Network Security Course | ETH Zurich - Autumn 2020

Firewalls, Intrusion Detection & Evasion

Dr. Stefan Frei

Security Officer @ SIX Digital Exchange www.sdx.com
frei@techzoom.net | Twitter @stefan_frei

Head of the working group "Supply Chain Security" @ ICT Switzerland

Firewall

Firewall – Types of Operation

Ingress vs Egress

A firewall is a system used to protect or separate a trusted network from an untrusted network, while allowing authorized communications to pass from one side to the other

- A firewall is configured to permit, deny, or proxy data through a network which has different levels of trust
- Firewalls enforce an access control policy between two networks

Firewall – Types of Operation

Network vs. Host

NETWORK FIREWALL

- Network firewalls are a software appliance running on specific hardware or as virtual instance that filter traffic between two or more networks
- Protect different network segments

HOST FIREWALL

- Host-based firewalls provide a layer of software on one host that controls network traffic in and out of that single machine
- Protect single host

Firewall

Filtering Rule

INGRESS

- Filter incoming traffic
- From **low security** to **high security** net

EGRESS

- Filter outgoing traffic
- From **high security** to **low security** net / outside
- Gets often forgotten

DEFAULT POLICY

- Define what to do when no rule matches
- **Default accept** versus **default reject** policy

DENY ACCESS

Techniques to deny access:

- **DROP** Silently drop packet
- **REJECT** Drop packet an inform sender (`ICMP Destination Unreachable`)

Firewall

Filtering Rule

Example: nmap scan against a host protected by firewall

SETUP

Firewall & Host

Port	Firewall Rule	Destination Host
tcp / 22	pass	Port open
tcp / 23	pass	Port closed
tcp / 24	block	anything
tcp / 25	reject	anything

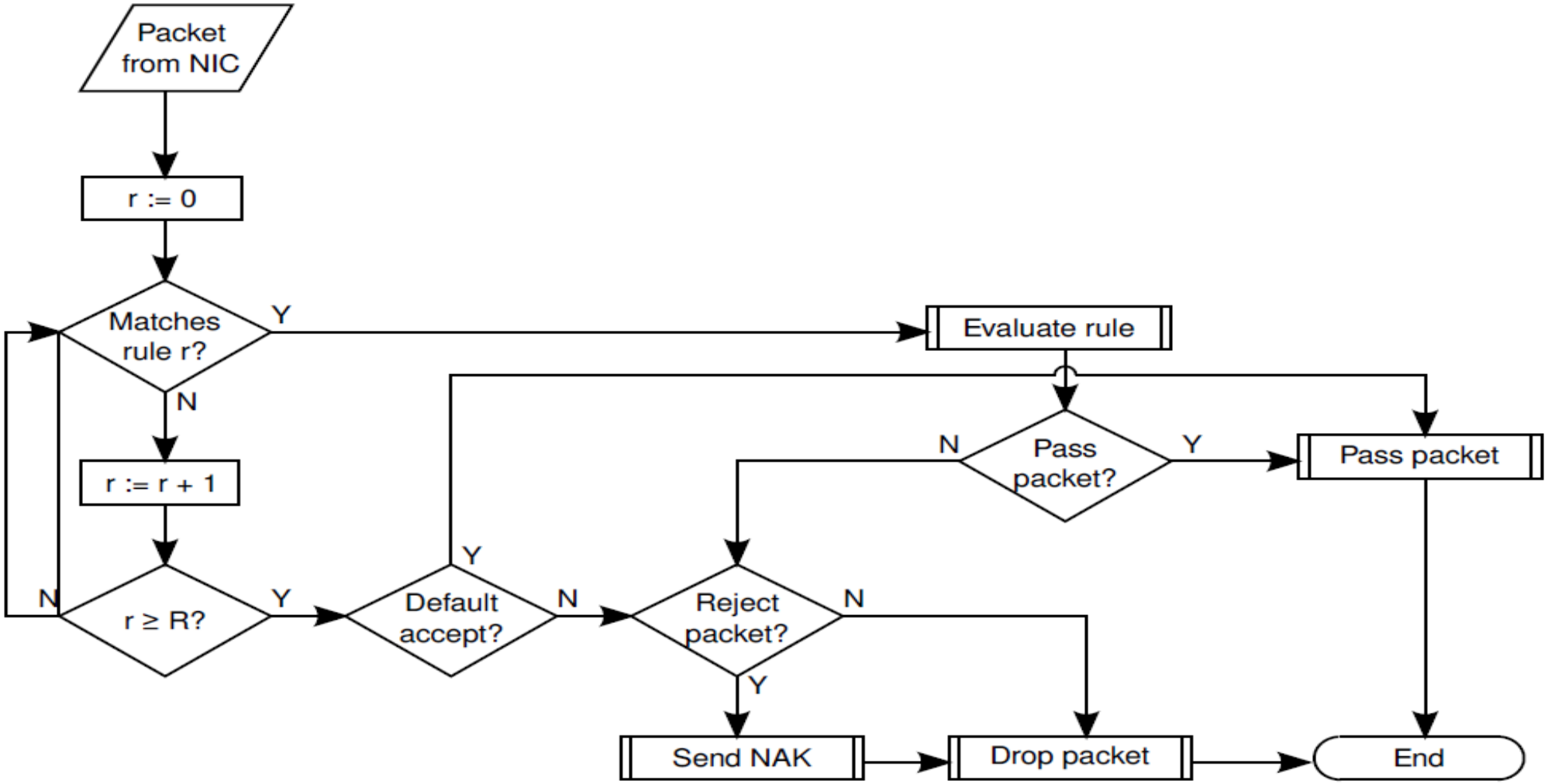
PORT SCAN

```
netsec> nmap -sT -p 22,23,24,25 10.0.0.18
Starting Nmap 7.80 ( https://nmap.org ) at 2020-11-09 14:21 CET
Nmap scan report for target.ethz.ch (10.0.0.18)
Host is up (0.0034s latency).

PORT   STATE    SERVICE
22/tcp open     ssh
23/tcp closed   telnet
24/tcp filtered priv-mail
25/tcp closed   smtp

Nmap done: 1 IP address (1 host up) scanned in 1.27 seconds
```

Firewall Rule Processing



Stateless Firewall

Packet Filter

Functionality

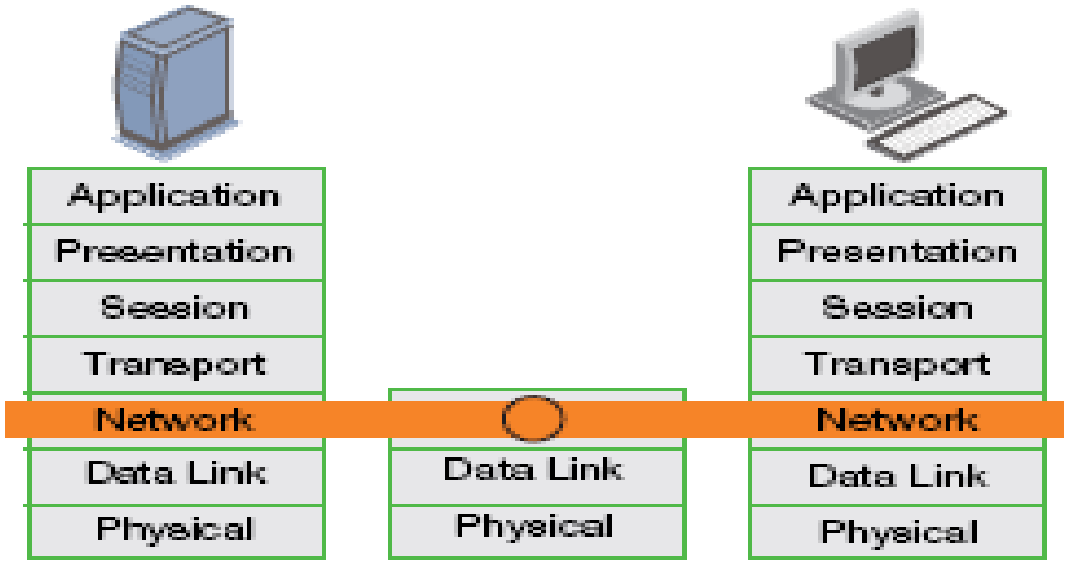
- Examine a packet at the network layer
- Decision based on packet header information (IP, port, flags)

