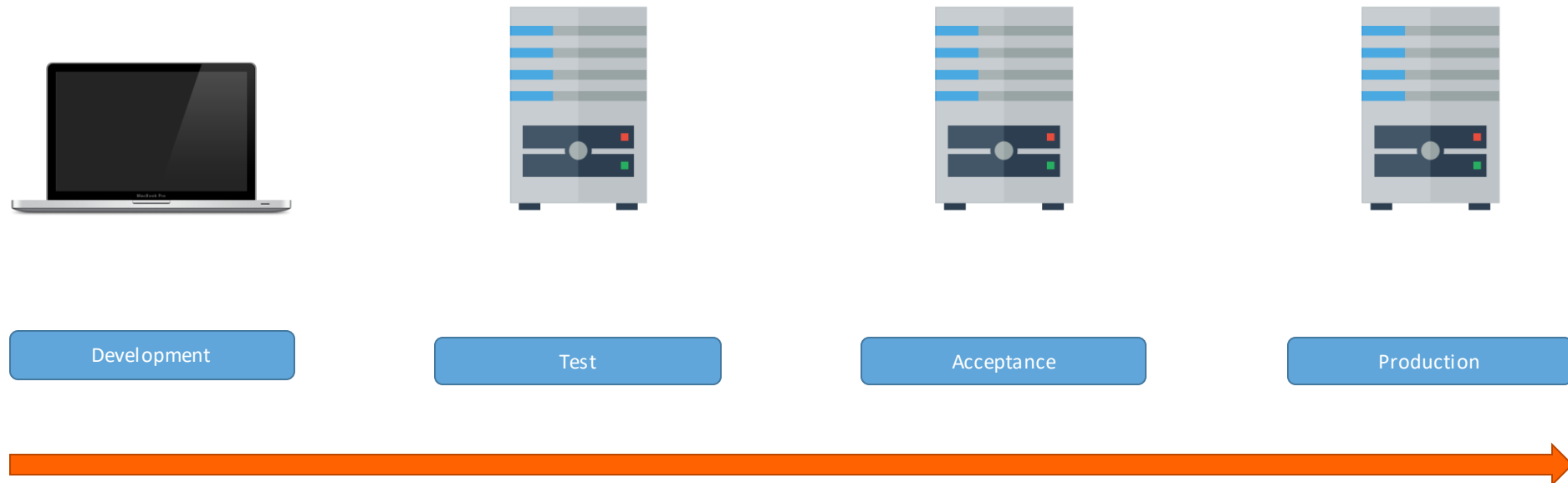


Davide Cioccia  
Security Engineer @ ING  
Trainer @ defdeveu

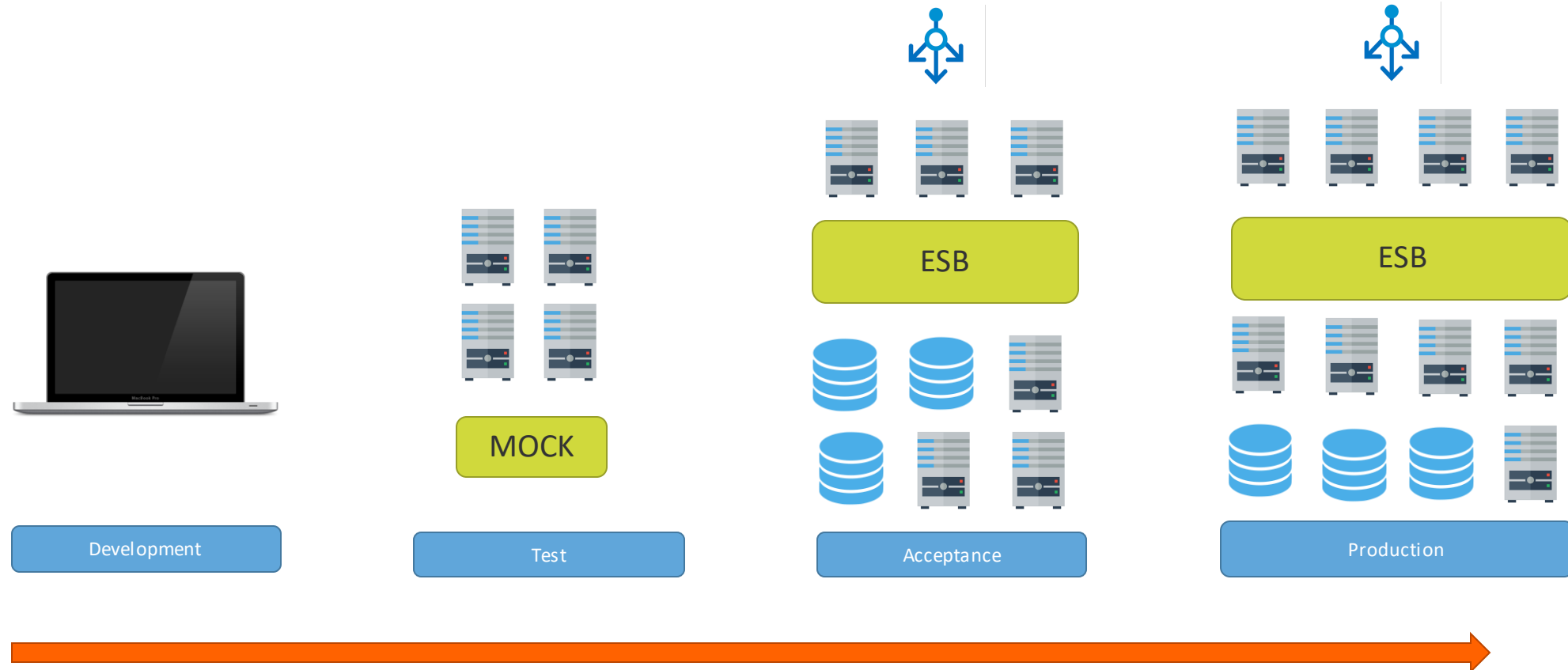
# What's this all about?

