Welcome to

9. Network Forensics

Communication and Network Security 2019

Henrik Lund Kramshøj hlk@zencurity.com

Slides are available as PDF, kramse@Github

9-Network-Forensics.tex in the repo security-courses

Plan for today

Subjects

- Centralized syslog
- Collect Network Evidence
- Netflow data
- Analyze Network data
- Network Forensics
- Create Incident Reports

Exercises

- Run forensics similar to ENISA examples
- Create a Kibana dashboard for looking at logs

Reading Summary

ANSM chapter 4,5,6 - 75 pages

4. Session Data
5. Full Packet Capture
6. Packet String Data

IP reputation

Zeek documentation Intel framework `https://docs.zeek.org/en/stable/frameworks/intel.html`

Suricata reputation support
`https://suricata.readthedocs.io/en/suricata-4.0.5/reputation/index.html`

Centralized syslog

Logfiler er en nødvendighed for at have et transaktionsspor

Logfiler giver mulighed for statistik

Logfiler er desuden nødvendige for at fejlfinde

Det kan være relevant at sammenholde logfiler fra:

- routere
- firewalls
- webservere
- intrusion detection systemer
- adgangskontrolsystemer
- ...

Husk - tiden er vigtig! Network Time Protocol (NTP) anbefales

Husk at logfilerne typisk kan slettes af en angriber - hvis denne får kontrol med systemet

syslog

syslog er system loggen på UNIX og den er effektiv

- man kan definere hvad man vil se og hvor man vil have det dirigeret hen
- man kan samle det i en fil eller opdele alt efter programmer og andre kriterier
- man kan ligeledes bruge named pipes - dvs filer i filsystemet som tunneller fra chroot'ed services til syslog i det centrale system!
- man kan nemt sende data til andre systemer

Man bør lave en centraliseret løsning

syslogd.conf eksempel

```
*.err;kern.debug;auth.notice;authpriv.none;mail.crit    /dev/console
*.notice;auth,authpriv,cron,ftp,kern,lpr,mail,user.none /var/log/messages
kern.debug;user.info;syslog.info                        /var/log/messages
auth.info                                               /var/log/authlog
authpriv.debug                                          /var/log/secure
...
# Uncomment to log to a central host named "loghost".
#*.notice;auth,authpriv,cron,ftp,kern,lpr,mail,user.none        @loghost
#kern.debug,user.info,syslog.info                               @loghost
#auth.info,authpriv.debug,daemon.info                           @loghost
```

Andre syslogs syslog-ng

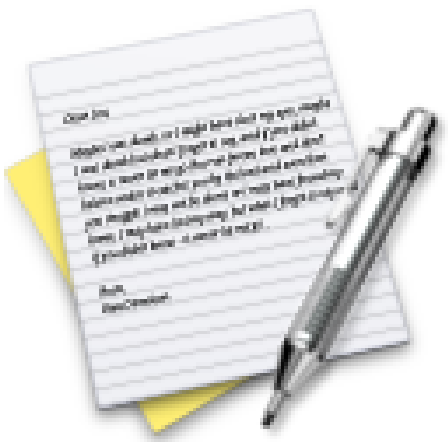
- der findes andre syslog systemer eksempelvis syslog-ng
- konfigureres gennem `/etc/syslog-ng/syslog-ng.conf`

```
options {
        long_hostnames(off);
        sync(0);
        stats(43200);
};

source src  unix-stream("/dev/log"); internal(); pipe("/proc/kmsg"); ;
destination messages  file("/var/log/messages"); ;
destination console_all  file("/dev/console"); ;
log  source(src); destination(messages); ;
log  source(src); destination(console_all); ;
```

Kan eksempelvis TCP og garanteret aflevering af beskeder

Exercise



Now lets do the exercise

Logning med syslogd og syslog.conf

which is number **38** in the exercise PDF.

Logfiler og computer forensics

Logfiler er en nødvendighed for at have et transaktionsspor

Logfiler er desuden nødvendige for at fejlfinde

Det kan være relevant at sammenholde logfiler fra:

- routere
- firewalls
- intrusion detection systemer
- adgangskontrolsystemer
- ...