Pros

- Application independent
- Good performance and scalability

Cons

- No state or application context



Stateful Firewall

Packet & Session Filter

Functionality

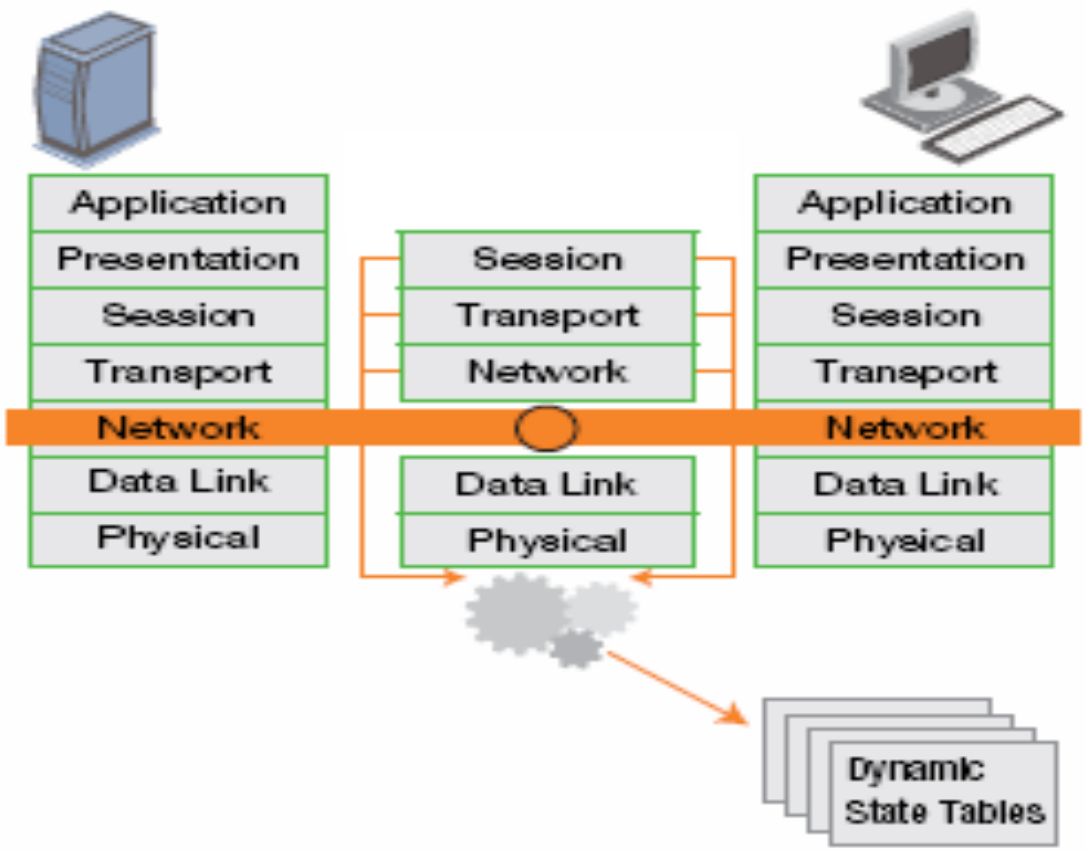
- Keep also track of the state of the network connections
- Decision also based on session state

Pros

- More powerful rules

Cons

- State for UDP?
- Inconsistent state: Host vs. Firewall
- State explosion



Evolution of Firewalls

Enforcing an access control is not enough

- Firewalls can't block all malicious traffic
- Many ports must be kept open for legitimate applications to run

- Users unwittingly download dangerous applications or other forms of malicious code
- Peer-to-peer and instant messaging have introduced new infection vectors

- Web 2.0 trends push critical business applications through firewall ports that were previously reserved for a single function (e.g. HTTP)

The legacy firewall technology is effectively blinded by this evolution

Next Generation Firewall (NGFW)

Payload inspection

Functionality

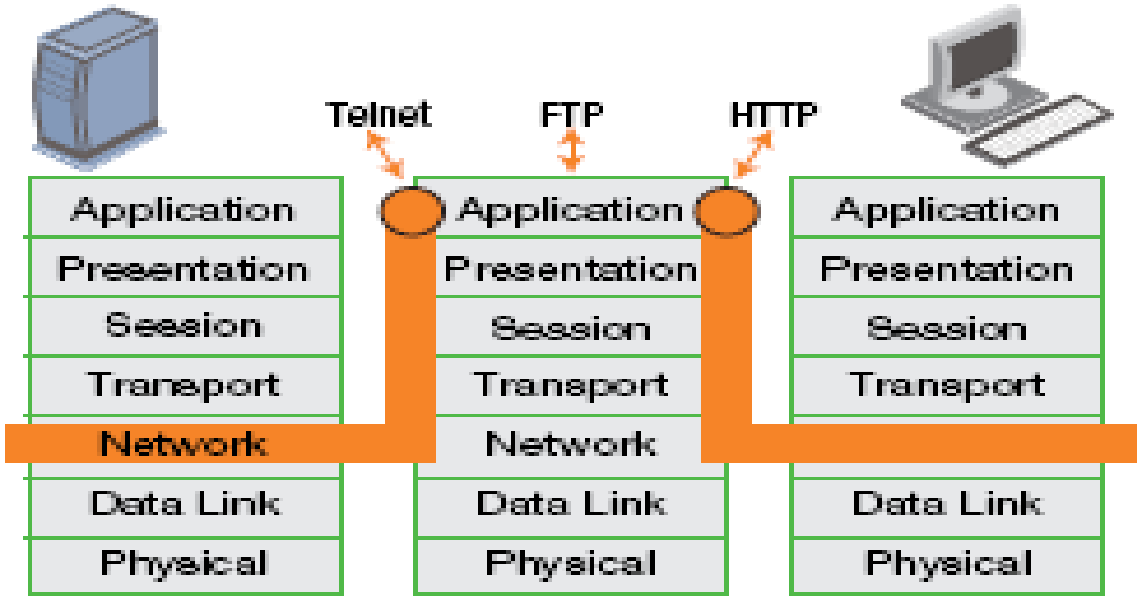
- Deep packet (content) inspection
- Take application and protocol state into account for security decision

Pros

- Even more powerful rules
- Application & protocol awareness

Cons

- Need to support many application protocols
- Performance, scalability
- Inconsistent state: Host vs. Firewall



Web Application Firewall (WAF)

Protect web-based applications from malicious requests

Request filtering

- Request patterns (signatures)
- SQL injection, cross-site scripting, buffer overflow attempts, checking number of form parameters, ...
- Static or dynamic blacklisting / whitelisting
- False positive problem

WAFs are often implemented as a reverse proxy to protect public facing web applications

- Reverse proxy: client is outside the internal network

Deployment Challenges

Scaling protection

Protecting large number of hosts, endpoints, network segments is not trivial

COMPLEXITY

- Firewall rulesets are complex and grow over time
- Thousands of rules on a single firewall are no exception
- Detection between channels out of sync (mail vs net vs proxy)

MANAGEMENT

- Tools are needed to manage hundreds of firewalls and their rules securely
- What is the process to change rulesets (orphan rules, expiration)?
- Who has permission, monitoring of changes

INCENTIVES

Infrastructure Team

- paid for providing connectivity, blamed for disruptions

Security Team

- paid to protect and disrupt connectivity

Firewall Attack Methods

Attack Techniques

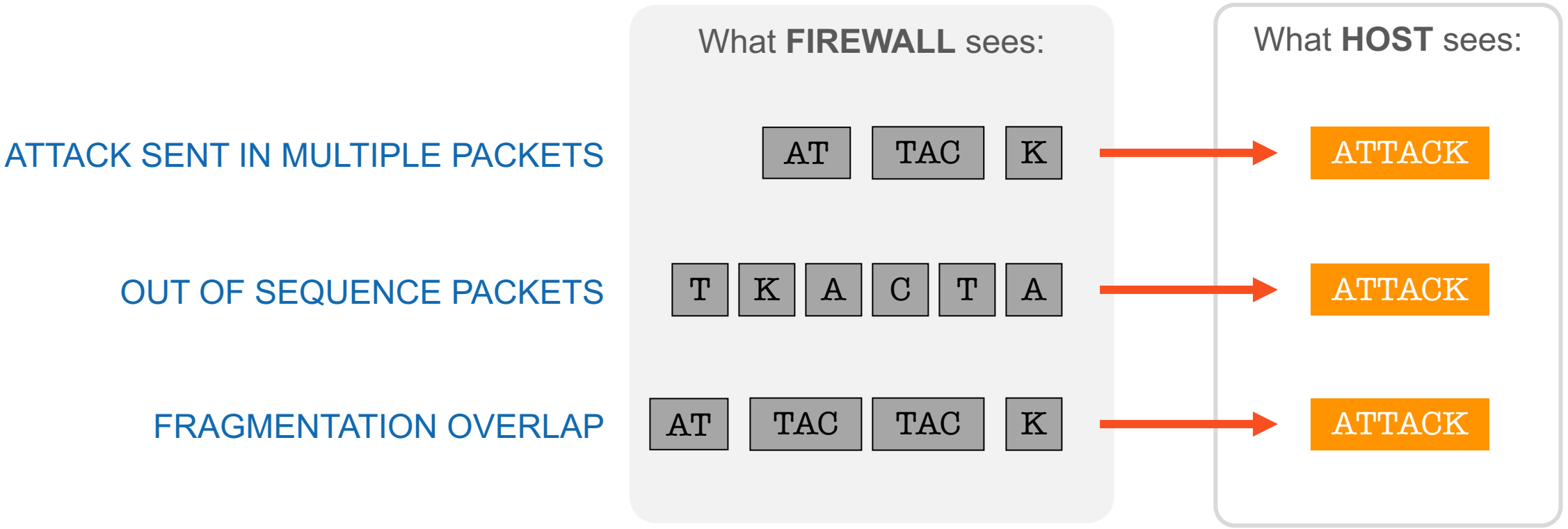
Firewalls are just complex machines, with vulnerabilities and assumptions

