IT-Security (ITS) B1

DIKU, E2023

Today’s agenda

Intrusion detection defined

Intrusion detection in theory

Intrusion detection in practice

Signature detection, anomaly detection, log analysis

Next time: Forensics

Intrusion Detection defined

Overall security goals

(Anticipate breaches and build to contain with defence in depth, segmentation, least privilege, etc.)

Prevent as much as possible with *best practices* such as secure coding, whitelisting, patching, secure configurations and more

Detect and **respond** when things go wrong

Learn and **repeat**

Intrusion Detection process / key activities

Intrusion Detection is the process of monitoring and analyzing system events, to identify and report such intrusions

Threat Assessment
How are we exposed (as a company, our business processes, and underlying IT)?

Visibility
What is the right level of insight we need in our systems and applications to detect intrusions?

Data Collection
How do we collect data to support our visibility needs?

Data Analysis
How do we analyse the data for signs of intrusions?

Incident Response
What do we do when we discover an attack?

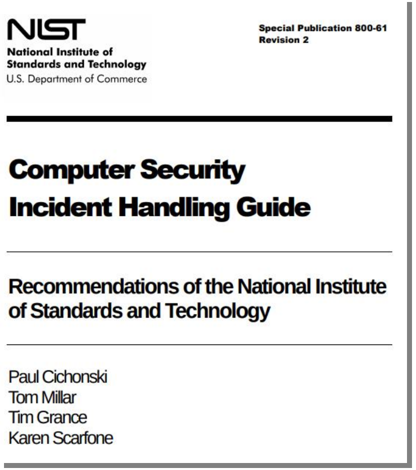
Intrusions defined

What is an **intrusion**? Or, when does it go from being an **event** to something more.

An intrusion or incident is an event on a host or network that violates security policy, or is an imminent threat to put a system in an unauthorized state.

Not all **intrusion attempts** are successful, not all **intrusions** lead to **compromise.** The criticality of an intrusion/incident depends, on the stage in which it was discovered (anything non-targeted before Initial Access is borderline relevant), on the systems affected, the accounts compromised, the type of adversary, their motivation, and more.

NIST Security Incident Handling process



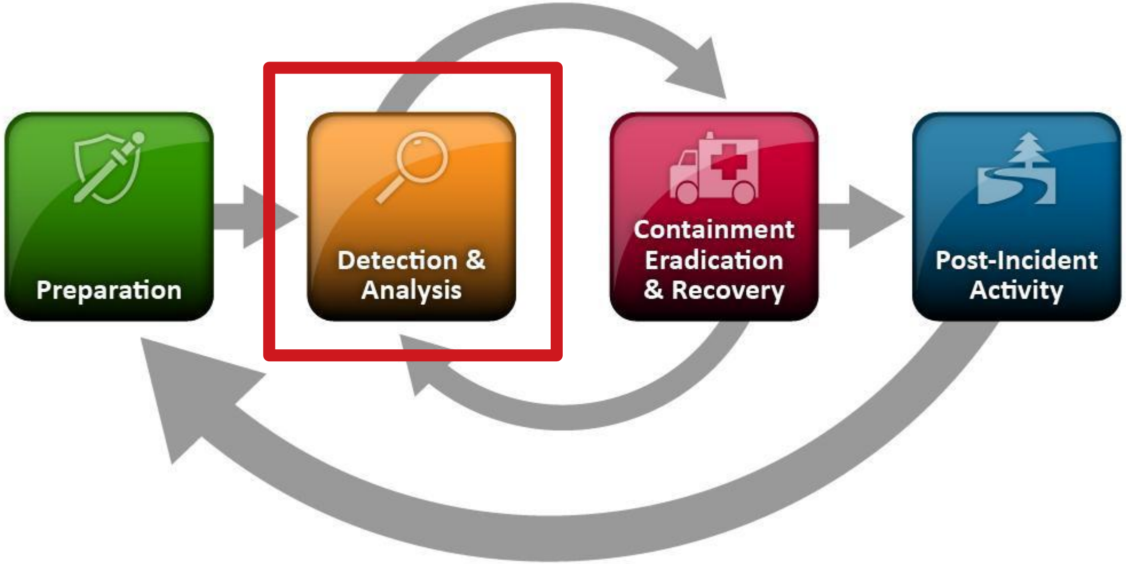
Special Publication 800-61
Revision 2

NIST
National Institute of
Standards and Technology
U.S. Department of Commerce

Computer Security
Incident Handling Guide

Recommendations of the National Institute
of Standards and Technology

Paul Cichonski
Tom Millar
Tim Grance
Karen Scarfone



Intrusions by the numbers

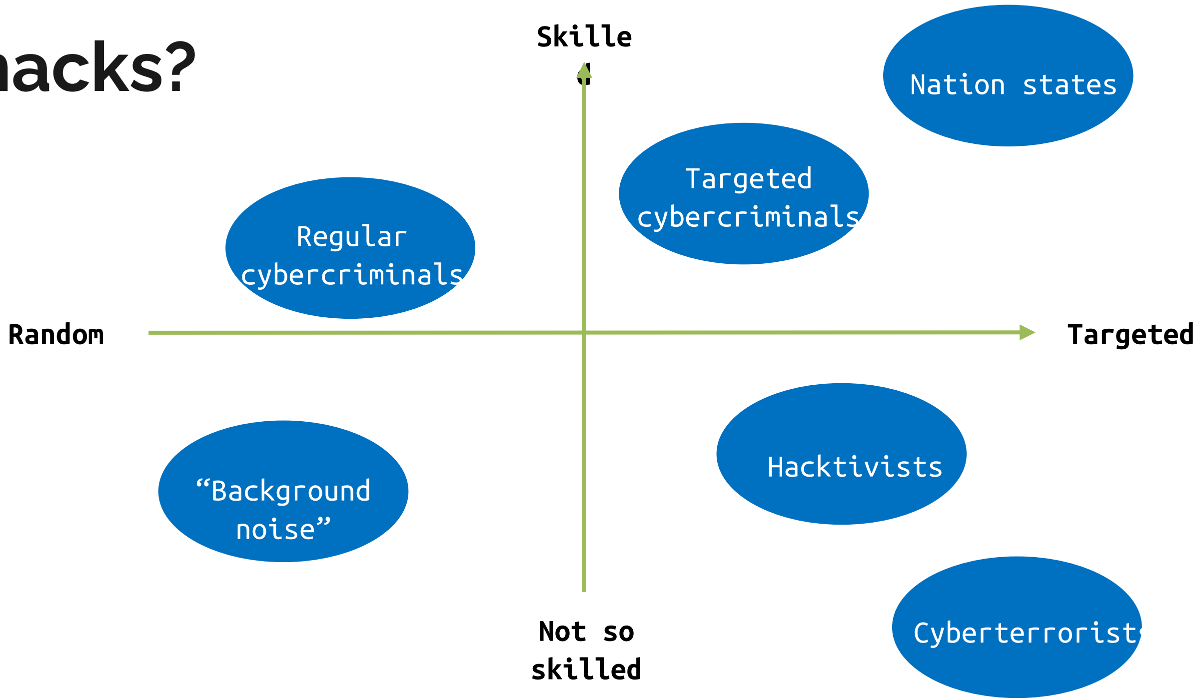
If or when?

“There are two kinds of companies.

There are those who've been hacked and those who don't know they've been hacked.”

Former FBI Director, James Comey

Who hacks?