Husk - tiden er vigtig! Network Time Protocol (NTP) anbefales

Husk at logfilerne typisk kan slettes af en angriber - hvis denne får kontrol med systemet

Web server access log

```
root# tail -f access_log
::1 - - [19/Feb/2004:09:05:33 +0100] "GET /images/IPv6ready.png
HTTP/1.1" 304 0
::1 - - [19/Feb/2004:09:05:33 +0100] "GET /images/valid-html401.png
HTTP/1.1" 304 0
::1 - - [19/Feb/2004:09:05:33 +0100] "GET /images/snowflake1.png
HTTP/1.1" 304 0
::1 - - [19/Feb/2004:09:05:33 +0100] "GET /~hlk/security6.net/images/logo-1.png
HTTP/1.1" 304 0
2001:1448:81:beef:20a:95ff:fef5:34df - - [19/Feb/2004:09:57:35 +0100]
"GET / HTTP/1.1" 200 1456
2001:1448:81:beef:20a:95ff:fef5:34df - - [19/Feb/2004:09:57:35 +0100]
"GET /apache_pb.gif HTTP/1.1" 200 2326
2001:1448:81:beef:20a:95ff:fef5:34df - - [19/Feb/2004:09:57:36 +0100]
"GET /favicon.ico HTTP/1.1" 404 209
2001:1448:81:beef:20a:95ff:fef5:34df - - [19/Feb/2004:09:57:36 +0100]
"GET /favicon.ico HTTP/1.1" 404 209
```

Web server logs are pretty standardized, common log format.

Logstash pipeline

```
input { stdin { } }
output {
  elasticsearch { host => localhost }
  stdout { codec => rubydebug }
}
```

- Logstash receives via **input**
- Processes with **filters** - grok
- Forward events with **output**

Logstash as SNMPtrap and syslog server

```
input {
  snmptrap {
    host => "0.0.0.0"
    type => "snmptrap"
    port => 1062
    community => "xxxxx"
  }
  tcp {
    port => 5000
    type => syslog
  }
  udp {
    port => 5000
    type => syslog
  }
}
```

- We run logstash on port 5000 - but use IPtables port forwarding

Have you even configured SNMP traps?

Maybe you have a device sending SNMP traps right now ...

IPtables forwarding

```
*nat
:PREROUTING ACCEPT [0:0]
# redirect all incoming requests on port 514 to port 5000
-A PREROUTING -p tcp --dport 514 -j REDIRECT --to-port 5000
-A PREROUTING -p udp --dport 514 -j REDIRECT --to-port 5000
-A PREROUTING -p udp --dport 162 -j REDIRECT --to-port 1062
COMMIT
```

Inserted near beginning of /etc/ufw/before.rules on Ubuntu

Remember defense in depth, dont run a priveleged Java VM as root ☺

Grok expresssions

```
filter {
  if [type] == "syslog" {
    grok {
      match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp}
      %{SYSLOGHOST:syslog_hostname} %{DATA:syslog_program}
      (?:\[%{POSINT:syslog_pid}\])?: %{GREEDYDATA:syslog_message}" }
      add_field => [ "received_at", "%{@timestamp}" ]
      add_field => [ "received_from", "%{host}" ]
    }
    syslog_pri { }
    date {
      match => [ "syslog_timestamp", "MMM  d HH:mm:ss", "MMM dd HH:mm:ss" ]
    }
  }
}
```

- Logstash filter expressions grok can normalize and split data into fields

Source: Config snippet from recommended link

`http://logstash.net/docs/1.4.1/tutorials/getting-started-with-logstash`

Grok expresssions, sample from my archive

```
filter {
# decode some SSHD
if [syslog_program] == "sshd" {
  grok {
# May 20 10:27:08 odn1-nsm-01 sshd[4554]: Accepted publickey for hlk from
10.50.11.17 port 50365 ssh2: DSA 9e:fd:3b:3d:fc:11:0e:b9:bd:22:71:a9:36:d8:06:c7

match => { "message" => "%{SYSLOGTIMESTAMP:timestamp} %{HOSTNAME:host_target}
sshd\[%{BASE10NUM}\]: Accepted publickey for %{USERNAME:username} from
  %{IP:src_ip} port %{BASE10NUM:port} ssh2" }

# "May 20 10:27:08 odn1-nsm-01 sshd[4554]: pam_unix(sshd:session):
session opened for user hlk by (uid=0)"
match => { "message" => "%{SYSLOGTIMESTAMP:timestamp} %{HOSTNAME:host_target}
sshd\[%{BASE10NUM}\]: pam_unix\(sshd:session\): session opened for user
%{USERNAME:username}" }
```

- Logstash filter expressions grok can normalize and split data into fields

Netflow data

Collect Network Evidence

How to get started

How to get started searching for security events?

