DETECTION AS CODE

Applying the Software Development Lifecycle to Blue Team Operations
Meet the Presenter

Chris Rothe

RED CANARY
Co-founder & Chief Product Officer

@crothe
Overview

Best Practices

1. For software development
2. Applied to infrastructure
3. Applied to security detection & response
4. Applied to security testing?
Y U TALKING ABOUT SOFTWARE DEV???
The one security question that can take you to a very dark place...
Will my security program actually detect bad s#*&? 

detection & response 

SOC/SIEM/CIRT/EDR
Will my **security program** actually detect bad s#*?&?

detection & response

SOC/SIEM/CIRT/EDR

a. Do I have coverage?
b. Do I have reliability?
c. Does it work when I upgrade my OS?
d. Does it work when I upgrade my data collection software?
e. Does it work when I upgrade my analytics (~Detection/~SIEM)?
a. Do I have coverage?
b. Do I have reliability?
c. Does it work when I upgrade my OS?
d. Does it work when I upgrade my data collection software?
e. Does it work when I upgrade my analytics (~Detection/~SIEM)?
Software engineering has solved improved this over four decades
CODE COMPLETE

A practical handbook of software construction

Steve McConnell

Two-time winner of the Software Development Magazine Jolt Award
Best practices from software engineering

1. Define your purpose
2. Tests!
3. Run tests continuously
4. Keep it simple!
5. Source control everything
6. Peer review
Proof from another domain:

Infrastructure as Code
Terraform
Write, Plan, and Create Infrastructure as Code

- HashiCorp
- Azure
- AWS
- Google Cloud
resource "aws_s3_bucket" "rmisc_bucket" {
    bucket = "rmisc-bucket"
    acl = "public-read"

    lifecycle_rule {
        id = "Glacier @30 Expire @365"
        enabled = true
        transition {
            days = "30"
            storage_class = "GLACIER"
        }
        expiration {
            days = "365"
        }
    }
}
How do I know this code/system will work as expected?
version: 2

jobs:
  
  terraform_plan:
    working_directory: /tmp/builddir
docker:
    - image: circleci/python:3.6

steps:
- checkout
- add_ssh_keys
- run:
    name: Terraform plan all repos
command: |
    terraform plan
    exit $?
A high quality complex system is made up of many simple systems
Source control everything
Peer review

Create Branch → Create Pull Request → Merge Branch

- 6

Peer review
resource "aws_s3_bucket" "rmisc_bucket" {
    bucket = "rmisc.bucket"
    acl = "public-read"

    lifecycle_rule {
        id = "Glacier @30 Expire @365"
        enabled = true
        transition {
            days = "30"
            storage_class = "GLACIER"
        }
        expiration {
            days = "365"
        }
    }
}
Apply to Security Detection Detection as Code
Apply to Security Detection

Detection as Code
class Detector925 < Detectors::Base

friendly_name 'WIN-Compiled-Powershell-ENCODEDCOMMAND'

public_description <<'ENDDOC'

# Description

The encodedCommand flag of PowerShell may be used to obfuscate the command that would otherwise be passed to the interpreter and thus caught by someone watching cmdline. For example, the command webclient.download may be encoded to obfuscate that Powershell is being used to download something potentially bad.

ENDDOC

mitre_attack_categories ['T1086']

detects do

  process_name.downcase != 'powershell.exe' &&
  modload_of_any?(*windows_powershell_dlls) &&
  process_command_line.present? &&
  (process_command_line_downcased.include?(' -ec') ||
   process_command_line_downcased.include?(' -en') ||
   process_command_line_downcased.include?(' enc ') ||
   process_command_line_downcased.include?('encodedcommand') ||
   process_command_line_downcased.include?('base64') ||
   process_command_line_downcased.include?(' -^e^c^')

Define your purpose

1
describe "#925 WIN-COMPILED-POWERSHELL-ENCODEDCOMMAND " do
[ 'NIGHTMARE.exe" -nP -w 1 -eC WwBSAEUAZgBdAC4',
'NIGHTMARE.exe" -nP -w 1 -eN WwBSAEUAZgBdAC4',
'NIGHTMARE.exe" -nP -w 1 -eNc WwBSAEUAZgBdAC4',
'NIGHTMARE.exe" -nP -w 1 -eNcodedCommandthing WwBSAEUAZgBdAC4',
'NIGHTMARE.exe" -nP -w 1 ooooBase64ooo WwBSAEUAZgBdAC4',
'NIGHTMARE.exe" -nP -w 1 -^e^c^ WwBSAEUAZgBdAC4',
].each do |command_line|
  it 'detects when NIGHTMARE.exe, with one of the cli options and none of the exclusions' do
    test_events = [
      StandardizedTestData::EXAMPLE_PROCESS_START.merge(
        process_name: 'NIGHTMARE.exe',
        process_path: 'C:\Windows\NIGHTMARE.exe',
        process_command_line: command_line,
        parent_process_name: 'iexist',
        parent_process_path: 'C:\iexist',
      )
    ]
    test_endpoint_info = StandardizedTestData::EXAMPLE_WINDOWS_ENDPOINT_WORKSTATION
    test_binary = {}
    expect_detector_to_match! test_process_start, test_endpoint_info, test_binary
  end
describe "#925 WIN-COMPILED-POWERSHELL-ENCODEDCOMMAND" do