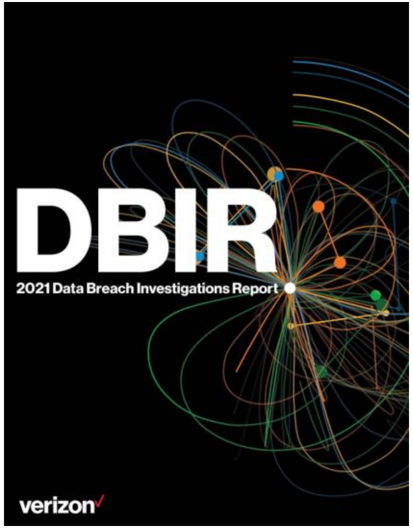

Skilled
Nation states
Targeted cybercriminals
Regular cybercriminals
Random
Targeted
Hacktivists
"Background noise"
Cyberterrorists
Not so skilled

Overall trends in Intrusion Detection

Global Median Dwell Time, 2011-2021

Compromise Notifications	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
All	416	243	229	205	146	99	101	78	56	24	21
External Notification	–	–	–	–	320	107	186	184	141	73	28
Internal Detection	–	–	–	–	56	80	57.5	50.5	30	12	18

Overall trends in Intrusion Detection



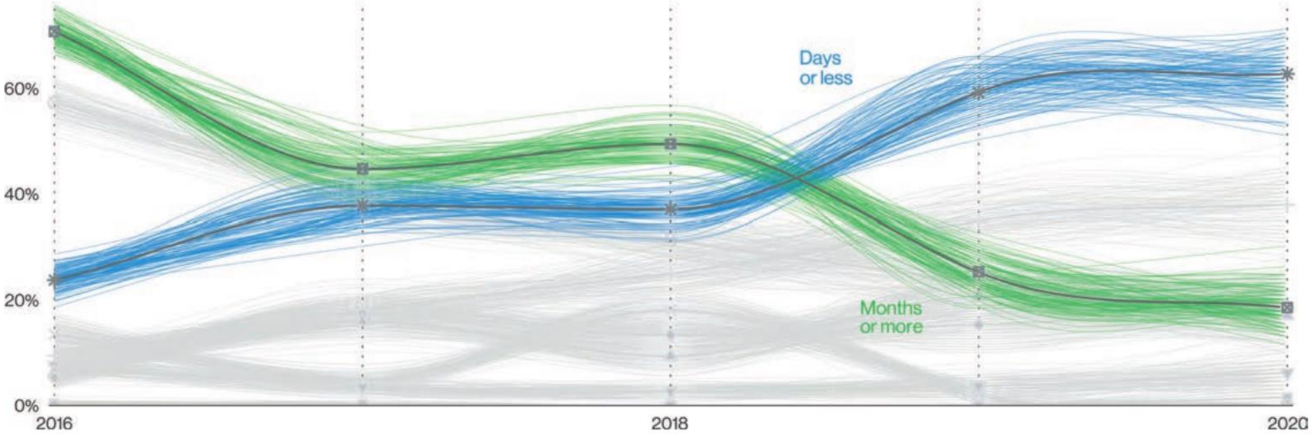
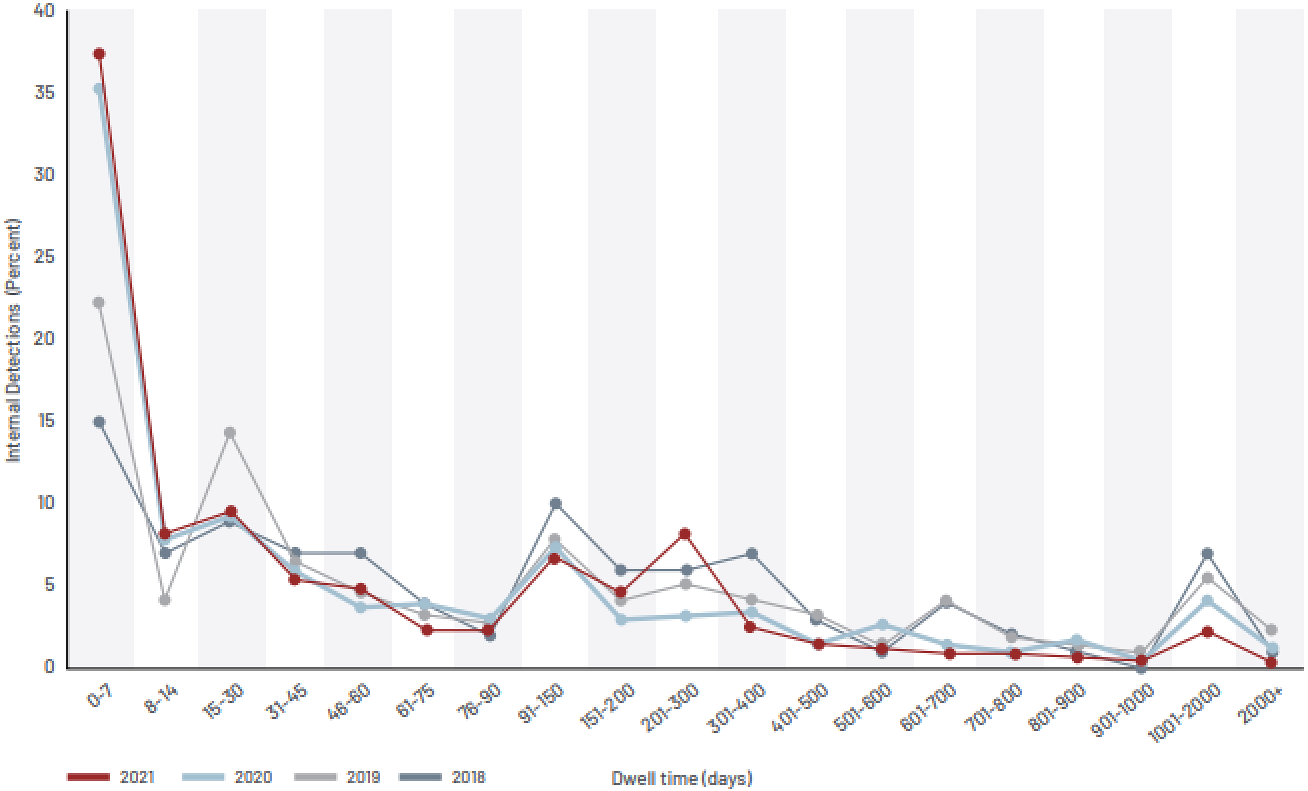


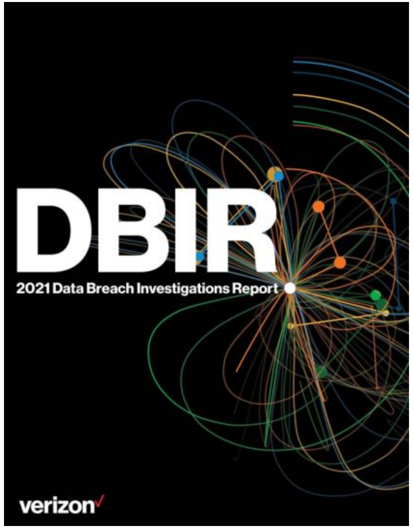
Figure 39. Discovery over time in breaches

Trends

Global Dwell Time Distribution, 2018–2021



Overall trends in Intrusion Detection



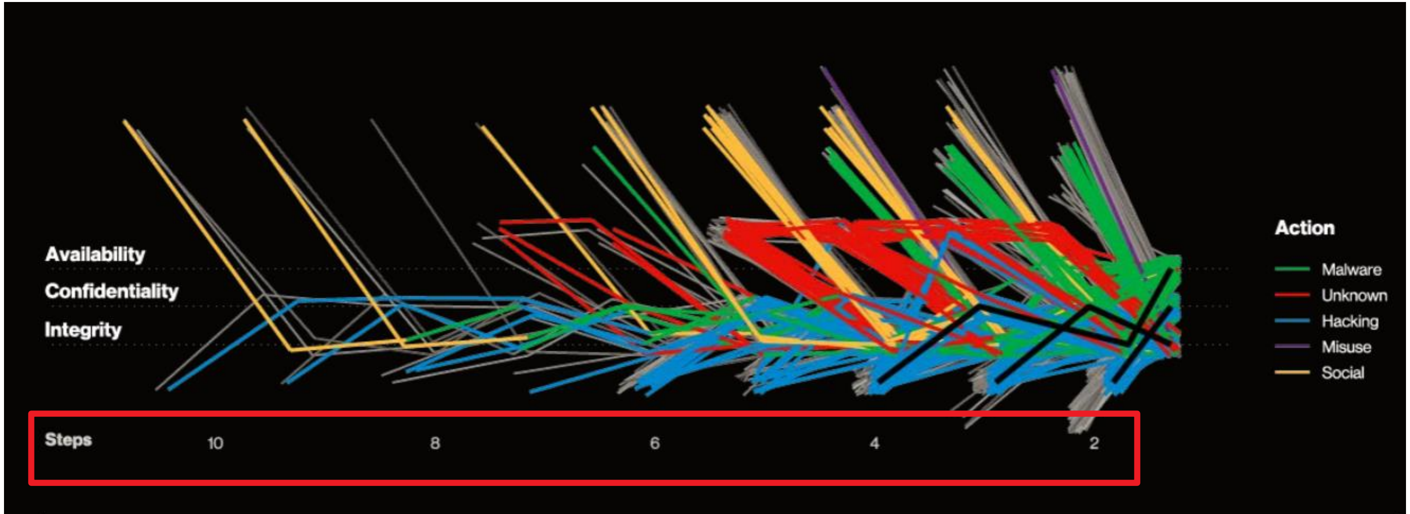


Figure 81. System Intrusion incident paths (n=251)

Intrusion Detection in Theory

True/false positive/negative

We have **events**, **sensors**, **HIDS** and **NIDS**: the **intrusion detection problem** is to determine whether an event is from a distribution of events of intruder behavior, or from a legitimate user distribution.

	intrusion	no intrusion
alarm raised	True Positive (TP) intrusion detected	False Positive (FP) false alarm
no alarm raised	False Negative (FN) intrusion missed	True Negative (TN) normal operation

False positive rate	$FPR = \frac{FP}{(FP+TN)}$
True negative rate	$TNR = 1 - FPR$
False negative rate	$FNR = 1 - TPR$
True positive rate	$TPR = \frac{TP}{(TP+FN)}$
Alarm precision	$AP = \frac{TP}{(TP+FP)}$

Figure 11.1: IDS event outcomes (left) and metrics (right). FP and FN (yellow) are the classification errors. TPR is also called the *detection rate*.