- Present our challenge
- Share our knowledge and experience on custom infrastructure security automation
- Share the framework we created in details
- Future work: embed the tool in the CI/CD pipeline
- Q&A

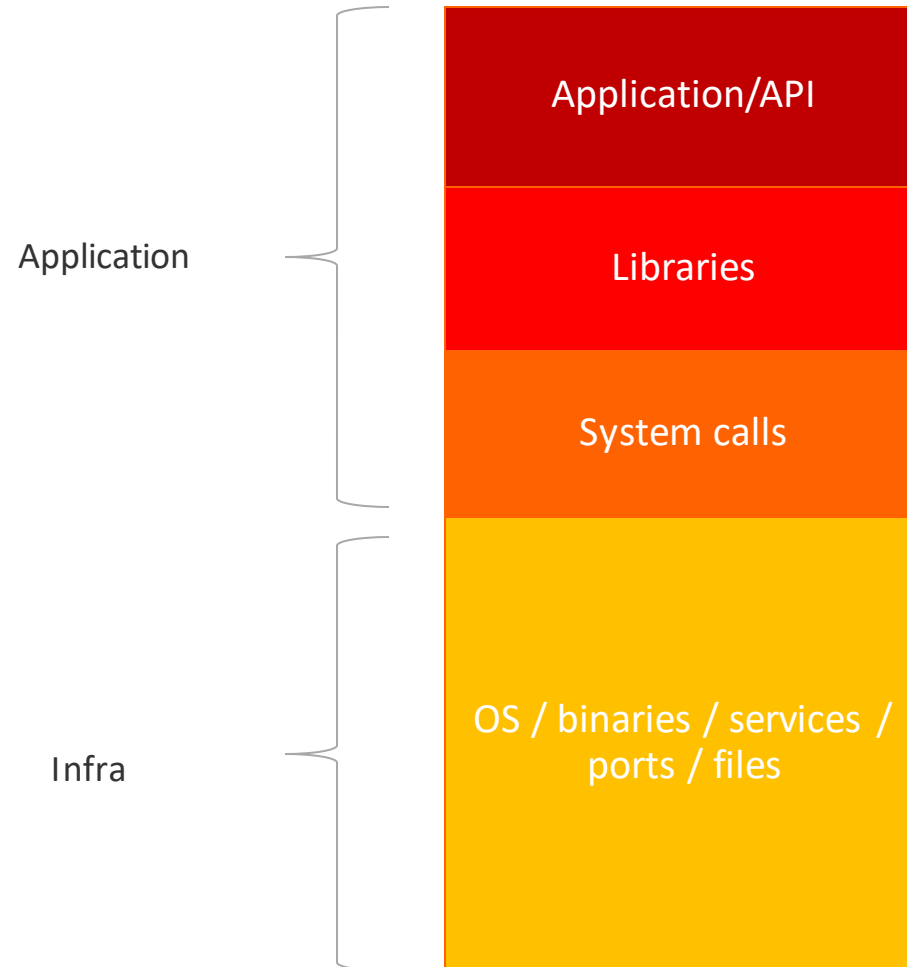
# A simple DTAP environment for one service



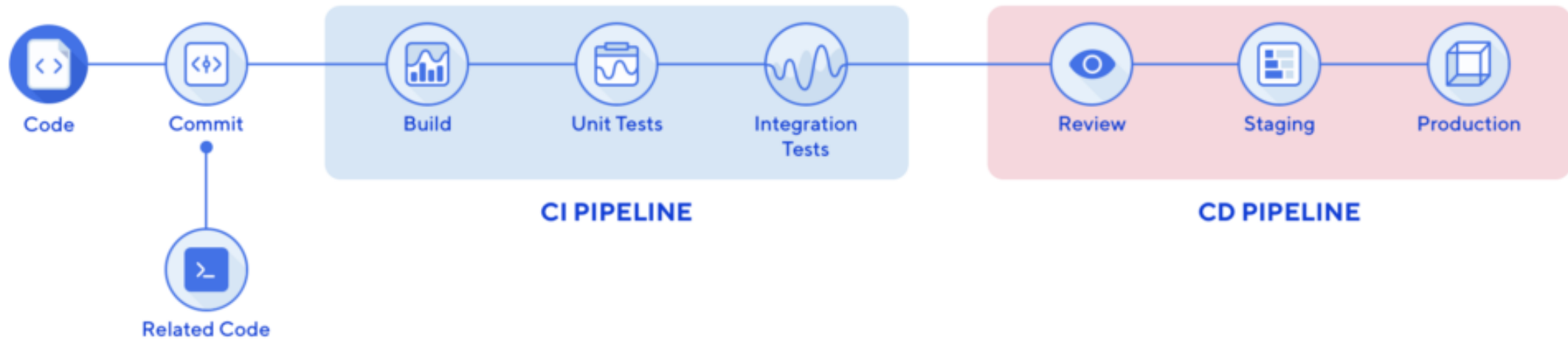
# The ING DTAP environment for one service



# What's on each server: the stack



# CI/CD secure pipeline and stack assessment coverage



Real time SAST in IDE

Incremental SAST

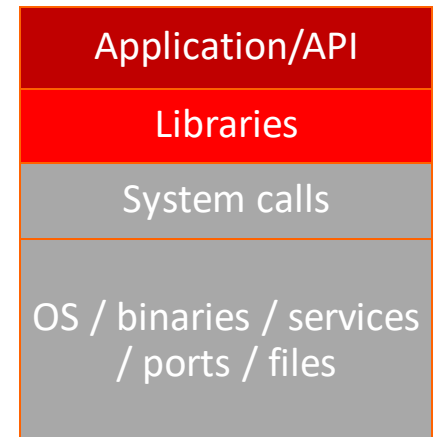
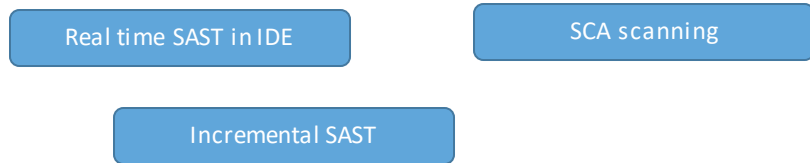
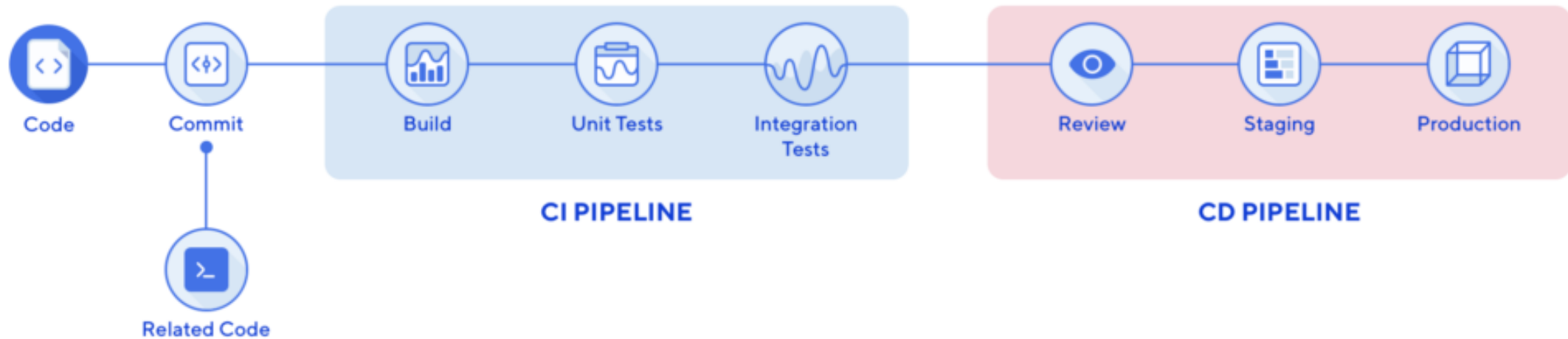
Application/API

