SPONSORS

ORACLE Cerner

ARISTA

corelight

red canary

TGS
TECHNOLOGY GROUP SOLUTIONS

tines

10D

OPTIV | okta

(ISC)²

Material Security

Synack

CISCO

KONICA MINOLTA

DEPTH SECURITY

MCC METROPOLITAN COMMUNITY COLLEGE

no starch press

KIDS VILLAGE

TreeTop Security
Stand Above.

STEM HARVEST

PASS THE HAT

MCC METROPOLITAN COMMUNITY COLLEGE

Testing Defensive Controls

with atomic-operator

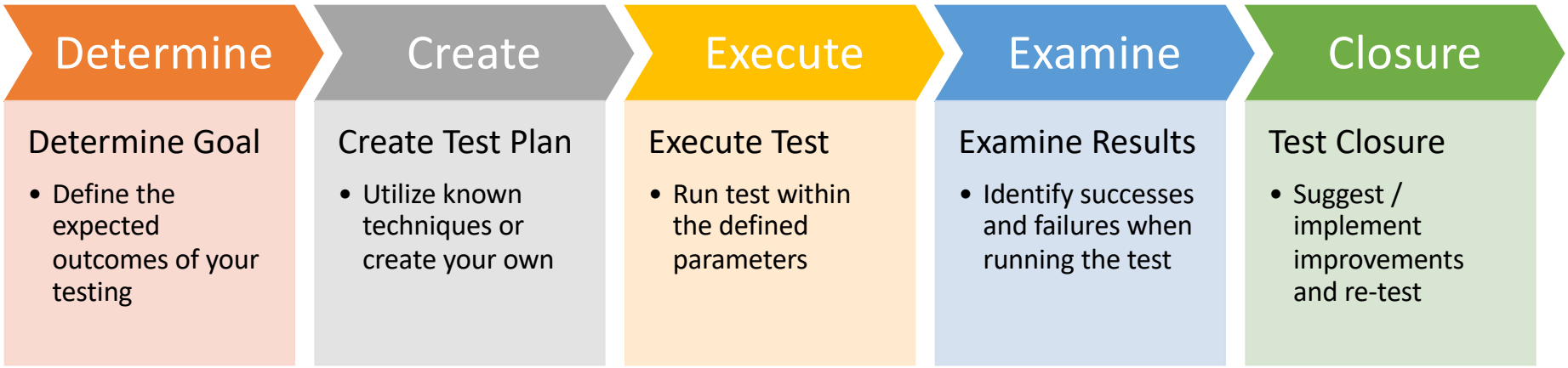
Josh Rickard

- Blue Team & DFIR
- Automate all the things
- Open Sorcerer

@MSAdministrator
github.com/MSAdministrator
letsautomate.it

Security testing is the process of verifying defensive controls are in place and are working as expected.

Simple testing process



Define Expected Outcomes

Why are you wanting to test?

AGAINST A SPECIFIC TOOL?

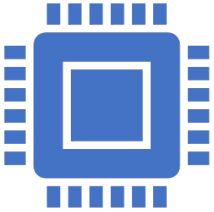
A SPECIFIC TECHNIQUE?

A SPECIFIC THREAT ACTOR?

Different Types

General security control testing

Ensuring that my organization has basic defensive controls in place and common attack techniques



Adversary Simulation & Emulation

More thorough and specific to each individual organization

emulation implies an **EXACTNESS** to the copy, whereas ***simulation*** only implies **SIMILARITY** with some freedom to be different

- Tim MalcomVetter

https://malcomvetter.medium.com/emulation-simulation-false-flags-b8f660734482

What are you wanting to test?

Endpoint configuration / hardening?

Endpoint Detection & Response product?

SIEM detection rules?

Strategy

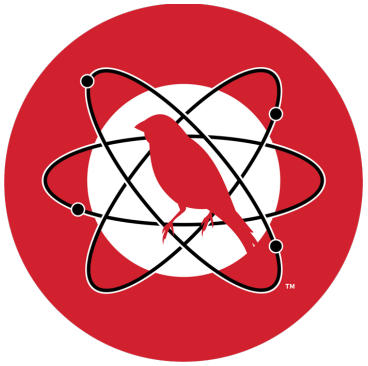
Utilize tests from third-party projects?

Atomic Red Team

Adversary emulation projects

Research and write your own tests?