Collect basic data from your devices and networks

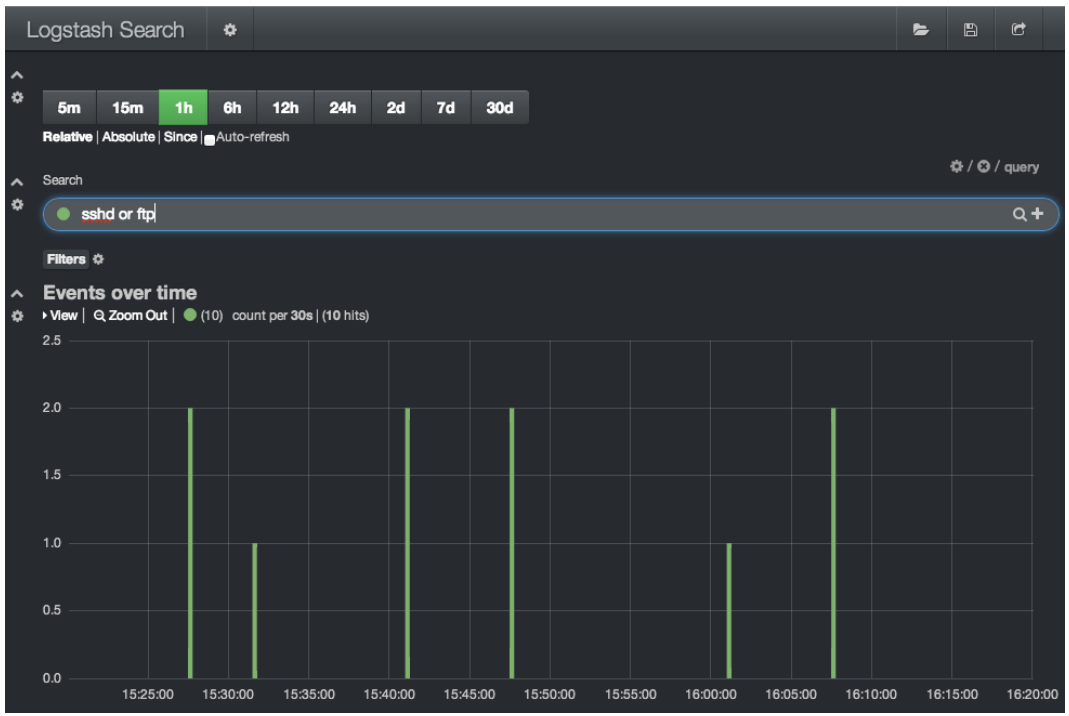
- Netflow data from routers
- Session data from firewalls
- Logging from applications: email, web, proxy systems

Centralize!

Process data

- Top 10: interesting due to high frequency, occurs often, brute-force attacks
- *ignore*
- Bottom 10: least-frequent messages are interesting