Libraries

System calls

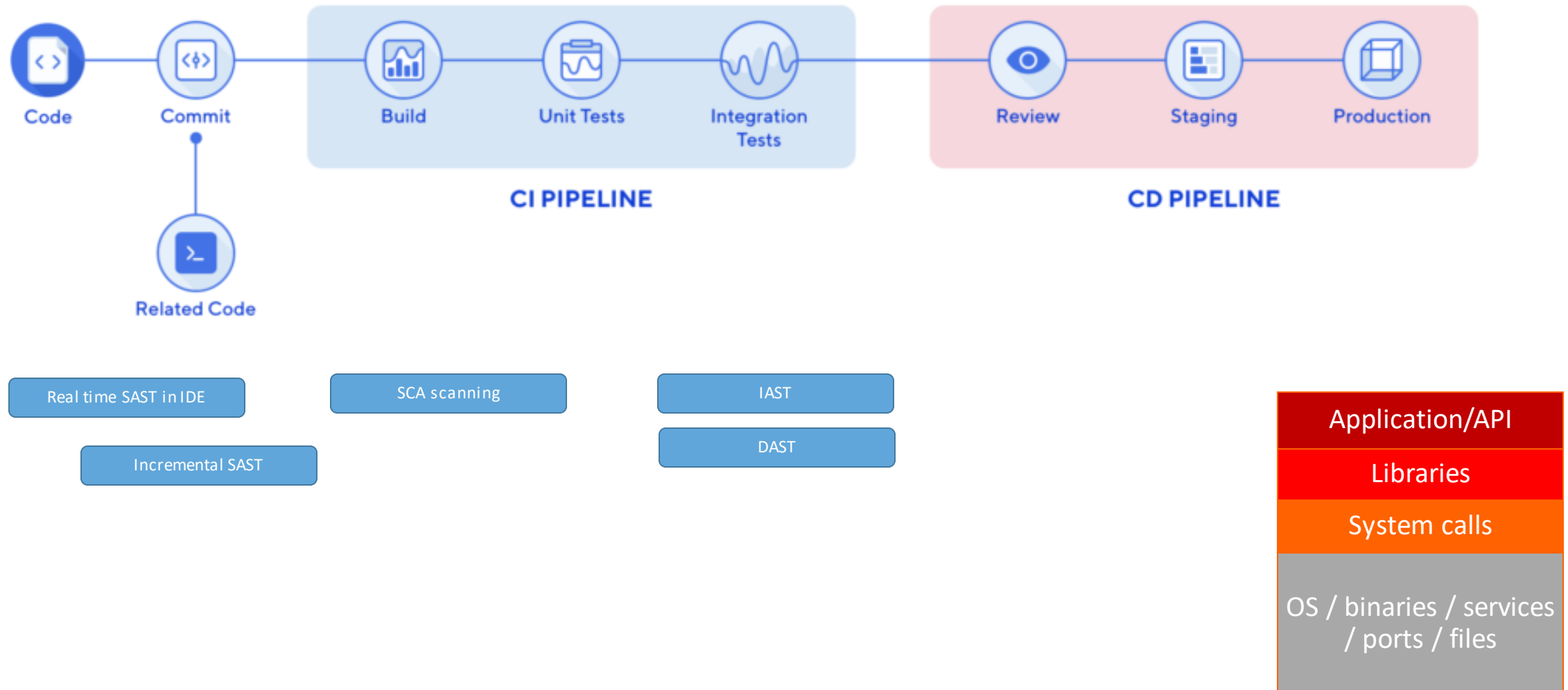
OS / binaries / services  
/ ports / files