Threat intelligence reports



Atomic Red Team

Attack Commands: Run with `powershell`! Elevation Required (e.g. root or admin)

```
# starting fake DC server, as SYSTEM (required)
$dc_output_file = "$env:TEMP\art-T1207-mimikatz-DC.log"
Remove-Item $dc_output_file -ErrorAction Ignore
$mimikatzParam ="`"log $dc_output_file`" `"lsadump::dcshadow /object:#{object} /attribute:#{attribute} /value:#{value}`" `
$dc = Start-Process -FilePath cmd.exe -Verb Runas -ArgumentList "/c #{psexec_path} /accepteula -d -s #{mimikatz_path} $min

# wait for fake DC server to be ready...
Start-Sleep -Seconds 5

# server ready, so trigger replication (push) and wait until it finished
& #{mimikatz_path} "lsadump::dcshadow /push" "exit"

Write-Host "`nWaiting for fake DC server to return"
Wait-Process $dc

Write-Host "`nOutput from fake DC server:"
Get-Content $dc_output_file
Start-Sleep 1 # wait a little until the file is not locked anymore so we can actually delete it
Remove-Item $dc_output_file -ErrorAction Ignore

Write-Host "End of DCShadow"
```

As a security analyst/detection engineer, I want to detect & prevent the use of mimikatz to create a rogue Active Directory Domain Controller within my environment.

Create a narrative

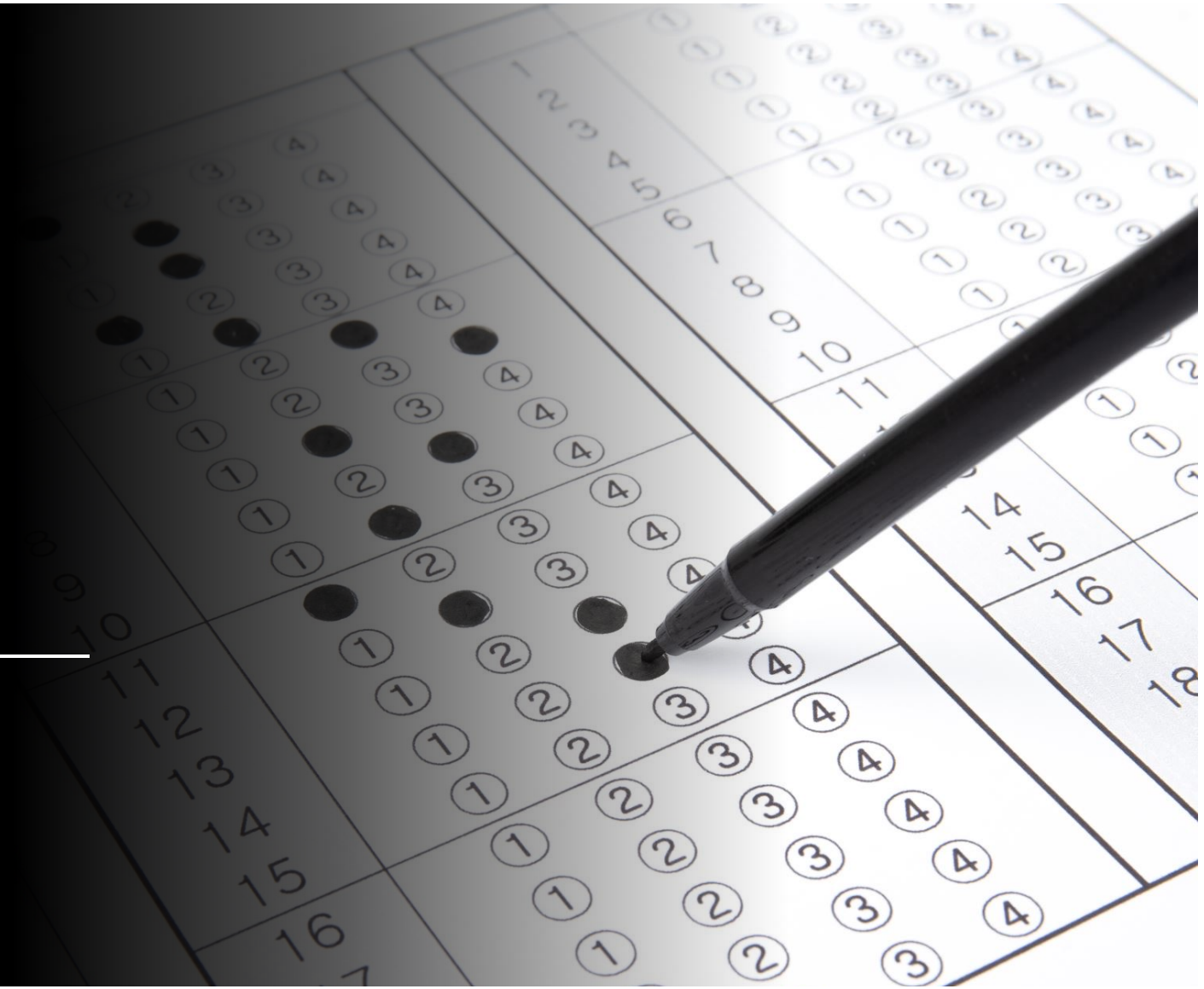
- User story format

> As a security analyst, I want to detect (or prevent) the use of mimikatz to create a rogue Active Directory Domain Controller within my environment so that a threat actor cannot change attributes of a user or system account

- GWT (Given-When-Then) format

- **GIVEN**: I am a threat actor
- **AND**: I am on a Windows Server 2019 system
- **WHEN**: I run `$env:TEMP\mimikatz\x64\mimikatz.exe lsadump::dcshadow /object:user.name /]attribute:badpwdcount /value:9999`
- **THEN**: I am unable to create a rogue domain controller
- **AND**: the security team gets alerted of this activity

Create a Test Plan



Define detailed requirements

In & out of scope

Deliverables

Environmental variables

In & out of scope

Certain products or tools?

