Forensics for System Administrators

Collection of Other Evidence

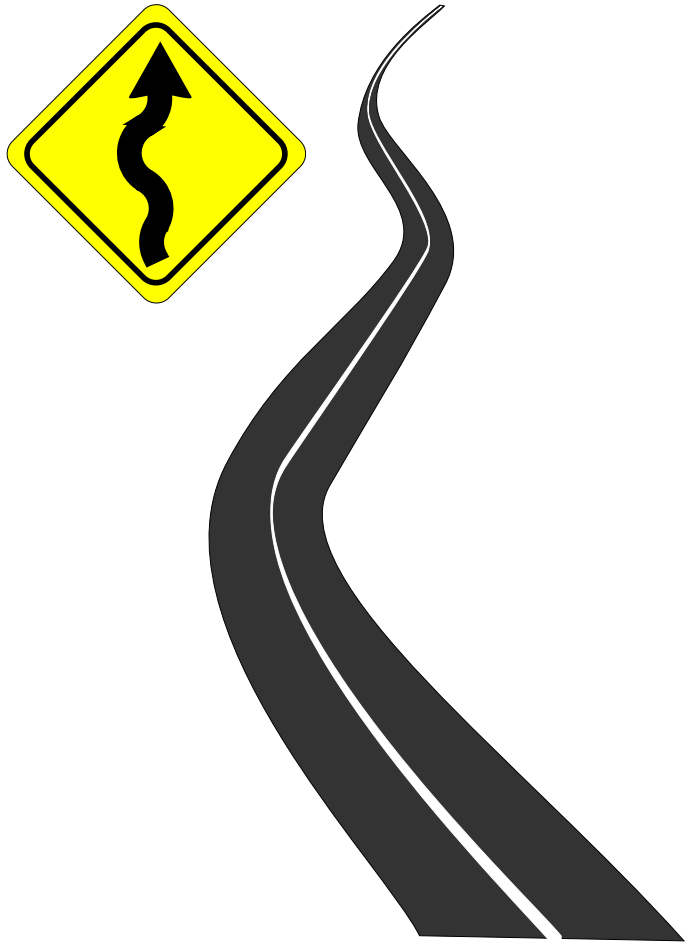
Klaus Möller
WP8-T1

Webinar, 27th of January 2022

Public

www.geant.org

Agenda



- Log(file) collection
  - Syslog
  - Systemd Journald
  - Audit Logs
  - Windows Eventlog
- Network traffic collection
  - Full packet dumps
  - NetFlows
- Other technical data
  - Hardware configuration
- Non-technical data
  - Users
  - Usage patterns
  - Role of the system

Log(file) Collection

Logfiles and where to find them

www.geant.org

Other Evidence?

- Log messages from the compromised system forwarded to a loghost
- Log messages from other systems
  - Firewalls, routers, switches, NAT gateways, VPN gateways, WLAN APs, …
  - Servers: DNS, DHCP, LDAP, CIFS, NFS, backup, KDcs, VM hosts, etc.
- Network traffic data
  - Packet captures
  - Flow data: NetFlow, etc.
- And more
  - Hardware configuration of affected systems
  - Non technical data
    - Users
    - Role of the compromised system (besides what is obvious)

Logging

- Log: Sequence of entries generated by the OS or application telling about its activities
  - What is being logged is determined by the programmers
  - Admins can only filter these down to a subset
  - Usefulness varies, but in general, log messages are indispensable
- Logs are typically stored on the system that generates them
  - When making a hard disk image, the logfiles are collected too
  - Ensure that the image(s) contains all system logfiles
    - Logs are sometimes kept on separate partitions/drives
- Logs may also be forwarded to central loghosts, SIEMs, etc.
  - Want to get a copy/subset of them without making disk images of the whole loghost

The Problem with Logs

- Once a system is compromised, the adversary can
  - Delete entries or complete logfiles
  - Change log entries or generate false entries, even for past events
  - Flood the log system with bogus entries
- Consequence: **Log entries by themselves are not trustworthy**
  - This holds true for every OS and every type of logging system
  - Need to compare with other, **independent** sources
- Exceptions
  - Log entries being forwarded to another system, i.e. loghost
    - Entries from before the compromise **may** be trustworthy
  - Minor incidents in which the adversary did not gain enough privileges to manipulate the log system