Intrusion detection: approaches

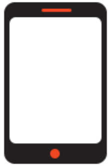
IDS approach	Alarm when...	Pros, cons, notes
signature-based (expert defines malicious patterns)	events match known-bad patterns	signatures built from known attacks; fast, accurate (fewer false positives); detects only already-known attacks
specification-based (expert defines allowed actions)	events deviate from per-application specifications of legitimate actions	manually developed spec of allowed; can detect new attacks; no alarm on newly seen allowed event; specs are protocol- or program-specific
anomaly-based (learning-based profile of normal)	events deviate from profiles of normal	need training period to build profiles; can detect new attacks; false alarms (abnormal may be benign); accuracy depends on features profiled

Table 11.1: IDS methodologies. Signature-based approaches use expert-built patterns (manual denylists). Specification approaches use expert-built specs (manual allowlists). Anomaly approaches define "normal" behavior from training data (empirical allowlists).

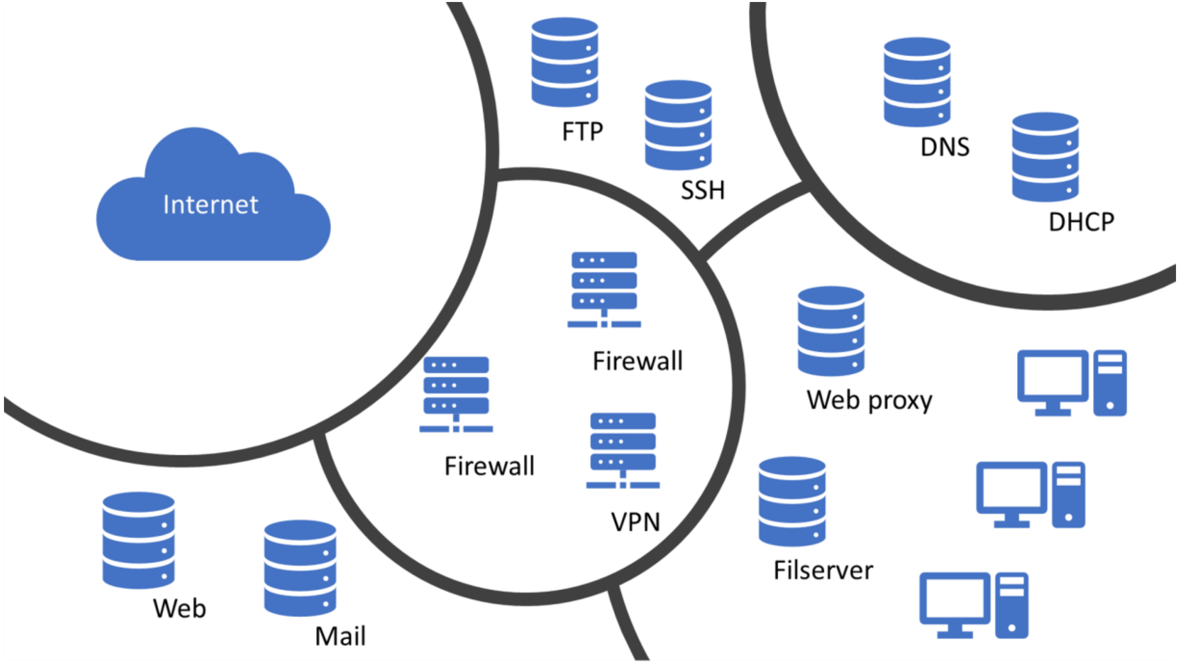
Intrusion Detection in Practice

Where should we focus?





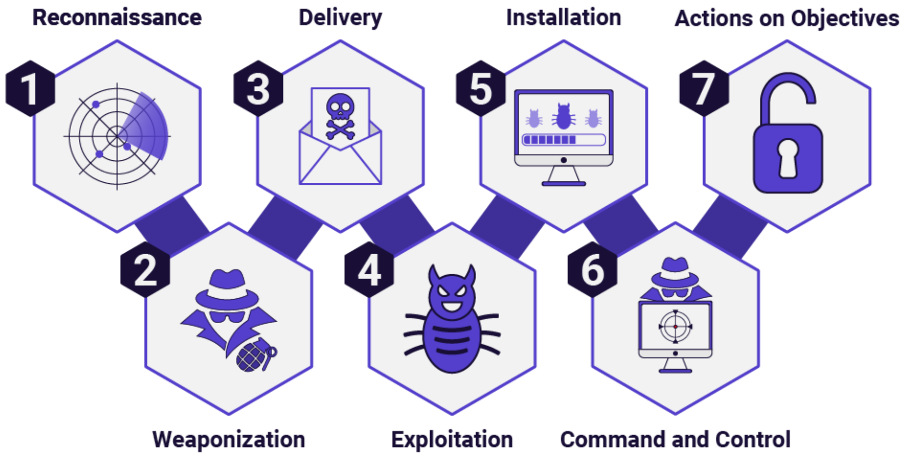

Possible data sources, include:



Visibility

Visibility

Q: What is the right level of insight we need in our systems and applications to detect intrusions?
A: Study how hackers actually hack: The **Cyber Kill Chain**:



MITRE ATT&CK

The Cyber Kill Chain is a good resource, but somewhat high-level. **MITRE ATT&CK** to the rescue:

ATT&CK Matrix for Enterprise

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Commonly Used Port	Automated Exfiltration	Data Destruction
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	Binary Padding	Bash History	Application Window Discovery	Application Deployment Software	Automated Collection	Communication Through Removable Media	Data Compressed	Data Encrypted for Impact
External Remote Services	Command-Line Interface	Account Manipulation	AppCert DLLs	BITS Jobs	Brute Force	Browser Bookmark Discovery	Distributed Component Object Model	Clipboard Data	Connection Proxy	Data Encrypted	Defacement
Hardware Additions	Compiled HTML File	AppCert DLLs	AppInit DLLs	Bypass User Account Control	Credential Dumping	Domain Trust Discovery	Exploitation of Remote Services	Data from Information Repositories	Custom Command and Control Protocol	Data Transfer Size Limits	Disk Content Wipe
Replication Through Removable Media	Control Panel Items	AppInit DLLs	Application Shimming	Clear Command History	Credentials in Files	File and Directory Discovery	Logon Scripts	Data from Local System	Custom Cryptographic Protocol	Exfiltration Over Alternative Protocol	Disk Structure Wipe
Spearphishing Attachment	Dynamic Data Exchange	Application Shimming	Bypass User Account Control	CMSTP	Credentials in Registry	Network Service Scanning	Pass the Hash	Data from Network Shared Drive	Data Encoding	Exfiltration Over Command and Control Channel	Endpoint Denial of Service
Spearphishing Link	Execution through API	Authentication Package	DLL Search Order Hijacking	Code Signing	Exploitation for Credential Access	Network Share Discovery	Pass the Ticket	Data from Removable Media	Data Obfuscation	Exfiltration Over Other Network Medium	Firmware Corruption
Spearphishing via Service	Execution through Module Load	BITS Jobs	Dylib Hijacking	Compile After Delivery	Forced Authentication	Network Sniffing	Remote Desktop Protocol	Data Staged	Domain Fronting	Exfiltration Over Physical Medium	Inhibit System Recovery
Supply Chain Compromise	Exploitation for Client Execution	Bootkit	Exploitation for Privilege Escalation	Compiled HTML File	Hooking	Password Policy Discovery	Remote File Copy	Email Collection	Domain Generation Algorithms	Scheduled Transfer	Network Denial of Service
Trusted Relationship	Graphical User Interface	Browser Extensions	Extra Window Memory Injection	Component Firmware	Input Capture	Peripheral Device Discovery	Remote Services	Input Capture	Fallback Channels		Resource Hijacking
[illegible]	[illegible]	[illegible]	File System Permissions	Component Object Model	[illegible]	[illegible]	Replication Through	[illegible]	[illegible]		Runtime Data

What are TTPs?