- E.g., EDR in scope but hardening out-of-scope

Certain operating systems?

- E.g., Windows in scope but Linux is out-of-scope

Execution context (remote or local)?

- E.g., remote and local execution are in scope

Define deliverables

- Improved detection rules?
- Improved configuration / hardening?
- Maybe just a report / graph?

Environmental variables

WHAT OPERATING SYSTEM ENVIRONMENT SHOULD BE USED?

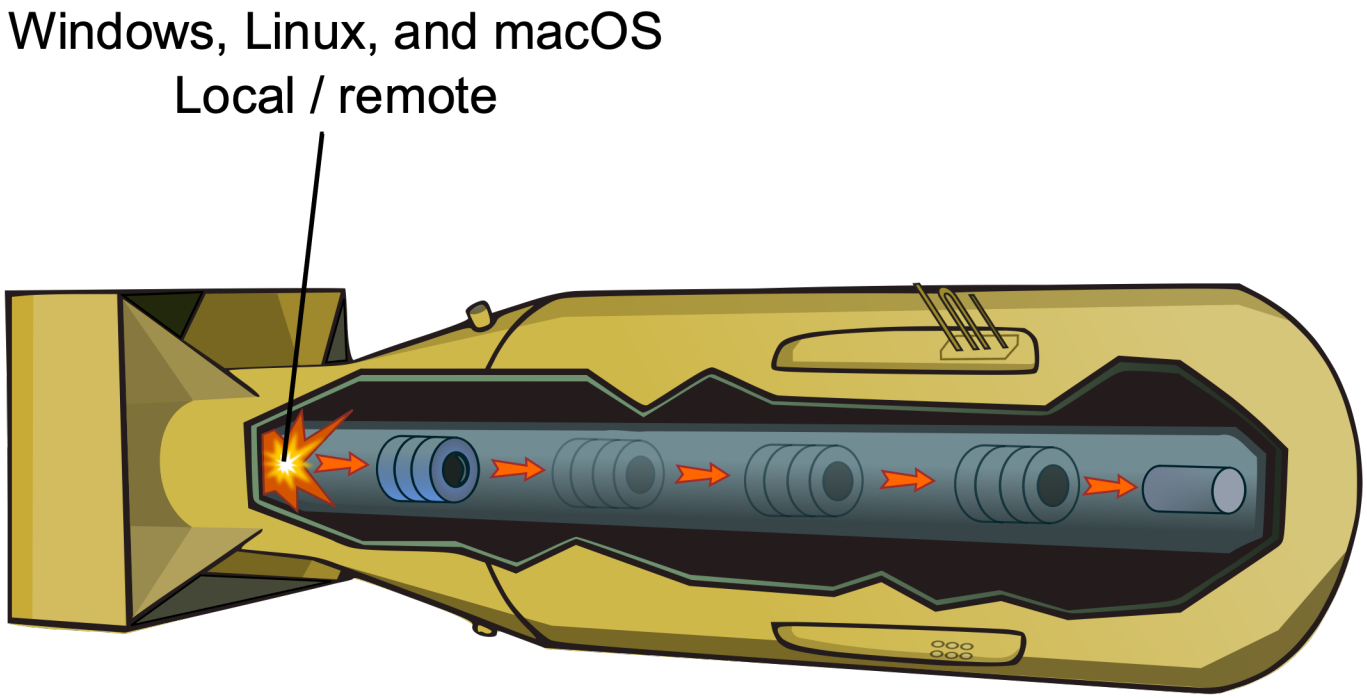
WHAT SOFTWARE SHOULD BE INSTALLED?