Syslog

- Primary source of logging information on Unix since the 80s
  - Log messages are unstructured ASCII text, max. 1024 characters long
  - Usually maintained through syslogd
  - First standard (RFC 3164) from 2001, current: RFC 5424 from 2009
- Logging before syslogd starts: Linux kernel log buffer
  - Accessible through `/dev/kmsg` or `dmesg` command
  - Size controllable through `log_buf_len` boot parameter
- Syslog on the network
  - UDP based (port 514/udp) - unreliable, messages may be lost
  - TCP (RFC 3195) introduces (some) reliability
  - TLS/DTLS (RFC 5425, 6012) add confidentiality on the wire

Syslog Forensics

- Collecting logfiles is easy
  - Just copy the files needed for the investigation
  - Usually in `/var/log` or `/var/adm/log` (see `/etc/syslog.conf`)
  - From the loghost or other system (firewall, domain controller, etc.)
- **Lots** of data on the central loghost - need to limit
  - Time, affected hosts, applications, etc.
  - Align with the scope of the forensic investigation
  - Data/privacy protection - do we have to say more?
- Syslog file examination is tricky
  - Lots of subtle differences make parsing log messages error prone
  - Check carefully whether the log parsing tool really gets what its looking for

Syslog Anti-Forensics

- Logfiles are just clear text files
  - Adversaries can read clear text logs too
  - And clear text network traffic
  - So they (can) know what is being logged about their activities
- Adversaries can modify clear text logfiles
  - Like removing messages about their doings
  - Or changing lines
  - Or delete the file entirely
- Adversaries may introduce false log messages too
  - Without being root

Linux: Systemd-Journald

- Journald as a syslogd replacement/supplement
  - Optional component of the systemd suite
  - Can not be used without systemd
  - Can forward entries by itself or through syslogd to another machine
- Addressing some shortcomings of syslogd
  - Indexed logfiles - for faster searching
  - Structured log format - easier to parse
  - (Log) message text catalogs - for internationalized log messages
  - Signed messages - manipulations of the logfile can be noticed
  - Per-user logfiles - users do not need access to the full system log to read their own log messages

Journald Forensics

Live Demo

- Collecting journald log(files)
  - Copy the journal directory/files
    - `/run/journal/`***`UUID`*** (in memory) (`Storage=volatile`)
    - `/var/log/journal/`***`UUID`*** (on disk) (Storage=persistent)
      - With `Storage=auto`, log will only be written if the directory exists
  - And the catalog database/source files
    - `/var/lib/systemd/catalog/database`
    - `/usr/lib/systemd/catalog/*.catalog`
- Examining journals on another machine
  - `journalctl --root=`***`PATH`*** (or `--file`)
  - Need to replicate directory structure under `/run, /var, /usr`
  - Do not forget to include the catalogs

Verifying Journald File Integrity

- Journald uses Forward Secure Sealing (FSS)
  - `Seal=yes` in `journald.conf`, and keys have been generated
  - Sealing key kept in the file system
    - `/var/log/journal/`***`UUID`***`/fss`
    - Not needed for verfication
  - **Verifcation key must not be kept on the system!**
- Sealing (i.e. signing) is done on time ranges of the log
  - Window of opportunity for adversaries
  - Can be changed with `journalctl --interval` (default: 15 minutes)
- Verifying journald logs
  - `journalctl --verify --verfication-key=`***`KEY`***
  - **Unsealed logs will also pass the verification!**

Audit Logs

- Logs of the audit system
  - Written by the audit daemon auditd
  - Default location: `/var/log/audit` (see `/etc/audit/auditd.conf`)
  - Independent from syslog or systemd-journald
- Collection
  - Like any other file
- Examination
  - Easier to parse than syslog
  - Different `auditd` logs (Linux, *BSD, Solaris, etc.) may not be compatible
- Anti-Forensics
  - Like any other file
  - More secure, if (and only if) mandatory access control (SeLinux) is active