TTPs = Tactics, Techniques, and Procedures

Tactics
The “why” of an ATT&CK technique. It is the adversary’s tactical goal: the reason for performing an action. For example, an adversary may want to achieve credential access.

Techniques
The “how” an adversary achieves a tactical goal by performing an action. For example, an adversary may dump credentials to achieve credential access.

Procedures
The “specific” implementation the adversary uses for a technique. For example, a procedure could be an adversary using PowerShell to inject into lsass.exe to dump credentials by scraping LSASS memory on a victim machine.

Example: SDelete

SDelete is an application that securely deletes data in a way that makes it unrecoverable. It is part of the Microsoft Sysinternals suite of tools.

Tactic:	Impact	(ID: TA0040)
Technique:	Data Destruction	(ID: T1485)
Procedure:	Sdelete	(ID: S0195)

Threat actor groups observed in the wild using this procedure:

* G0053 FIN5
* G0080 Cobalt Group
* G0016 APT29
* G0091 Silence

https://attack.mitre.org/software/S0195/

On Linux, **shred** is comparable. (Forensics is the topic of next lecture.)

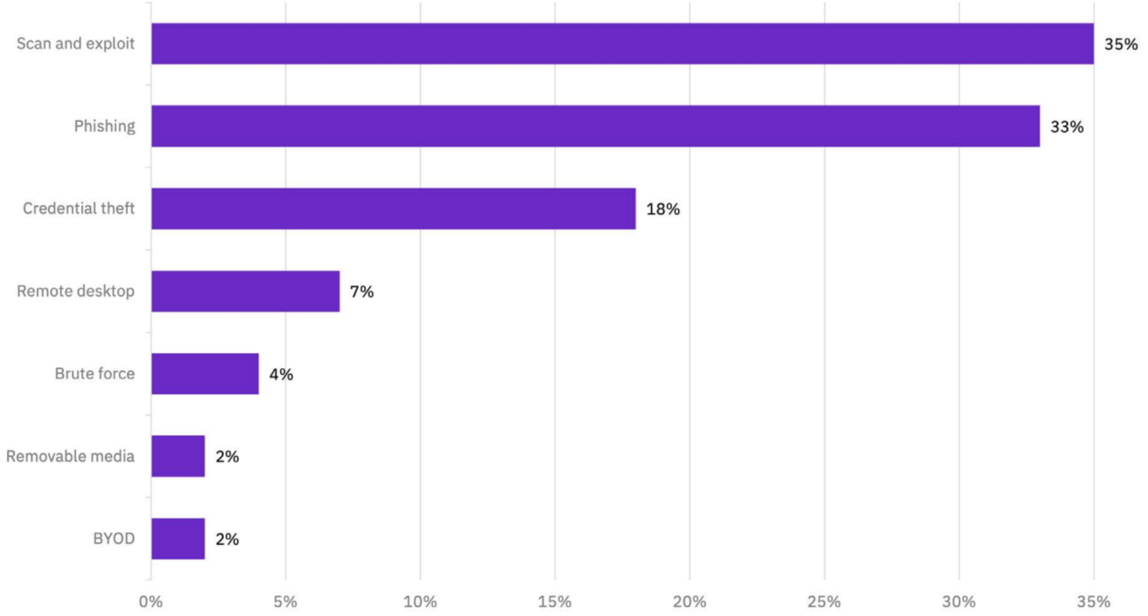
Which ATT&CK Techniques to focus on?

All?

Or, **some**? And if so, then **which**?

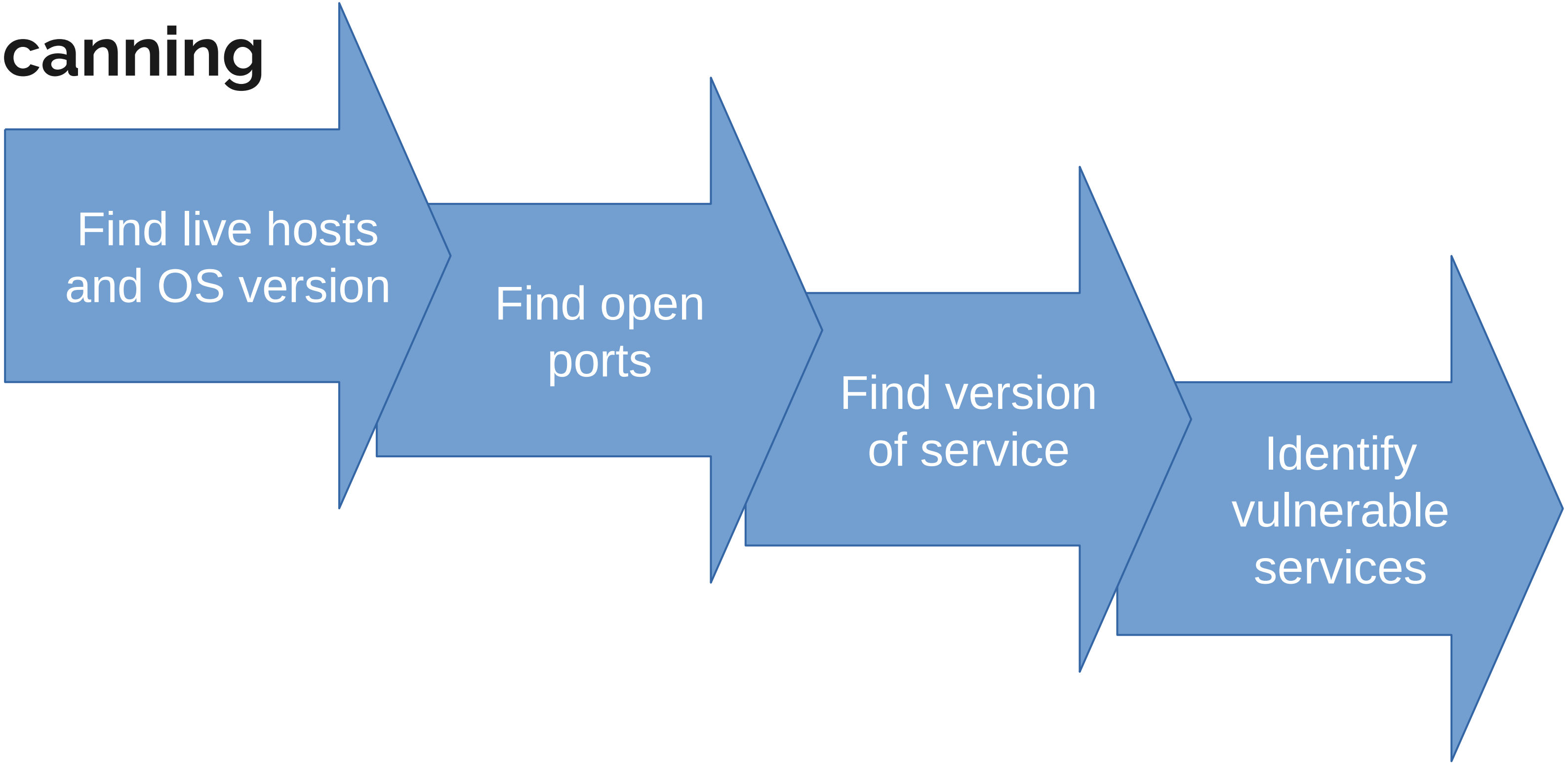
Look at the evidence, i.e. **tecniques observed in the wild** – either by ourself or reported in freely available information aka **(open source) threat intelligence**.

