Integrity Monitoring

Detecting changes in the filesystem

Klaus Möller
WP8-T1

Webinar, 7th of August 2020

Public

www.geant.org

What is Integrity Monitoring?

- “the process of validating the integrity of operating system files and directories”
- Integrity: file/directory content and metadata are unchanged with regards to a given “known good” state
- However: changes to files and directories are intented
  - Updates: OS, software etc.
  - Configuration changes: users, network (addresses), settings, etc.
- Detecting unauthorized or unintended changes
  - Those made by attacks or mistakes

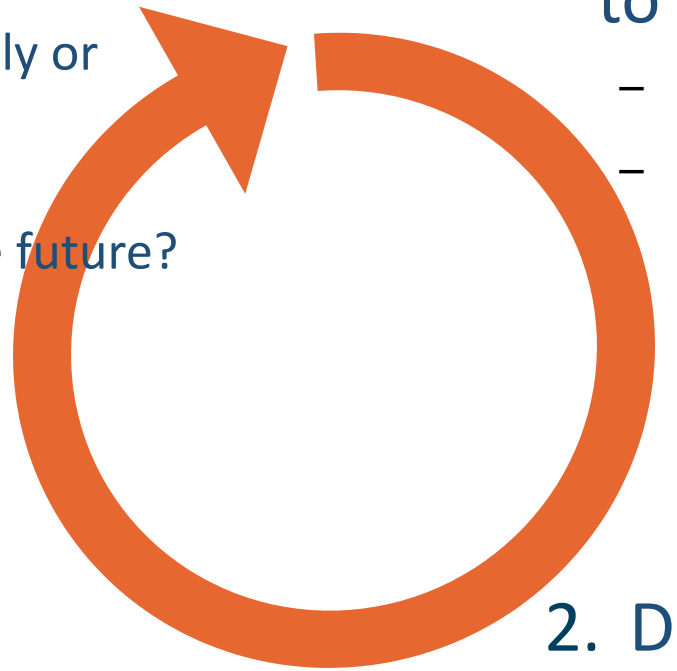
⟶ **File Integrity Monitoring (FIM)**

Why is (File) Integrity Monitoring useful?

- Assessing the impact of integrity violations
  - Which changes were made?
    - I.e. new firewall rules, new users, changed daemon/service configurations, unparseble configurations/libraries
    - Changed binaries/libraries/kernel (modules/drivers), ...
    - Additionally installed or removed files?
      - Crypto-miners, Spam-SW, phishing pages, AV, firewall, FIM, …
  - What happened? How did it happen?
- Detecting unintended changes
  - Critial: configuration mistakes that open weaknesses
    - Empty passwords, disabling authentication, …

Integrity monitoring workflow

1. Plan: what to monitor, how to monitor
   - Systems, files, directories
   - Attributes: content, permissions, etc.
2. Do: Take Baseline
   - Record valid state(s)
3. Check: at regular intervals
   - Examine attributes of monitored files & directories
   - Compare exam results with baseline → Report
4. Act
   - Valid state?
   - Change deliberately or unintentionally?
   - Consequences?
   - Preventable in the future?



Plan: what systems should be monitored?

- Rule of thumb: By order of impact/mission criticality
  - Look at your Business Impact Analysis (if present)
  - Identity management, authentication databases/servers
    - I.e. KDCs, Domain Controllers, LDAP servers with authentication information
    - Compromising these will compromise most other systems
  - Systems storing your mission critical data
    - Database servers, file servers, backup servers
  - Security critical systems
    - Firewalls, SIEM, loghost, …
  - Other mission critical systems
    - Webservers, application servers, load balancers, VM-hosts, central switches/routers, central DNS, central Email, HR, CRM, …

Plan: what files should be monitored?

- Trusted computing base
  - Kernel, kernel modules/drivers
    - `/boot, /lib/modules, C:\BOOTMGR, C:\Boot\BCD`
  - Binaries, libraries
    - `/bin, /usr/bin, /lib, /usr/lib`
    - `C:\Windows\System32`
    - Directories in $PATH (Linux) or %PATH% (Windows)
  - System configuration
    - Linux/Unix: `/etc`
    - Windows: Registry
  - Critical files in Home directories
    - `~/.ssh/authorized_keys, ~/.config`

Plan: Limited checks

- Sockets, named pipes, IPC objects
  - Reading (i.e. checksumming) will likely block
  - Inode number will change when socket gets re-created at boot
  - Permissions, ownership, major/minor device number can be monitored
- Symlinks
  - Not all FIMs will monitor where the symlink points to
- Confidential data
  - Key material, esp. private keys
  - No text diffs
  - May show up in text diffs or logs
- Temporary filesystems/directories
  - `/tmp, /usr/tmp, /var/tmp, /dev/shm, /run/user/, /etc/mntab`
  - Permissions (sticky bit) are OK

Plan: What to exclude from checking?