# CI/CD secure pipeline and stack assessment coverage

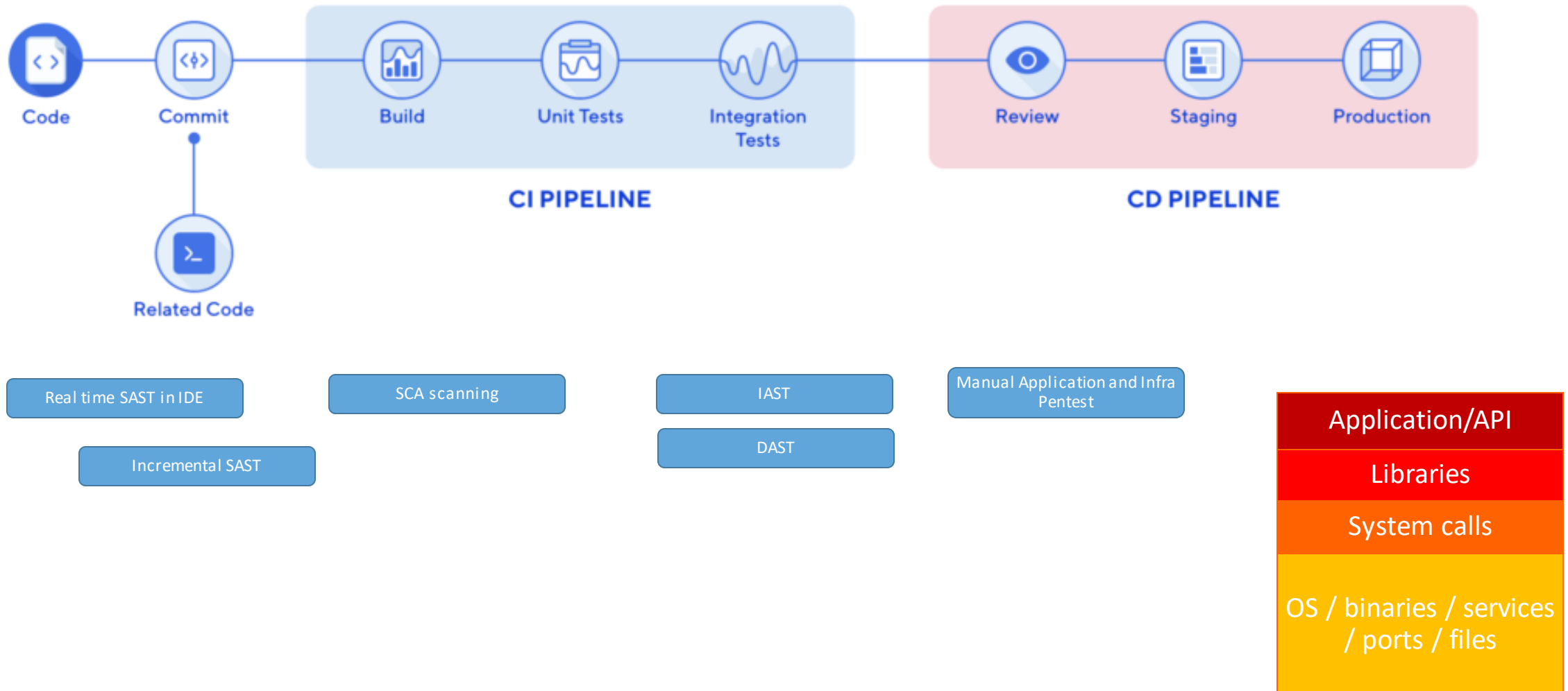




# CI/CD secure pipeline and stack assessment coverage



# CI/CD secure pipeline and stack assessment coverage




**SAY "MANUAL TESTER" AGAIN.**



**I DARE YOU! I DOUBLE DARE YOU!**

# The challenge

## Problems:

- Manual infrastructure security testing overheded
- Machines, VM images, Docker images are only assessed on provisioning
- Infrastructure incremental changes are not checked
- VA tools not valuable
- 2000 applications = 10x 2000 machines =  per testers

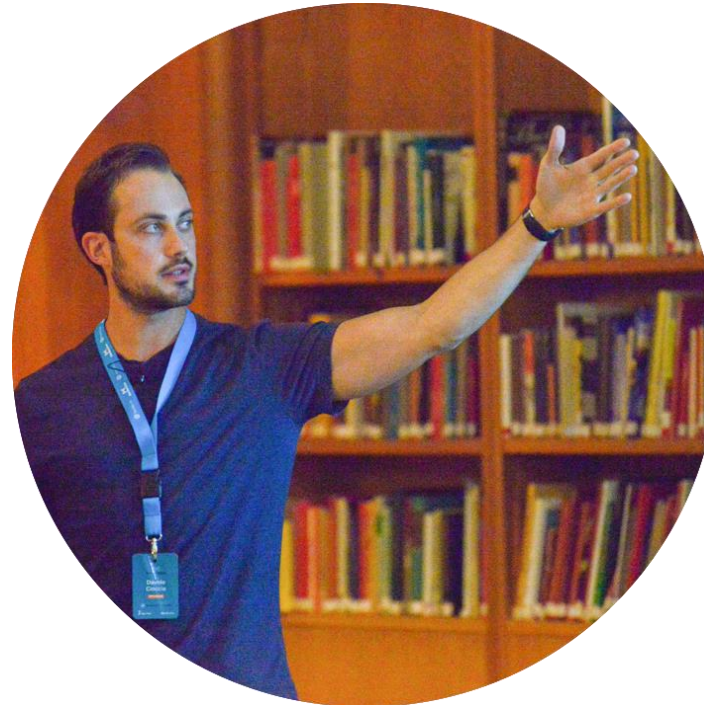
## Question:

- How can we check the security of the deltas on infra level?

#whoarewe



Spyros Manglis  
Security Engineer @ ING



Davide Cioccia  
Security Engineer @ ING

Let's automate



## Why Ansible

Agentless

Ssh/wirnm

Keeps state

Extensible

Easy to scale

Running based on facts (preconditions)



# Open source community

<https://galaxy.ansible.com/home>

## Well known authors

**Info**

Minimum Ansible Version **2.5**

Installation `$ ansible-galaxy install redhatofficial.rhel7_pci_dss`

Last Commit a month ago

Last Import a month ago

Tags **compliance** **compliancecode** **hardening** **openscap** **pcidss** **redhat** **redhatofficial** **scap** **security** **ssg** **system**

**Content Score**

**Quality Score**  **4.1 / 5**

Last scored a month ago. [Show Details](#)

**Community Score** **No Surveys** **0 / 5**

Based on 0 surveys. [Show Details](#)

**Tell us about this role**

Quality of docs? ☐ ☐ ☐ ☐ ☐


Ease of use? ☐ ☐ ☐ ☐ ☐

Does what it promises? ☒ Y ☐ N


Works without change? ☒ Y ☐ N


Ready for production? ☒ Y ☐ N

## You can share yours

 **rhel7\_pci\_dss**

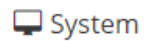
PCI-DSS v3.2.1 Control Baseline for Red Hat Enterprise Linux 7

 **4.1 / 5** Score

 **866** Downloads

## Most Popular

## Playbook for everything




System


</> Development

 Networking

 Cloud


 Database

 Monitoring

 Packaging

 Playbook Bundles

 Security

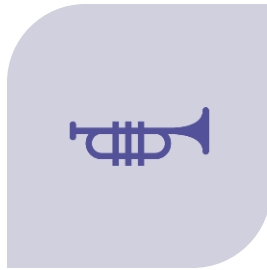
 Web



# Ansible use cases



PROVISIONING



ORCHESTRATION



CONFIGURATION



DEPLOYMENT

## How is Ansible mostly used in security

- STIG (Security Technical Implementation Guide)
- PCI DSS (Payment Card Industry – Data Security Standards)
- Networking Hardening
- Internal Security Standards
- Threat Hunting (0 days)