For example, in their 2021 X-Force Threat Intelligence Index, IBM notes their observations on the Initial Access tactic:

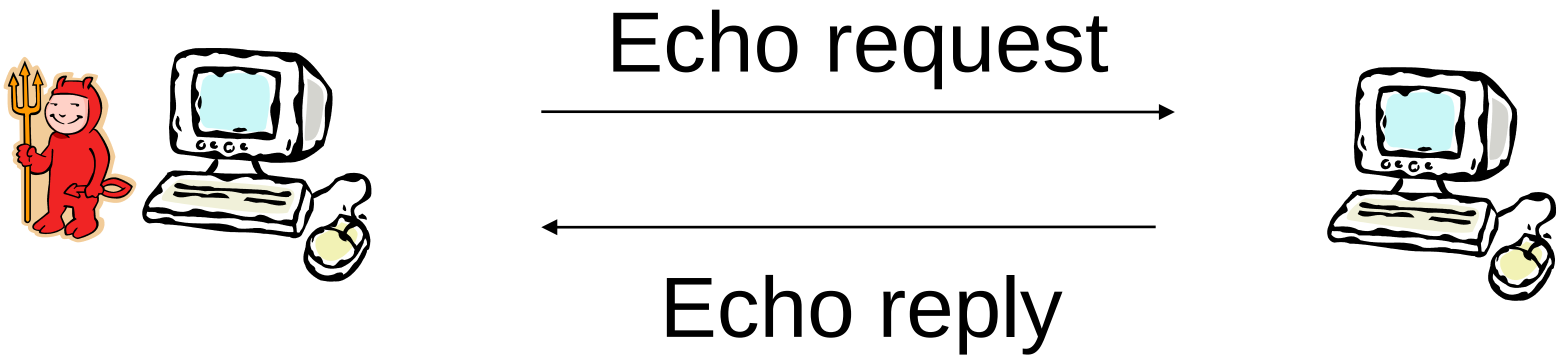


Scanning

Scanning


Find live hosts and OS version
Find open ports
Find version of service
Identify vulnerable services

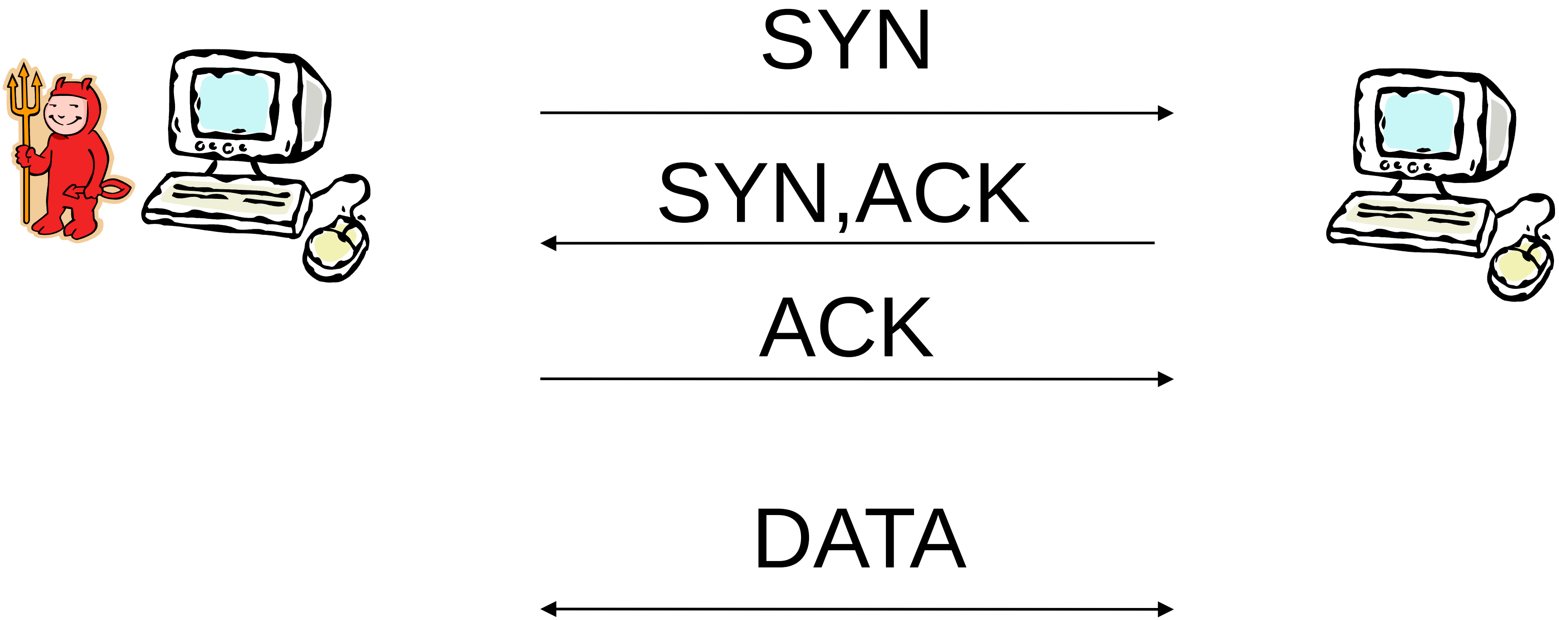
Scanning



Scapy / Ping sweep

```
$ sudo python
>>> from scapy.all import *
>>> conf.verb = 0
>>> for i in range(1, 256):
...     packet = IP(dst="192.168.184." + str(i), ttl=20)/ICMP()
...     reply = sr1(packet, timeout=1)
...     if not (reply is None):
...          print reply.src
...     else:
...          print "timeout " + str(i)
...
192.168.184.140
192.168.184.148
```

Port scanning

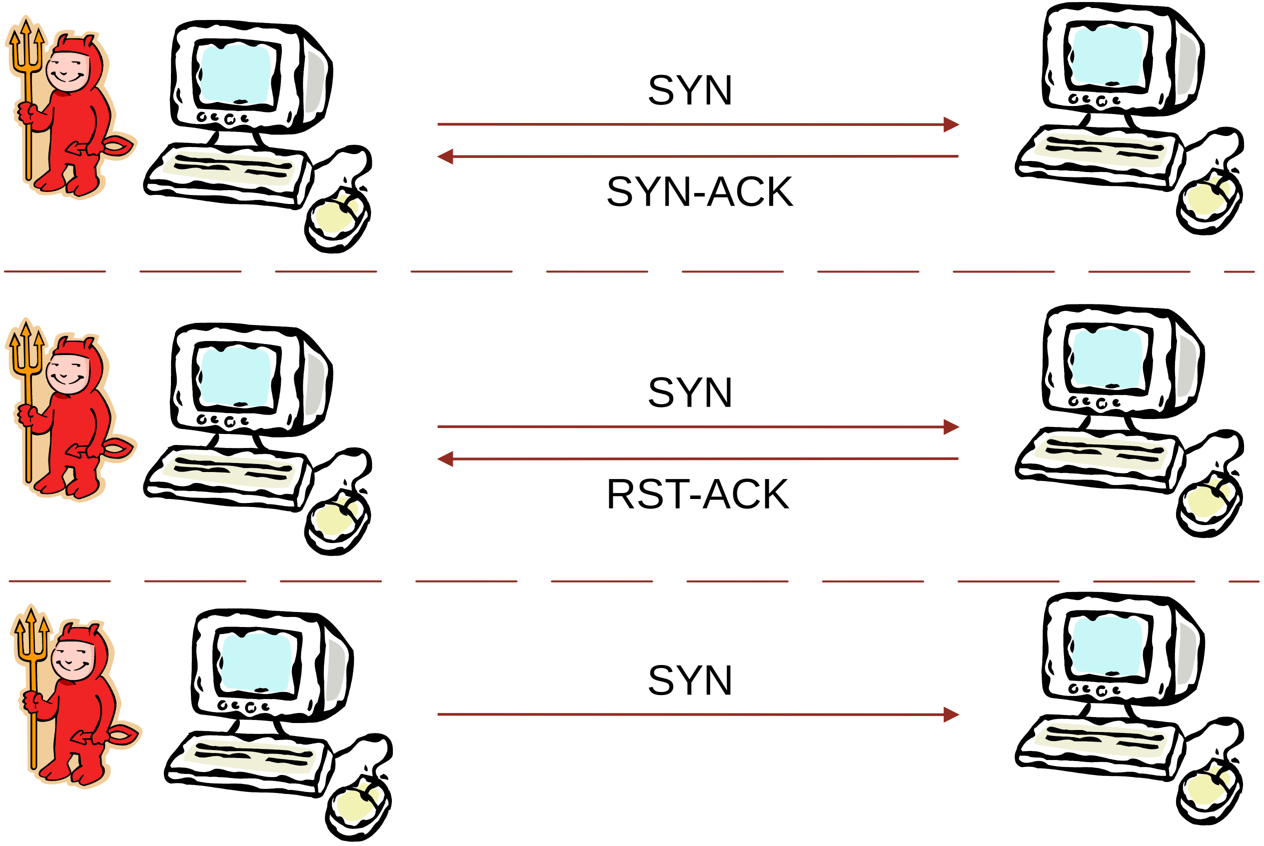


TCP

Port open!