  [ 'NIGHTMARE.exe' -noP -w 1 -eC WwBSAEUAZgBdAC4',
    'NIGHTMARE.exe' -noP -w 1 -eN WwBSAEUAZgBdAC4',
    'NIGHTMARE.exe' -noP -w 1 -eNc WwBSAEUAZgBdAC4',
    'NIGHTMARE.exe' -noP -w 1 -eNcodedCommandthing WwBSAEUAZgBdAC4',
    'NIGHTMARE.exe' -noP -w 1 o000Base6400o WwBSAEUAZgBdAC4',
    'NIGHTMARE.exe' -noP -w 1 -^e^c WwBSAEUAZgBdAC4',
  ].each do |command_line|

  it 'detects when NIGHTMARE.exe, with one of the cli options and none of the exclusions' do

    test_events = [
      StandardizedTestData::EXAMPLE_PROCESS_START.merge(
        process_name: 'NIGHTMARE.exe',
        process_path: 'C:\Windows\NIGHTMARE.exe',
        process_command_line: command_line,
        parent_process_name: 'iexist',
        parent_process_path: 'C:\iexist',
      )
    ]

    test_endpoint_info = StandardizedTestData::EXAMPLE_WINDOWS_ENDPOINT_WORKSTATION
    test_binary = {}
    expect_detector_to_match! test_process_start, test_endpoint_info, test_binary
  end
end
Run tests continuously
Test Summary

Your build ran **3901** tests in RSpec with **0 failures**

**Slowest test:** spec.detectors.detector1213_spec #12 (took 1.65 seconds).

3

Run tests continuously
<table>
<thead>
<tr>
<th>DETECTOR</th>
<th>OS</th>
<th>SUMMARY</th>
<th>STATUS</th>
<th>CHANGE DESCRIPTION</th>
<th>ISSUE #</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIN-PHP-SPAWNING-CMD</td>
<td>WIN</td>
<td>Identify php spawning cmd</td>
<td>Active</td>
<td>Added coverage for php-cgi.exe</td>
<td>#763</td>
</tr>
<tr>
<td>WIN-PHP-SPAWNING-POWERSHELL</td>
<td>WIN</td>
<td>Identify php spawning powershell</td>
<td>Active</td>
<td>Added coverage for php-cgi.exe</td>
<td>#763</td>
</tr>
<tr>
<td>WIN-PHP-SPAWNING-SCRIPT</td>
<td>WIN</td>
<td>Identify php spawning scripts</td>
<td>Active</td>
<td>Added coverage for php-cgi.exe</td>
<td>#763</td>
</tr>
<tr>
<td>WIN-PHP-SPAWNING-SH</td>
<td>WIN</td>
<td>Identify php spawning sh</td>
<td>Active</td>
<td>Added coverage for php-cgi.exe</td>
<td>#763</td>
</tr>
<tr>
<td>WIN-PHP-SPAWNING-KSH</td>
<td>WIN</td>
<td>Identify php spawning ksh</td>
<td>Active</td>
<td>Added coverage for php-cgi.exe</td>
<td>#763</td>
</tr>
<tr>
<td>WIN-PHP-SPAWNING-ZSH</td>
<td>WIN</td>
<td>Identify php spawning zsh</td>
<td>Active</td>
<td>Added coverage for php-cgi.exe</td>
<td>#763</td>
</tr>
<tr>
<td>WIN-PHP-SPAWNING-BASH</td>
<td>WIN</td>
<td>Identify php spawning bash</td>
<td>Active</td>
<td>Added coverage for php-cgi.exe</td>
<td>#763</td>
</tr>
</tbody>
</table>
require 'detectors/base'

module Detectors
  class Detector925 < Detectors::Base
    friendly_name 'WIN-COMPiled-POWERSHELL-ENCodedCOMmand'

    # If the detector is 'active', 'inactive', or 'archived'
    state 'active'

    # If the detector is being tuned (and thus scored lower)
    tuning? false

    # A description of what this detector looks for, **SAF**E FOR CUSTOMER / WORLD CONSUMPTION**
    public_description '<==ENDOC'
    # Description

end
<table>
<thead>
<tr>
<th>DETECTOR</th>
<th>OS</th>
<th>SUMMARY</th>
<th>STATUS</th>
<th>CHANGE DESCRIPTION</th>
<th>ISSUE #</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIX-JAVA-SPAWNING-SHELL</td>
<td>*NIX</td>
<td>Detects Java spawning shells</td>
<td>New/Inactive</td>
<td>Create</td>
<td>#769</td>
</tr>
</tbody>
</table>

Commit NIX-JAVA-SPAWNING-SHELL, not active yet

ForensicITGuy added the "webshells" label on Mar 18

ForensicITGuy added this to the Milestone Carnage milestone on Mar 18

ForensicITGuy requested review from canaryk and kyle-rainey-rc on Mar 18
Due to the noise, potential I want to invite additional scrutiny. I've excluded CLI containing TeamCity and "build".

I'm thinking we need to split this for Linux? and OSX? I have a bad feeling about OSX, it would be nice to track the differences.

@kyle-rainey-rc I can get along with a split. It would definitely help distinguish "client-side" Java use rather than infrastructure-side use.
Can we apply to Security Testing?

Atomic Security Tests

1. Define your purpose
2. Tests!
3. Run tests continuously
4. Keep it simple!
5. Source control everything
6. Peer review