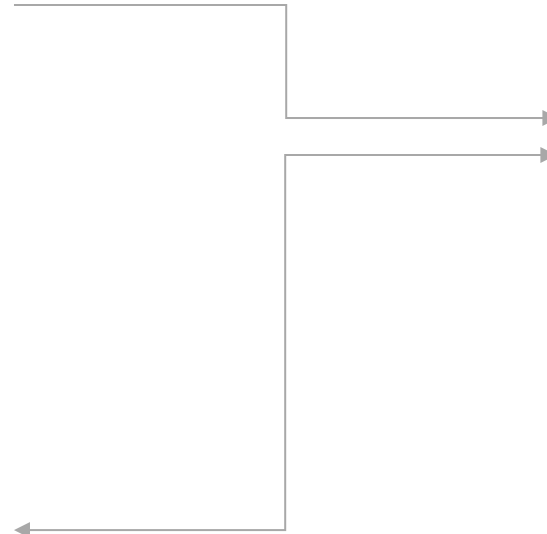
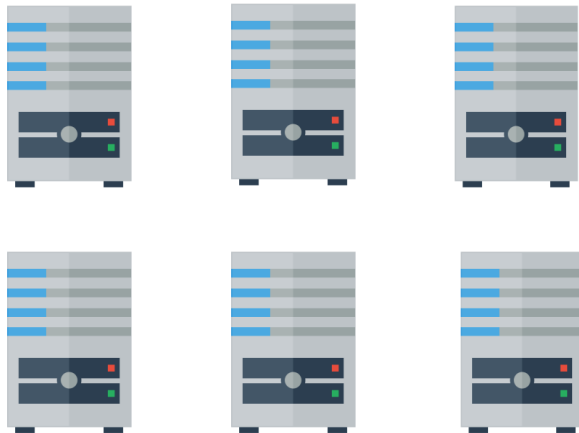
## Why not other tools?

- Can't ABC do the XYZ
- Developers and Ops can understand
- Developers and Ops can use



# Why not other tools?

- I can take advantage of all those



# The process

1. Understand our infrastructure landscape and how it changes
2. Adopt well know industry standard (PCI DSS, MITRE, STIG)
3. Define threats for Windows and Linux machines based on the STRIDE model
4. Map the threats with the MITRE ATT&CK, STIG
5. Define tests (create playbooks) based on the threats.
6. Assess the base images (at provisioning)
7. Incremental scans on changes

# The process

## Bussiness Q&A

### Questions

When these default images change?

How these default images change?

What does the change process look like?

How often changed?

What testing was done on the changes?

Which optional packages do you require the most?

What are the possible external threats?

# The process

## Technical Q&A

Questions
Do you require extra disk
Config files
File permissions
Nessus scan
Additional Network scans
Installed packages
Custom Builds / Deploys
CronJobs
Internal Builds / Artifactory
Installed Patches
Tools
Certificates
Ports
Critical folders
Groups/User
Possible external threats?
SEM-A

# How is been used within ING

- Define infrastructure security standards (guided always STIG, PCI DSS, MITRE ATT&CK ...etc)



Initial Access	Execution	Persistence (1)	Persistence (2)	Privilege Escalation	Defence Evasion (1)	Defence Evasion (2)	Credential Access	Discovery	Lateral Movement	Collection	Exfiltration	Command And Control
Drive-by Compromise	CME/DP	Accessibility Features	Login Scripts	Access Token Manipulation	Access Token Manipulation	Install Root Certificate	Account Manipulation	Account Discovery	Application Deployment Software	Audio Capture	Automated Exfiltration	Command and Control
Exploit Public Facing Application	Command Line Interface	Account Manipulation	LSASS Driver	Accessibility Features	Binary Padding	InstallU32	Brute Force	Application Window Discovery	Distributed Component Object Model	Automated Collection	Data Compressed	Communication Through Removable Media
Hardware Additions	Compiled HTML File	AppCert DLLs	Modify Existing Service	AppCert DLLs	BITS Jobs	Maneuvering	Credential Dumping	Browser Bookmark Discovery	Exploitation of Remote Services	Clipboard Data	Data Encrypted	Connection Proxy
Malicious Through Removable Media	Control Panel Items	Applet DLLs	Netsh Helper DLL	Applet DLLs	Applet DLLs	Modify Registry	Credentials in Files	Network Service Scanning	Logon Scripts	Data from Information	Data Transfer Size Limits	Custom Command and Control Protocol
Spezifischer Attachment	Dynamic Data Exchange	Application Shimming	New Service	Application Shimming	CME/DP	Mobile	Credentials in Registry	Network Service Scanning	Pass the Hash	Data from Local System	Exfiltration Over Alternative Protocol	Custom Cryptographic Protocol
Spezifischer Link	Execution Through API	Authentication Package	Office Application Startup	Repair User Account Control	Code Signing	Network Share	Exploitation for Credential Access	Network Share Discovery	Pass the Ticket	Data from Network Shared Drive	Exfiltration Over Command and Control Channel	Data Encoding
Spezifischer via Service	Execution Through Module Load	BITS Jobs	Path Interception	DLL Search Order Hijacking	Compiled HTML File	NTFS File Attributes	Forced Authentication	Network Sniffing	Network Sniffing	Data from Removable Media	Exfiltration Over Other Network Medium	Data Obfuscation
Supply Chain Compromise	Exploitation for Client Execution	Booklet	Port Monitors	Exploitation for Privilege Escalation	Component Firmware	Obscured Files or Information	Hooking	Password Policy Discovery	Remote File Copy	Data Staged	Exfiltration Over Physical Medium	Domain Fronting
Trusted Relationship	Graphical User Interface	Browser Extensions	Redundant Access	Extra Window Memory Injection	Component Object Model Hijacking	Process Dropping/Linking	Input Capture	Permission Groups Discovery	Remote Services	Email Collection	Scheduled Transfer	Fallback Channels
Valid Accounts	InstallU32	Change Default File Association	Registry Run Key / Startup Folder	File System Permissions Weakness	Control Panel Items	Process Hijacking	Kerberos/NTLM AS Poisoning	Remote System Discovery	Replication Through Removable Media	Input Capture	Multi-Step Proxy	
	LSASS Driver	Component Firmware	Scheduled Task	Hooking	DCShadow	Process Injection	LLMNR/NT NS Poisoning	Shared Webroot	Man in the Browser		Multi-Stage Channels	
	MMIO	Component Object Model Hijacking	Screenreader	Image File Execution Options Injection	Extra Window Memory Injection	Redundant Access	Network Sniffing	Silent Shared Content	Screen Capture		Multiband Communication	
	PowerShell	Create Account	Security Support Provider	New Service	Regedit/Regmon	Regedit/Regmon	Password Filter DLL	Remote System Discovery	Third-party Software	Video Capture	Multi-layer Encryption	
	Regedit/Regmon	DLL Search Order Hijacking	Service Registry Permissions Weakness	Path Interception	DLL Search Order Hijacking	Regedit/Regmon	Private Keys	Windows Admin Shares	Remote Access Tools			
	Regedit/Regmon	External Remote Services	Shortcut Modification	Port Monitors	DLL Side Loading	Regedit/Regmon	Two Factor Authentication Interception	System Information Discovery	Windows Remote Management			
	RunU32	File System Permissions Weakness	SP and Trust Provider Hijacking	Process Injection	Exploitation for Defence Evasion	RunU32		System Network Configuration Discovery				Remote File Copy
	Scheduled Task	Hidden Files and Directories	System Firmware	Scheduled Task	Extra Window Memory Injection	Scripting		System Network Connections Discovery				Standard Application Layer Protocol
	Scripting	Hooking	Time Providers	Service Registry Permissions Weakness	File Deletion	Signed Binary Proxy Execution		System Service Discovery				Standard Non-Application Layer Protocol
	Service Execution	Hypervisor	Valid Accounts	SD-History Injection	File Permissions Modification	Signed Script Proxy Execution		System Time Discovery				Unclassified Used Protocol
	Signed Binary Proxy Execution	Web Shell	Web Shell	Valid Accounts	File System Logical Offsets	SP and Trust Provider Hijacking						Web Service
	Signed Script Proxy Execution	Windows Management Instrumentation Event Subscription	Web Shell	Web Shell	Hidden Files and Directories	Software Packing						
	Third-party Software	Windows Helper DLL			Image File Execution Options Injection	Template Injection						
	Trusted Developer Utilities				Indicator Blocking	Timestamp						
	User Execution				Indicator Removal	Trusted Developer Utilities						
	Windows Management Instrumentation				Root Tools	Valid Accounts						
	Windows Remote Management				Root	Web Service						
	XSL Script Processing				Indirect Command Execution	XSL Script Processing						