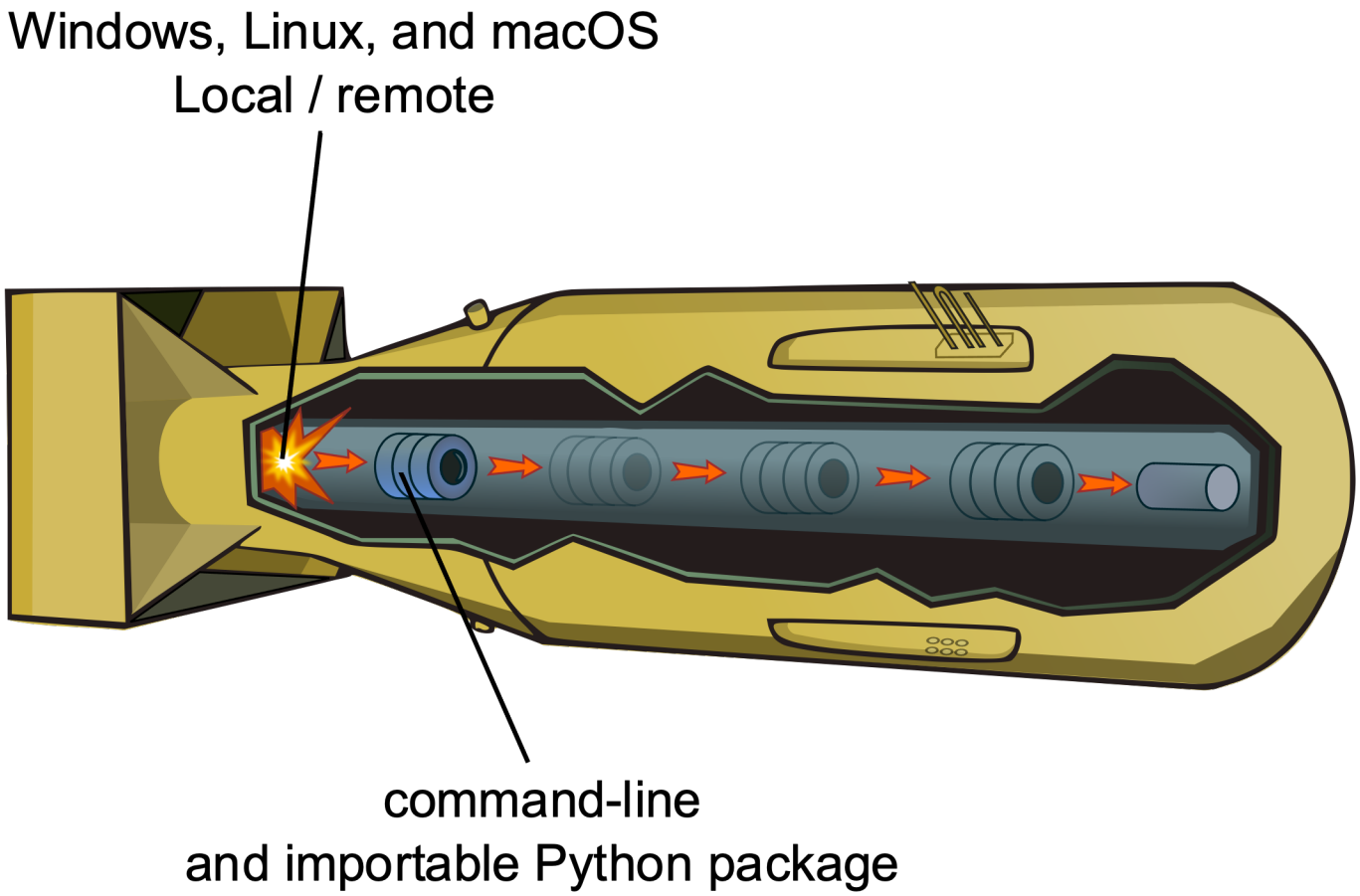
HOW DO YOU VERIFY (SUCCESS OR FAILURE) OF A TEST?

