Nationaal Cyber Security Centrum
Ministerie van Justitie en Veiligheid

From NCSC-NL to Static Analysis

Jan Rooduijn

Software Security, Radboud University
17-10-2024

NCSC

About me

- I’m part of the research team at the NCSC
- I also like programming and ethical hacking
- I have a background in theoretical computer science

Overview

A brief history of the NCSC

Legal base

Different roles

Different phases

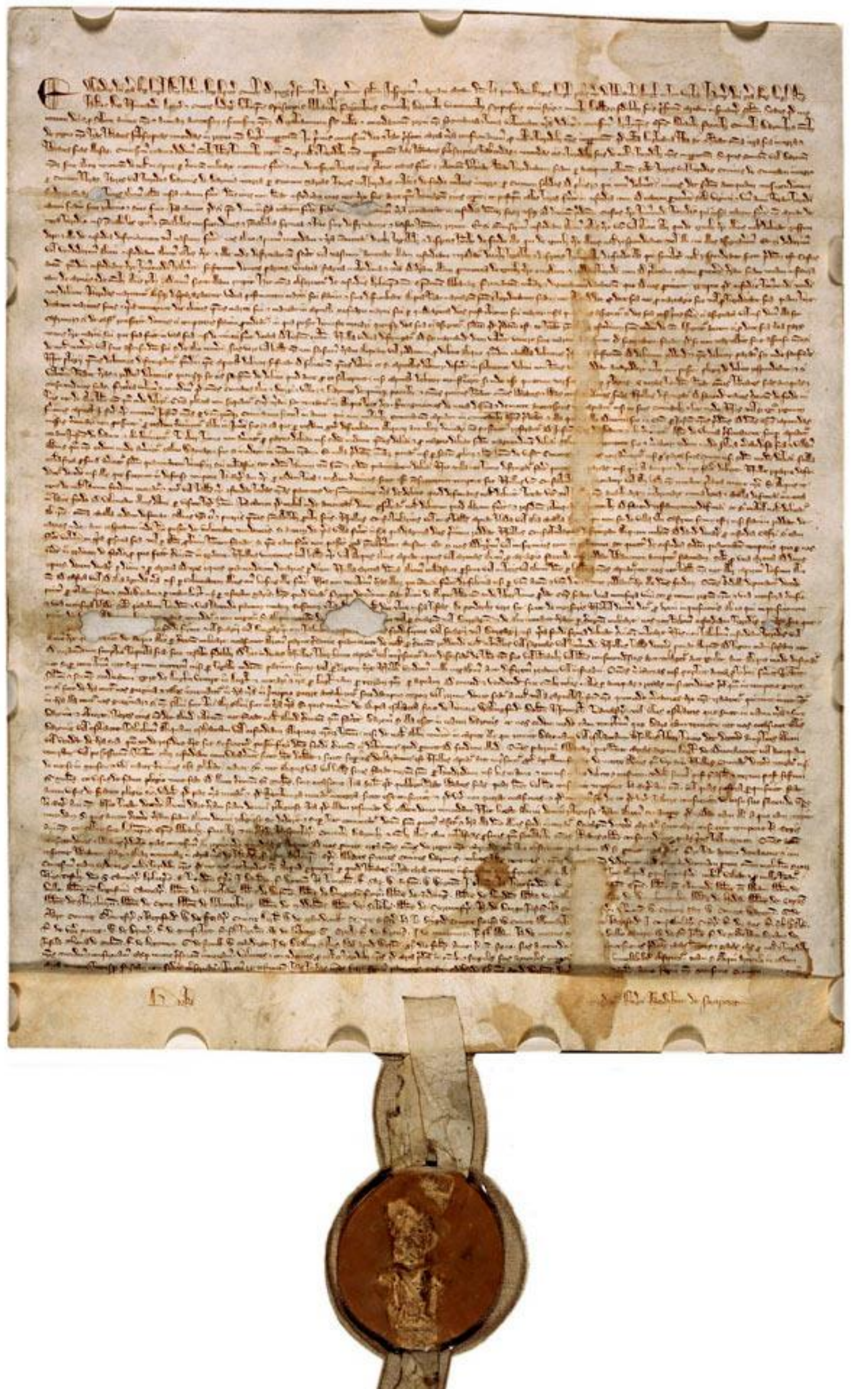
My research on static analysis

A brief history of the NCSC

- 2002 – CERT-RO (Central government)
- 2004 – GOVCERT (Government)
- 2012 – NCSC (Central Government & Vital Infrastructure)

  (part of NCTV - National Coordinator for Counterterrorism & Security)
- 2019 – NCSC (Central Government & Vital Infrastructure)

  (independent executive organization within the Ministry of Justice & Security)
- 2025 – NCSC merges with CSIRT-DSP and DTC



The legal base of NCSC-NL

…June 2016 ………

July 2016 – EU Directive on Network and Information Security (NIS)

Jan 2018 – Law on data processing and reporting obligation on cybersecurity (WGMC)

Nov 2018 – Law on protection of network and information systems (WBNI)

Dec 2022 – WBNI 1.1

Jan 2023 – EU Directive NIS2

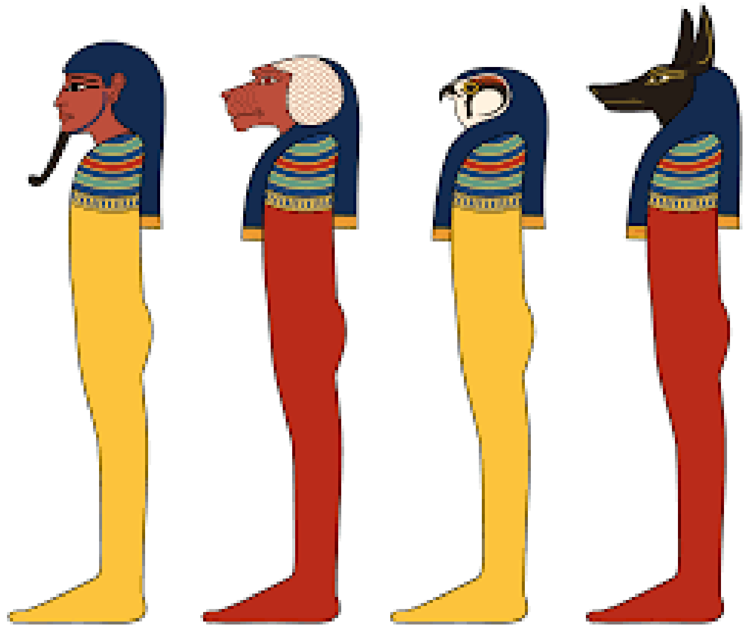
Oct 2024 – NIS2 implemented

Spring 2025 – Cyberbeveiligingswet (‘Cybersecuritylaw’: Internet consultation ended on 1 July 2024)