**Legend**

- 150 observations
- 450 observations
- 350 observations
- 50 observations
- 10 observations



# Internal Threat model



ThreatID	Threat	Description	Mitigations	Mitigations sufficient?	Tests
INF-T-WND-1	Attacker adds or edits system users or groups	An attacker edits system, service or regular users accounts gain persistence An attacker adds new low or high privileged account to gains persistence An attacker edits or adds a system group.			Done
INF-T-WND-2	Attacker installs malicious package/software	An attack installs custom package with vulnerabilities to aid him with gaining persistence or elevate privileges as a stepping stone towards further attacks or lateral movement.			Done
INF-T-WND-3	Attacker exploits known vulnerability of a running service	An attacker exploits a vulnerability of a services installed and/or running on the system to get foothold, elevate privileges or continue further attacks. Such as: network services, usb, thunderbolt, smartcard, ...	Keep your system up-to-date, check for new discovered vulnerabilities		done
INF-T-WND-4	Attacker exploits known vulnerability on installed software in order to hurt Confidentiality, Integrity or Availability	Attacker exploits a known vulnerability in the installed software on the system to elevate privileges and put himself in position to further his attack. (Think: buffer overflows, privilege escalation, unquoted service path, ...)			done
INF-T-WND-5	Attacker steals and reuses credentials and continues lateral movement	Attacker find credentials that aid him in furthering his attack from configuration files, memory, logs, network traffic, etc. Can be combined with WND-3			Done
INF-T-WND-6	Attacker disables system security controls	An attacker disables system/os security controls such as ASLR, DEP in order to weaken the system in his further malicious actions such as exploiting a vulnerability such as buffer overflow on an application present on the system in order to elevate his privileges and continue his attack.			done



ThreatID	Threat	Description	Mitigations	Mitigations sufficient?	Tests	CIO Security Scope
INF-T-LNX-1	Attacker manipulates confidential application related data	Applications that are hosted on the server handle confidential data. Attacker that is unauthorised gains access to the data and enables him to manipulate it in any way possible. This could mean reading application configuration files, databases config files, gains access to source code to understand business logic, decompiles application, ...	SEM-E will trigger alert on sensitive file modification.	NO, because the teams specifies the files to monitor (can miss). Does SEM-I notify read activities?	Done	Yes
INF-T-LNX-2	Attacker hosts a rogue application to the web/app server to steal credentials and confidential info	Attacker uploads/writes rogue application with replica functionalities as the ones genuine ones on the same web/app server and lures victims to use his application instead. This could also mean taking down or changing the endpoint/urls of the genuine application.	SEM-A		Done	Yes
INF-T-LNX-3	Attacker extracts confidential data from application logs	Attacker gets read access to all application logs of all log levels; He/She then looks for confidential and sensitive information in the log files.	CIO Security guidelines does not allow to write sensitive info in the log files	No, teams that do not go through the SCR/LA procedure can bypass the check	Done	Yes
INF-T-LNX-4	Attacker fills up disk space by flooding the application logs	Attacker realises that the logs are being stored on a local/remote disk with limited capacity and tries to fill the disk up by triggering logging functionality on the hosted application.			Done	Yes
INF-T-LNX-5	Attacker manipulates application logs	Attacker has gained write access to the applications logs; he/she uses that to cover or clear his malicious previously executed activities by selectively deleting or modifying existing logs.			Done	Yes

# ING ISTG (Infrastructure Security Testing Guide)

## Mapping Threats to Ansible tasks

### 1. Information leaks in application properties files.

Application `.properties` and `.config` file could leak sensitive information such as access credentials to local JavaKeyStore files. The following example shows how to locate such leaks.

- First, locate all the readable `.properties` files on the server.

```
find / -readable -type f -iname *.prop* 2>/dev/null > readable-properties-files.list
```

- Next, identify if sensitive information is leaked in any of these files.

```
for file in $(cat readable-properties-files.list); do printf "\nChecking File $file:\n" && grep -iE "pass|key|ntlm|hash|user" --color=always $file;done > properties-files-check.list
```

This results to the following disclosure:

```
com.ibm.ssl.trustStore=${user.root}/keys/TWSClientTrustFile.jks
com.ibm.ssl.trustStorePassword={xor}somevalue==
```

#### Steps

- Attacker find where the tomcat instance is located by searching through the server. Example: `find / -name "tomcat"` `find / -name "tcserver"` `find / -name "server.xml"`
- After finding out the location of the tomcat server, the attacker proceeds and look for applications logs for the targeted application by observing the log configuration on the server and on the application. Example, Given the target application is named `BigTransactionsAPI`, the attacker tries to locate the logs by: `find /location/logs/usually/go -type f -iname "*BigTransactionsAPI*.log"`
- Attacker then checks or confirms if he has read access to the logs by `ls -la` the log files and observing file permissions. (This can be combined in one command)
- Once confirmed, attacker digs through the log files to find/steal confidential, sensitive information. Example: `grep -E "account|creditcard|password|username|iban|phonenummer" /var/log/tcserver/logs/bigtransactionsapi-application.log`

#### Result

- Depending on the CIA Rating of the application in some cases logging sensitive and confidential data is allowed. However, this should be taken good care of. The access to the logs should be limited to few NPA accounts at most.
- C3 and C4 regarded data are the ones attackers will be targeting and they should be not available on an application that has lower C rating
- Accessing log files should trigger a SEM-A alert
- The logs found should comply with retention policy

INF-T-LNX-3.md 3.34 KB

[Edit](#) [Web IDE](#) [Replace](#) [Delete](#)

### INF-T-LNX-3

Attacker extracts confidential data from application logs

#### Threat description

Attacker has gained read access to all application logs of all log levels. He/She proceeds further and digs through the log files looking for confidential and sensitive information.