IP SOURCE SPOOFING	▪ Spoofing the source IP address to bypass filters ▪ Works for stateless protocols (e.g. UDP, ineffective for TCP)
ARTIFICIAL FRAGMENTATION	▪ Fragment packets to bypass rules ▪ Without proper reassembly at the firewall the attack gets through ▪ *The port number is only in the first fragment*
VULNERABILITIES	▪ Exploiting vulnerabilities in firewall software / firmware / OS ▪ Exploit vulnerabilities in target application
DENIAL OF SERVICE	▪ Firewall state explosion ▪ What's the firewall fallback policy?
TUNNELING COVERT CHANNELS	▪ Data in ICMP ping packets, or use DNS requests as channel ▪ Attack through VPN virtual private network
ENCODINGS	▪ Different encodings and addition of noise

Evasion by Fragmentation

Attack only visible after correct reassembly of packets

Detection will fail to recognize the signature ATTACK without proper reassembly of packets



Payload Encoding

Different encodings or mappings confuse detection

Confuse the inspection engine into allowing an otherwise blocked attacks to pass using various encoding options allowed on the protocol or application in use

Examples:

URL ENCODINGS

- Escape encoding (% encoding),
- Microsoft (%u) encoding, path character transformations (`/./` , `//`, `\`)

HTML OBFUSCATIONS

- UTF-7, UTF-16 and UTF-32 character set encoding
- Chunked encodings (different chunk sizes, chaffing/arbitrary numbers inserted)
- Different compression schemas (Deflate, GZip)
- Base64 encoding

https://en.wikipedia.org/wiki/Chunked_transfer_encoding

Payload Encoding

Different encodings or mappings confuse detection

ORIGINAL ATTACK STRING (CVE-2004-0121)

- the following original attack string represented in different encodings:

```
<img src="mailto:aa&amp;quot; /select javascript:alert('vulnerable')">
```

7-BIT UNICODE ENCODED

```
+ADw-html+AD4 +ADw-body+AD4+ADw-img src+AD0AIg-mailto:aa+ACY-quot; /select
javascript:alert('vulnerable')+ACIAPg+ADw-/body+AD4 +ADw-/html+AD4
```

BASE64 ENCODED WITH CHAFF

- chaff is insertion of noise with the purpose to confuse systems

```
P[G;.?h0bW☖{#w_+%_~&%]I<Dxib!&2$R'5|Pg,^o8(;aW1nI:$H );_N'-
?>yYz$0i\(*~?bWF>p^b.&HRv}OmF#.hJn%#:F1b3Q`7_IC{9(#@z#.Z☖W}xl☖Y&3Qg[amF*2YX#N^}
|^?^`j()cm$]>☖l%w,dD"$p](hb.☖\^#GVy'>d@!!☖~Cgnd`n[ Vsb](m'VyYW☖JsZS#c`
!)#"p'I@%j4KP'C9i`~b.:2]R5'{P?$i';A_8L *,2)h}0)@bWw☖+Cgo=
```

Payload Encoding

Infinite Combinations

Different encoding and obfuscation schemas can be combined with noise insertion in millions of ways

- Encodings are not necessarily unique
- Undefined or border cases are very effective for detection evasion
- Different implementations of decoding on **target application** vs. **detection engine decoding**

Firewall Vulnerabilities

Complex Software has Vulnerabilities

Firewalls are complex software, which is riddled by vulnerabilities as any other software product. Being a security product from a security vendor does not imply better code quality.

- **Juniper Networks** revealed that it had found “unauthorized” code embedded in an operating system running on some of its firewalls - Dec 2015
- Network security vendor **Fortinet** has identified an authentication issue that could give remote attackers administrative control over some of its products - Jan 2016
- A vulnerability in the Internet Key Exchange (IKE) version 1 (v1) and IKE version 2 (v2) code of **Cisco ASA Software** could allow an unauthenticated, remote attacker to cause a reload of the affected system or to remotely execute code - Feb 2016
- Numerous **antivirus end-point protection** issues: https://googleprojectzero.blogspot.com/2016/06/how-to-compromise-enterprise-endpoint.html

Intrusion Detection Methods

Intrusion / Attack Detection

Key Features

Protecting a large number of hosts, end-points, or network segments is not trivial

REACTIVE
- System can only detect already known attacks

PROACTIVE
- System can detect known and new, yet unknown attacks

DETEMINISTIC
- System always performs the same given the same input
- The same stimuli always result in same action
- Reason for alert is known

NON-DETERMINISTIC
- System detection is fuzzy (heuristics, machine learning, sandboxing) Depends on current state of the world
- Reason for alert typically not known

Detection Techniques

Basic Detection Approaches

Static (Signatures)

- Static analysis and identification of known malware.
- Create and distribute signatures of known malware.

- Reliable, low rate of false positives.
- Reactive, only known malware can be detected.
- Problem of large scale distribution of signatures.

Behavior (Dynamic)

- Catalogue and identify suspicious behavior.
- Run samples, classify upon observed behavior.

- Proactive, detection of unknown malware.
- Complex and computationally expensive.
- Higher rate of false positives.

Detection Techniques

Key Features

PROTOCOL ANALYSIS

- Analysis and decoding of protocols
- Reassembly and normalization of traffic

SIGNATURES

Compare attributes of observables

- to a blacklist or whitelist
- to patterns of known attacks / exploits / malware

SANDBOXING

- Suspicious file is executed within a virtual environment
- Specific actions are categorized and labeled as good or bad

MACHINE LEARNING

- Recognize complex patterns
- Make decisions based on the data and assumptions formed from previous data

Signature Based Detection

PROACTIVE	DETERMINISTIC
NO	YES

Key Features

Over the years, signature-based systems have changed and advanced. **The core concepts still lie at the heart of all modern detection systems** .. and will continue to be integral for the foreseeable future

Characteristics

- Able to promptly identify and label a threat
- Different signature systems are used together so they can most accurately label a known threat
- For each new threat, a unique signature or signature artifact is created by a skilled engineer or security researcher
- Frequent updates to signature database or online lookups

Progression and sophistication of signature-based detection systems depend upon human signature writers

Source: Machine Learning In Security: Good & Bad News About Signatures (1) - http://ubm.io/21Vi7G6

Signature Based Detection

One Dimensional

PROACTIVE	DETERMINISTIC
NO	YES

What / How

- Exists in practically all detection and protection technologies
- By far the fastest and most efficient way of categorizing a data artifact
- Boolean output: known good (e.g. whitelist), known bad (e.g. from blacklist)

Pros

- Low resource requirements
- Fast
- Low false positives

Cons

- Reactive – threat must be known before
- Frequent update of signatures
- Humans in the loop

Signature Based Detection

Two Dimensional

PROACTIVE	DETERMINISTIC
NO	YES

What / How

- Classic regular-expression functions and string matching
- Fundamental building blocks of anti-malware, IDS, and DLP systems
- Easily capable of identifying previously known exploits and host enumeration techniques

Pros

- Low/medium resource requirements
- More flexibility with patterns
- Low false positives

Cons

- Reactive – threat must be known before
- Frequent update of signatures
- Humans in the loop

DLP = Data Loss Prevention

Signature Based Detection

Multi Dimensional

PROACTIVE	DETERMINISTIC
PARTIAL	YES

Hybrid system as the threat spectrum grew and attackers found new ways to obfuscate attacks

What / How

- Instead of triggering on a single signature, a multi-dimensional signature was created
- In both sandboxing and network behavioral monitoring, certain actions and activities are labeled as either suspicious or bad
- When a threshold of good or bad activities is reached, the threat is classified and labeled

Pros & Cons

- Best mix of one and two dimensional systems
- More efficient and effective than single approach

DLP = Data Loss Prevention

Sandboxing Based Detection

Run malware in detonation chamber

PROACTIVE	DETERMINISTIC
YES	MOSTLY

Sandboxing products typically run a samples in a sandbox (instrumented) environment

What / How

- Examine / monitor runtime behavior of sample (e.g. malware)
- Compare the behavior against:
  - A list or rules previously developed by the vendor in their lab
  - Apply machine learning for behavior classification

Pros

- Proactive, can detect unknown threats
- No signature updates required

Cons

- Resource intensive
- High latency
- Difficult to scale

Machine Learning

Run malware in detonation chamber

PROACTIVE	DETERMINISTIC
YES	PARTIAL

Security vendors are now applying increasingly sophisticated machine learning elements into their cloud-based analysis and classification systems, and into their products

PROVEN

- These techniques have already proven their value in Internet search, targeted advertising, and social networking

NO HUMANS

- Machine learning largely removes humans and their biases to the development of an n-dimensional signature (or a “classification model”)

Machine Learning

Supervised Learning

Instead of manually trying to figure out and label all the good, bad, and suspicious behaviors, a machine is fed a bunch of “known bad” and “known good” samples (this could be be binary files or network traffic)

- The machine compares all the observable behaviors of the collected samples, automatically determines which behaviors were more prevalent or less prevalent to each class of samples
- It then calculates a weighting factor for each behavior, and combines all that intelligence in to a single model of n-dimensions

Machine Learning

Unsupervised Learning

The machine infers a function to describe hidden structure from unlabeled data. Since the examples are unlabeled, there is no error or reward signal to evaluate a potential solution.