- Ephemeral/dynamic file systems
  - `/proc, /sys, /dev, /etc/mntab`
  - Too many changes in operation to be useful
- Network file systems
  - NFS, CIFS, AFS, etc.
  - Check these on the server – not over the network
- Removable media
  - USB/flash drives, CD/DVD/BD, Floppy(?)
  - Content will change with different media mounted

Plan: what attributes should be monitored?

- Content, of course
  - Complete file? – that's called a backup;)
  - Usually cryptographic checksums: SHA256, … (too often still MD5, SHA1)
  - For very large files (> 1 GByte), checksumming may take too long
  - Full content for small (vital) text files - allows diff to show changes
- Permissions/ACLs
  - S-UID/S-GID bits
  - Write permissions on configuration files for ordinary users?
  - Read permissions for world appearing on confidential data?
- Owner, Group
  - System binaries/libraries should be owned by root
- Size
  - Binaries, libraries should not change size – except through updates
  - Others (log files) should only grow – what about log rotation?

Plan: what attributes could be monitored?

- Device ID, Inode:
  - Somebody might have replaced stuff with mounts to another filesystem
- Number of links:
  - Each file in a directory has one link to it, plus itself and the parent directory
  - Hidden files/directories will show up as mismatch on link count
  - Works well on Ext2/3/4, vfat, and (old) standard Unix filesystems
  - Does not work with modern filesystems: XFS, Btrfs, ZFS
- MACtimes
  - Modification
  - Access
  - Creation (Windows), Change (of metadata/inode: Linux/Unix)
  - B(orn) or D(eleted) timestamps – if supported by filesystem
  - Timestamps can be changed by attackers
    - Even creation with root privileges & anti-forensic tools