Windows Eventlog

- Logging framework for Windows NT family
- Until Windows XP/Server 2003
  - Binary format, `.evt` extension
- Windows Event Log , since Windows Vista/Server 2008
  - XML-based format, `.evtx` extension
  - Events collected by Windows Event Collector Service and written to disk
- Message catalogs as Dynamic Link Library (DLL) files needed for message texts
- Windows Event Forwarding (EWF) possible to an Windows Event Collector (WEC) server
  - Push or pull (collector initiated) forwarding

EVT Forensics

- EVT Collection
  - Default location: `%SYSTEMROOT%\System32\config\*.evt`
  - See `HKLM\SYSTEM\CurrentControlSet\Services\Eventlog\`
  - Files are locked by SYSTEM account, direct access (i.e. copy) is possible only when offline
- Do not forget the message catalogs
  - DLL locations kept in registry: `HKLM\SYSTEM\CurrentControlSet\Services\Eventlog\`

EVTX Forensics

- Collection
  - Default location: `%SYSTEMROOT%\System32\winevt\Logs\*.evtx`
  - See `HKLM\SOFTWARE\Policies\Microsoft\Windows\Eventlog\`***`LOG`***`\File`
  - Files can be copied with admin privileges
  - Or exported (with filtering) through `wevtutil epl` ***`Logfile Exportfile`***
- Do not forget the message catalogs
  - DLL Locations in `HKLM\SYSTEM\CurrentControlSet\Services\EventLog\<channel>\<provider>\EventMessageFile`
    - Inside the DLL, look for the `MessageTable` resource

Apple Unified Logging

- Standard log format on macOS since 2016
  - iOS 10. , macOS 10.12, tvOS 10.0, and watchOS 3.0
  - `/private/var/db/diagnostics/*.tracev3`
  - `/private/var/db/uuidtext`
- Recommended archiving
  - Copy contents of both directories into a directory whose name has the .logarchive extension

Application Logs

- Examples
  - Apache logfiles, e. g. `access_log`, `error_log` or Tomcat logfiles, e. g. `catalina.log`
- Log4j - standard logging API orginally developed for Java
  - Has been ported to many other languages
  - Mostly ports of Log4j version 1, not the current version 2
- Language - Library
  - Java - Log4j 2, `https://logging.apache.org/log4j/2.x/`
  - Perl - Log4perl, `http://search.cpan.org/dist/Log-Log4perl/`
  - Python - Logging module, `https://docs.python.org/3/library/logging.html`
  - Ruby - Logger class, `https://github.com/lenny/log4jruby`, `https://ruby-doc.org/stdlib-2.1.2/libdoc/logger/rdoc/Logger.html`
  - C - log4c, `http://log4c.sourceforge.net/`

GÉANT
Networks • Services • People

Network Traffic Collection

www.geant.org

Network Evidence – Upsides

Observing network traffic has tremendous advantages:

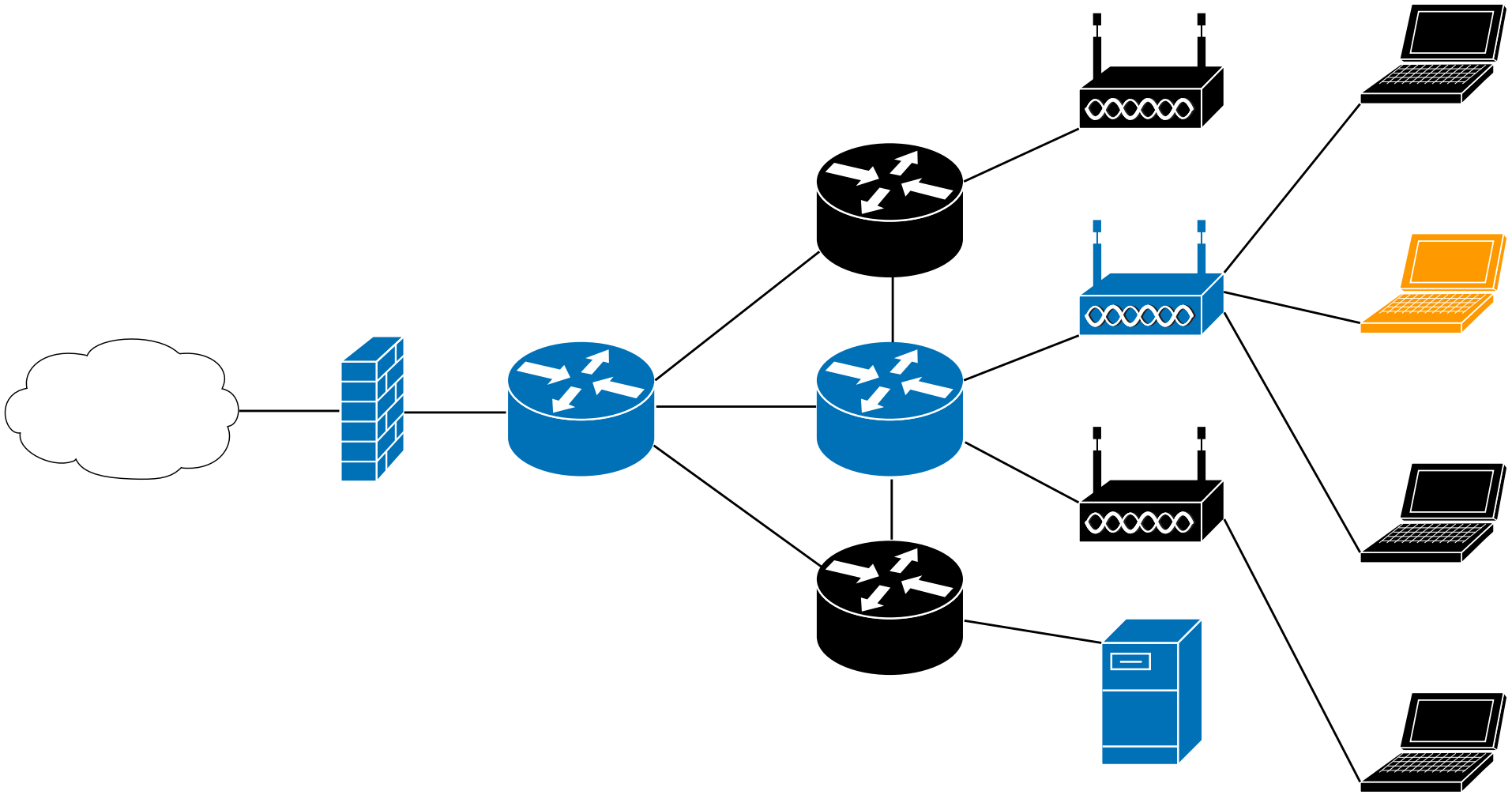
- Usually, the network itself cannot be manipulated by attackers
- No way to make traffic disappear – only camouflage
- Even if content is encrypted, communication relations are apparent

Network Evidence – Downsides

However, every coin has two sides:

- These days, there is a **lot** of network traffic (depending on where exactly you look, you might see many, many, many bits fly past you)
- Looking into network traffic without good reason is likely to be at least controversial, if not outright illegal where you live

Where to Look



What to Grab

Two general approaches:

- Packet capturing:
  - Full network traffic (“bits copied off the wire”)
  - Mostly stored in **pcap** or **pcapng** files
- Flow capturing:
  - Only communication metadata, no payloads
  - Mostly captured in NetFlow or sFlow format
  - Storage format is not standardized

Capturing Packets

- A record of the complete communication
- Limit the capture:
  - **Privacy vs. wiretapping – be very careful, thin ice!**
  - Capture files can get large rather quickly
  - More traffic in the capture → not “interesting” traffic
- What is really necessary for the investigation?
  - Filter *at the source* accordingly!
  - Explore other means to reach the goal:
    - Is there a less intrusive alternative?
    - In legal terms, is there an option less risky?

Capturing Packets at the Endpoint

Where:

- At the “interesting” device or
- at the “other” side (e. g., DNS servers)

How:

- Wireshark/**tshark:** `https://wireshark.org/`
- **tcpdump:** `https://tcpdump.org/`
- **ntopng:** `https://ntop.org/`
- Specialized software, e. g., DNS loggers

Capturing Packets Along the Way

Where:

- At hubs (**cough**), switches, routers with mirror ports
- at dedicated inline capture points

How:

- Any of the methods above
- Sniffing software:
  - Zeek (formerly Bro): `https://zeek.org/`
  - Snort: `https://snort.org/`
  - Suricata: `https://suricata.io/`

What *Exactly* Are Flows?