#### Tests

Assuming that the attacker has gained limited or root access to the server, he/she can move forward and look for confidential and sensitive data in application logs. The application logs can differ but most commonly we have java and scala applications running on tomcat server. However, there are other flavours of applications and depending on the application type and the server type application log files will differ. The steps explained below are applicable mostly to tomcat application logs but the same approach should be adopted and followed for other/similar environments as well.

The default permissions are:

```
<mapping>
  <directory>${dist.install.dir}/log</directory>
  <filemode>750</filemode>
  <username>apprun</username>
  <groupname>applog</groupname>
</mapping>
```

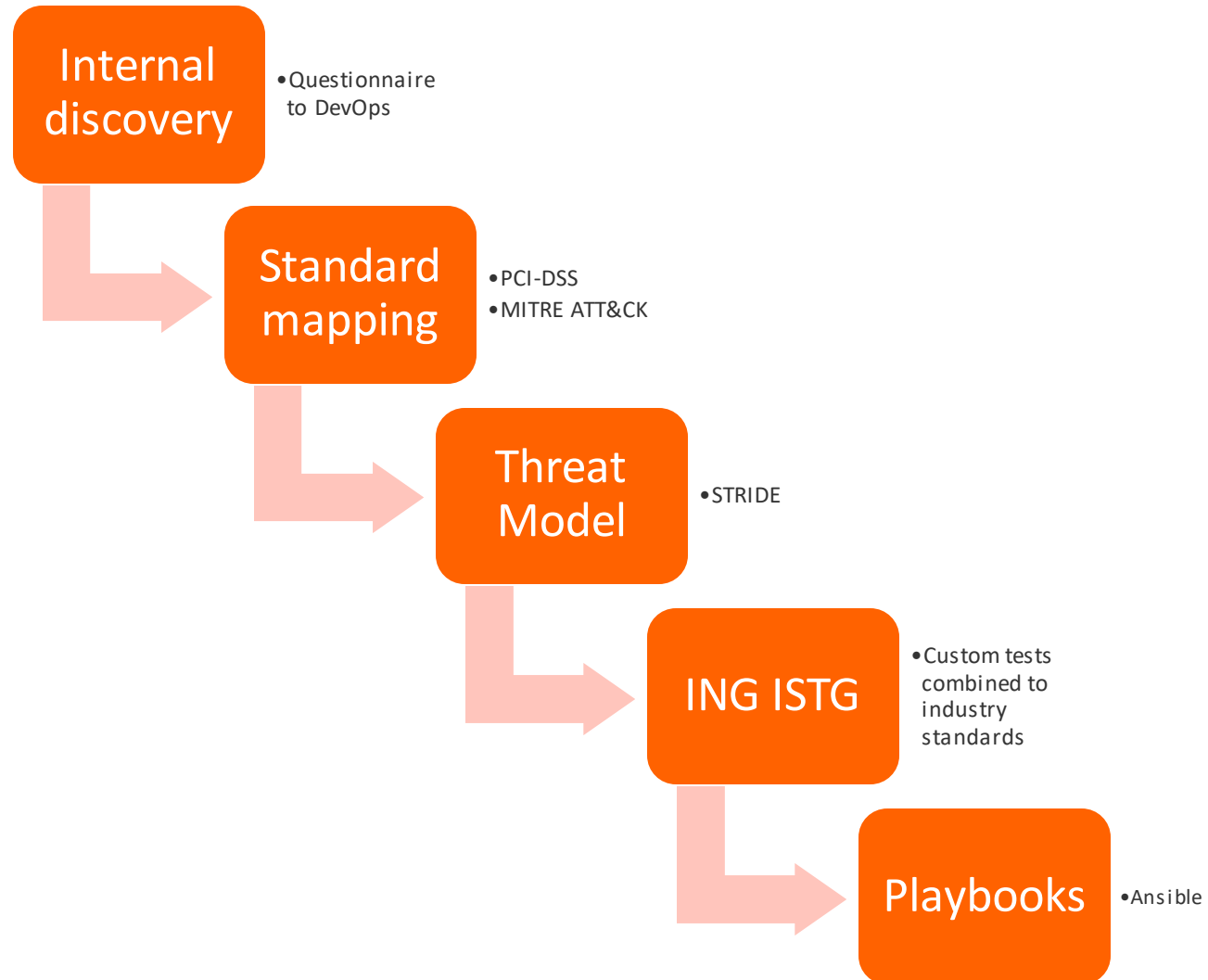
# Playbooks

## Ansible in action

```
1 # Attacker extracts sensitive data
2 ---
3 - name: ING | Check for sensitive data in the readable logs
4   find:
5     paths: /
6     file_type: file
7     recurse: yes
8     hidden: yes
9     patterns:
10      - '(?i)^.*{{ application }}.*\.logs$'
11     use_regex: yes
12     contains: '^(*firstname|.*user|.*pass|.*address|.*st
13      creditcard|.*surname|.*iban|.*phone|.*ntlm|.*hash|.*k
14   register: secret_app_logs
15   check_mode: yes
16   ignore_errors: yes
17
18 - set_fact:
19   app_secret_logs: "{{ secret_app_logs.files | to_json
20     gr_name, mode: mode, path: path }}"
21
22 - debug:
23   var: app_secret_logs
24   run_once: True
25
```

```
1 # Attacker manipulates application logs
2 ---
3 - name: ING | Check for readable - writable application log files
4   find:
5     paths: /
6     file_type: file
7     # follow: yes
8     recurse: yes
9     hidden: yes
10    use_regex: yes
11    patterns:
12     - '(?i)^.*{{ application }}.*\.logs$'
13   register: application_logs
14   check_mode: yes
15   ignore_errors: yes
16
17
18 - set_fact:
19   app_logs_writeable: "{{ application_logs.files | to_json | from_json |
20     [?ends_with(mode, '2') == 'true' || ends_with(mode, '3') == 'true' ||
21     == 'true' || ends_with(mode, '7') == 'true'].{gr_name: gr_name, mode:
22     }}"
23
24 - debug:
25   var: app_logs_writeable
26   run_once: True
27
28 - set_fact:
29   app_logs_readable: "{{ application_logs.files | to_json | from_json |
30     [?ends_with(mode, '4') == 'true' || ends_with(mode, '5') == 'true' ||
31     == 'true' || ends_with(mode, '7') == 'true'].{gr_name: gr_name, mode:
32     }}"
33
34 - debug:
35   var: app_logs_readable
36   run_once: True
37
```