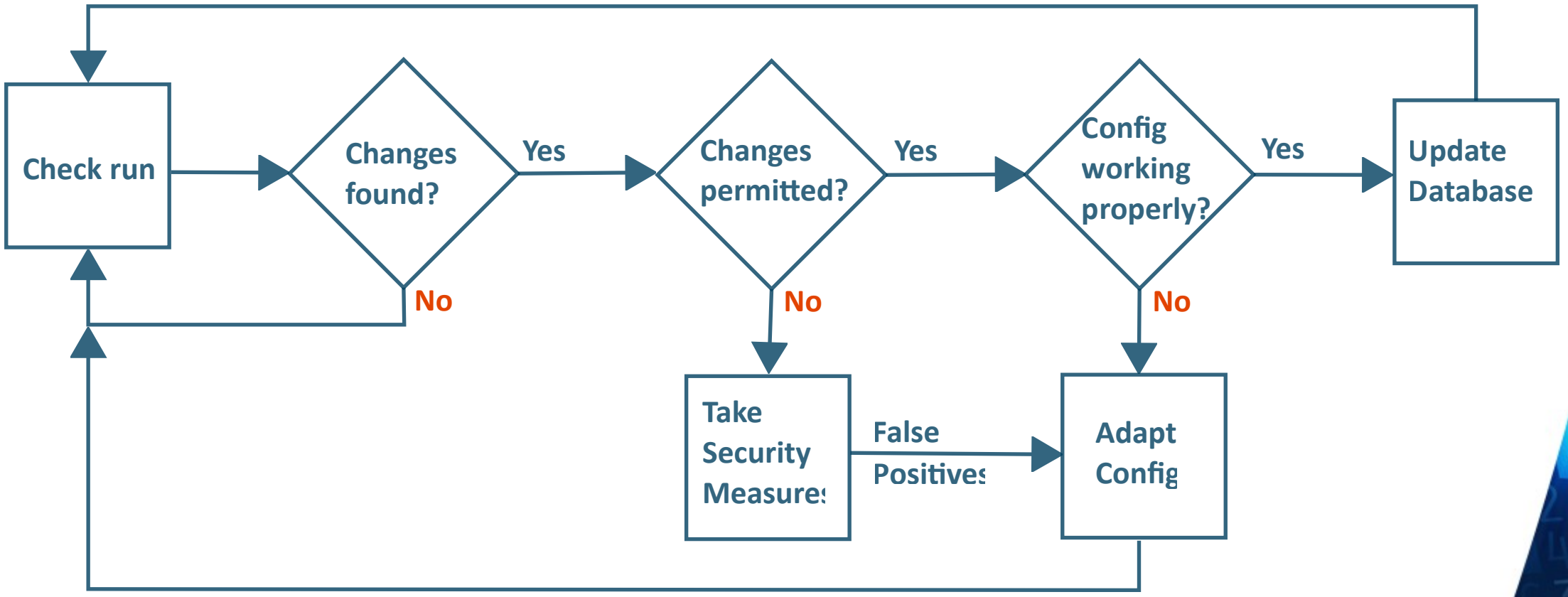
Do: Baselining

- Naive: `find / -print0 | xargs -0 sha256sum > /tmp/my.db`
- Baseline must be secured against tampering/loss
  - Best done by keeping on a central server
  - Same goes for configuration of the FIM
  - If kept locally, sign digitally, check before use
  - Availability issues: deleted locally, no network, how to act then...
- Baseline must be taken form a “known good”/valid/legal state
  - After a fresh/complete install?
  - After initial setup?
  - Patches, updates, later installs?

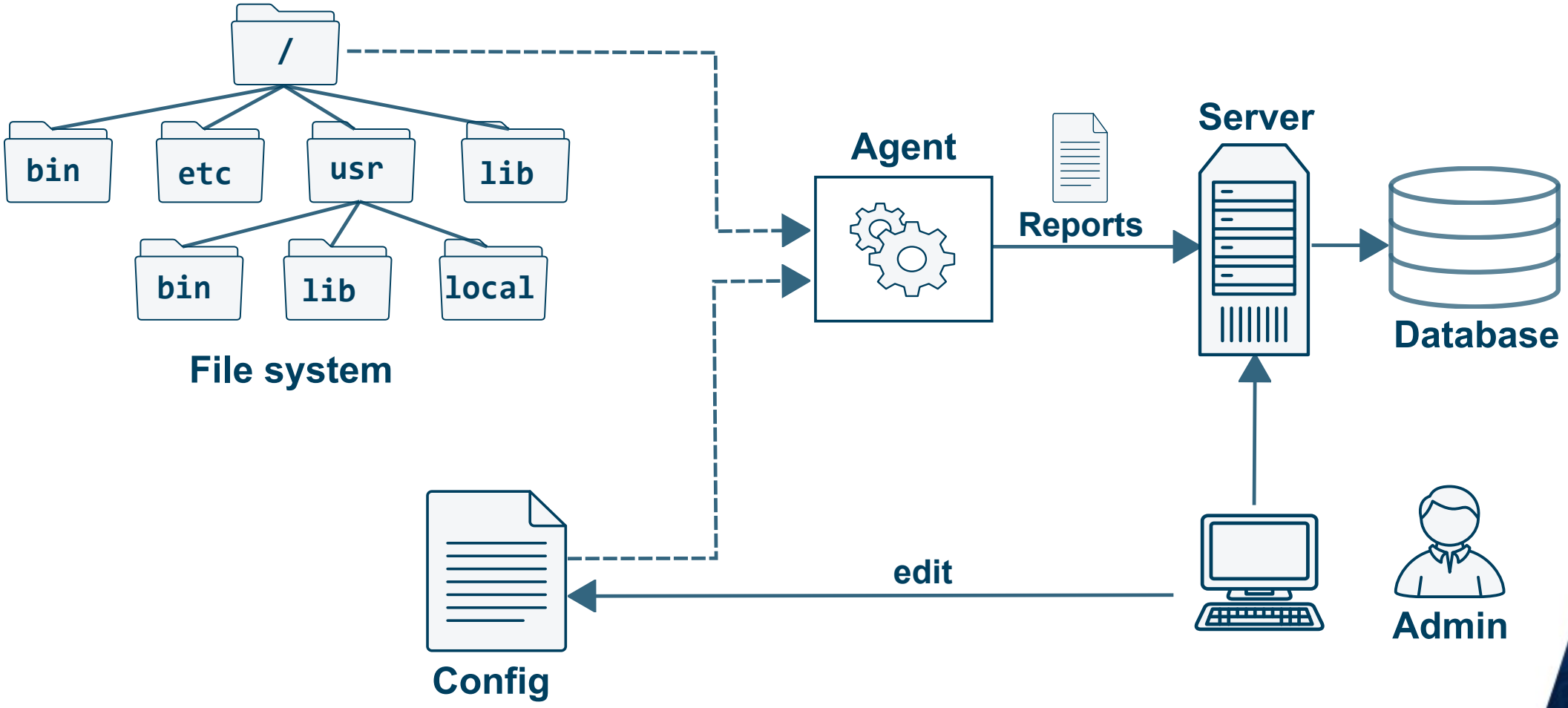
Check

- How often to check?
  - Depends, anywhere between once/hour and once/day
  - More checks – more work, more load on the systems
  - OTOH: checking more often may spot attacks earlier
  - Ideal: real-time monitoring for changes (Linux: inotify system call)
- What to report?
  - Need **actionable** data: Report + Background = enough information to draft a plan to act upon
- How to report changes?
  - Log messages (syslog, eventlog) – best to SIEM/central loghost
  - Email (standalone systems)
  - Console log?