- Flow: Sequence of IP packets observed in a time slot sharing a number of properties
  - Source address/port
  - Destination address/port
  - Protocol (TCP, UDP, ICMP, etc.)
  - Ingress/egress interface, AS number, etc. (varies with protocol version)
- Each flow carries additional information
  - Number of packets & bytes
  - Timestamp when the flow started/ended/expired
- Flows do not contain packet payloads
  - Storage and (somewhat) privacy friendly

Flow Formats/Standards

- NetFlow
  - Defined by Cisco
  - Relevant in practice: Versions 5 and 9 (RFC 3954)
- Internet Protocol Flow Information Export (IPFIX)
  - Based on NetFlow, but incompatible
  - Standardized by IRTF (RFCs 7011 and 7012)
- Sampled Flow (sFlow)
  - Defined by sFlow.org (RFC 3176)
  - Does **not** produce flow data, just *samples* of flows → of lesser value in forensics
- Numerous other standards by other vendors
  - jFlow, NetStream, Cflowd, Rflowd, AppFlow, Traffic Flow, ...
  - Remote Network Monitoring (RMON2) MIB (RFC 4502)

How to Capture Flows

- Collection over the network
  - Export flow data directly to the investigator's collector
  - … or via a samplicator system if necessary
    - `https://github.com/sleinen/samplicator`
  - What information is transmitted varies by version
- Collection from storage
  - On-disk storage format is not standardized
  - Files can be copied, databases, etc. might have to be exported
    - Do not forget to include the schema when exporting
  - Can the investigators tools read the format?
    - Can be difficult with closed formats/non-open source tools
    - Alternatively, re-export the flows to a dedicated (investigators) collector

NetFlow Tools

Some useful tools to collect and analyze flows:

- **nfsen/nfdump**: https://nfsen.sourceforge.net/, https://github.com/phaag/nfdump
- **cflowd**: https://www.caida.org/catalog/software/cflowd
- SiLK: https://tools.netsa.cert.org/silk
- vFlow: https://github.com/EdgeCast/vflow
- IPFIXcol2: https://github.com/CESNET/ipfixcol2

Related tools:

- ARGUS: https://openargus.org/
- YAF: https://tools.netsa.cert.org/yaf

Other Networks to Consider

Some other sources to capture network traffic from:

- DECT (ETSI EN 300175, 300444, 102527)
- Mobile phone: GSM, GPRS, EDGE, LTE, 5G, …
- Wireless Personal Area Networks (WPAN)
  - Bluetooth (IEEE 802.15-1) - wireless peripherals
  - Zigbee (IEEE 802.15-4) - IoT
- NFC, RFID (ISO 18000-3)
- Infrared (IrDa)
- Loopback interface