- The detection models and engines still require an element of baselining, but continually learn and reassess that baseline on an hourly or daily basis.
- Capable of identifying attack vectors such as “low-and-slow” data exfiltration, lateral movement, and staging servers

Evolution

Attack & Malware Detection

SIGNATURE-BASED detection systems will continue to be valuable in to the future – not as a replacement, but as a companion to the new advancements in unsupervised machine learning

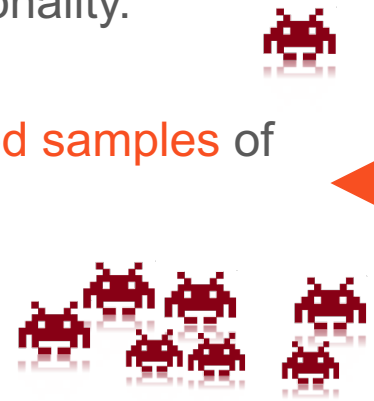
MACHINE LEARNING APPROACHES will play an increasingly important role in future generations of threat detection technology

Detection Evasion by Design

Detection Evasion by Design

Malware Development Lifecycle

1. Develop new malware with desired functionality.
2. Automatically create numerous permutated samples of the initial malware at massive scale.
3. Protect samples from analysis.
4. Make samples aware of sandboxing / detection technologies.
5. Quality Assurance: Test samples against all current anti-malware solutions before deployment.



If sample is detected by antivirus

Malware used in a targeted attack will not be detected by anti-malware tools at the time of attack because it was tested for detection beforehand

An arms race ...

57.6 Million

Malware samples counted in 2017/Q3

640,000	samples / day
26,667	samples / hour
444	samples / minute
9	samples / second

Source: https://www.mcafee.com/us/resources/reports/rp-quarterly-threats-dec-2017.pdf

Step 1 - Create Initial Malware

Create malware core functionality, such as key logger, web-injection, attacks, spreading, communication, ..

- Hire a coder
- Buy malware on dark market
- Steal malware (no trust among thieves)
- Buy / use a do-it-yourself malware construction kit

Step 2 – Crypter & Serial Variants

- Encrypt malware so that signature detection systems and static analysis processes are ineffectual
- Use different keys to create serial variants / permutations
- Typically encrypt the contents of the malware executable.
- Upon execution only decrypt sections of code that are in the process of being executed on the victim's computer.

Step 3 & 4 – Protect malware against debugging

Add anti-debugging features to malware that prevent security researchers and automated sandbox analysis technologies from dissecting samples.

- "Protector" technology was originally designed as a **DRM protection** technology.
- Ironically it is now more commonly employed by criminals.

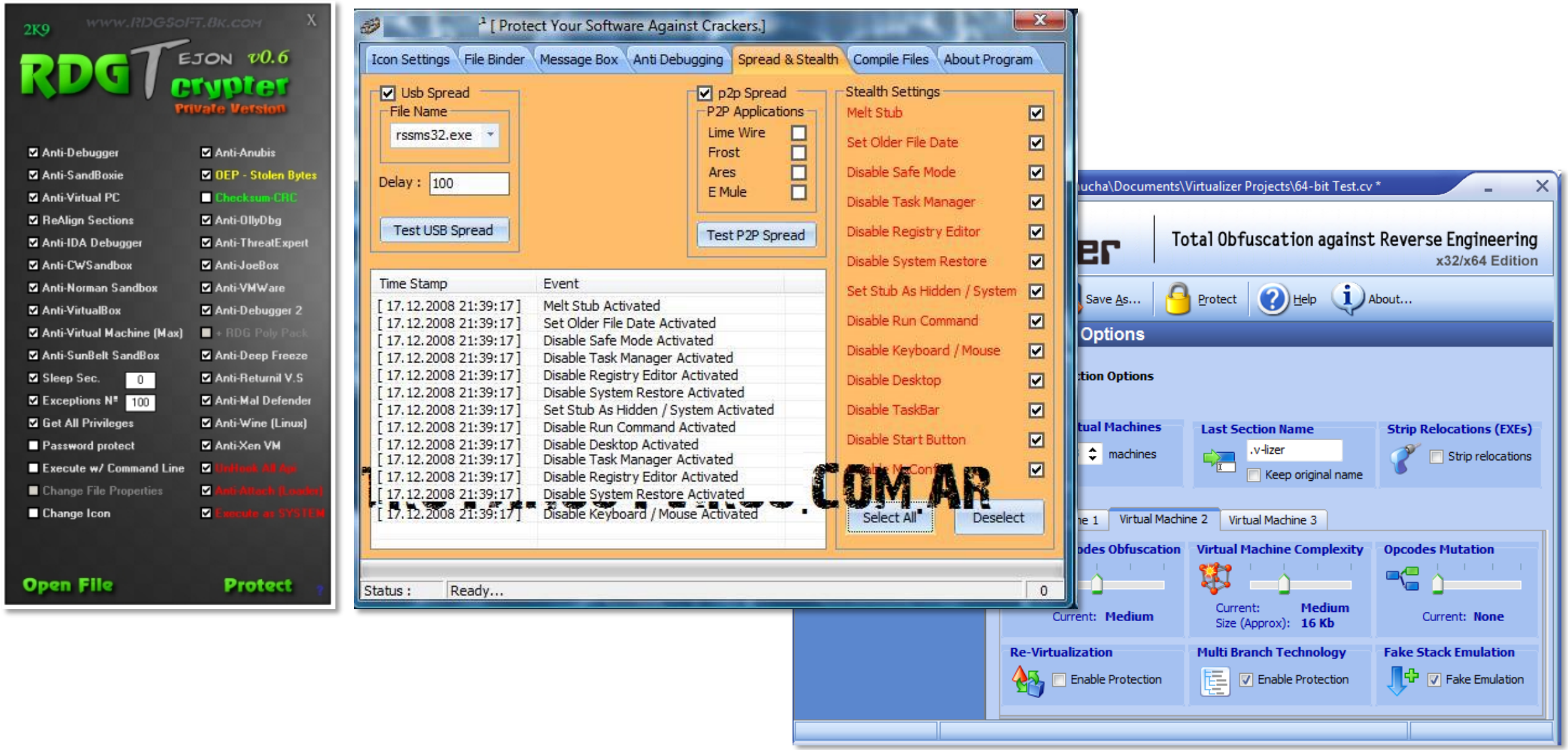
Protectors detect the use of debuggers or virtualization techniques

- If seen, the malware then causes different operations.
- Hiding the malicious intention, or trigger exploitation of virtual environment.

Commercial Code Virtualizer for USD 99.-
http://oreans.com/codevirtualizer.php

DRM = Digital Rights Management

Step 3 & 4 – Protector Tools



Polymorphism Techniques

Polymorphism Techniques

Mutate code while keeping the original algorithm intact.

Tools manipulate the structures of the source code of the malware by reordering and replacing common programmatic routines.

EXAMPLE TECHNIQUES:

- Swapping of equivalent code constructs
- Changing the order of code
- Inserting noise
- Compiler modulation

Step 3 & 4 – Polymorphism

Swapping equivalent code constructs

```
while (condition) {
    statements;
}
```

=

```
while (true) {
    if (!condition) break;
    statements;
}
```

```
if (condition) {
    do {
        statements;
    } while (condition);
}
```

```
goto TEST;
START:
    statements;
TEST:
    if (condition) goto START;
```

Step 3 & 4 – Polymorphism

Changing the order objects in code

- Swapping of registers
- Reordering of instructions
- Defining functions in a different order

```
function alfa() {
    statements;
}
function bravo(b1,b2) {
    statements;
}
function charlie(c) {
    statements;
}
```

=

```
function charlie(c) {
    statements;
}
function alfa() {
    statements;
}
function bravo(b1,b2) {
    statements;
}
```

Step 3 & 4 – Polymorphism

Insert noise

- Insert redundant code that either does nothing or is interpreted as doing nothing, and has no negative impact upon compiled code
- Inclusion of code with exceptions that will never occur or will never be called.
- Include code that calculates π but is never called.

```
if (1==1) {
   statements;
}
```

```
Sleep(0);
statements;
```

```
if (!sqrt(a)<0) {
   statements;
}
```

Step 3 & 4 – Polymorphism

Compiler setting modulation

- The compiler(s) used can have a significant effect on the final output code.
- Different compilers, different versions of compiler, and even different compiler settings can result in substantially different binary code output.
- Minor tweaks to compiler settings are easy to automate and provide a reasonable degree of flexibility in the creation of serial variants.