The main goal of the NCSC

Make the Netherlands

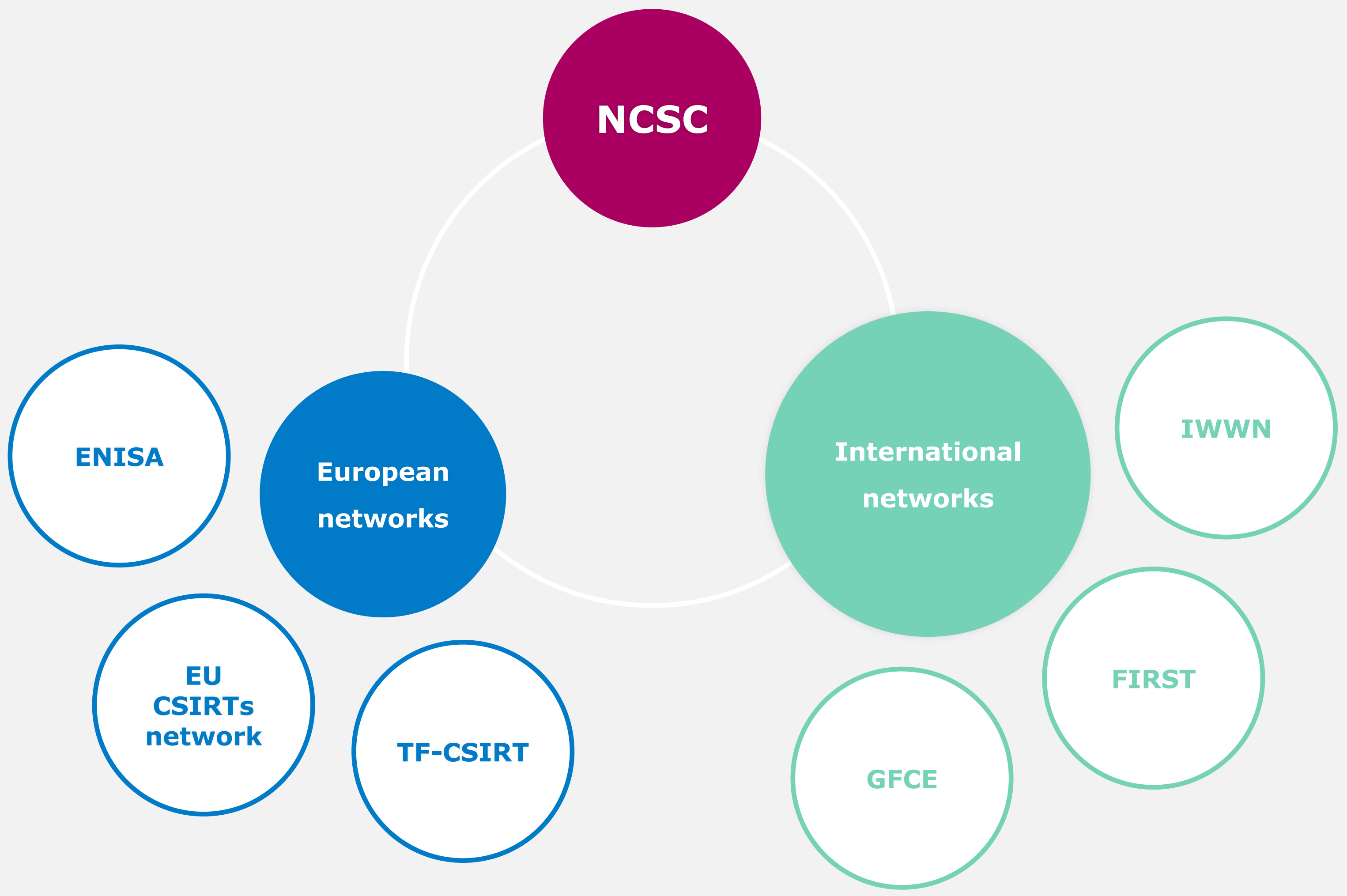
cyber resilient



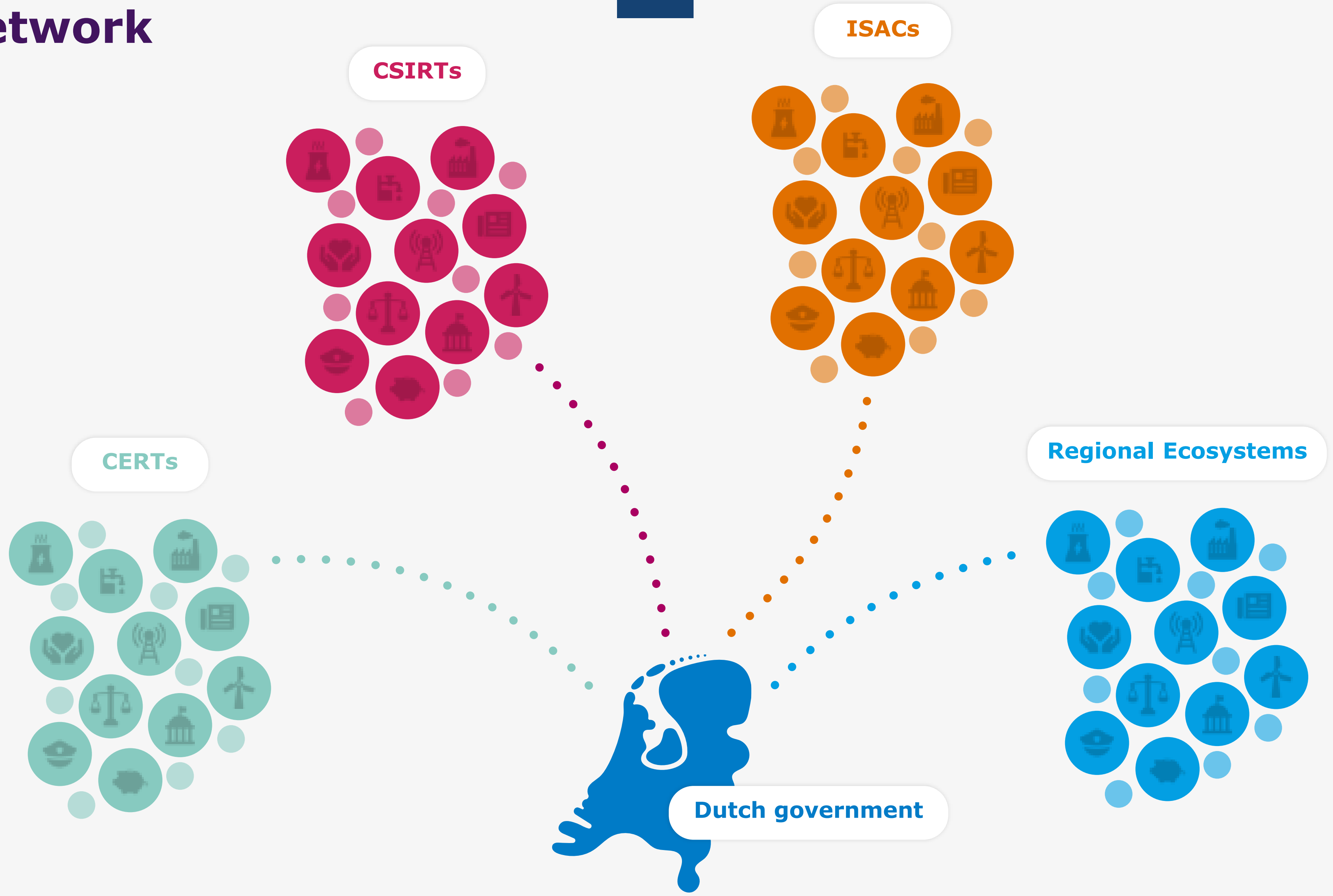
NCSC – 4 main roles

- National CSIRT
- Sectoral CSIRT
- Operational coordinator
- Centre of Knowledge and Expertise

Collaboration: **International partners**



Collaboration:
National network



Inform
(Cold phase)

Crisis response
(Warm phase)

Advice
(Cold phase)

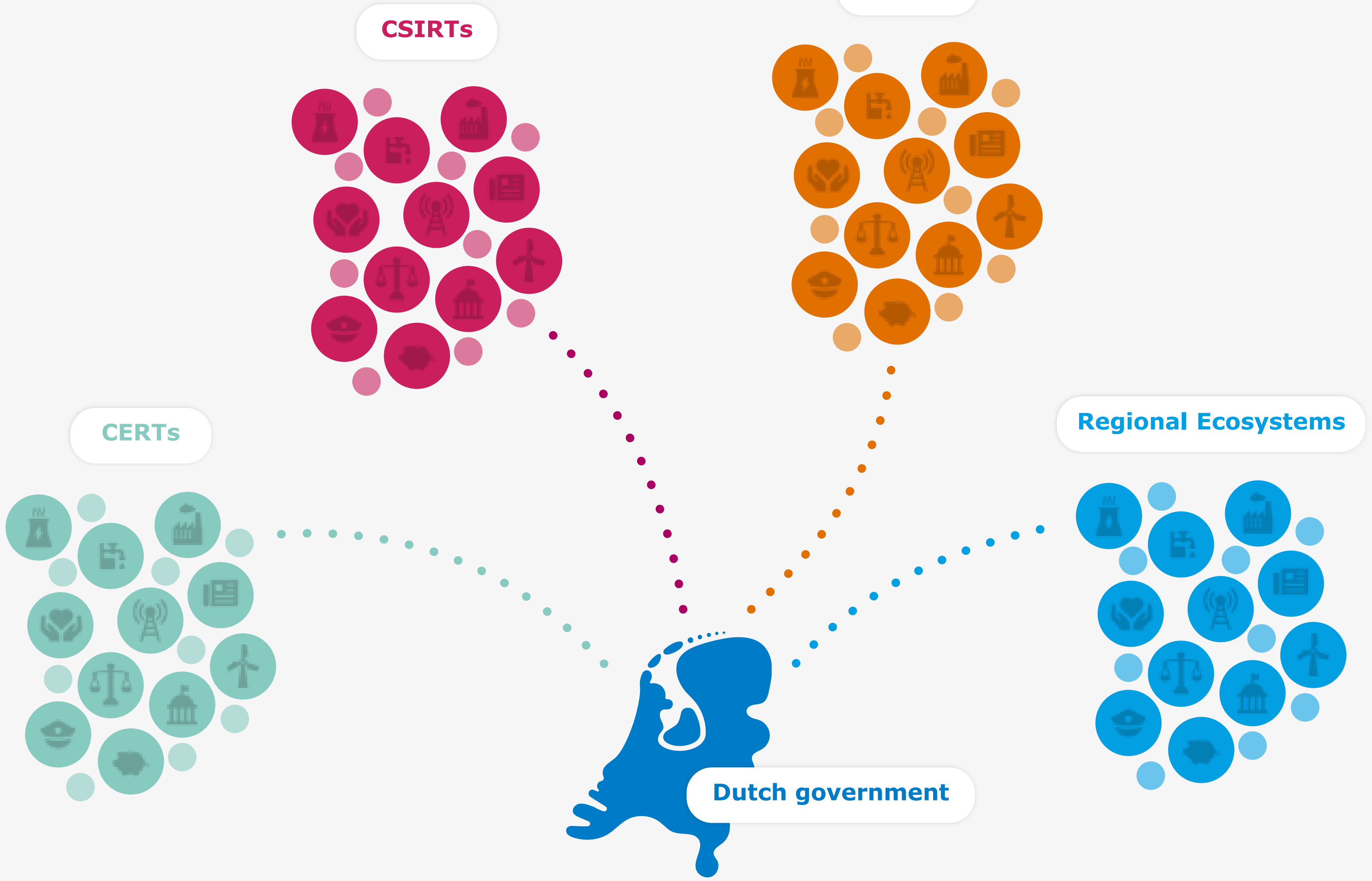
Crisis response
(Warm phase)
Inform
(Cold phase)
Advice
(Cold phase)

Early this year: active exploitation of zero-day vulnerabilities in Ivanti Connect Secure

- What is Ivanti Connect Secure?
- Active exploitation of zero-day exploits
- Why is this a critical case?