Port closed!

Blocked by firewall?

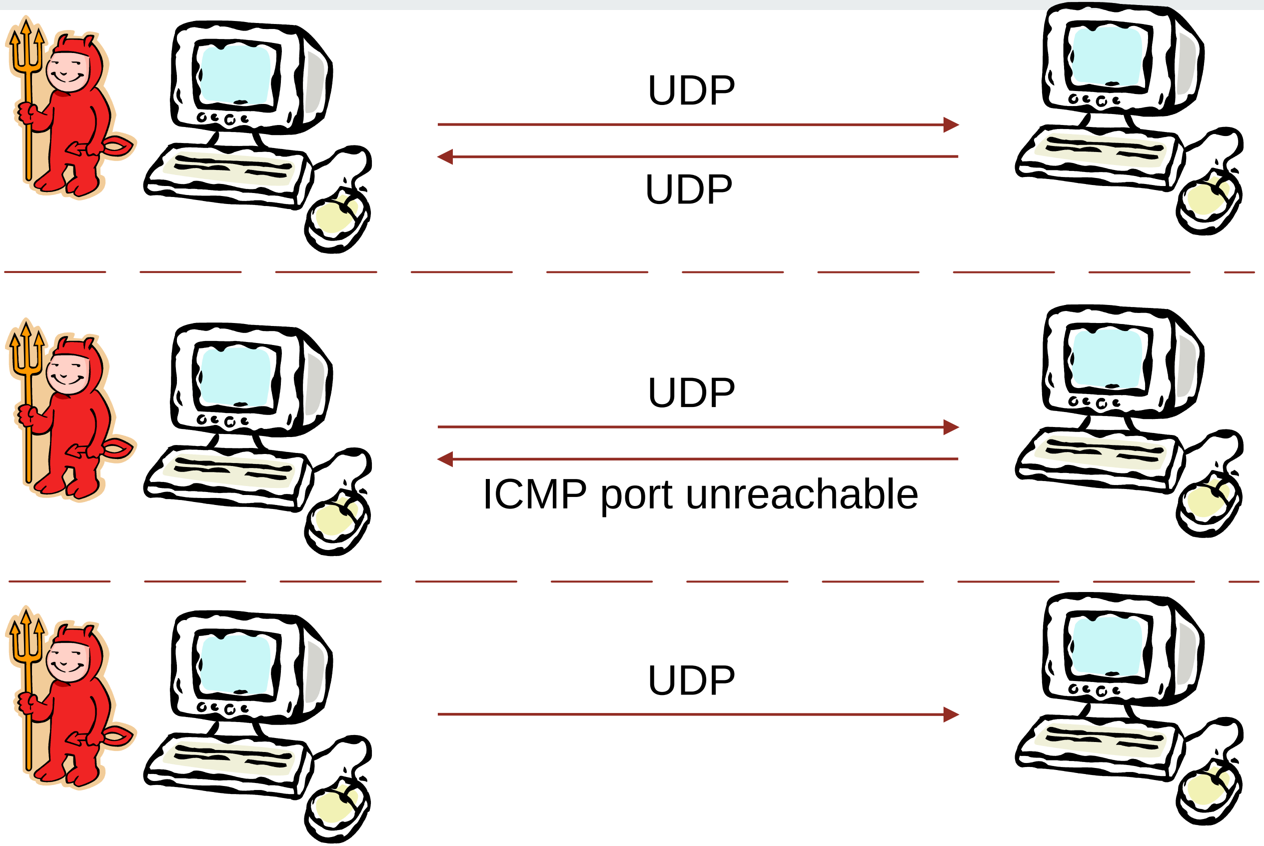


UDP

Port open!

Port closed (blocked by firewall?)!

Port closed or blocked by firewall or port open but expecting specific data?



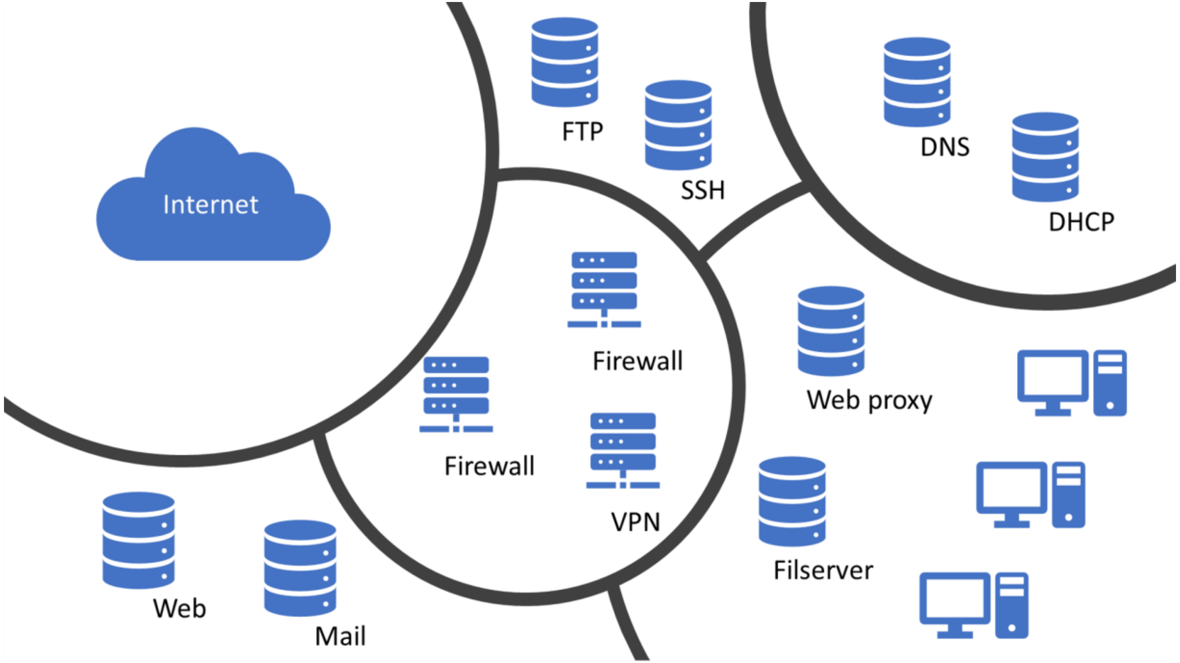
Snort

```
# Snort rule to detect the packet used to exploit a vulnerability
in CVS.

alert tcp $EXTERNAL_NET any -> 130.225.254.12 2401 (msg:"CVS
server heap overflow attempt"; flow:to_server,established;
content:"|45 6e 74 72 79 20 43 43 43 43 43 43 43 43 43 2f 43
43|"; offset:0; depth:20; dsize: >512; threshold: type limit,
track by_dst, count 1, seconds 60 ; sid:1000000; rev:1;
classtype:attempted-admin;)
```

More on logs and visibility

Possible log sources, include:



Example mail server log

```
2022-10-10T11:29:49.0000000Z,user@company.com,UserLoggedIn,[Details]
2022-10-10T11:31:34.0000000Z,user@company.com,UserLoggedIn,[Details]
2022-10-10T11:31:42.0000000Z,user@company.com,FilePreviewed,[Details]
2022-10-10T11:31:45.0000000Z,user@company.com,UserLoggedIn,[Details]
2022-10-10T11:31:47.0000000Z,user@company.com,UserLoggedIn,[Details]
2022-10-10T11:32:44.0000000Z,user@company.com,UserLoggedIn,[Details]
2022-10-10T11:32:54.0000000Z,user@company.com,UserLoggedIn,[Details]
2022-10-10T11:42:30.0000000Z,user@company.com,Set-Mailbox,[Details]
2022-10-10T11:49:33.0000000Z,user@company.com,New-InboxRule,[Details]
2022-10-10T11:55:24.0000000Z,user@company.com,UserLoggedIn,[Details]
```