gcc compiler options

Optimization Options

-falign-functions[=*n*] **-falign-jumps**[=*n*] **-falign-labels**[=*n*] **-falign-loops**[=*n*] **-fassociative-math -fauto-inc-dec -fbranch-probabilities -fbranch-target-load-optimize -fbranch-target-load-optimize2 -fbtr-bb-exclusive -fcaller-saves -fcheck-data-deps -fconserve-stack -fcprop-registers -fcrossjumping -fcse-follow-jumps -fcse-skip-blocks -fcx-fortran-rules -fcx-limited-range -fdata-sections -fdce -fdce -fdelayed-branch -fdelete-null-pointer-checks -fdse -fdse -fearly-inlining -fexpensive-optimizations -ffast-math -ffinite-math-only -ffloat-store -fforward-propagate -ffunction-sections -fgcse -fgcse-after-reload -fgcse-las -fgcse-lm -fgcse-sm -fif-conversion -fif-conversion2 -findirect-inlining -finline-functions -finline-functions-called-once -finline-limit=*n* -finline-small-functions -fipa-cp -fipa-cp-clone -fipa-matrix-reorg -fipa-pta -fipa-pure-const -fipa-reference -fipa-struct-reorg -fipa-type-escape -fira-algorithm=*algorithm* -fira-region=*region* -fira-coalesce -fno-ira-share-save-slots -fno-ira-share-spill-slots -fira-verbose=*n* -fivopts -fkeep-inline-functions -fkeep-static-consts -floop-block -floop-interchange -floop-strip-mine -fmerge-all-constants -fmerge-constants -fmodulo-sched -fmodulo-sched-allow-regmoves -fmove-loop-invariants -fmudflap -fmudflapir -fmudflapth -fno-branch-count-reg -fno-default-inline -fno-defer-pop -fno-function-cse -fno-guess-branch-probability -fno-inline -fno-math-errno -fno-peephole -fno-peephole2 -fno-sched-interblock -fno-sched-spec -fno-signed-zeros -fno-toplevel-reorder -fno-trapping-math -fno-zero-initialized-in-bss -fomit-frame-pointer -foptimize-register-move -foptimize-sibling-calls -fpeel-loops -fpredictive-commoning -fprefetch-loop-arrays -fprofile-correction -fprofile-dir=*path* -fprofile-generate -fprofile-generate=*path* -fprofile-use -fprofile-use=*path* -fprofile-values -freciprocal-math -fregmove -frename-registers -freorder-blocks -freorder-blocks-and-partition -freorder-functions -frerun-cse-after-loop -freschedule-modulo-scheduled-loops -frounding-math -frtl-abstract-sequences -fsched2-use-superblocks -fsched2-use-traces -fsched-spec-load -fsched-spec-load-dangerous -fsched-stalled-insns-dep[=*n*] -fsched-stalled-insns[=*n*] -fschedule-insns -fschedule-insns2 -fsection-anchors -fsee -fselective-scheduling -fselective-scheduling2 -fsel-sched-pipelining -fsel-sched-pipelining-outer-loops -fsignaling-nans -fsingle-precision-constant -fsplit-ivs-in-unroller -fsplit-wide-types -fstack-protector -fstack-protector-all -fstrict-aliasing -fstrict-overflow -fthread-jumps -ftracer -ftree-builtin-call-dce -ftree-ccp -ftree-ch -ftree-coalesce-inline-vars -ftree-coalesce-vars -ftree-copy-prop -ftree-copyrename -ftree-dce -ftree-dominator-opts -ftree-dse -ftree-fre -ftree-loop-im -ftree-loop-distribution -ftree-loop-ivcanon -ftree-loop-linear -ftree-loop-optimize -ftree-parallelize-loops=*n* -ftree-pre -ftree-reassoc -ftree-sink -ftree-sra -ftree-switch-conversion -ftree-ter -ftree-vect-loop-version -ftree-vectorize -ftree-vrp -funit-at-a-time -funroll-all-loops -funroll-loops -funsafe-loop-optimizations -funsafe-math-optimizations -funswitch-loops -fvariable-expansion-in-unroller -fvect-cost-model -fvpt -fweb -fwhole-program --param *name*=*value* **-O -O0 -O1 -O2 -O3 -Os**

Preprocessor Options

-A*question*=*answer* **-A-***question*[=*answer*] **-C -dD -dI -dM -dN -D***macro*[=*defn*] **-E -H -idirafter** *dir* **-include** *file* **-imacros** *file* **-iprefix** *file* **-iwithprefix** *dir* **-iwithprefixbefore** *dir* **-isystem** *dir* **-imultilib** *dir* **-isysroot** *dir* **-M -MM -MF -MG -MP -MQ -MT -nostdinc -P -fworking-directory -remap -trigraphs -undef -U***macro* **-Wp,***option* **-Xpreprocessor** *option*

Assembler Option

-Wa,*option* **-Xassembler** *option*

Linker Options

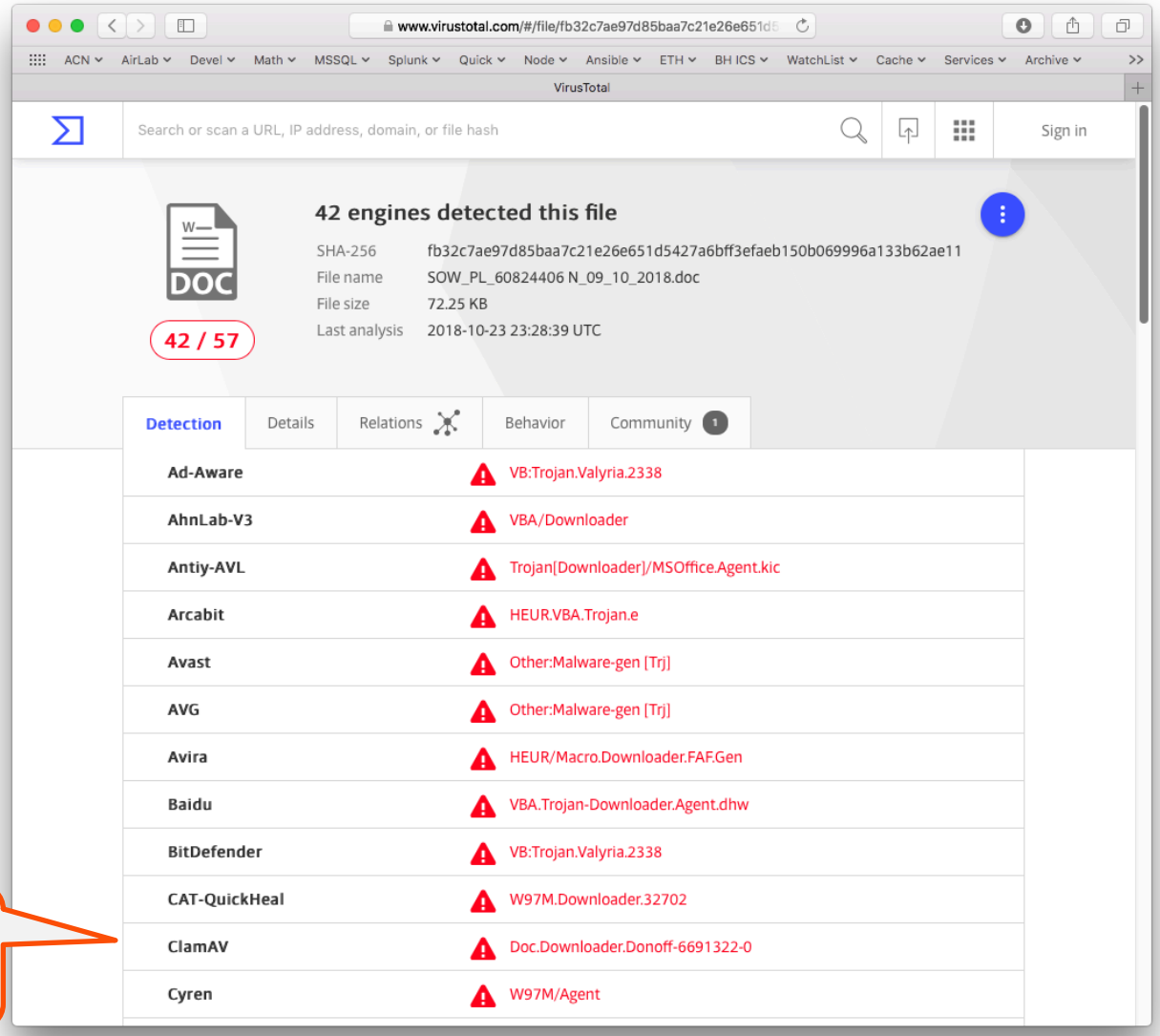
object-file-name **-l***library* **-nostartfiles -nodefaultlibs -nostdlib -pie -rdynamic -s -static -static-libgcc -shared -shared-libgcc -symbolic -T** *script* **-Wl,***option* **-Xlinker** *option* **-u** *symbol*