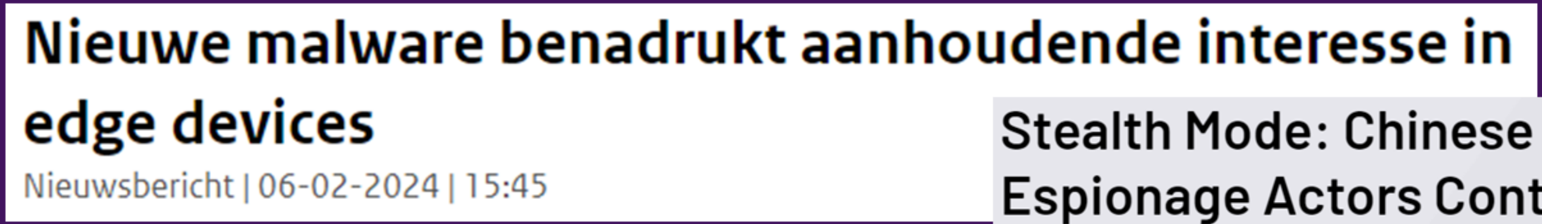
Timeline NCSC

- 09-01: NCSC is notified by international partner
- 10-01: NCSC notifies its constituents
- 10-01 (later in the evening): Ivanti goes public
- 11-01: NCSC publishes a HIGH/HIGH advisory, from this point on NCSC actively monitors and alerts for compromised systems and malware infections. We are also available for incident response to our constituents.
- 16-01: Metasploit exploit released, largescale exploitation is expected
- 31-01: Patches released

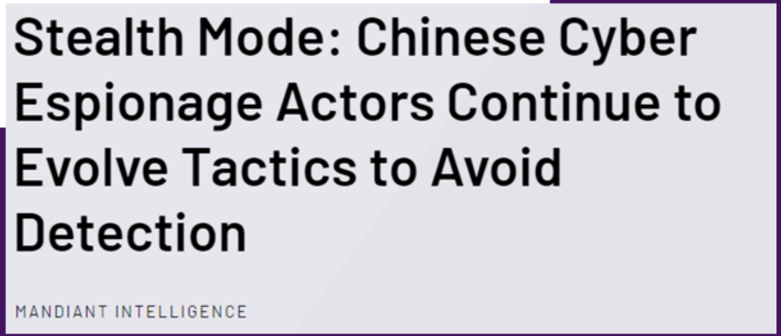

CSIRTs
ISACs
CERTs
Regional Ecosystems
Dutch government

Trend analysis

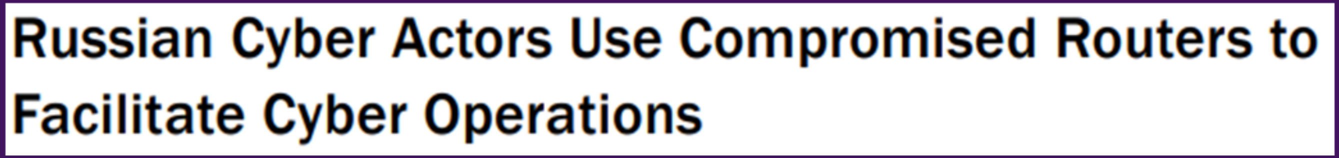
NCSC-NL constateert een trend waarin actoren in aanhoudende mate VPN-oplossingen en andere publiek beschikbare edge oplossingen, zoals Ivanti Connect Secure misbruiken.


Nieuwe malware benadrukt aanhoudende interesse in edge devices

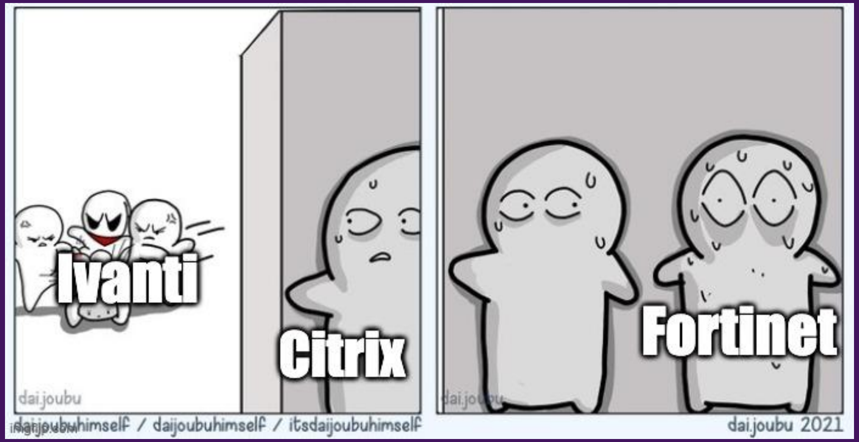
Nieuwsbericht | 06-02-2024 | 15:45


Stealth Mode: Chinese Cyber Espionage Actors Continue to Evolve Tactics to Avoid Detection

MANDIANT INTELLIGENCE

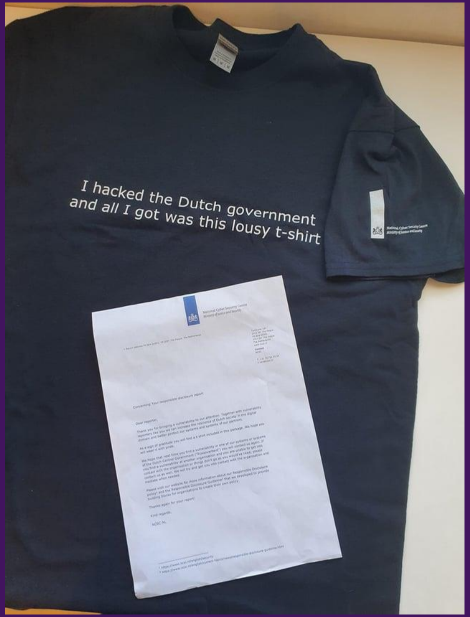

Russian Cyber Actors Use Compromised Routers to Facilitate Cyber Operations

Trend analysis



Found a vulnerability?

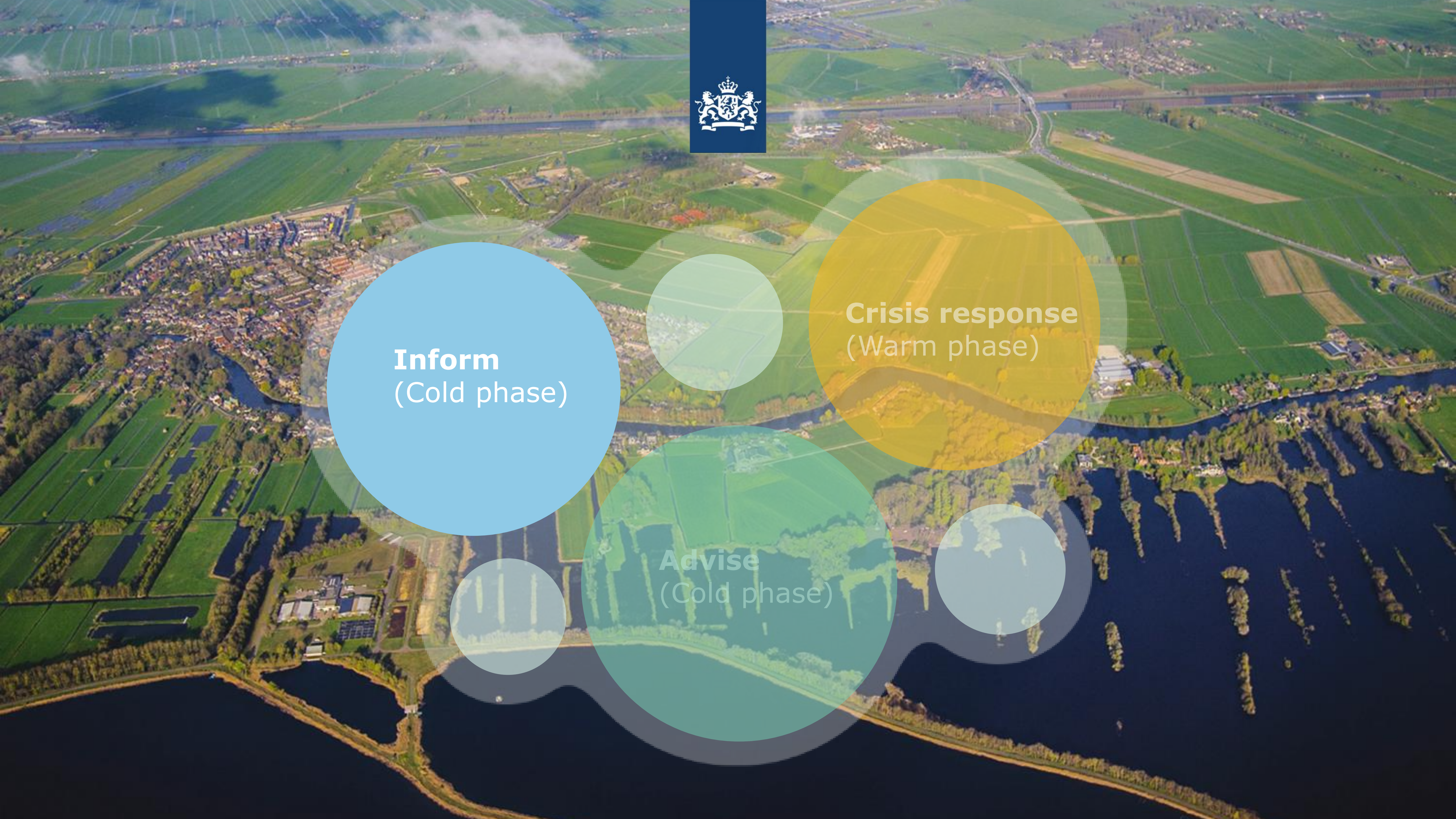
Coordinated Vulnerability Disclosure (CVD)