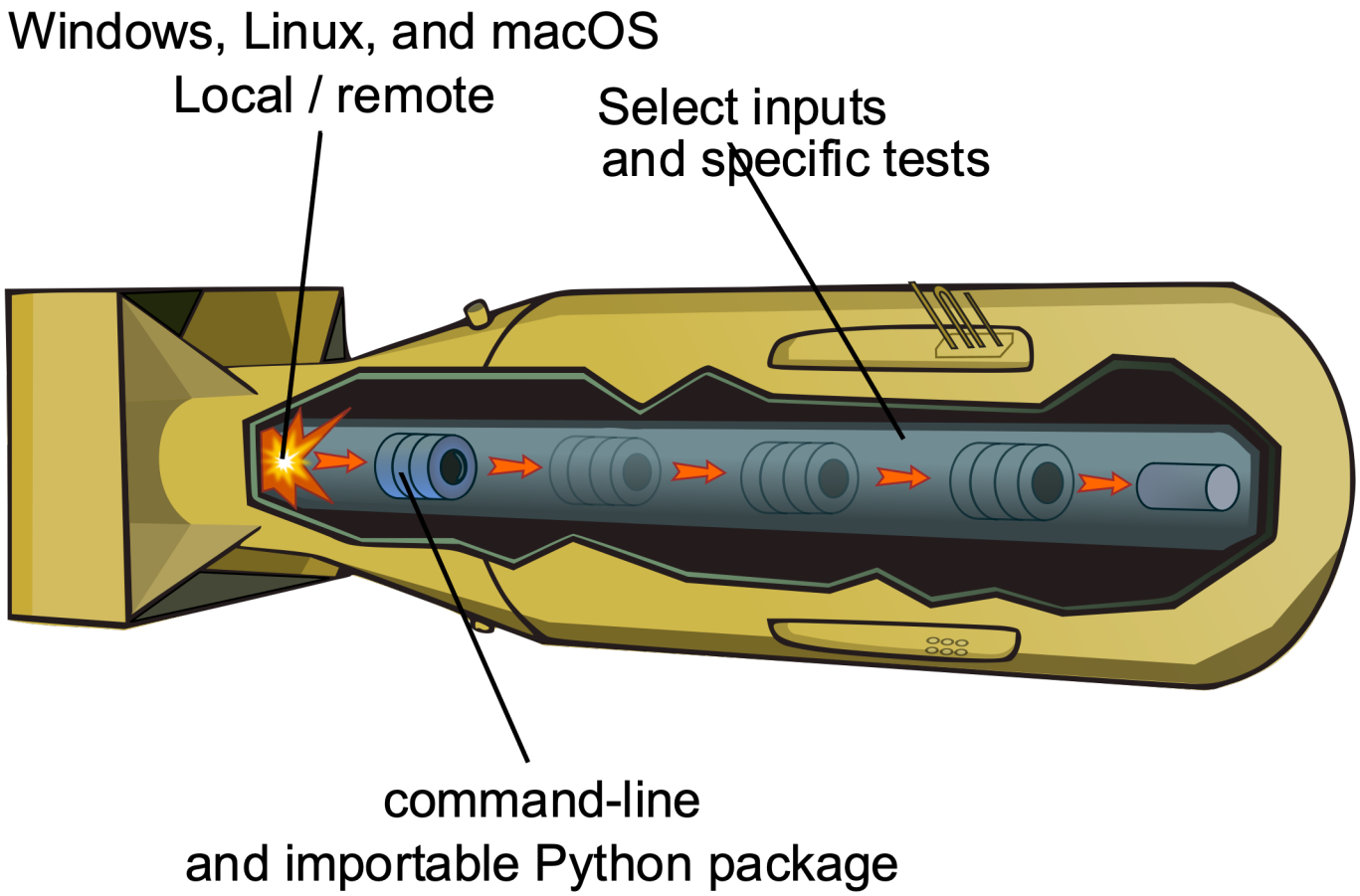
Execute Test

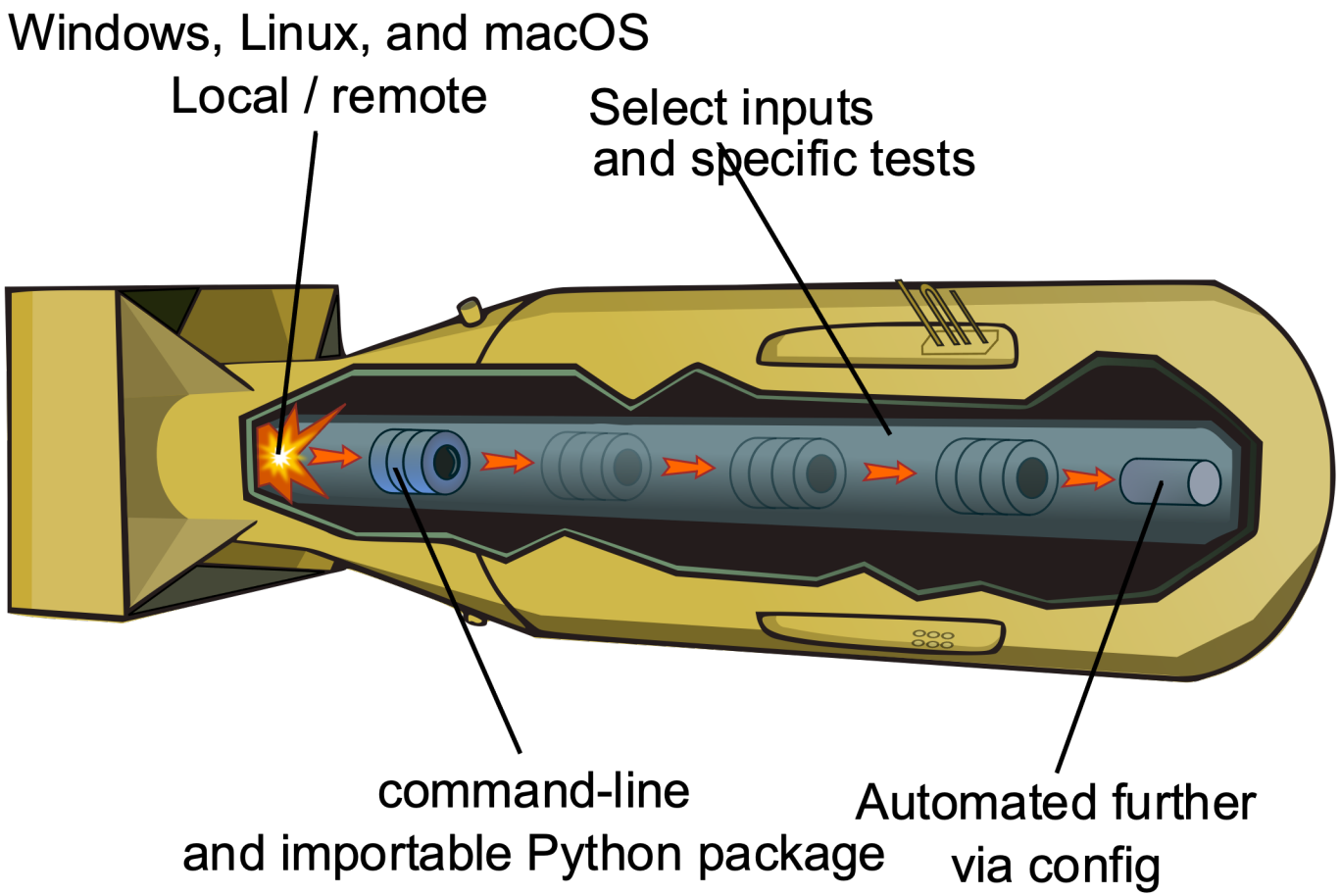
A Python package is used to execute Atomic Red Team tests (Atomics) across multiple operating system environments.