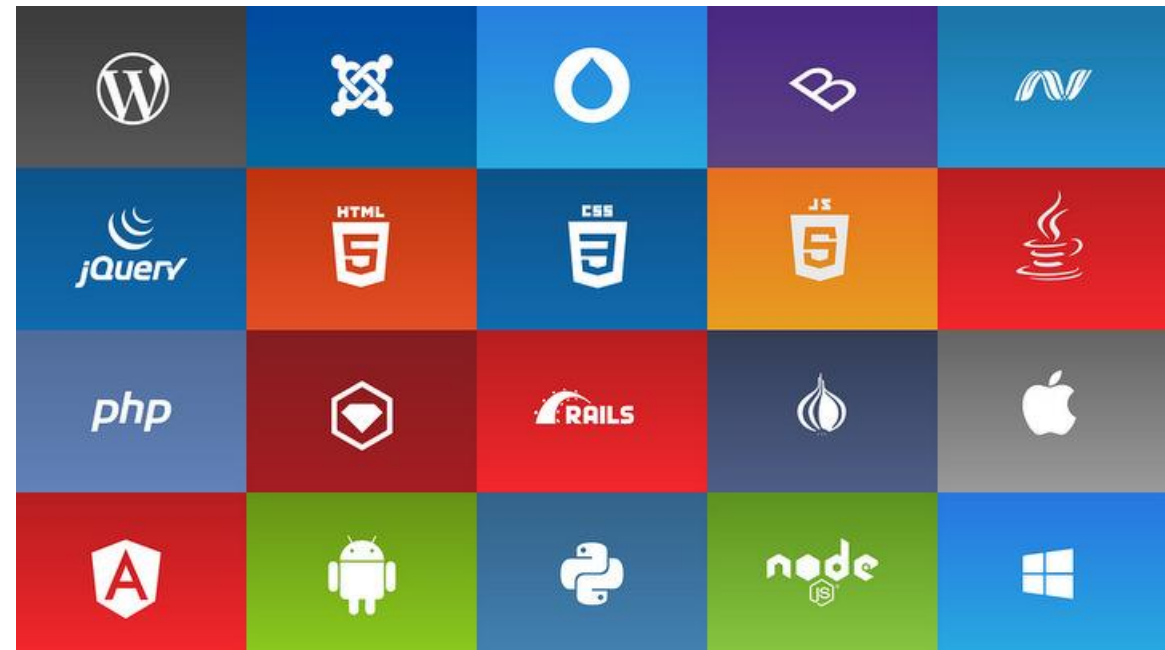
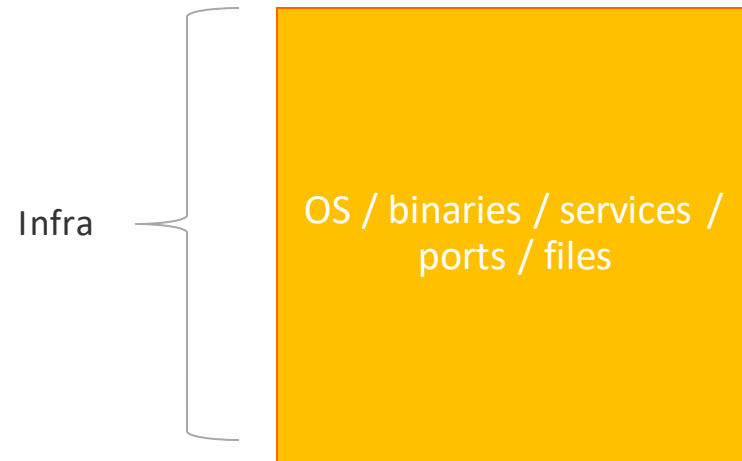
# Recap



# How is been used within ING

## Define the `Default` Stack

- Map your systems, get the default stack
  - Can be used for hardening
  - Can be used for provisioning
  - Can be used to keep state



# How is been used within ING

## Define the `Changes`

- Map the changes
- Changes per application stack (php, javascript, java ...etc)
- What extra component are used from teams



# How is been used within ING

## What about Pentesters



# How is been used within ING

## An example of what we look for

- Vulnerable binaries on the machine
- DoS vulnerabilities
- Privilege escalation techniques
- Sensitive information stored in clear text
- File and Folder Permissions
- Cronjobs
- Report (simple example {take it from our current report} )
- ....



Which security problem(s) we can solve

Catch the log hanging fruits

Replicate 0 Days

Enforce configuration

Automate the obvious

A close-up of Gene Wilder as Willy Wonka, wearing his signature black-rimmed glasses and a red velvet suit. He has a slight, knowing smile and is looking directly at the camera. The background is blurred, showing what appears to be a party or event with other people and lights.

**LIVE DEMO**

**I TOO LIKE TO LIVE DANGEROUSLY**

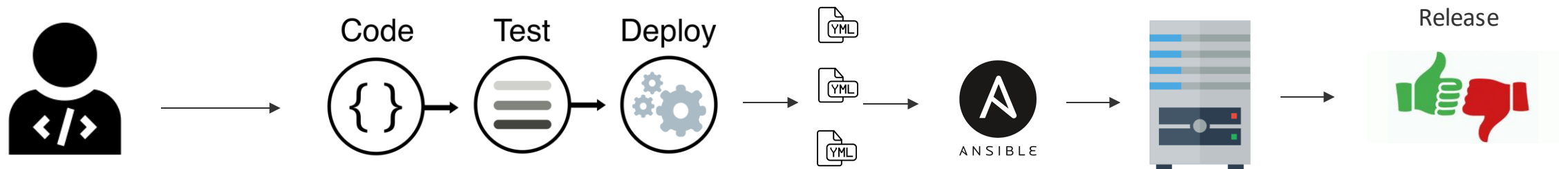
## Future work

- Release the playbooks as open source project
- Embed in Azure CI/CD pipeline
- Assign Risk score to tasks
- Automate the approval/disapproval

## How it will look in the near future



CI/CD pipeline (future)



## Sit Back and Enjoy