I hacked the Dutch government
and all I got was this lousy t-shirt

https://www.reddit.com/r/hacking/comments/wdohl1/i_hacked_the_dutch_government_and_they_sent_me/

How my team contributes

1. Software Bill of Materials starterguide: https://english.ncsc.nl/publications/publications/2024/july/30/software-bill-of-materials-starter-guide
2. Automatic security playbooks: https://www.ncsc.nl/wat-doet-het-ncsc-voor-jou/onderzoek/onderzoeksresultaten/tno-
3. SoC of the future: https://english.ncsc.nl/publications/publications/2024/june/27/index


Inform
(Cold phase)
Crisis response
(Warm phase)
Advise
(Cold phase)

Actionable information

What does the NCSC share?

Indicators of Compromise (IoC’s) for het Security Operations Center (SoC)

Threat Alyses voor de Chief Information Security Officer (CISO)

Advisories from open and closed sources voor IT / SOC

Reports en advisories

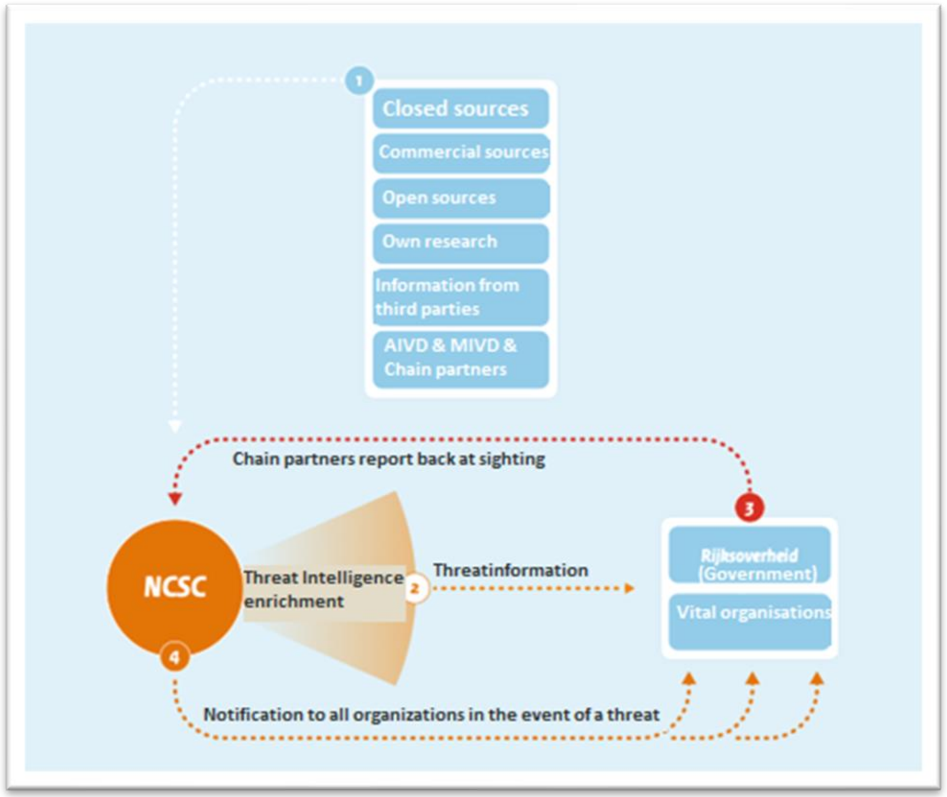
24/7 alert and ready for escalation (up to the level of the prime minister)

cert@ncsc.nl open for reports

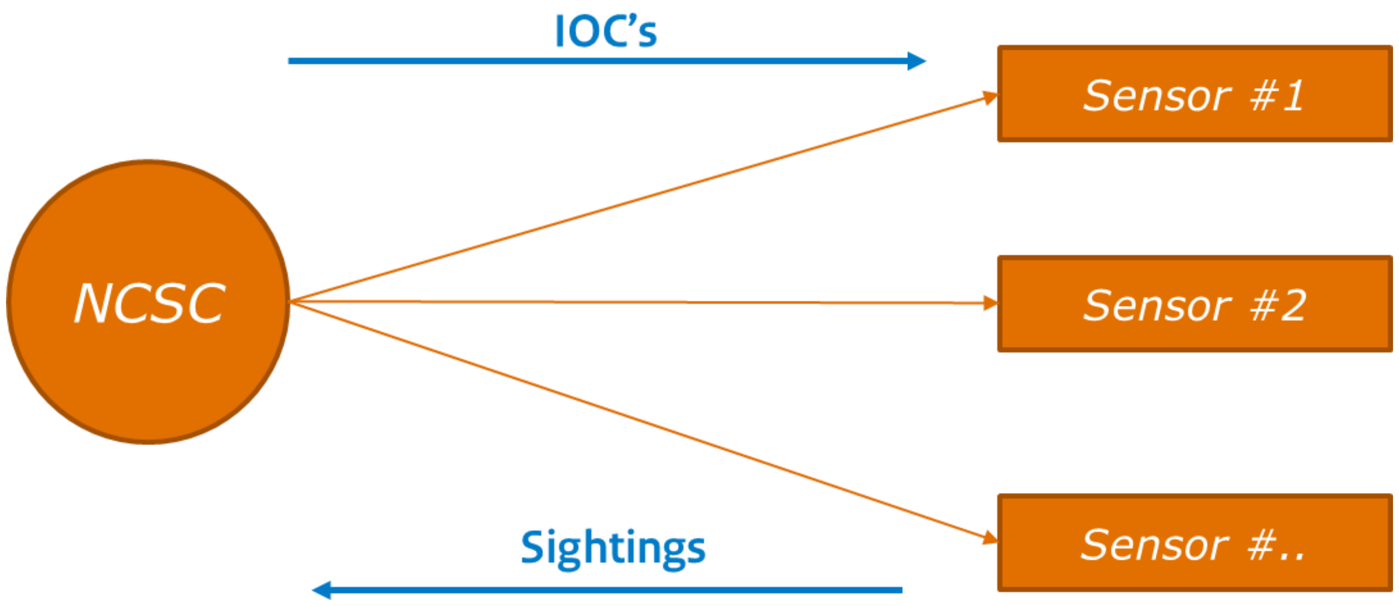
Daily advisories https://advisories.ncsc.nl/advisories

Shifting towards vulnerability API https://vulnerabilities.ncsc.nl/

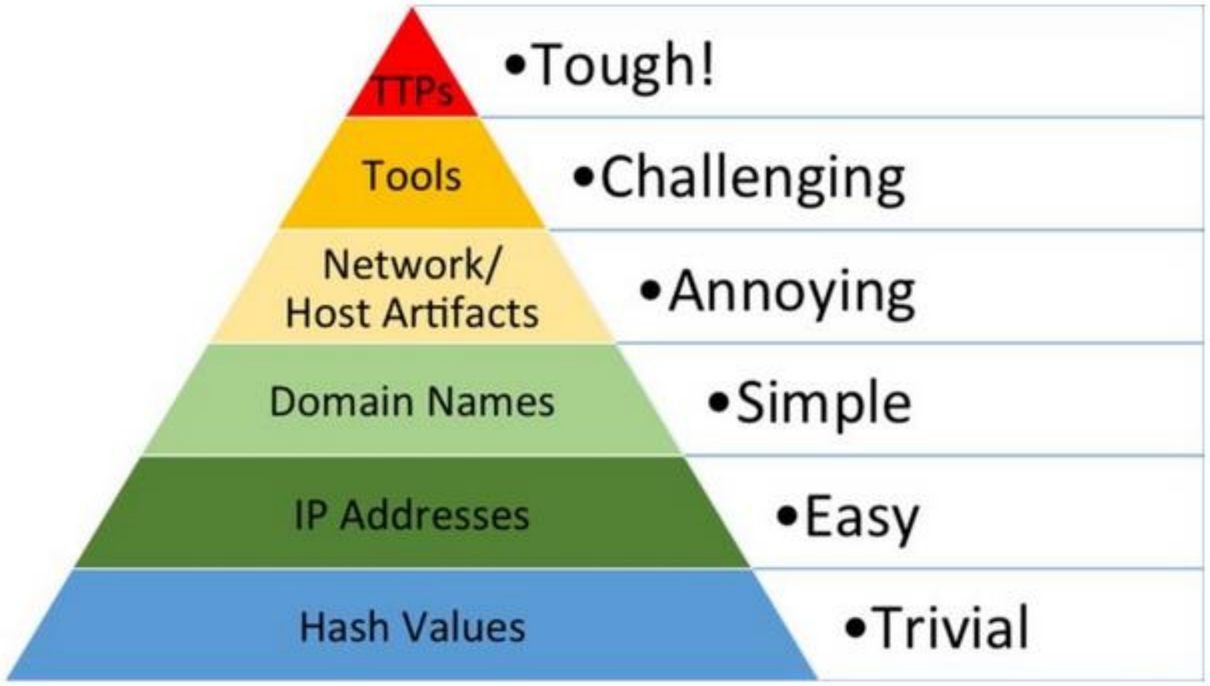
National Detection Network (NDN)