View data efficiently

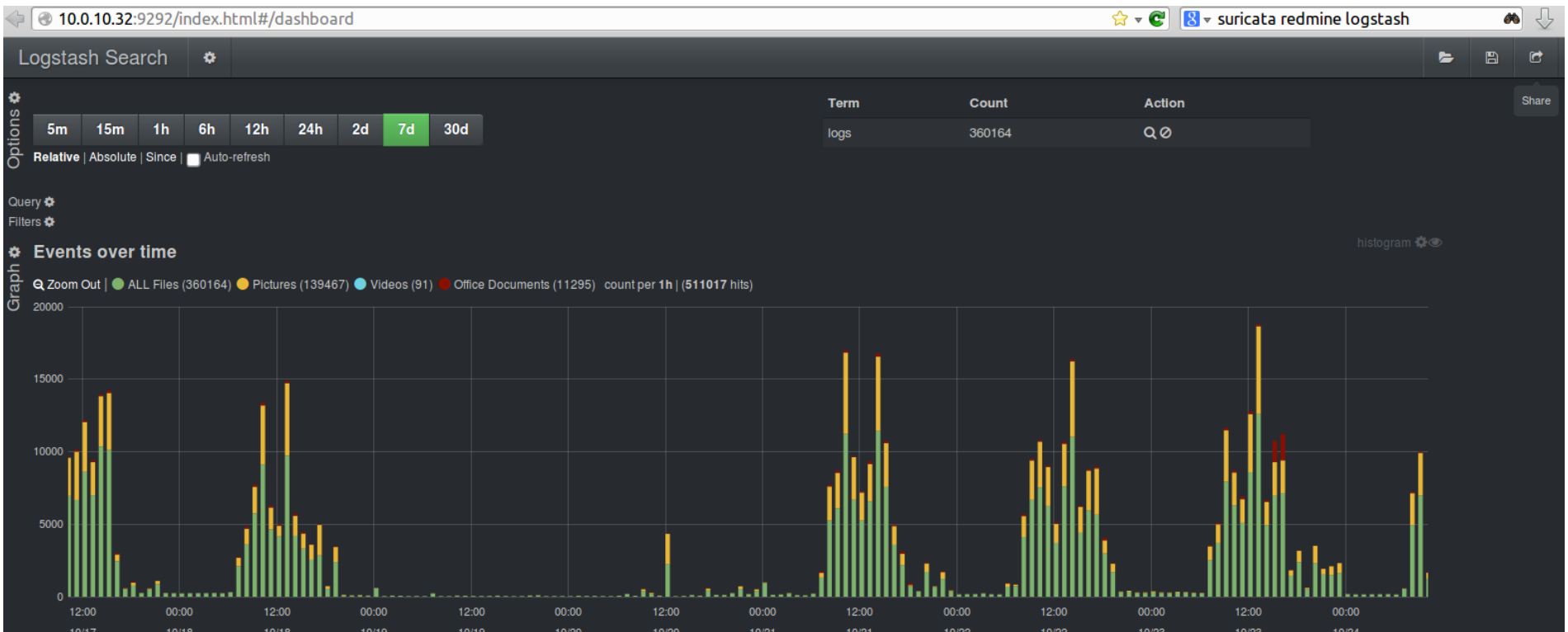


View data by digging into it easily - must be fast

Logstash and Kibana are just examples, but use indexing to make it fast!

Other popular examples include Graylog and Grafana

Graphs and Dashboards!



- Screenshot from Peter Manev, OISF
- Shown are Suricata IDS alerts processed by Logstash and Kibana

Suricata with Dashboards

Picture from Twitter
`https://twitter.com/nullthreat/status/445969209840128000`

`http://suricata-ids.org/`

Security Onion

- Security Onion is a Linux distro for IDS, NSM, and log management
- `http://securityonion.blogspot.dk`
- `http://blog.securityonion.net/p/securityonion.html`
- Not so great in production, focus on fewer tools, or buy BIG CPU ☺

Nice starting point for researching dashboards/network packets

Next steps

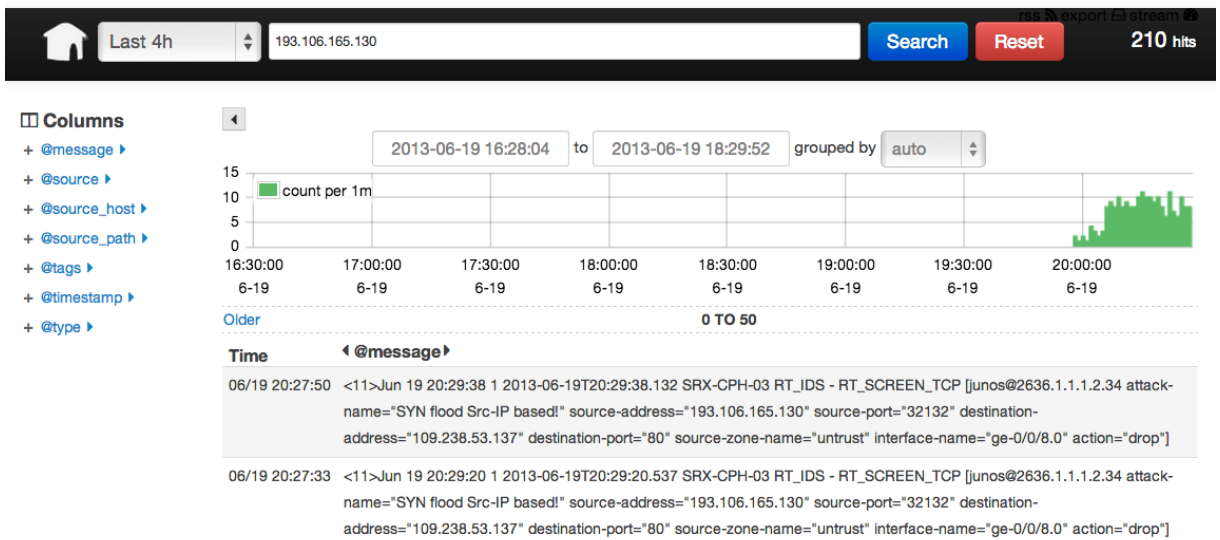
In our network we are always improving things:

Suricata IDS `http://www.openinfosecfoundation.org/`
More graphs, with **automatic identification** of IPs under attack
Identification of **short sessions without data** - spoofed addresses
Alerting from **existing** devices
Dashboards with key measurements

Conclusion: Combine tools!

Analyze Network data

Network tools - examples



Net: Bro `http://www.bro-ids.org` Suricata `http://suricata-ids.org`

DNS: DSC and PacketQ `https://github.com/dotse/packetq/wiki`

Syslog: Elasticsearch, Logstash, and Kibana, called ELK stack or Elastic stack

Packetbeat `https://www.elastic.co/products/beats/packetbeat`

Collect and present data more easily - non-programmers

Example tool, let see what BRO IDS is

The Bro Network Security Monitor

Bro is a powerful network analysis framework that is much different from the typical IDS you may know.

While focusing on network security monitoring, Bro provides a comprehensive platform for more general network traffic analysis as well. Well grounded in more than 15 years of research, Bro has successfully bridged the traditional gap between academia and operations since its inception.

`https://www.bro.org/`

BRO more than an IDS

> The key point that helped me understand was the explanation that Bro is a domain-specific language for networking applications and that Bro-IDS (http://bro-ids.org/) is an application written with Bro.

Why I think you should try Bro
`https://isc.sans.edu/diary.html?storyid=15259`

Bro scripts

```
global dns_A_reply_count=0;
global dns_AAAA_reply_count=0;
...
event dns_A_reply(c: connection, msg: dns_msg, ans: dns_answer, a: addr)
{
++dns_A_reply_count;
}

event dns_AAAA_reply(c: connection, msg: dns_msg, ans: dns_answer, a: addr)
{
++dns_AAAA_reply_count;
}
```

Source: dns-fire-count.bro from

`https://github.com/LiamRandall/bro-scripts/tree/master/fire-scripts`

Trust me, this IS better than having to write network parsers in C ☺

Bro demo (on a Mac)

```
kunoichi:~ root# brew install bro

kunoichi:~ root# broctl
Hint: Run the broctl "deploy" command to get started.

Welcome to BroControl 1.5

Type "help" for help.

[BroControl] > install
creating policy directories ...
installing site policies ...
generating standalone-layout.bro ...
generating local-networks.bro ...
generating broctl-config.bro ...
generating broctl-config.sh ...
```

Bro demo: Run bro

```
kunoichi:etc root# pwd
/usr/local/etc
kunoichi:etc root# grep eth0 node.cfg
interface=eth0
#interface=eth0
#interface=eth0
// My mac is not a Linux system, uses another interface naming scheme
kunoichi:etc root# vi node.cfg
kunoichi:etc root# grep en0 node.cfg
interface=en0
```

Bro demo: Run bro

```
// back to Broctl and start it
[BroControl] > start
starting bro
// and then
kunoichi:bro root# pwd
/usr/local/var/spool/bro
kunoichi:bro root# tail -f dns.log
```

More examples at:

https://www.bro.org/sphinx/script-reference/log-files.html

Example, Using tools similar to PacketQ

Using PacketQ

Let's have a practical look at how PacketQ works by trying to figure out what kind of DNS ANY queries are being sent towards our name-server.