Collection of Other Data

www.geant.org

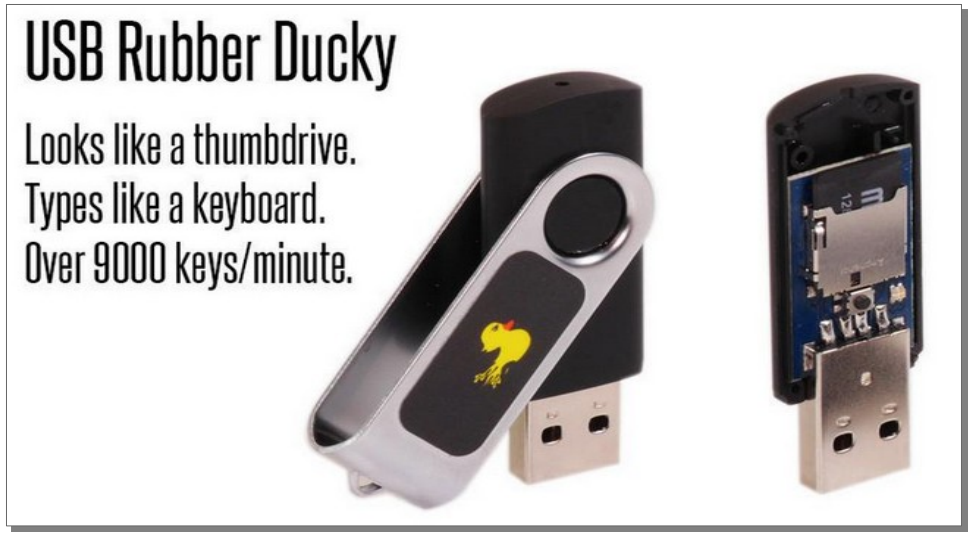
Other Data

- Data that does not reside on on of the imaged systems and is not necessarily available in databases, logs, or similar
  - May have to interview people to get to this data
  - Limit questions strictly to the investigations goal - do not be too nosy
- If assigned names or addresses change, collect “historical” data
  - Public DNS databases
  - WHOIS database
  - Internet routing databases
  - Locally: ARP/CAM table history (possibly in the logs), DHCP mappings, NAT IP-address mappings (don’t forget port numbers), etc.
  - Yes, that’s ***data preservation*** - with all implications:(
- Point here: Collect the target state as defined by polices, etc. and correlate with what the investigation will uncover on the systems

Technical Data

- Location of the compromised system(s) (building, room, rack, etc.)
  - “Surrounding data”: Building keys, phone number of someone to let the investigator in, etc.
- State of the hardware, i.e. are there traces of break-ins or manipulation?
  - Scratches on the casing or screws
    - Easy to spot on new ones, almost impossible to tell on heavily used ones
  - Cabling: Where to the connected cables lead to?
  - Other cables that end near the system, unconnected, but were they in the past?
  - Attached peripherals: diskettes, SD-cards, USB-sticks, other USB-devices (rubber ducky, key-logger, …)
  - Possibly other hardware that has been build in, e. g. different/additional hard drivers, additional cards, devices attached to internal buses
- **Take pictures before and along the investigation!**

Images not shown to scale



USB Keylogger

Non-technical Data: Users

- User-ID ↔ Username mappings
  - From LDAP or Active Directory (NIS for very old Unix installations)
  - Other directory services
- Login patterns (of the users)i.e.
  - Typical work hours/shifts
  - During the investigated time frame, where were they?
    - Home office, holiday, sick leave, traveling (timezone, location)
- Home-users: IP-address pool of their home ISP
  - To rule out legitimate logins from there

Non-technical Data: Usage Patterns

- Type of programs used
  - Non-developer usually do not use compilers, adversaries compiling their toolkit do
- Documents used
  - Accesses outside the users home directory, e. g. other home directories
- There are patterns of network usage also
  - Like websites needed to visit
  - Remote logins
- Peak times
  - Number of logged-in users
  - Amount of data transferred (mirrors, backups, etc.)

Non-technical Data: Role of the System

- I.e. database/web/DNS/etc. -server,
  - What kind of services should be running on the system and what should not
  - Details of that configuration, e. g.
    - Installed databases, schemas
    - Web base directory, web-server modules, content management system, etc.
    - Authoritative/caching/forwarding name server, zones serviced, …
- Network configuration
  - IP-Addresses, including IPv6, and other protocols
  - Type of configuration (static, DHCP, SLAAC, …)
  - Interfaces, including tunnel, VPN, virtual, etc. ones
  - Local routing table (default gateway)
  - Local resolver, including /etc/hosts, /etc/nsswitch.conf
  - Network-mounted drives

Wrapping Up

www.geant.org

What have you learned?

- There is more evidence than memory and disk images
  - Collect log information from central log hosts
  - Packet captures and IP flow information
  - And more
  - Do not forget that name or IP-address mappings change over time - may need to collect historical data also
- We need your input & feedback → please fill out the survey
- Coming soon: Part 2 of this series: Evidence analysis
  - Autopsy, Volatility, and more
  - Tools you would like to know about (or recommend)?
    - Open source only, please

Thank you

Any questions?

Next Webinar: ***TBD***

TBD Nth, 2022

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

- C. Lonvick: RFC 3164 - The BSD Syslog Protocol, 2001, https://datatracker.ietf.org/doc/rfc3164/
- R. Gerhards: RFC 5424 - The Syslog Protocol, 2009, https://datatracker.ietf.org/doc/rfc5424/
- Journald logfile format https://systemd.io/JOURNAL_FILE_FORMAT/
- G. A. Marson and B. Poettering: *Practical Secure Logging: Seekable Sequential Key Generators,* https://eprint.iacr.org/2013/397.pdf