1
Closed sources
Commercial sources
Open sources
Own research
Information from third parties
AIVD & MIVD & Chain partners
Chain partners report back at sighting
3
NCSC
Threat Intelligence enrichment
2
Threatinformation
Rijksoverheid (Government)
Vital organisations
4
Notification to all organizations in the event of a threat



Pyramid of Pain



https://www.attackiq.com/glossary/pyramid-of-pain/

How my team contributes

1. Research into the use of advisories
   https://pure.tudelft.nl/ws/portalfiles/portal/134568654/No_One_Drinks_From_the_Firehose.pdf
2. Research into the automation of cyber security operations

Inform
(Cold fase)
Crisis response
(Warm phase)
Advice
(Cold phase)

Resilience

How can the NCSC contribute to the *prevention* of incidents?

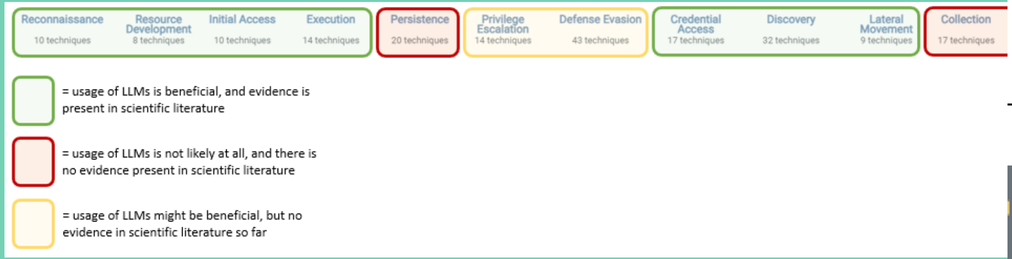
The NCSC stimulates, advices and supports organisations in strengthening their resilience.

- Cyber essentials
- Insights on threats, e.g. malware, phishing, ransomware
- Advice on risk management
- Advice on detection and response
- **Guidelines for developing safe software**

https://www.ncsc.nl/documenten

How my team contributes

1. Study into threat modelling with the KU Leuven (https://www.ncsc.nl/documenten/publicaties/2024/mei/7/index)
2. Study into the risks of AI in the hands of adversaries (https://www.ncsc.nl/documenten/publicaties/2024/mei/21/index)
3. **Study into new tools for static analysis**

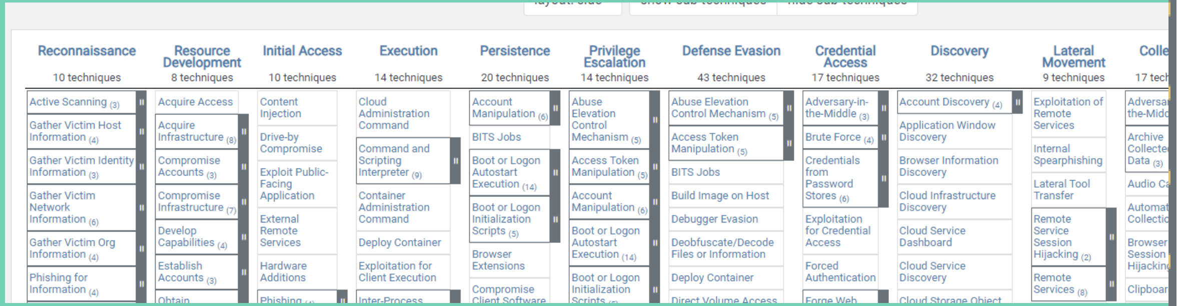


Reconnaissance	Resource Development	Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection
10 techniques	8 techniques	10 techniques	14 techniques	20 techniques	14 techniques	43 techniques	17 techniques	32 techniques	9 techniques	17 techniques

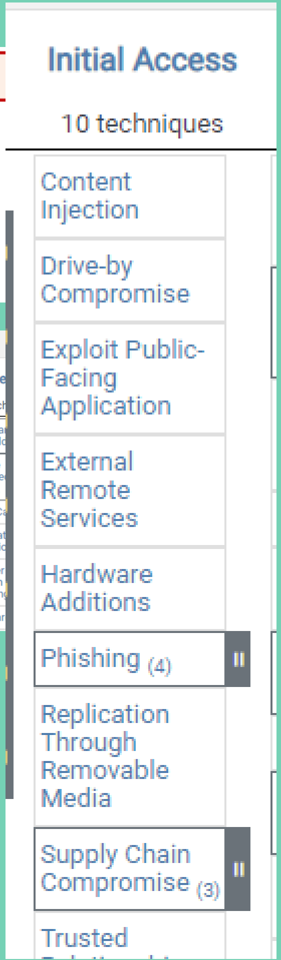
🟩 = usage of LLMs is beneficial, and evidence is present in scientific literature

🟥 = usage of LLMs is not likely at all, and there is no evidence present in scientific literature

🟨 = usage of LLMs might be beneficial, but no evidence in scientific literature so far



Reconnaissance 10 techniques	Resource Development 8 techniques	Initial Access 10 techniques	Execution 14 techniques	Persistence 20 techniques	Privilege Escalation 14 techniques	Defense Evasion 43 techniques	Credential Access 17 techniques	Discovery 32 techniques	Lateral Movement 9 techniques	Colle 17 tech
Active Scanning (3)	Acquire Access	Content Injection	Cloud Administration Command	Account Manipulation (6)	Abuse Elevation Control Mechanism (5)	Abuse Elevation Control Mechanism (5)	Adversary-in-the-Middle (3)	Account Discovery (4)	Exploitation of Remote Services	Adversa the-Midd
Gather Victim Host Information (4)	Acquire Infrastructure (8)	Drive-by Compromise	Command and Scripting Interpreter (9)	BITS Jobs	Access Token Manipulation (5)	Access Token Manipulation (5)	Brute Force (4)	Application Window Discovery	Internal Spearphishing	Archive Collecte Data (3)
Gather Victim Identity Information (3)	Compromise Accounts (3)	Exploit Public-Facing Application	Container Administration Command	Boot or Logon Autostart Execution (14)	Account Manipulation (6)	BITS Jobs	Credentials from Password Stores (6)	Browser Information Discovery	Lateral Tool Transfer	Audio Ca
Gather Victim Network Information (6)	Compromise Infrastructure (7)	External Remote Services	Deploy Container	Boot or Logon Initialization Scripts (5)	Boot or Logon Autostart Execution (14)	Build Image on Host	Exploitation for Credential Access	Cloud Infrastructure Discovery	Remote Service Session Hijacking (2)	Automat Collectio
Gather Victim Org Information (4)	Develop Capabilities (4)	Hardware Additions	Exploitation for Client Execution	Browser Extensions	Boot or Logon Initialization Scripts	Debugger Evasion	Forced Authentication	Cloud Service Dashboard	Remote Services (8)	Browser Session Hijackin
Phishing for Information (4)	Establish Accounts (3)	Phishing	Inter-Process	Compromise Client Software		Deobfuscate/Decode Files or Information	Forge Web	Cloud Service Discovery		Clipboar
	Obtain					Deploy Container		Cloud Storage Object		
						Direct Volume Access				



Initial Access

10 techniques

- Content Injection
- Drive-by Compromise
- Exploit Public-Facing Application
- External Remote Services
- Hardware Additions
- Phishing (4)
- Replication Through Removable Media
- Supply Chain Compromise (3)
- Trusted

https://attack.mitre.org/

What is static analysis and why is it important?

- Static analysis is about analysing software without running it
- Mijn research is about automatic static analysis tools (ASATs), such as SonarQube.
- ASATs discover vulnerabilities relatively early in the software development process.
- Another advantage compared to dynamic analysis: it is easier to cover the whole code.
- The use of ASATs is a widely accepted best practice for secure software development (Microsoft SDL, NCSC guidelines)

My research is about ASATs which:

- Support at least one popular programming language
- Find more than just formatting and dependency bugs
- Use novel techniques
- Are relatively popular

1 CodeQL

2 Infer

3 Semgrep OSS

4 Snyk Code

CodeQL

- Transforms code into a database
- This database can be queried using a Datalog-based query language
- This query language is object-oriented and supports recursive queries

Infer

- Uses separation logic for efficient summary-based reasoning with no false negatives.
- Is actually a *verification* tool, automated through the use of bi-abduction.