Act: Workflow



FIM: Schema

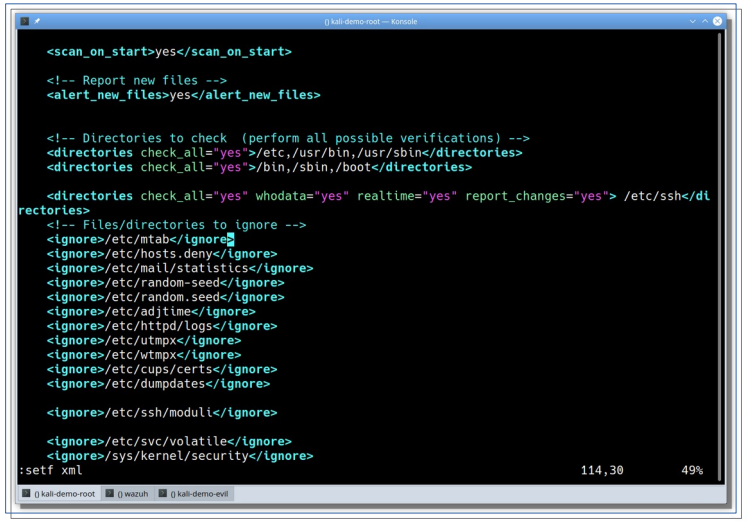

/
bin
etc
usr
lib
bin
lib
local
File system
Agent
Reports
Server
Database
Config
edit
Admin

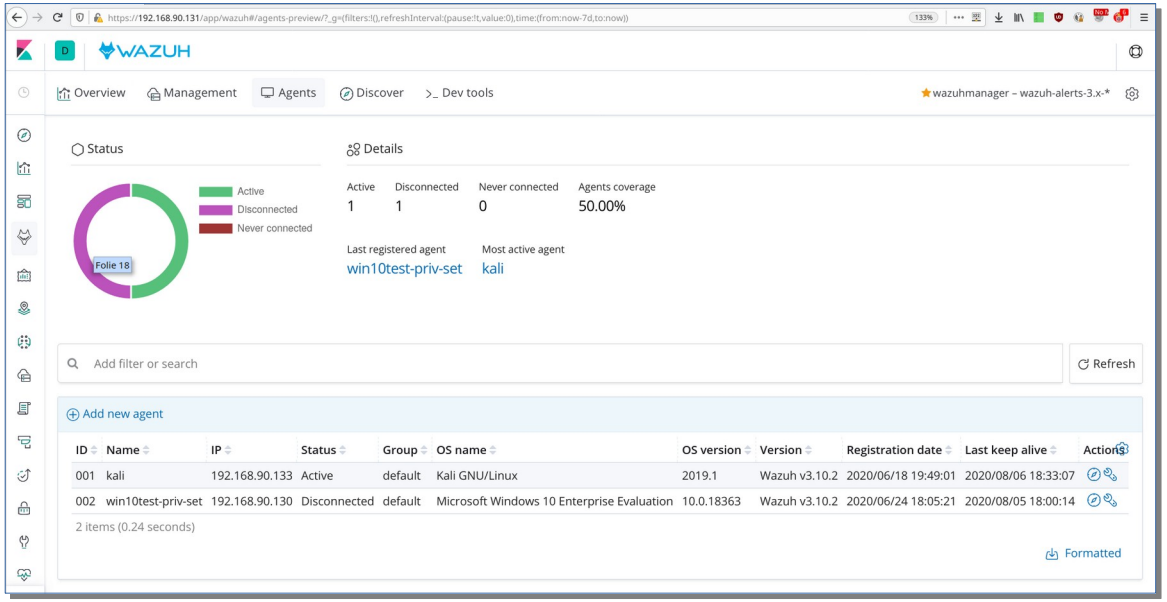
How to start?