Example web server log

```
[Oct 1 12:47:57 2022] 87.118.116.103:46928 [200]: /pressroom.php
[Oct 1 12:47:57 2022] 87.118.116.103:46930 [404]: /favicon.ico - No such file or directory
[Oct 1 12:47:57 2022] Notice:  Undefined index: tag in /tmp/php/pressroom.php on line 17
[Oct 1 12:48:05 2022] 87.118.116.103:46932 [200]: /pressroom.php?tag=news
[Oct 1 12:48:14 2022] 87.118.116.103:46934 [200]: /pressroom.php?tag=events
[Oct 1 12:48:14 2022] 87.118.116.103:46936 [200]: /pressroom.php?tag=research
[Oct 1 12:48:18 2022] 87.118.116.103:46938 [200]: /pressroom.php?tag=foo
[Oct 1 12:48:18 2022] Notice:  Non-existent tag requested: foo
[Oct 1 12:48:55 2022] 87.118.116.103:46946 [200]: /pressroom.php?tag=error.log
[Oct 1 12:49:10 2022] 87.118.116.103:46950 [200]: /pressroom.php?tag=../../etc/passwd
```

Example DNS log

```
30-09-2022 01:29:55 UDP Rcv 10.232.65.43    Q (3)www(7)gstatic(3)com(0)
30-09-2022 01:29:55 UDP Snd 10.232.65.43  R Q (3)www(7)gstatic(3)com(0)
30-09-2022 01:29:55 UDP Rcv 10.201.120.30   Q (5)login(4)live(3)com(0)
30-09-2022 01:29:55 UDP Snd 10.201.120.30 R Q (5)login(4)live(3)com(0)
30-09-2022 01:29:55 UDP Rcv 10.230.20.106   Q (2)gg(6)google(3)com(0)
30-09-2022 01:29:55 UDP Snd 10.230.20.106 R Q (2)gg(6)google(3)com(0)
30-09-2022 01:29:55 UDP Rcv 10.201.100.45   Q (4)pool(3)ntp(3)org(0)
30-09-2022 01:29:55 UDP Snd 10.201.100.45 R Q (4)pool(3)ntp(3)org(0)
30-09-2022 01:29:55 UDP Rcv 10.201.100.65   Q (5)yahoo(3)com(0)
30-09-2022 01:29:55 UDP Snd 10.201.100.65 R Q (5)yahoo(3)com(0)
```

Example DHCP log

```
10,2022/09/09,08:30:01,Assign,10.201.22.101,WS10012A,8c164566564e
10,2022/09/09,08:33:12,Assign,10.201.22.108,WS10022A,8c1645665a4b,
10,2022/09/09,08:33:55,Assign,10.201.22.109,WS10052A,8c164566779e,
10,2022/09/09,08:34:01,Assign,10.201.22.110,WS10044A,8c164566464c,
11,2022/09/09,08:34:32,Renew,10.201.22.122,VM10081A,005056c00001,
10,2022/09/09,08:34:34,Assign,10.201.22.130,WS10012A,8c16456651aa
11,2022/09/09,08:35:45,Renew,10.201.22.133,VM10110A,005056cee001,
10,2022/09/09,08:35:53,Assign,10.201.22.134,WS10072A,8c16456ab1a4b,
12,2022/09/09,08:37:01,Release,10.201.22.110,WS10048A,8c16456694c,
10,2022/09/09,08:37:10,Assign,10.201.22.110,WS10097A,8c164561239e,
```

Example firewall log

```
Mar  1 11:28:47 Built inbound UDP id 4253 from 192.38.84.35/7179 to 130.226.237.14/53
Mar  1 11:28:47 Teardown TCP id 4198 duration 0:00:00 bytes 7194 TCP FINs from in
Mar  1 11:28:47 Deny TCP from 10.150.96.249/54735 to 130.226.237.153/4433 flags RST ACK
Mar  1 11:28:47 Built inbound UDP id 4254 from 192.38.84.42/61918 to 130.226.237.14/53
Mar  1 11:28:47 Built inbound UDP id 4257 from 10.202.55.102/64651 to 130.226.237.14/53
Mar  1 11:28:47 Built outbound UDP id 4259 from 130.226.142.7/53 to 130.226.237.14/20238
Mar  1 11:28:47 Built inbound UDP id 4258 from 10.202.55.21/53921 to 130.226.237.14/53
Mar  1 11:28:47 Built outbound UDP id 4255 from 130.226.237.173/53 to 130.226.237.14/27800
Mar  1 11:28:47 Teardown TCP id 4210 duration 0:00:00 bytes 0 TCP FINs
Mar  1 11:28:47 Built inbound id 4260 TCP from 10.209.100.121/62921 to 130.226.237.153/4433
```

Example firewall log

```
Jun 4 14:23:01 src=192.168.30.143 dst=46.30.215.95 tcp spt=42449 dpt=80 len=60 syn [Details]
Jun 4 14:23:01 src=46.30.215.95 dst=192.168.30.143 tcp spt=80 dpt=42449 len=60 ack syn [Details]
Jun 4 14:23:01 src=192.168.30.143 dst=46.30.215.95 tcp spt=42449 dpt=80 len=52 ack [Details]
Jun 4 14:23:01 src=192.168.30.143 dst=46.30.215.95 tcp spt=42449 dpt=80 len=39 ack psh [Details]
Jun 4 14:23:01 src=46.30.215.95 dst=192.168.30.143 tcp spt=80 dpt=42449 len=52 ack [Details]
Jun 4 14:23:01 src=46.30.215.95 dst=192.168.30.143 tcp spt=80 dpt=42449 len=67 ack psh [Details]
Jun 4 14:23:01 src=192.168.30.143 dst=46.30.215.95 tcp spt=42449 dpt=80 len=52 ack [Details]
Jun 4 14:23:01 src=46.30.215.95 dst=192.168.30.143 tcp spt=80 dpt=42449 len=64 ack psh [Details]
Jun 4 14:23:01 src=192.168.30.143 dst=46.30.215.95 tcp spt=42449 dpt=80 len=52 ack [Details]
Jun 4 14:23:01 src=46.30.215.95 dst=192.168.30.143 tcp spt=80 dpt=42449 len=52 ack fin [Details]
```

Log analysis

Log analysis approach

The Question
The specific question we are trying to answer

The Understanding
Understand the log

The Pattern
Find a pattern in the log that helps answer the question

The Search
Find all log entries that contain the pattern

The Extract
Output all or some elements of the log entries

The Clustering
If needed, aggregate the output to answer the question

For example, if we want to find all successful logins to our mail server (**the question**), and we know that the corresponding mail server log entry looks like this (**the understanding**):

```
2021-09-21T08:29:49.00Z,user@company.com,UserLoggedIn,[Details]
```

Then we need to search for “UserLoggedIn” (**the pattern**) to find all entries relevant to answer our question.

Find the pattern

Logs are **different**.

Patterns for successful logins in other logs will look different.

Which patterns should we search for:

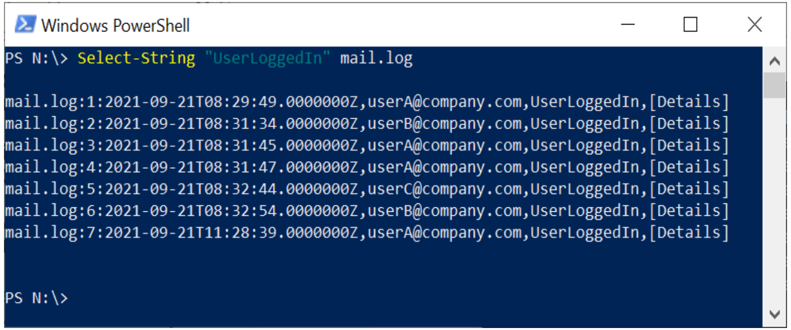
FTP: Sep 30 13:09:52 - ftpuser (80.62.115.191) **! Successfully logged in.**

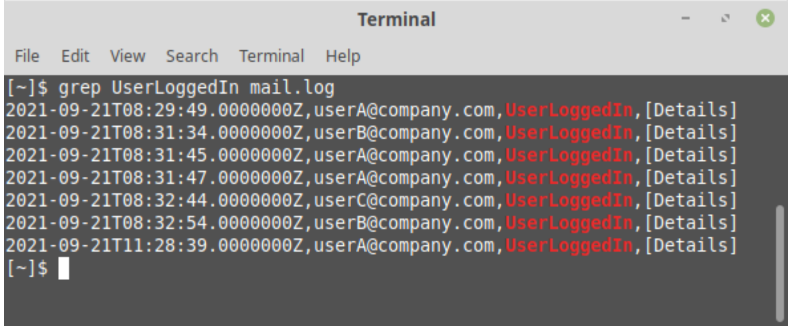
SSH: Sep 30 13:31:07 **Accepted password** for root from 62.44.128.103 port 35901 ssh2

DHCP: Sep 30 13:53:00 User REGH IP 80.62.115.191 IPv4 192.168.200.103 **assigned to session**

Search

Given a pattern, we can use **custom-built tools** (like SIEMs), **write our own tools** using more or less well-chosen string matching algorithms or use **command-line shells**.