Semgrep OSS

- Rules are written in a YAML file format
- Has an easy-to-use but capable pattern matching engine
- Supports taint-tracking
- Supports autofixing

Some preliminary comparative findings

- Infer detects fewer issue-types compared to SonarQube, but detect those issue-types with higher precision. (Liu et al. 2023).
- Infer outperforms SonarQube both in TPR and in FPR (Ablasser 2019)
- Semgrep and CodeQL greatly outperform SonarQube on a synthetic benchmark, but perform similar on actual vulnerabilities (Li et al. 2023, Li et al. 2024)
- On C++ programs, CodeQL outperforms Semgrep both on real and synthetic benchmarks (Li et al. 2024)
- Incrementalisation (one of the strong suits of Infer) is difficult to achieve with CodeQL (Szabo 2024)

Query-based SAST-tools

- Rules are external rather than hardcoded into the tool
- Easier to quickly incorporate new vulnerability classes
- It also makes the tools suitable for security research

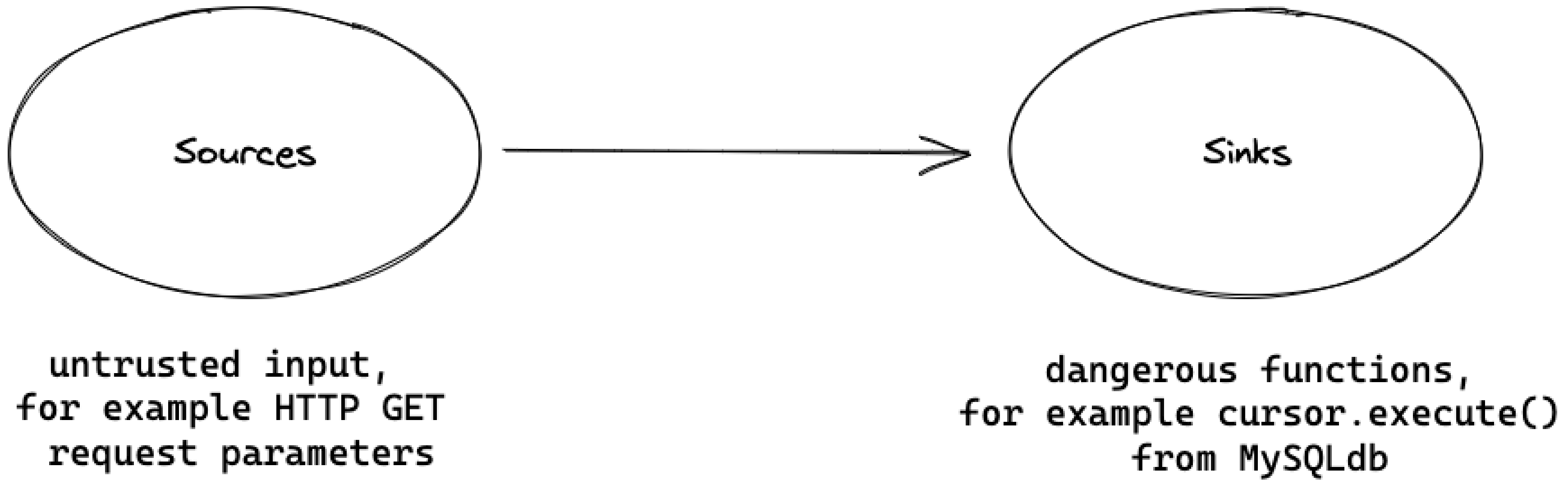
1 **CodeQL**

2 Infer

3 **Semgrep OSS**

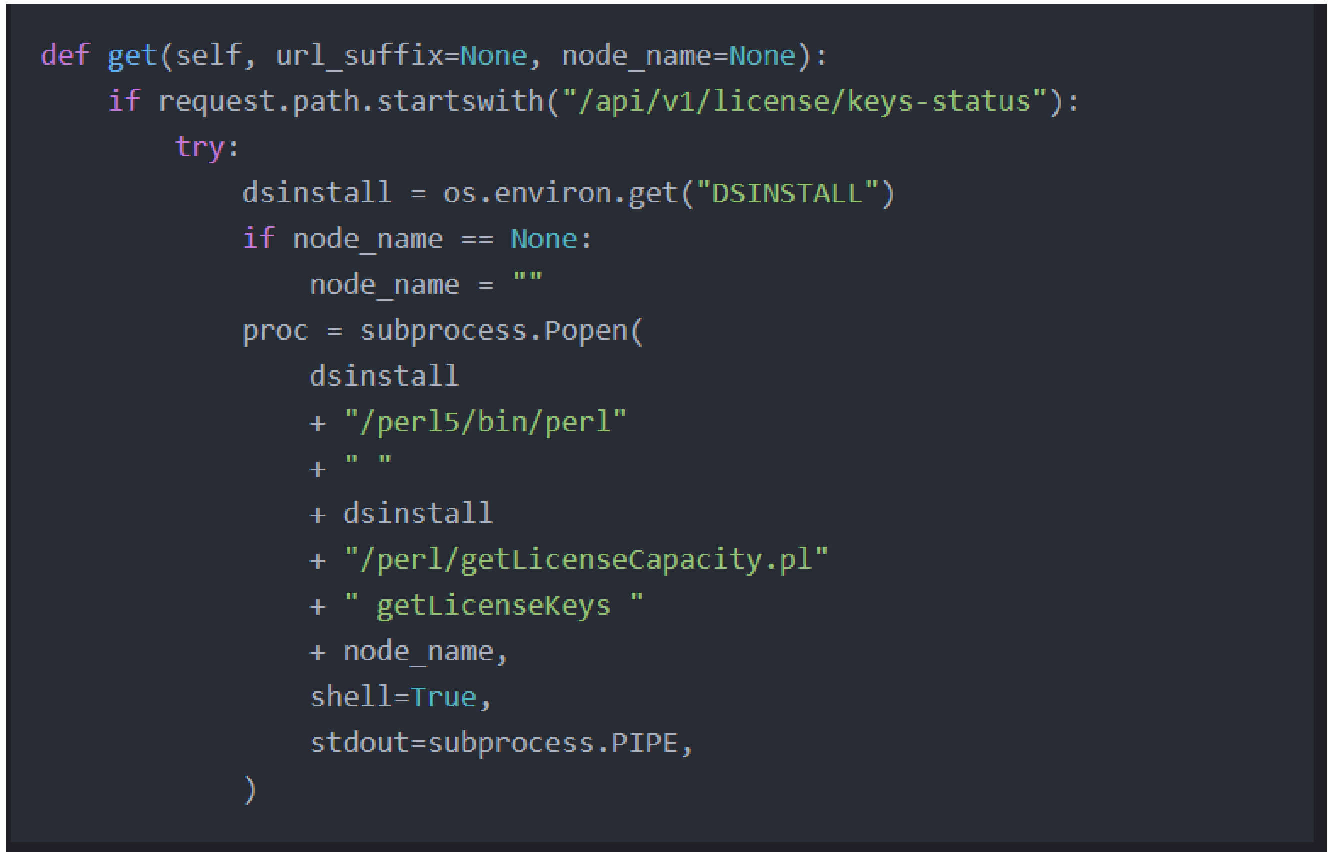
4 Snyk Code

General pattern for vulnerabilities


Sources
Sinks
untrusted input, for example HTTP GET request parameters
dangerous functions, for example cursor.execute() from MySQLdb

https://github.blog/developer-skills/github/codeql-zero-to-hero-part-1-the-fundamentals-of-static-analysis-for-vulnerability-research/