- Begin small
  - One or two servers, only a handful of files
  - Can be implemented on spare hardware
- Observe, adapt, expand
  - Learn how and when changes happen and why
  - Adapt your configuration
  - Write down in knowledge base
- Expand bit-by-bit
  - Have a plan (what to monitor)
  - It’s better to observe too few things than too much

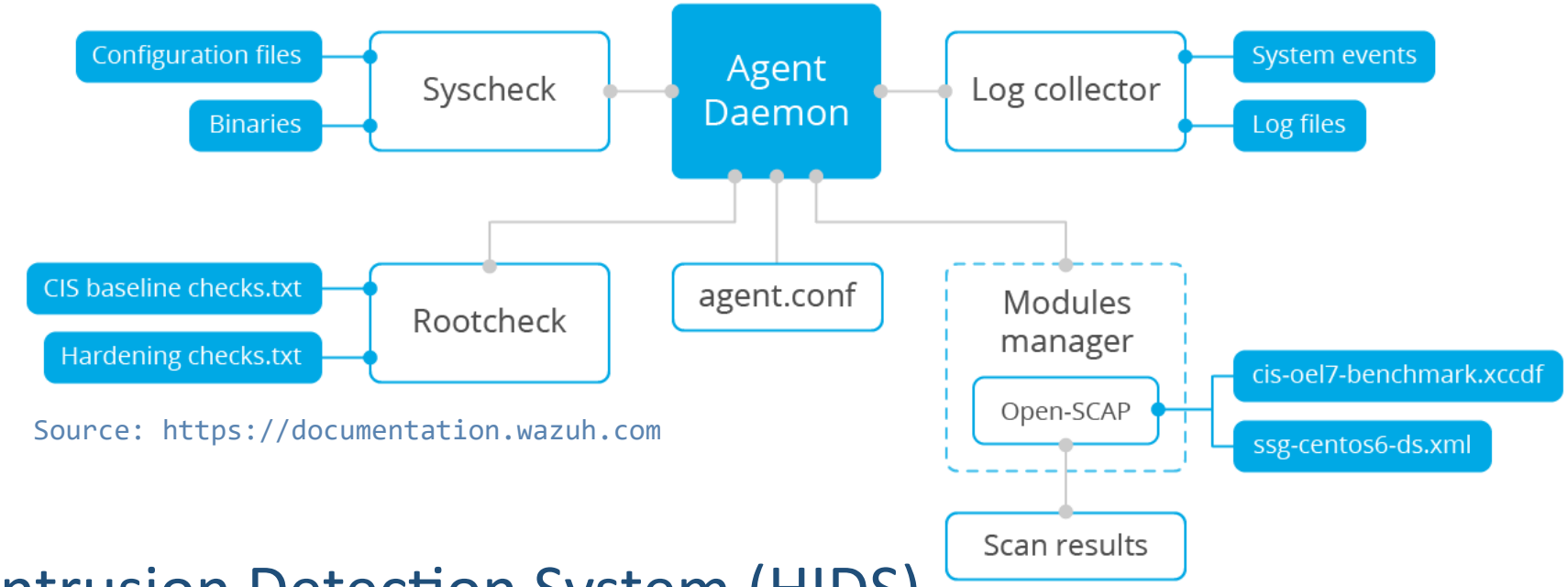
Wazuh Live Demonstration

- Configuring syscheck
- Adding/deleting a file
- Changing the content of a file
- Looking into events/reports