Step 5 - Quality Assurance

Check out www.virustotal.com

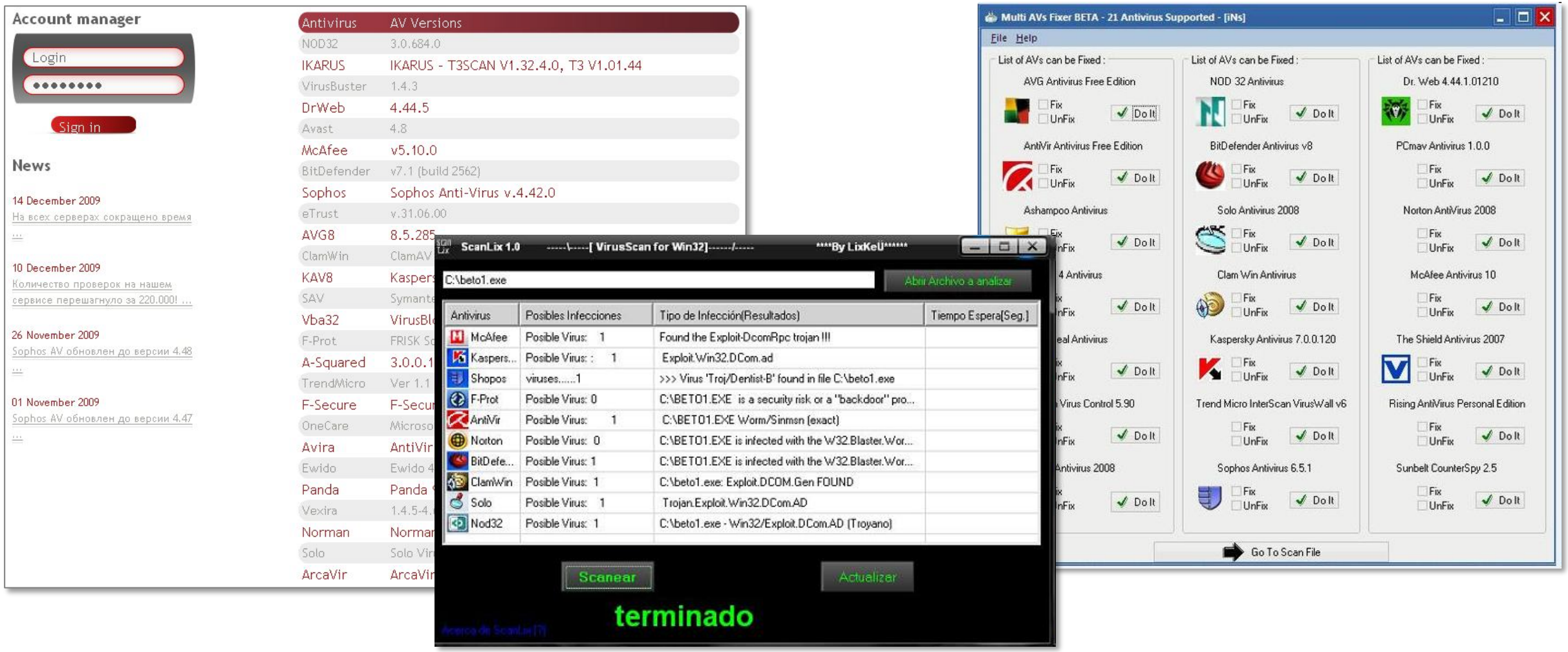
Malware samples created are passed **through multiple commercial antivirus products** to verify that they will not be detected prior to their criminal deployment.

- The underground provides tools and services for **automated testing and detection notification** for entire malware collections.
- Only testing services that do not submit malware samples to antivirus vendor are used (so www.virustotal.com is out).



Scan of an attachment received 2018-09-10

Step 5 - Automated Anti-Malware Checking Tools



Source: http://www.wired.com/threatlevel/2009/12/virus-check
http://pandalabs.pandasecurity.com/archive/Multi-AVs-Scanners.aspx

Detection Example

(Mail attachment: info.zip)

Day 1 at 13:13h – Detected by 2%

Testing of infected info.zip, sent to the author as mail attachment by a business partner.

Upon **initial upload, 1 out of 57** up-to-date anti-malware engines flagged the file as malicious.

SHA256: bf84db71be81fa27d0d796d000347d47ef0dcd814062663d556726bf15e15678

File name: info.zip

Detection ratio: 1 / 57

Analysis date: 2015-04-16 13:13:39 UTC (9 minutes ago)

Analysis | Relationships | Additional information | Comments 5 | Votes

Antivirus	Result	Update
Norman	Kryptik.CLASS	20150416
ALYac	✔	20150416
AVG	✔	20150416
AVware	✔	20150416
Ad-Aware	✔	20150416
AegisLab	✔	20150416
Agnitum	✔	20150416
AhnLab-V3	✔	20150416
Alibaba	✔	20150416
Antiy-AVL	✔	20150416
Avast	✔	20150416

Day 1 at 14:55h – Detected by 7%

Testing of infected info.zip, sent to the author as mail attachment by a business partner.

1.8 hours later, 4 out of 57 up-to-date anti-malware engines flagged the file as malicious.

SHA256: bf84db71be81fa27d0d796d000347d47ef0dcd814062663d556726bf15e15678

File name: info.zip

Detection ratio: 4 / 53

Analysis date: 2015-04-16 14:55:27 UTC (4 minutes ago)

12 0

Analysis | Relationships | Additional information | Comments 5 | Votes

Antivirus	Result	Update
Ikarus	Win32.Outbreak	20150416
Norman	Kryptik.CLASS	20150416
Panda	Trj/Genetic.gen	20150416
TrendMicro-HouseCall	Suspici.304ED7F9	20150416
ALYac		20150416
AVG		20150416
AVware		20150416
Ad-Aware		20150416
AegisLab		20150416
Agnitum		20150416

6 months later – Detected by 80%

Testing of infected info.zip, sent to the author as mail attachment by a business partner.

Half a year later, still not detected by all engines:
44 of 55 anti-malware engines flagged the file as malicious.

SHA256:	bf84db71be81fa27d0d796d000347d47ef0dcd814062663d556726bf15e15678
File name:	info.zip
Detection ratio:	44 / 55
Analysis date:	2015-12-25 09:14:25 UTC (4 months, 2 weeks ago)

Analysis | File detail | Relationships | Additional information | Comments 5 | Votes

Antivirus	Result	Update
ALYac	Trojan.GenericKD.2301190	20151225
AVG	Generic_s.ENO	20151225
AVware	Trojan.Win32.Upatre.qv (v)	20151225
Ad-Aware	Trojan.GenericKD.2301190	20151224
Yandex	Trojan.DL.Upatre!	20151225
Antiy-AVL	Trojan[Downloader]/Win32.Upatre	20151225
Arcabit	Trojan.Generic.D231D06	20151225
Avast	Win32:Crypt-SAL [Trj]	20151225
Avira (no cloud)	TR/Crypt.ZPACK.138875	20151224
Baidu-International	Trojan.Win32.Upatre.vyh	20151224
BitDefender	Trojan.GenericKD.2301190	20151225

Attack Detection & Defense Effectiveness

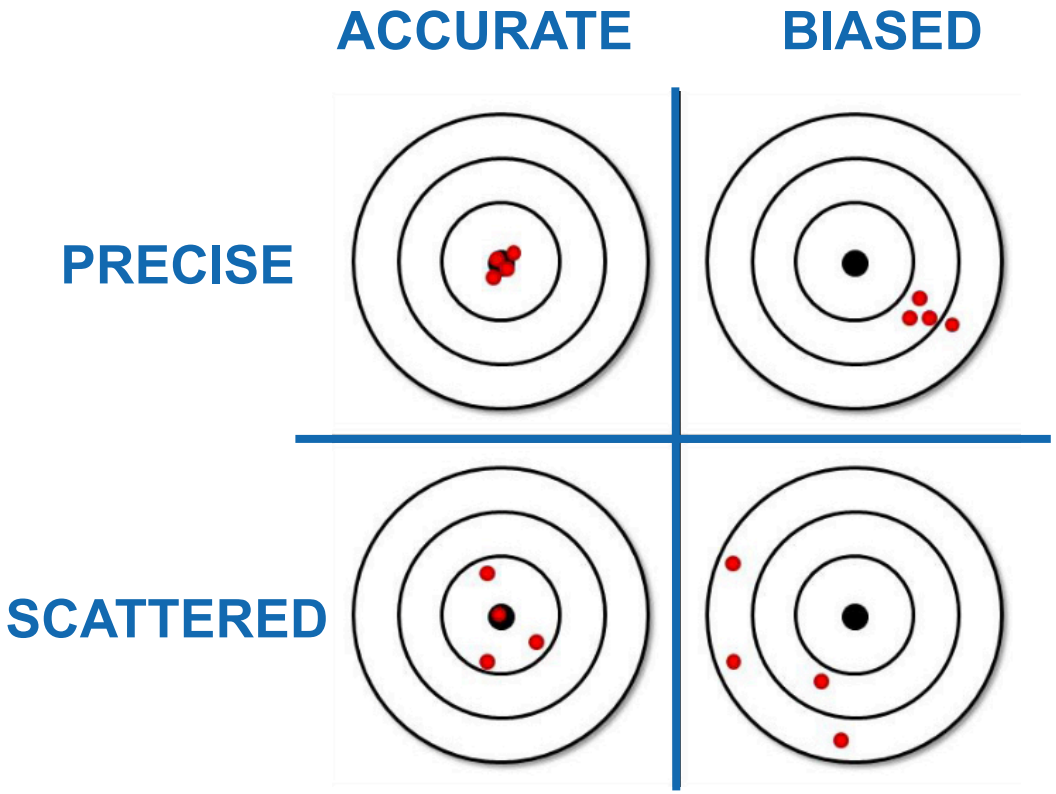
Intrusion Detection

Challenges and limitations of intrusion detection

- Unable to inspect encrypted traffic (encryption becomes widespread SSL/TLS, Skype, VPN, ..)
- High number of false positives
- Packet capturing and analysis at high link speed (tens of Gbit/s)
- Latency introduced by inspection engine
- Application level attacks (JavaScript, ..)
- Policy / signature management

Accuracy vs. Precision

For decision making (block / pass) the consistency of accuracy is most important



ACCURACY

How **close** the measured values are to the **target value**.

PRECISION

Values of repeated measurements are clustered and have **little scatter**.