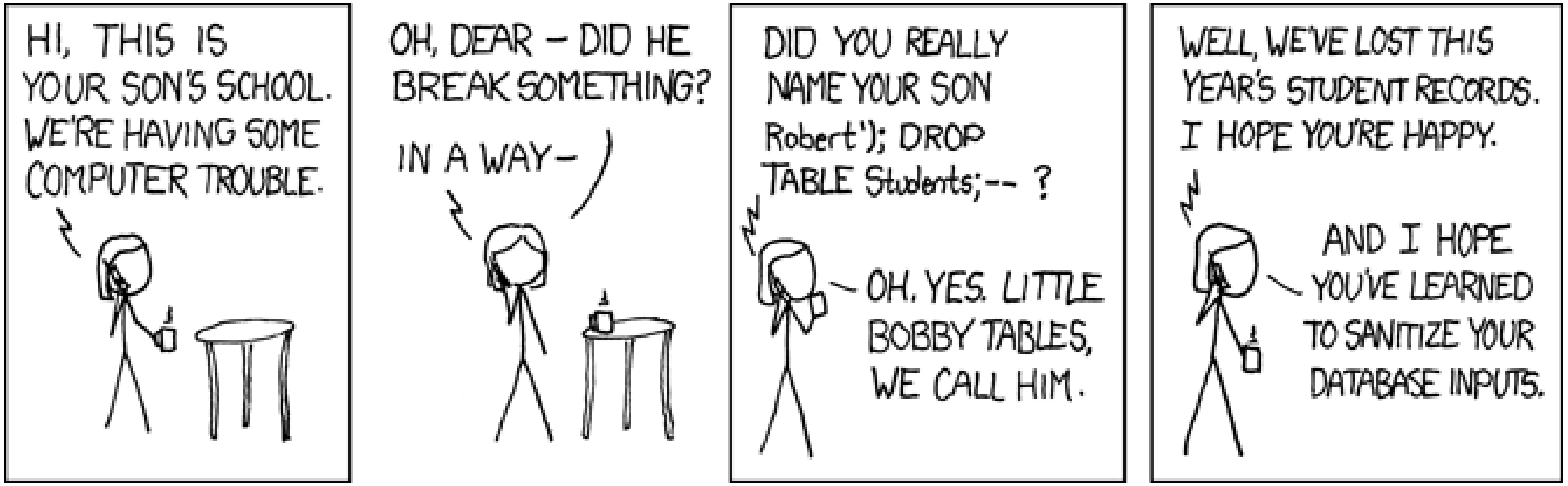
Code from Ivanti REST API



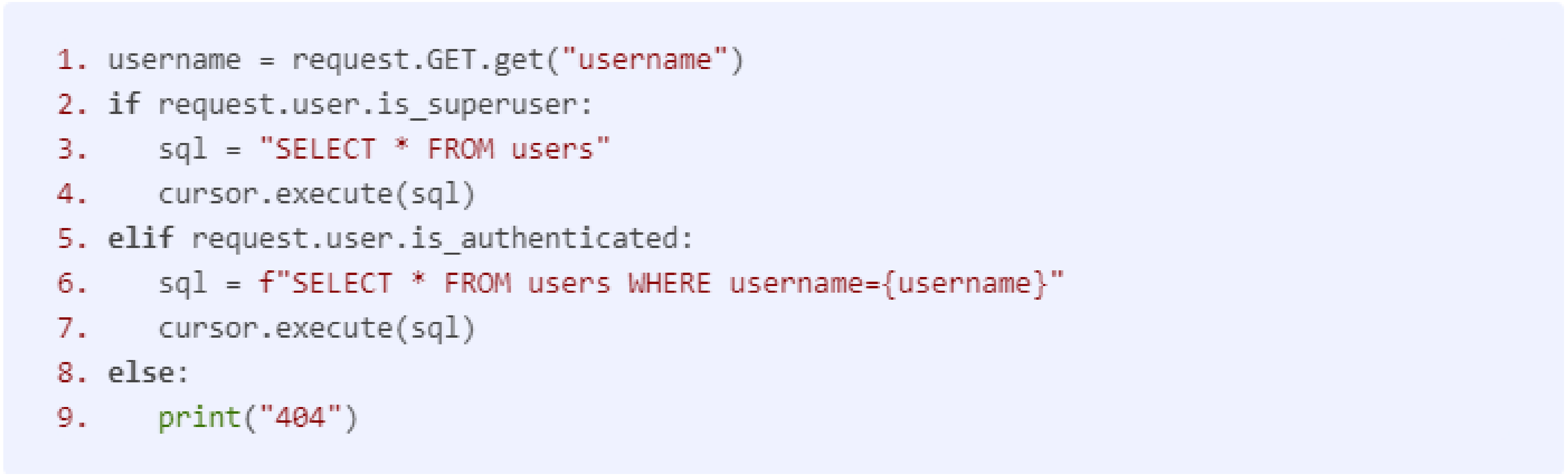
```
def get(self, url_suffix=None, node_name=None):
    if request.path.startswith("/api/v1/license/keys-status"):
        try:
            dsinstall = os.environ.get("DSINSTALL")
            if node_name == None:
                node_name = ""
            proc = subprocess.Popen(
                dsinstall
                + "/perl5/bin/perl"
                + " "
                + dsinstall
                + "/perl/getLicenseCapacity.pl"
                + " getLicenseKeys "
                + node_name,
                shell=True,
                stdout=subprocess.PIPE,
            )
```

https://www.assetnote.io/resources/research/high-signal-detection-and-exploitation-of-ivantis-pulse-connect-secure-auth-bypass-rce

SQL Injection

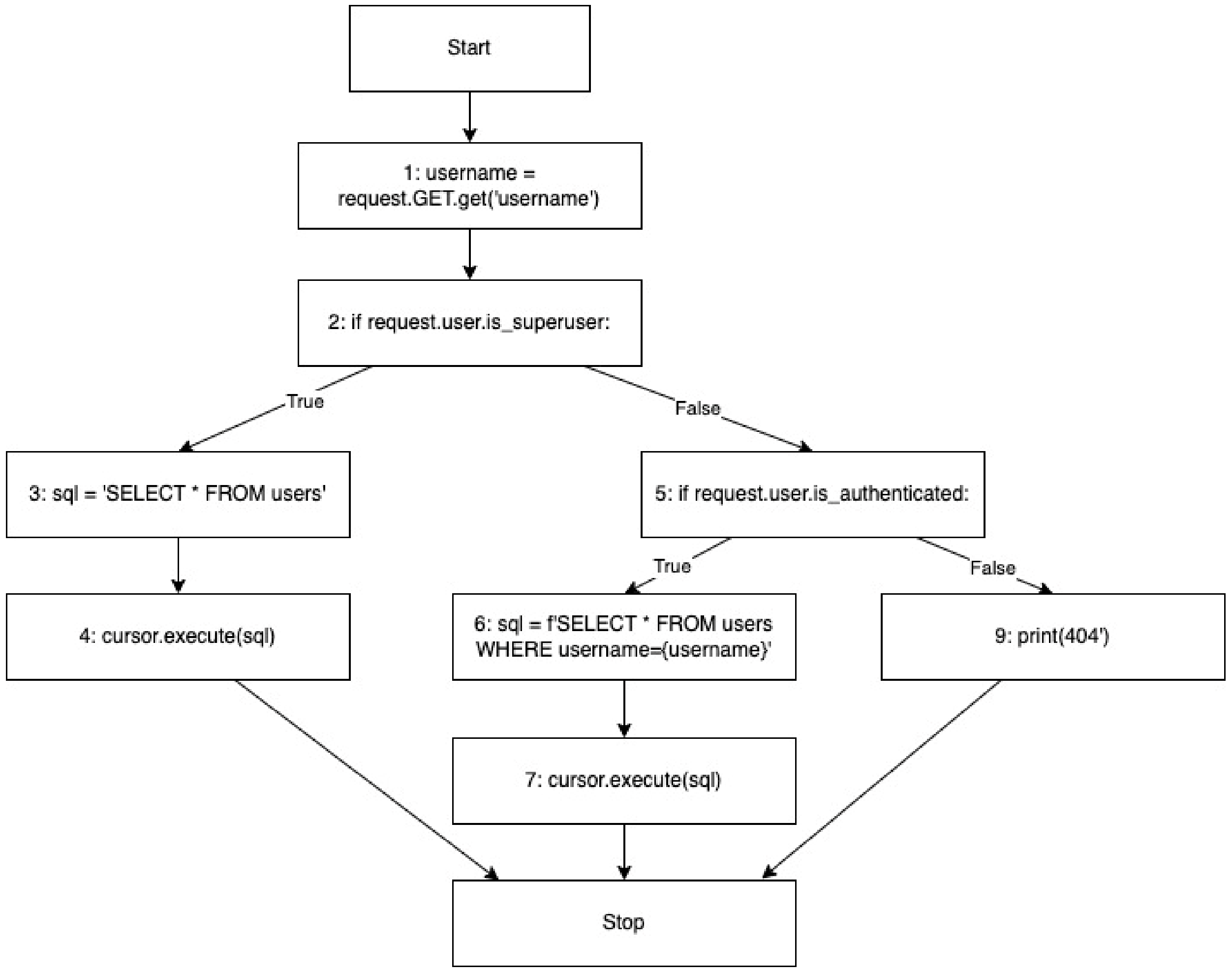

HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

https://xkcd.com/327/



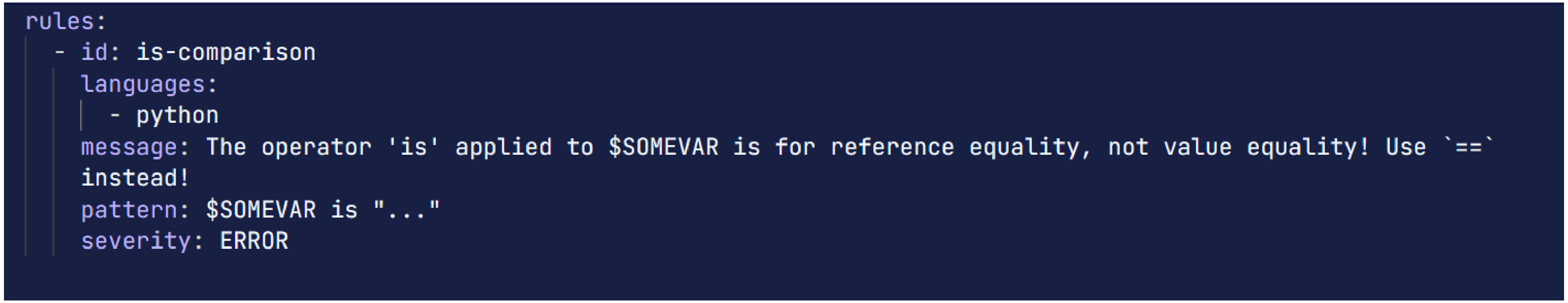
```python
username = request.GET.get("username")
if request.user.is_superuser:
    sql = "SELECT * FROM users"
    cursor.execute(sql)
elif request.user.is_authenticated:
    sql = f"SELECT * FROM users WHERE username={username}"
    cursor.execute(sql)
else:
    print("404")
```

https://github.blog/developer-skills/github/codeql-zero-to-hero-part-1-the-fundamentals-of-static-analysis-for-vulnerability-research/