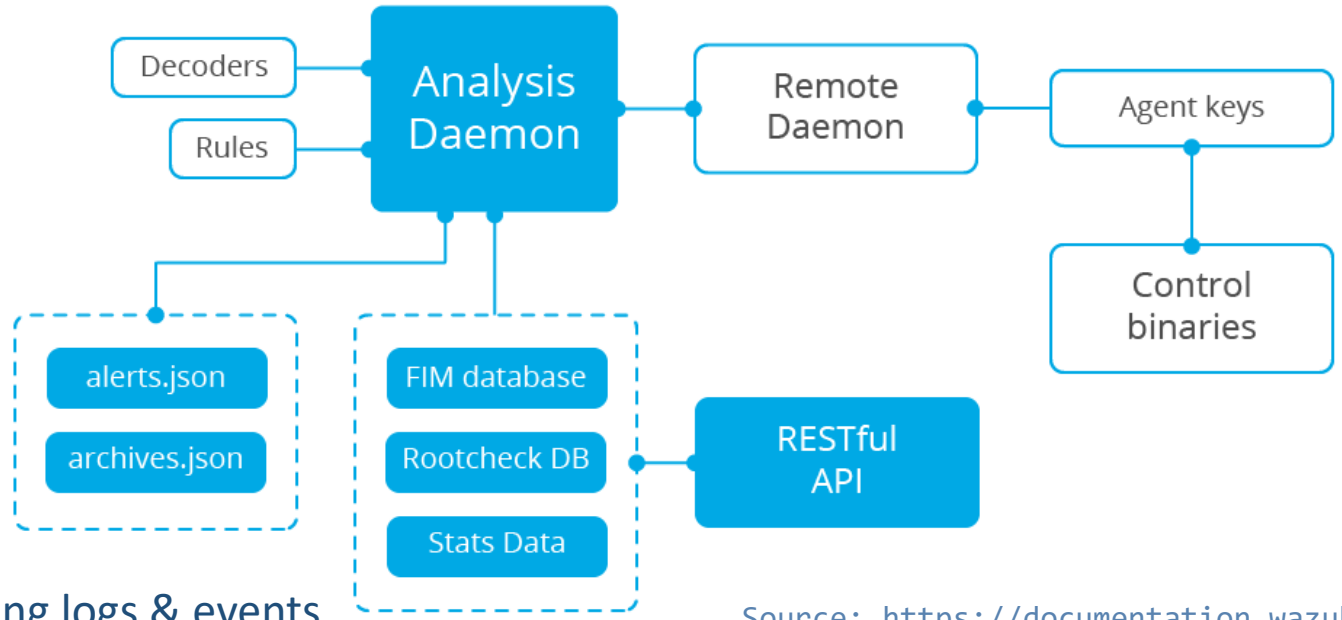
Wazuh: Agent



Source: https://documentation.wazuh.com

- Full Host Intrusion Detection System (HIDS)
  - Syscheck: Integrated FIM
  - Rootcheck: configuration check & rootkit detection
  - Log collector: Event & log file monitoring/forwarding (Filebeat)
  - Modules Manager: Place to plug-in user defined (scan) modules

Wazuh: Server



Source: https://documentation.wazuh.com

- Analysis Daemon
  - Decodes and analyses incoming logs & events
- Remote Daemon: Agent management
- Elasticstack: Kibana, Filebeat
  - Analysis (ElasticSearch)
  - Log/Event forwarding reception (Filebeat)
  - Dashboard (Kibana)

Limits of FIM: Malware

- Some malware doesn't write anything to the filesystem
  - What's not there, can't be found
  - But most malware needs a means of persistence: Autostart keys, kernel modules, boot loader/parameters, etc.
  - This will leave traces
- Rootkits hide files/directories from **every** user
  - What is not visible can't be checked or seen
  - But hiding a file/subdir also alters the parent directory: Timestamps, Link counts, etc.
- A thorough check will detect ***something***
- But it's up to the admin to pick up on strange reports

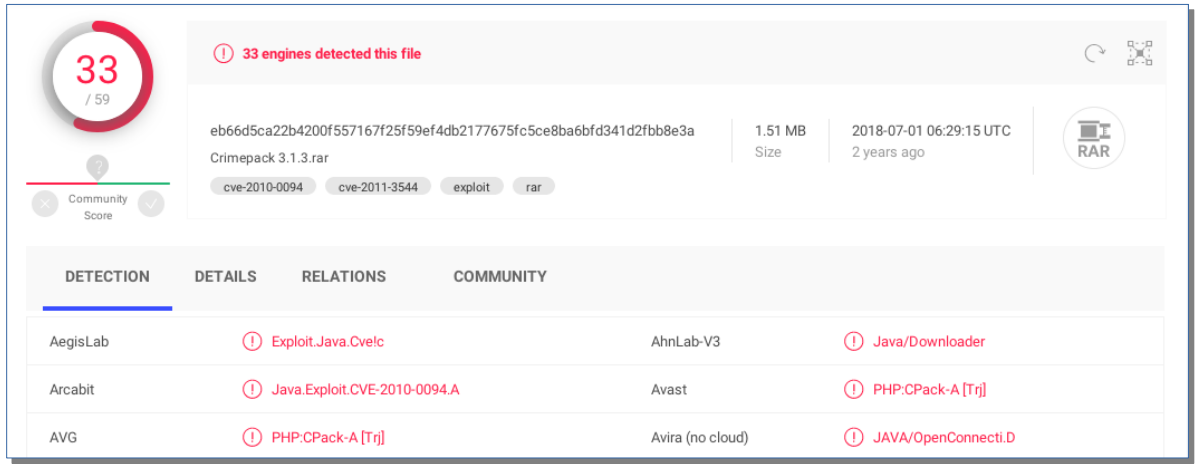
Limits of FIM: File signature evasion

- Find a collision,
  - I.e. a file that has the same cryptographic hash sum as the original
  - Can be done with weak/broken hash algorithms, like MD5 or SHA1
  - Very rare in practice
    - Do not confuse with cases where valid Authenticode signatures were used
    - These were made with leaked/stolen certificates
- Mitigation
  - Multiple checksums – attacker has to find collisions for all employed hash algorithms
  - Stronger hash algorithms: SHA256, SHA-512, SHA-3, etc.
  - Full content comparision, i.e. diff