ATOMIC-OPERATOR


Windows, Linux, and macOS
Local / remote


Windows, Linux, and macOS
Local / remote
command-line
and importable Python package


Windows, Linux, and macOS
Local / remote
Select inputs
and specific tests
command-line
and importable Python package


Windows, Linux, and macOS
Local / remote
Select inputs
and specific tests
command-line
and importable Python package
Automated further
via config


~ pip3 upgrade
~ atomic-operator run --techniqu username "admin" --password "pass"
Package name
Command
Techn to run
command remotely
from atomic_operator import AtomicOperator
at = AtomicOperator()
at.run(
techniques=["all"],
test_guids=[],
select_tests=False,
check_prereqs=False,
get_prereqs=False,
cleanup=False,
copy_source_files=True,
command_timeout=20,
debug=False,
prompt_for_input_args=False,
return_atomics=False,
config_file=None,
config_file_only=False,
hosts=[],
username=None,
password=None,
ssh_key_path=None,
private_key_string=None,
verify_ssl=False,
ssh_port=22,
ssh_timeout=5
)

Resources

Repository: https://github.com/swimlane/atomic-operator

Docs: https://www.atomic-operator.com/

RedCanary Blog: https://redcanary.com/blog/atomic-operator/

Examine Results

What happened?

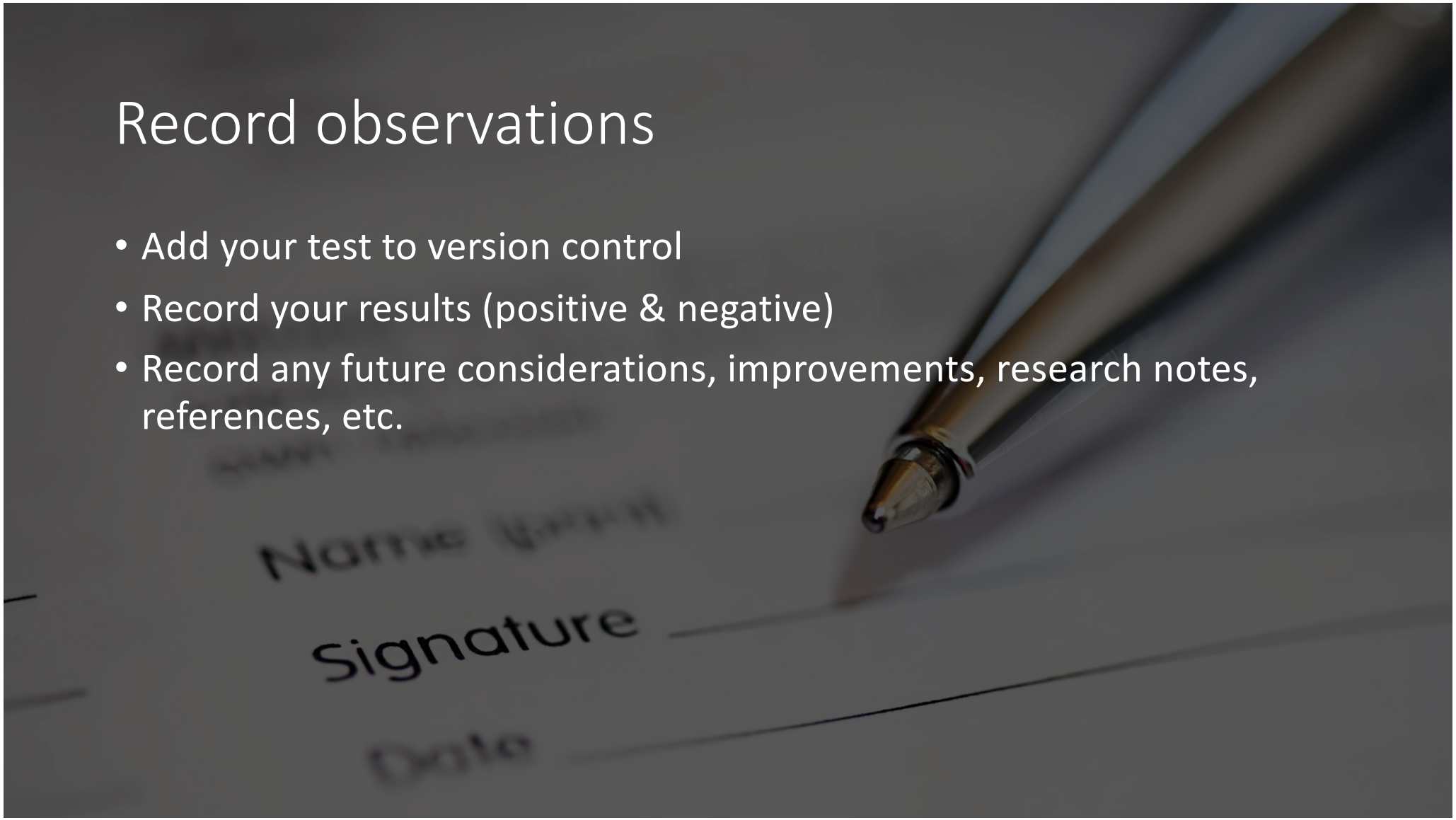
Did your test execute as expected?