"It is better to be roughly right than precisely wrong."
- John Maynard Keynes

If you get all measurements very precise, it does not mean that the measurements are close to the target value

Intrusion Detection / Testing

FACT

Attack / No attack

MEASURED *Detection, Alert*	Yes	No	
Yes	*a*	*b*	*a+b*
No	*c*	*d*	*c+d*
	a+c	*b+d*	*Total*

a **TRUE POSITIVE (TP)**
Alert on true intrusion

b **False Positive (FP)**
Alert on non-intrusion

c **False Negative (FN)**
No alert on true intrusion

d **TRUE NEGATIVE (TN)**
No alert on non-intrusion

Accuracy = (a+d) / Total

Firewall with Zero False Negatives

This firewall is guaranteed to block all attacks ..

Perfect Detection

The difficulty: Build a detector with optimal balance between FP and FN

0% FALSE **NEGATIVES**

```
bool is_attack(char *packet) {
  printf( "Alert: An attack!\n");
  return True;
}
```

0% FALSE **POSITIVES**

```
bool is_attack(char *packet) {
  printf( "All fine, no attack!\n");
  return False;
}
```

Costs of detection errors

- **FALSE POSITIVE**: Mobilize emergency response team, stop service, interrupt business, ..
- **FALSE NEGATIVE**: Getting compromised, forensics, downtime, cleanup

Base Rate Fallacy

Accurate detection is very challenging when rate of attacks is very low

Detector with FP rate of **0.1%** and FN rate of **2.0%**, with **5 attacks** per hour.

METRIC	SCENARIO 1	SCENARIO 2
Total Traffic packets / hour	1,000	
Good Traffic packets / hour	995	
Malicious Traffic packets / hour	5	
False Alerts (0.1%) False Positives / hour	**~ 1** (0.1% x 995)	
Missed Alerts (2.0%) False Negatives / hour	**~ 0.1** (2.0% x 5)	

Base Rate Fallacy

Accurate detection is very challenging when rate of attacks is very low

Detector with FP rate of **0.1%** and FN rate of **2.0%**, with **5 attacks** per hour.

METRIC	SCENARIO 1	SCENARIO 2
Total Traffic packets / hour	1,000	10,000,000
Good Traffic packets / hour	995	9,999,995
Malicious Traffic packets / hour	5	5
False Alerts (0.1%) False Positives / hour	**~ 1** (0.1% x 995)	**~ 10,000** (0.1% x 9,999,995)
Missed Alerts (2.0%) False Negatives / hour	**~ 0.1** (2.0% x 5)	**~ 0.1** (2.0 % x 5)

Note: Only the environment changed between scenarios

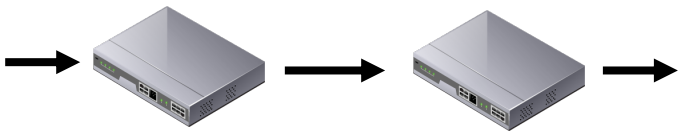
Multiple Detectors

Combining two independent detectors, do we get a better detector?

PARALLEL COMPOSITION

Alerting Logic

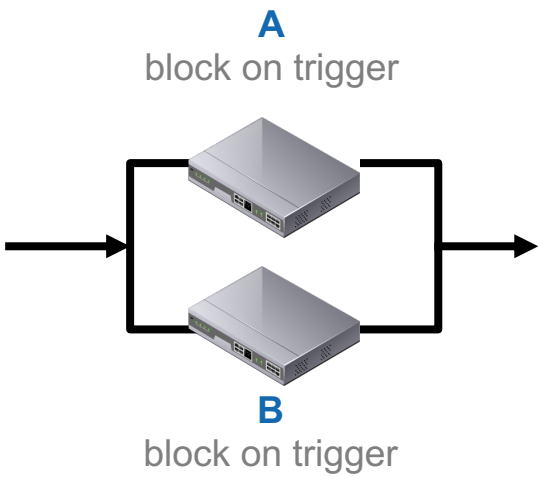
- Either detector A or B triggers alert
- `Alert = (A or B)`
- Reduced false negative rate
- Increased false positive rate



SERIAL COMPOSITION

Alerting Logic

- Both detectors A and B must trigger for alert
- `Alert = (A and B)`
- Increased false negative rate
- Decreased false positive rate



Detection Performance

A high number of false positives is a major challenge for detection systems

Accurate detection is very challenging when rate of attacks is very low

Different detection techniques achieve different sensitivity and specificity

Vendors claims to detect almost 100% of the samples are meaningless without indicating rate of false positives

- Use a combination of diverse tests to increase precision
- Context is important (host vs. network based detection)

Layered Security
Filtering & Protection

TYPES OF ATTACKS

Targeted Attacks vs. Opportunistic Attacks

TARGETED ATTACK

- Actors have **clearly defined objectives and targets** - *that they pursue consistently.*
- Usually with the backing of **considerable resources** - *finances, expertise, human resources, tools & materials*

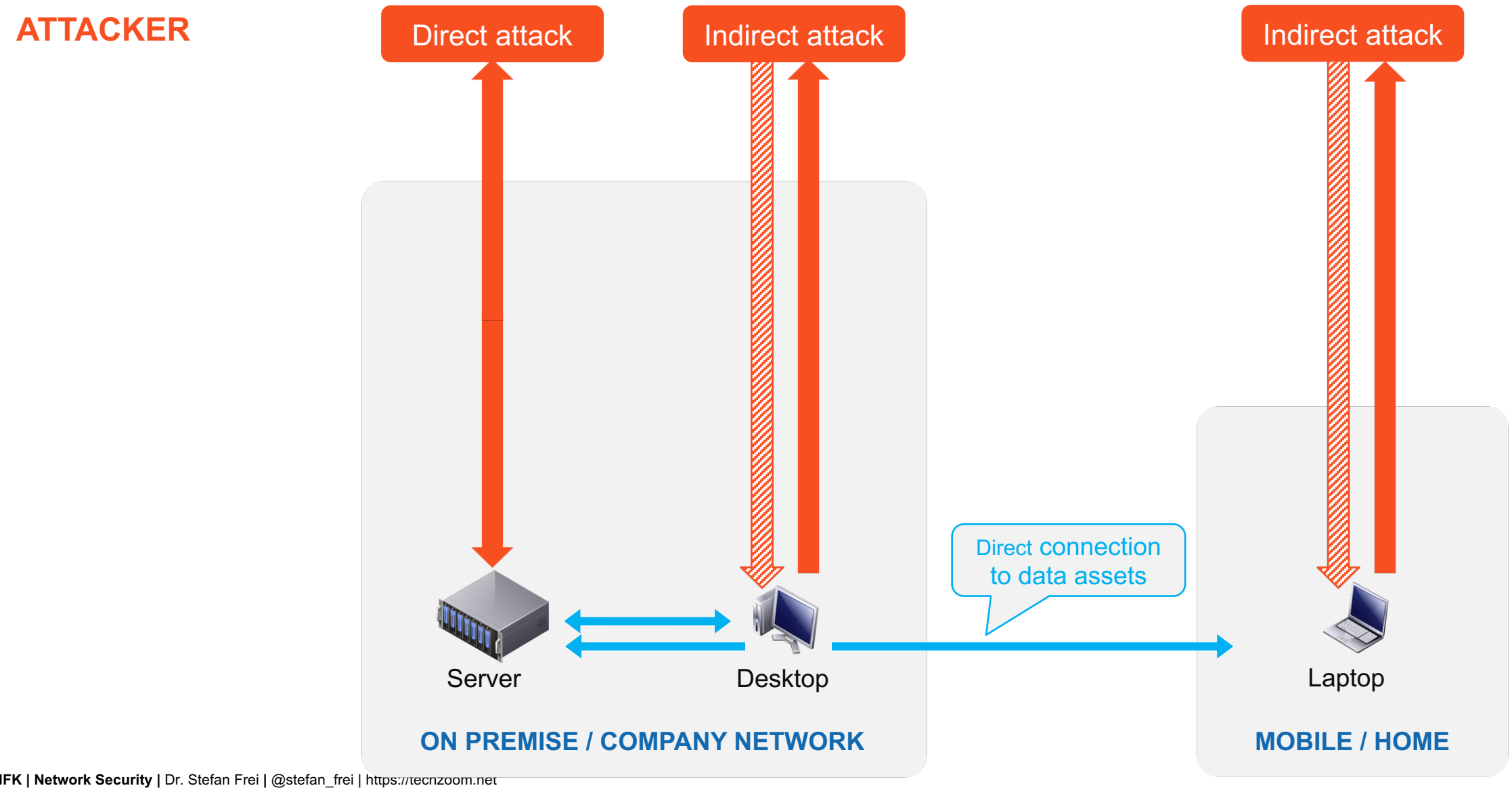
OPPORTUNISTIC ATTACK

- Actors **take opportunities** online - either by chance or because the target is not adequately protected.