Other uses for file hashes: Virustotal

- Unknown file, good or malicious?
  - Scan with your own Anti-Virus
    - What if it says nothing?
  - Use more AV-Scanner
  - **`https://virustotal.com`**
- Can’t/won’t send file
  - Malware upload blocked
  - May contain sensitive information
  - Search by cryptographic hash
  - md5, sha1, sha256

```
> sha256sum Crimepack\ 3.1.3.rar
eb66d5ca22b4200f557167f25f59ef4db2177675fc5ce8ba6bfd341d2fbb8e3a  Crimepack 3.1.3.rar
>
```



What have you learned?

- What integrity monitoring (at the OS level) is
- How to do integrity monitoring
- How to configure the integrity monitoring software

What has been left out?

- Boot process integrity
  - TPM, secure boot (MS), EVM/LMA (Linux)
- Binary signing under (elfsign - Linux, Authenticode - Windows)
- Cryptographic signing of files (PGP, S/MIME)

Thank you

Any questions?

Next module: ***Network 1st Hop Security***, 11th of August 2020

www.geant.org

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 731122 (GN4-2).

References

- Kim, Gene H.; Spafford, Eugene H. (1994). "The Design and Implementation of Tripwire: A File System Integrity Checker"
  - https://dl.acm.org/doi/10.1145/191177.191183
- Lawrence Grim: “IDS: File Integrity Checking”
  - https://www.sans.org/reading-room/whitepapers/detection/ids-file-integrity-checking-35327
- OSSEC Host-Based Intrusion Detection Guide
  - Rory Bray, Daniel Cid, Andrew Hay, Syngress, 2008, ISBN: 978-1597492409
- Host integrity monitoring using OSIRIS and Samhain
  - Brian Wotring, Syngress, 2005, ISBN-13: 978-1597490184

Some Open Source FIM software

- Tripwire: the grandparent of many FIM software (1992)
  - https://github.com/Tripwire/tripwire-open-source
- Aide: Advanced Intrusion Detection Environment
  - https://aide.github.io/
- Afick: Another File Integrity CheckKer
  - http://afick.sourceforge.net/
- Samhain: Linux FIM with additional monitoring of kernel data structures
  - https://www.la-samhna.de/samhain/
- OSSEC, Wazuh: Full open source HIDS with FIM (syscheck)
  - https://www.ossec.net/
  - https://wazuh.com/

Wazuh Live Demonstration

- Wazuh Server Appliance
  - https://documentation.wazuh.com/3.10/installation-guide/virtual-machine.html
  - https://packages.wazuh.com/vm/wazuh3.10.2_7.3.2.ova
- Kali Linux
  - https://www.kali.org/downloads/
- Windows 10 from Microsoft Evaluation Center
  - https://www.microsoft.com/en-us/microsoft-365/windows

Backup material

Stuff that didn’t make it due to time constraints

www.geant.org

The research leading to these results has received funding from the European Union’s Horizon 2020 research and innovation programme under Grant Agreement No. 731122 (GN4-2).

Decoding Wazuh file modes

```
/* File types.  */
#define __S_IFDIR       0040000 /* Directory.  */
#define __S_IFCHR       0020000 /* Character device.  */
#define __S_IFBLK       0060000 /* Block device.  */
#define __S_IFREG       0100000 /* Regular file.  */
#define __S_IFIFO       0010000 /* FIFO.  */
#define __S_IFLNK       0120000 /* Symbolic link.  */
#define __S_IFSOCK      0140000 /* Socket.  */

/* Protection bits.  */
#define __S_ISUID       04000   /* Set user ID on execution.  */
#define __S_ISGID       02000   /* Set group ID on execution.  */
#define __S_ISVTX       01000   /* Save swapped text after use (sticky).  */

#define __S_IREAD       0400    /* Read by owner.  */
#define __S_IWRITE      0200    /* Write by owner.  */
#define __S_IEXEC       0100    /* Execute by owner.  */
```

In-House Tools

- What if no FIM software on the system?
  - By default, there is none, or it's not active
- Some tools come with the operating system
  - Linux: Package database (rpm, dpkg)
    - Already has checksums, permissions, sizes, and more
  - Windows: sfc, sigverif, sigcheck
    - Checks the Authenticode signatures on executables and DLLs
- None of them will replace an FIM
  - Meant for system administration, not security
  - But better than nothing in emergencies (see shortcomings)