Did your security controls prevent execution?

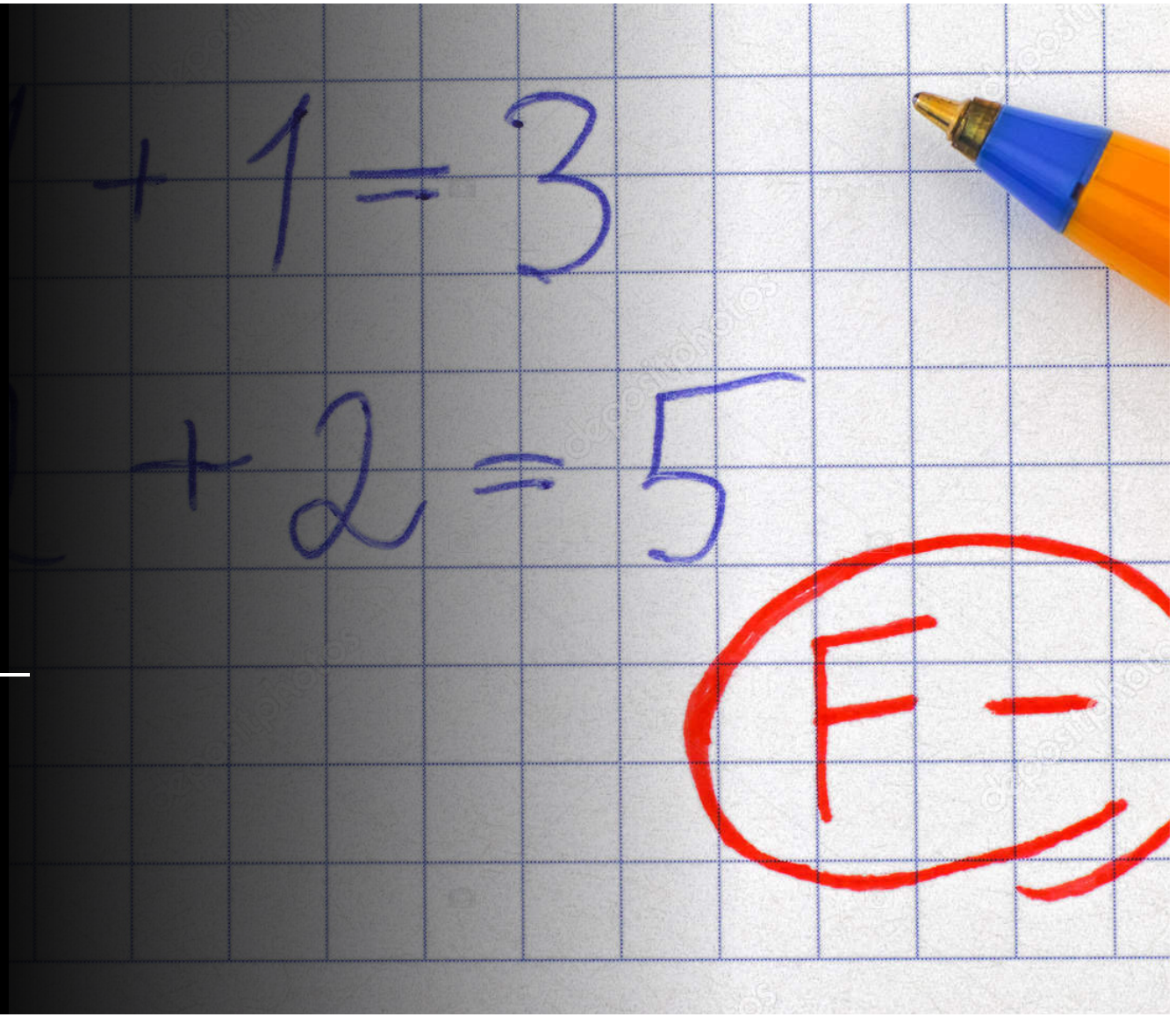
Did your security controls alert on this behavior?