Start
1: username = request.GET.get('username')
2: if request.user.is_superuser:
True
False
3: sql = 'SELECT * FROM users'
5: if request.user.is_authenticated:
True
False
4: cursor.execute(sql)
6: sql = f'SELECT * FROM users WHERE username={username}'
9: print(404')
7: cursor.execute(sql)
Stop

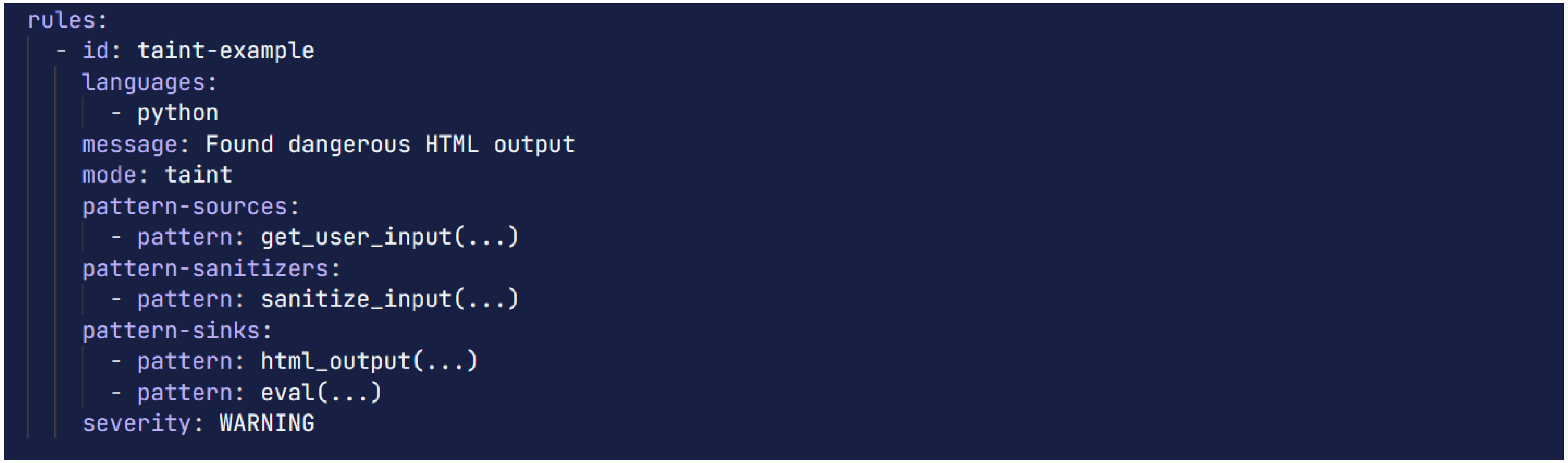
https://github.blog/developer-skills/github/codeql-zero-to-hero-part-1-the-fundamentals-of-static-analysis-for-vulnerability-research/

An example rule



```yaml
rules:
  - id: is-comparison
    languages:
      - python
    message: The operator 'is' applied to $SOMEVAR is for reference equality, not value equality! Use `==`
    instead!
    pattern: $SOMEVAR is "..."
    severity: ERROR
```

Taint tracking



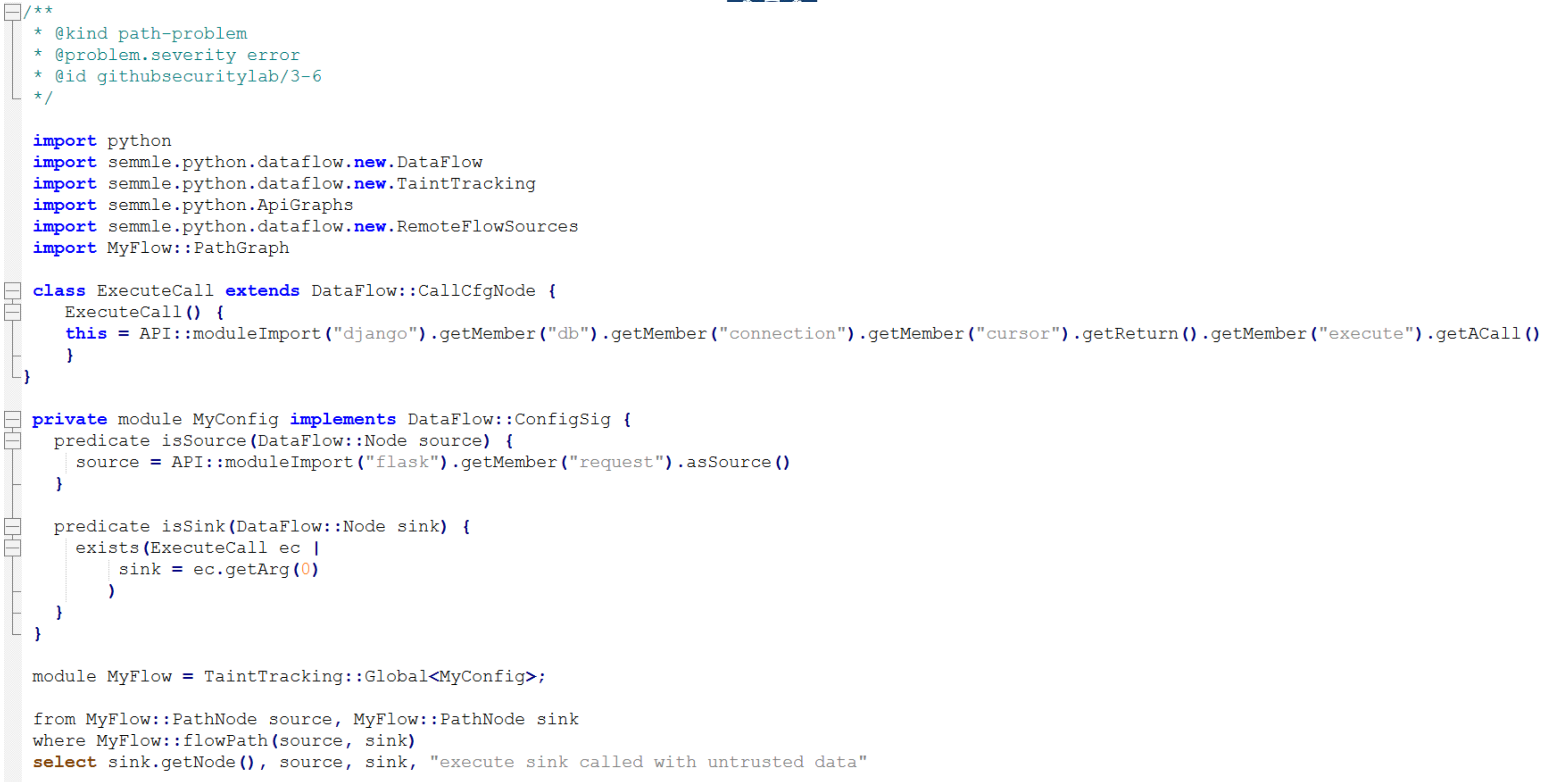
```
rules:
  - id: taint-example
    languages:
      - python
    message: Found dangerous HTML output
    mode: taint
    pattern-sources:
      - pattern: get_user_input(...)
    pattern-sanitizers:
      - pattern: sanitize_input(...)
    pattern-sinks:
      - pattern: html_output(...)
      - pattern: eval(...)
    severity: WARNING
```

An example query

```
if cond():
    pass
else:
    do_something
```

```
import python

from If i, StmtList l
where (l = i.getBody() or l = i.getOrelse())
  and forall(Stmt p | p = l.getAnItem() | p instanceof Pass)
select i
```



```
/**
 * @kind path-problem
 * @problem.severity error
 * @id githubsecuritylab/3-6
 */

import python
import semmle.python.dataflow.new.DataFlow
import semmle.python.dataflow.new.TaintTracking
import semmle.python.ApiGraphs
import semmle.python.dataflow.new.RemoteFlowSources
import MyFlow::PathGraph

class ExecuteCall extends DataFlow::CallCfgNode {
    ExecuteCall() {
    this = API::moduleImport("django").getMember("db").getMember("connection").getMember("cursor").getReturn().getMember("execute").getACall()
    }
}

private module MyConfig implements DataFlow::ConfigSig {
  predicate isSource(DataFlow::Node source) {
    source = API::moduleImport("flask").getMember("request").asSource()
  }

  predicate isSink(DataFlow::Node sink) {
    exists(ExecuteCall ec |
        sink = ec.getArg(0)
        )
  }
}

module MyFlow = TaintTracking::Global<MyConfig>;

from MyFlow::PathNode source, MyFlow::PathNode sink
where MyFlow::flowPath(source, sink)
select sink.getNode(), source, sink, "execute sink called with untrusted data"
```

Tutorial

github.com/JanRooduijn/tutorial-radboud

Nationaal Cyber Security Centrum

Nederland Digitaal Veilig