DNS ANY traffic is currently commonly abused for DNS amplification attacks (See Blog post "DDoS-Angriffe durch Reflektierende DNS-Amplifikation vermeiden" in German). The first thing I want to know is what are the IP addresses of the victims of this potential DNS amplification attack:

```
packetq -t -s "select src_addr,count(*) as count from dns where qtype=255 group
by src_addr order by count desc limit 3" lolo.20130118.070000.000179
"src_addr" ,"count"
"216.245.221.243",933825
"85.126.233.70" ,16802
"80.74.130.55" ,91
```

Are you using existing tools? or build your own specialised tools from scratch?

http://securityblog.switch.ch/2013/01/22/using-packetq/

http://jpmens.net/2013/05/27/server-agnostic-logging-of-dns-queries-responses/

Storing query logs, old school or needed?

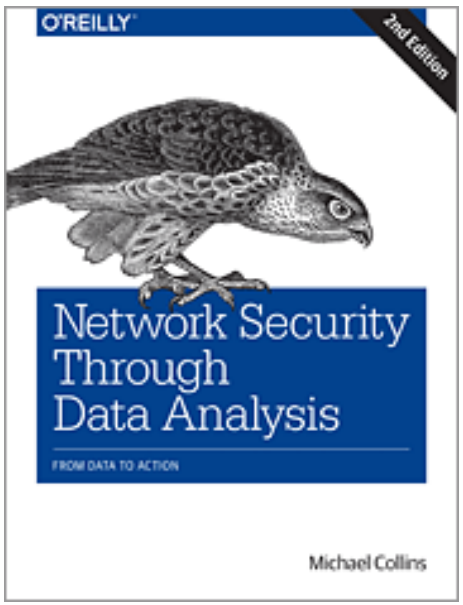
- policy/protocols/ssl/expiring-certs.bro
- policy/protocols/ssl/extract-certs-pem.bro
- policy/protocols/ssl/heartbleed.bro
- policy/protocols/ssl/known-certs.bro
- policy/protocols/ssl/log-hostcerts-only.bro
- policy/protocols/ssl/validate-certs.bro
- policy/protocols/ssl/validate-ocsp.bro
- policy/protocols/ssl/weak-keys.bro

Looking at DNS PacketQ it was an Older link, but thinking the time is now for doing:

- DNS query logs, keep it for at least a week? - with DSC and PacketQ
- SSL/TLS full logs over sessions, certs, keys - with Bro/Suricata
  `https://www.bro.org/sphinx-git/script-reference/scripts.html`
- Log and search with Elasticsearch?
  `https://www.elastic.co/guide/en/elasticsearch/guide/current/index.html`

- Even netflow session logging, full 1:1 - NFSen or Suricata Flow mode

Network Security Through Data Analysis



Low page count, but high value! Recommended.

Network Security through Data Analysis, 2nd Edition By Michael S Collins Publisher: O'Reilly Media 2015-05-01: Second release, 348 Pages

New Release Date: August 2017

Network Forensics

ENISA

The European Union Agency for Network and Information Security (ENISA) is a centre of expertise for cyber security in Europe.

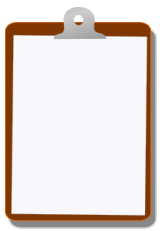
ENISA is contributing to a high level of network and information security (NIS) within the European Union, by developing and promoting a culture of NIS in society to assist in the proper functioning of the internal market.

`https://www.enisa.europa.eu/`

ENISA exercises

- We will use these as examples:
- ENISA Presenting, correlating and filtering various feeds Handbook, Document for teachers
  `https://www.enisa.europa.eu/topics/trainings-for-cybersecurity-specialists/online-training-m`
  `documents/presenting-correlating-and-filtering-various-feeds-handbook`
- ENISA Forensic analysis, Network Incident Response
  `https://www.enisa.europa.eu/topics/trainings-for-cybersecurity-specialists/online-training-m`
  `documents/2016-resources/exe2_forensic_analysis_ii-handbook`
- ENISA Network Forensics, Handbook, Document for teachers
  `https://www.enisa.europa.eu/topics/trainings-for-cybersecurity-specialists/online-training-m`
  `documents/network-forensics-handbook`

For Next Time



Think about the subjects from this time, write down questions

Check the plan for chapters to read in the books
Most days have about 100 pages or less, but one day has 4 chapters to read!

Visit web sites and download papers if needed

Retry the exercises to get more confident using the tools