Linux In-House Tools: rpm & dpkg

- **rpm:** package manager for Redhat-based systems
  - CentOS, Fedora, openSUSE, …
- **dpkg:** package manager for Debian-based systems
  - Ubuntu, Kali, …
- Verify option: -V
  - Checks against information in the local database of installed package
  - Example: size and modification time have changed

```
> rpm -V openssh
S.?....T.  c /etc/ssh/sshd_config
>
```

rpm & dpkg -V: Output Format

- Output format for differences from package database information

```
.........   ← Test passed
S           ← file Size differs
 M          ← Mode differs (includes permissions and file type)
  5         ← digest (formerly MD5 sum) differs
   D        ← Device major/minor number mismatch
    L       ← readLink(2) path mismatch
     U      ← User ownership differs
      G     ← Group ownership differs
       T    ← mTime differs
        P   ← caPabilities differ
?           ← Information not in the database
```

Linux In-House Tools: Shortcomings

- Does not cover
  - Other package formats (for example self-extracting software)
  - Manually installed files (.tar.gz)
  - Files copied to different locations (chroot jails)
  - Files added by the attacker
- Local system only
- No automation/reporting
- Dpkg implements only the checksum part
- On a live system, lots of deviations from install
  - No way to flag changes as good and include them in the database
- Database is not secured against attackers with root privileges

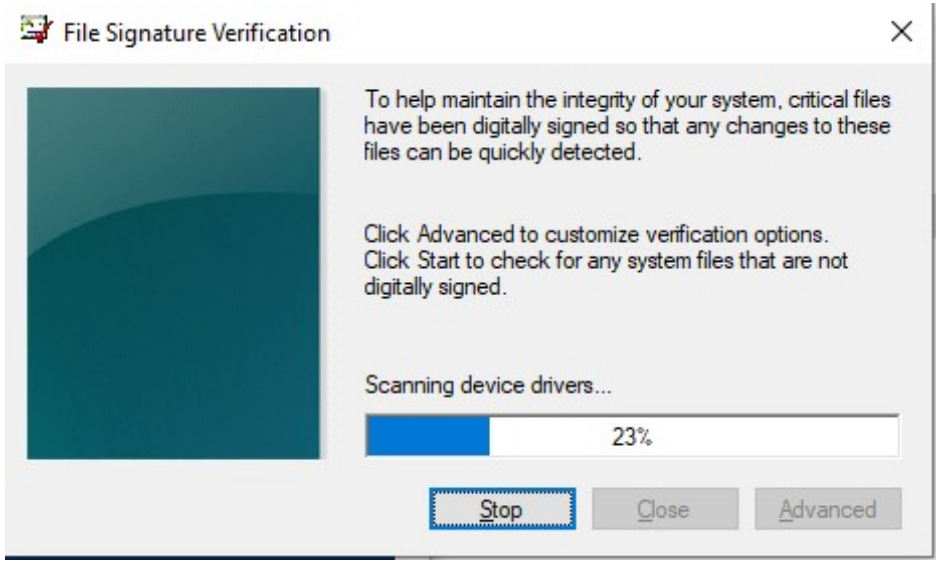
Windows In-House Tools: System File Checker

- Check if protected system files have been altered
  - Just verify: `sfc.exe /verifyonly`
  - Verify and restore: `sfc.exe /scannow`
  - Backups in `%windir%\system32\dllcache`
  - Or installation source
  - Not enabled by default
  - Log: `%windir%\Logs\CBS.log`
- Often bypassed by attackers
- Shortcomings
  - What is protected is not configurable by users/admins
  - Local system only

```
PS C:\Windows\system32> sfc /verifyonly

Beginning system scan.  This process will take
some time.

Beginning verification phase of system scan.
Verification 100% complete.

Windows Resource Protection did not find any
integrity violations.
PS C:\Windows\system32>
```

Windows In-House Tools: Sigverif

- Tool to verify signatures of device drivers in Windows
  - Device drivers (i.e. kernel modules) must be cryptographically signed to be loaded by the kernel
  - Reports to local log (default: `C:\Users\Public\Public Documents\Sigverif.txt`)
- Shortcomings
  - Checks only fixed list of drivers
  - List not configurable
  - No config file checks
  - No registry key checks



Windows In-House Tools: Sysinternals Sigcheck

- Verifies signatures like sigverif
- More fine-grained controls
  - CLI tool (scripting)
  - CSV output
  - Can show unsigned files only: `sigcheck -u`
  - Can check with virustotal
- Shortcomings
  - Still nothing for configuration/registry checks