Record observations

- Add your test to version control
- Record your results (positive & negative)
- Record any future considerations, improvements, research notes, references, etc.



Test Closure



Closure

01

Verify requirements were met

02

Suggest or create improvements

- Hardening configuration changes
- Create / improve detection rules
- Add new log source to SIEM
- Etc.

03

Provide summary report of

- Your test
- Your findings
- Suggested improvements

As a security analyst/detection engineer, I want to detect & prevent the use of mimikatz to create a rogue Active Directory Domain Controller within my environment.

Verify requirements were met

- Some questions to ask
  - Were your security controls able to prevent and/or detect this action?
  - Do you have visibility to see this activity?
  - How was this activity identified?
  - Are there variants of this activity that could bypass prevention/detection?
  - Etc.

Improvements

Are there any changes to configuration that would improve defenses?

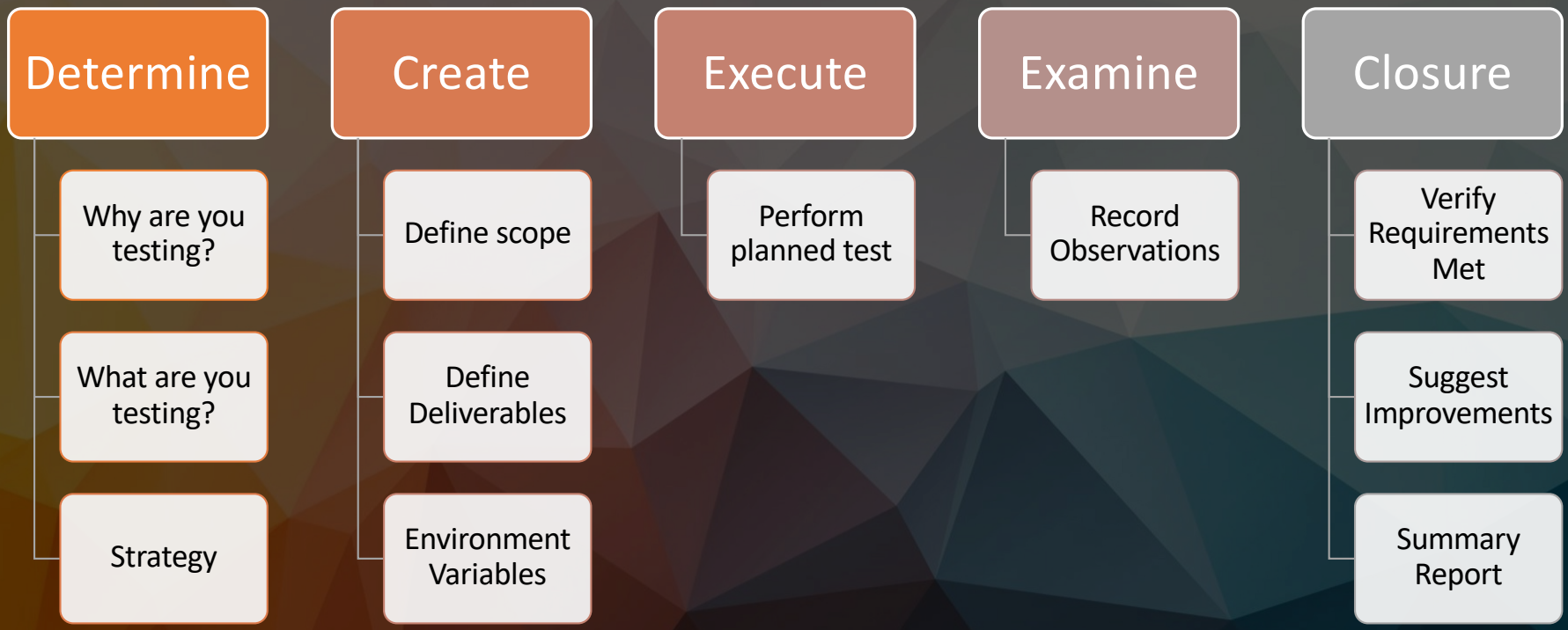
Can you implement or improve any detection rules?

Are you lacking visibility? If so, how should we improve it?

Summary report

- Document your findings in a central location like JIRA, internal wiki, etc.
- Document
  - Your test
  - The results of your test
  - Suggested improvement(s)

Testing Process Flow




Start Testing

Resources

- Twitter: https://twitter.com/MSAdministrator
- GitHub: https://github.com/MSAdministrator
- Blog: https://letsautomate.it
- Repository: https://github.com/swimlane/atomic-operator
- Docs: https://www.atomic-operator.com/
- RedCanary Blog: https://redcanary.com/blog/atomic-operator/

Help us get better!

Please provide feedback on…

my talk	the conference	anything else
https://bit.ly/KC22talk	https://bit.ly/KC22event	https://bit.ly/IqT6zt