Extract

Suppose we want to zoom in on **who logged on**.

Then, in the **terminal**, we can use **cut** to output the second column (**-f2**) delimited by comma (**-d’,’**):

```
[~]$ grep UserLoggedIn mail.log | cut -d',' -f2
userA@company.com
userB@company.com
userA@company.com
userA@company.com
userC@company.com
userB@company.com
userA@company.com
[~]$
```

Clustering

Suppose we want to count **how many times** each user logged on.

Then, in the **terminal**, we can **sort** the output of cut and use **uniq** (**-c**) to occurrence of each user:

```
[~]$ grep UserLoggedIn mail.log | cut -d',' -f2 | sort | uniq -c
      4 userA@company.com
      2 userB@company.com
      1 userC@company.com
[~]$
```

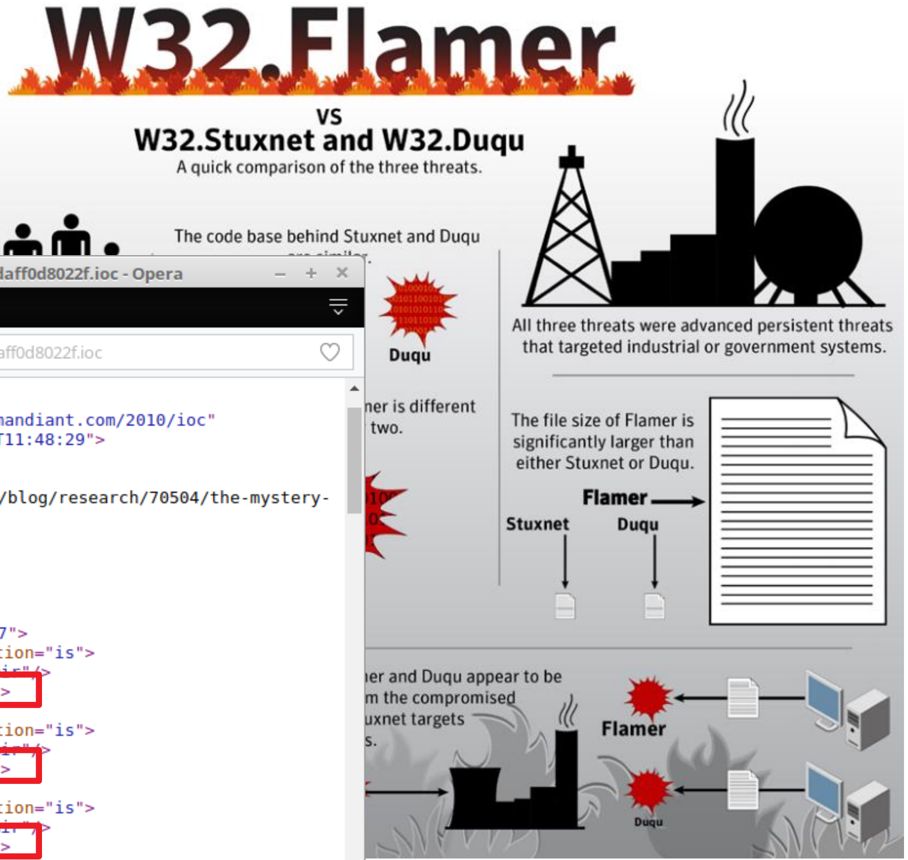
Indicators of compromise in Intrusion Detection

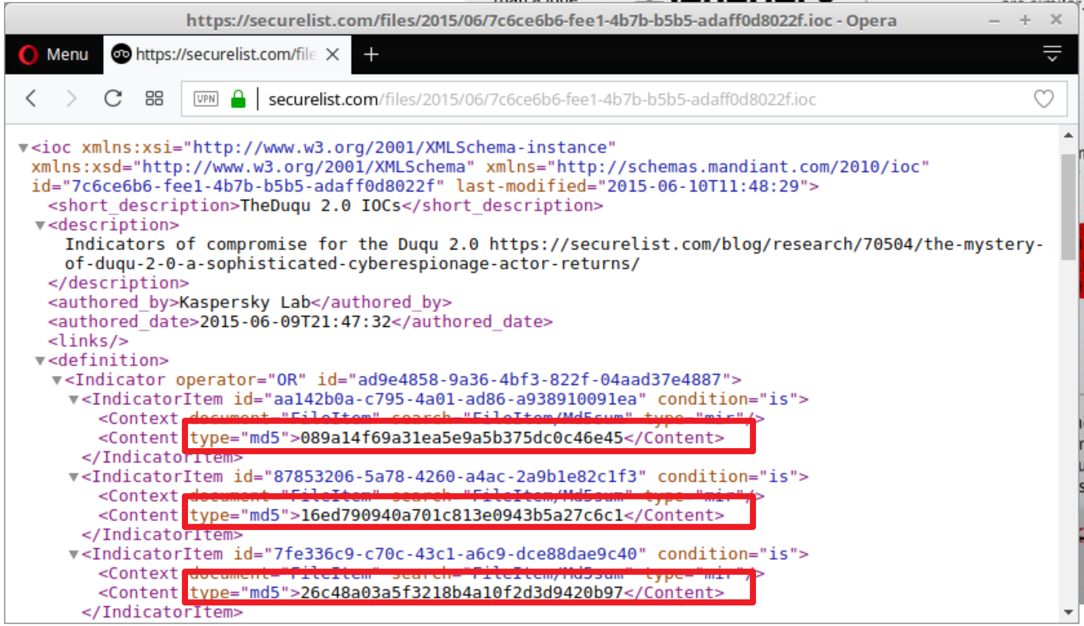
Indicators of compromise (IOCs)

Technical characteristics that identify a known threat, attacker methodology, or other evidence of compromise, e.g.:

- C2 domains
- IPs used in attack
- Special GET requests
- Malware file system locations
- Malware hashes
- Memory artifacts

Duqu IOCs


W32.Flamer
vs
W32.Stuxnet and W32.Duqu
A quick comparison of the three threats.
All three threats appear to be developed by teams of attackers, rather than a lone
The code base behind Stuxnet and Duqu
Duqu
ner is different
two.
All three threats were advanced persistent threats that targeted industrial or government systems.
The file size of Flamer is significantly larger than either Stuxnet or Duqu.
Flamer
Stuxnet
Duqu
ner and Duqu appear to be
m the compromised
uxnet targets
s.
Flamer
Duqu


https://securelist.com/files/2015/06/7c6ce6b6-fee1-4b7b-b5b5-adaff0d8022f.ioc - Opera
Menu
https://securelist.com/file
securelist.com/files/2015/06/7c6ce6b6-fee1-4b7b-b5b5-adaff0d8022f.ioc
<ioc xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.mandiant.com/2010/ioc" id="7c6ce6b6-fee1-4b7b-b5b5-adaff0d8022f" last-modified="2015-06-10T11:48:29">
<short_description>TheDuqu 2.0 IOCs</short_description>
<description>
Indicators of compromise for the Duqu 2.0 https://securelist.com/blog/research/70504/the-mystery-of-duqu-2-0-a-sophisticated-cyberespionage-actor-returns/
</description>
<authored_by>Kaspersky Lab</authored_by>
<authored_date>2015-06-09T21:47:32</authored_date>
<links/>
<definition>
<Indicator operator="OR" id="ad9e4858-9a36-4bf3-822f-04aad37e4887">
<IndicatorItem id="aa142b0a-c795-4a01-ad86-a938910091ea" condition="is">
<Context document="FileItem" search="FileItem/Md5sum" type="mir"/>
<Content type="md5">089a14f69a31ea5e9a5b375dc0c46e45</Content>
</IndicatorItem>
<IndicatorItem id="87853206-5a78-4260-a4ac-2a9b1e82c1f3" condition="is">
<Context document="FileItem" search="FileItem/Md5sum" type="mir"/>
<Content type="md5">16ed790940a701c813e0943b5a27c6c1</Content>
</IndicatorItem>
<IndicatorItem id="7fe336c9-c70c-43c1-a6c9-dce88dae9c40" condition="is">
<Context document="FileItem" search="FileItem/Md5sum" type="mir"/>
<Content type="md5">26c48a03a5f3218b4a10f2d3d9420b97</Content>
</IndicatorItem>

IOCs and “The Pyramid of Pain”