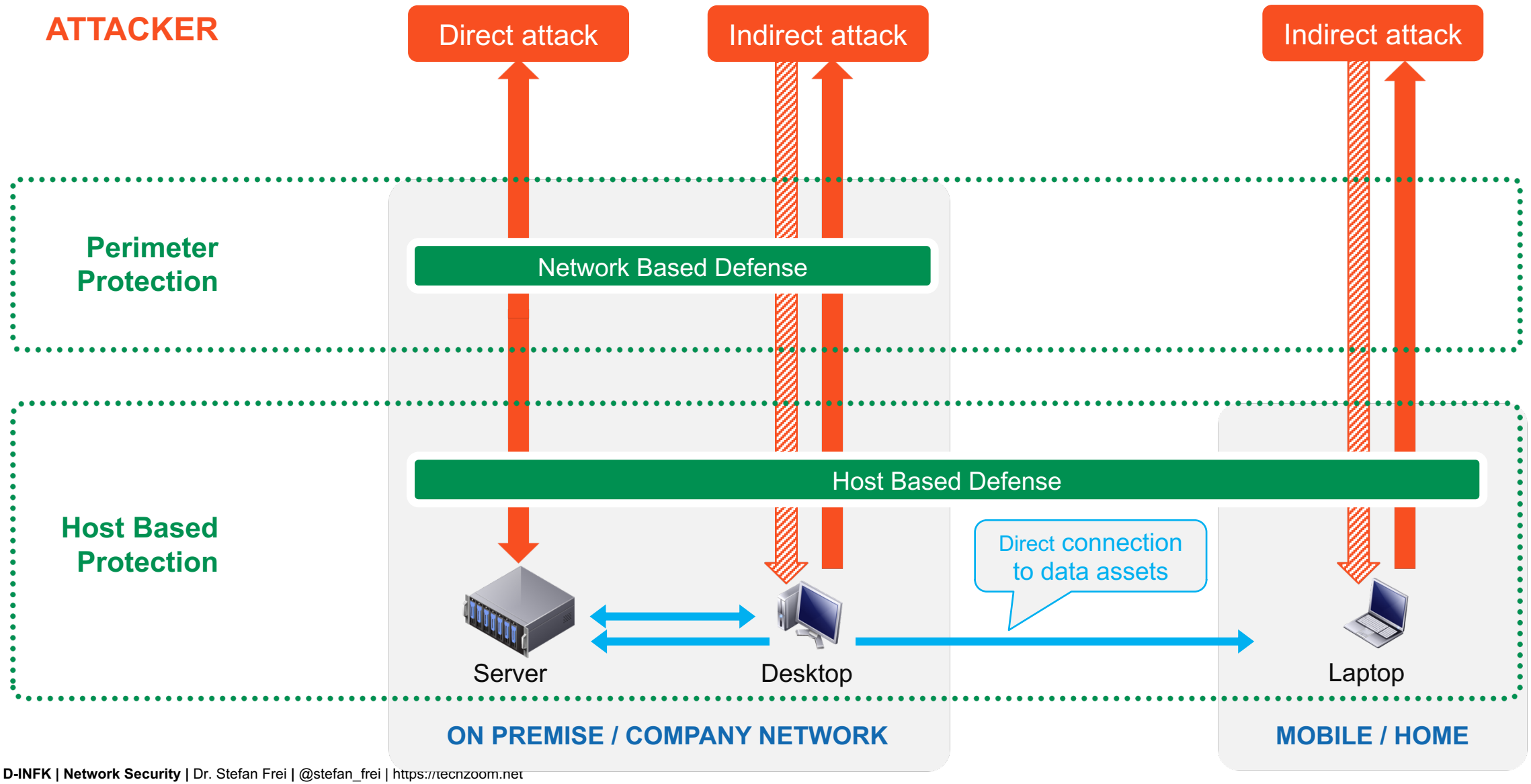
PERSISTENT ATTACK

- The attack is capable of **constantly increasing its penetration** - *of systems and resources.*
- Attackers pursue their goals **over a longer period of time** - *via multiple parallel channels.*
- **Reinfection** - *following failed or inadequate attempts*

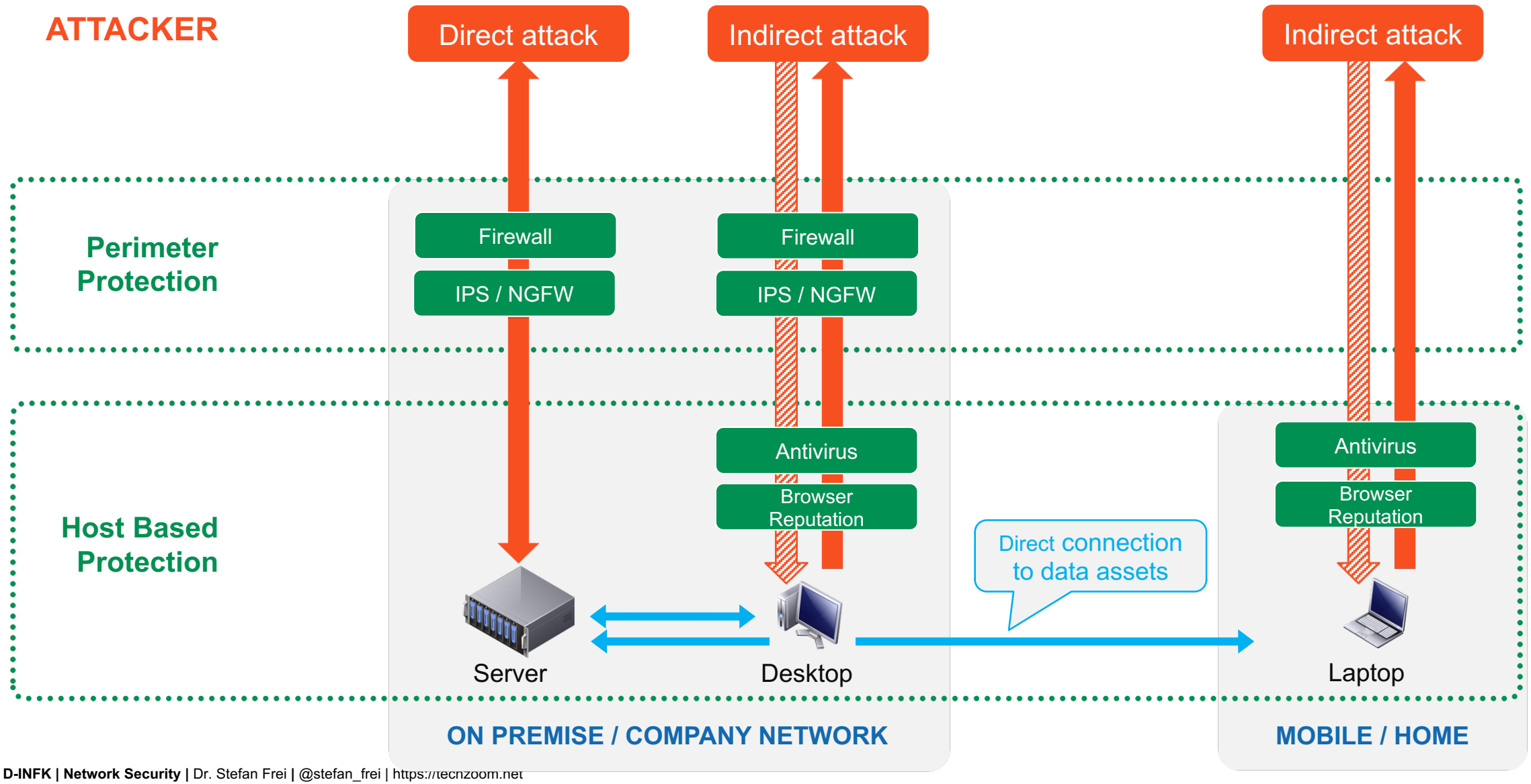
Attack types: Server vs. End-point Targets



Protection Layers – Network vs. Host Based Defense

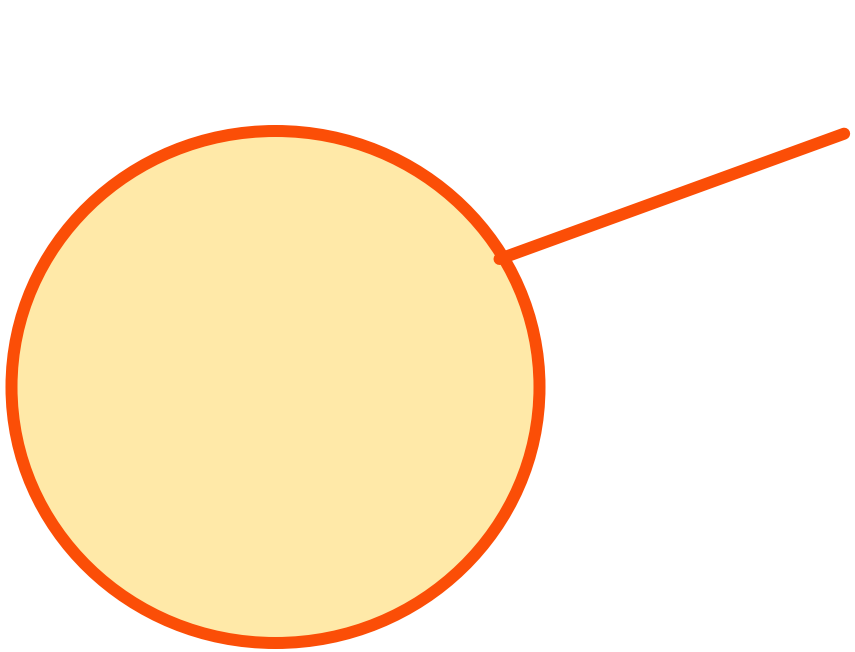


Protection Layers – On-Premise vs. Off-Premise



Level 1 – Some attacks always get through

Exploits and malware not detected



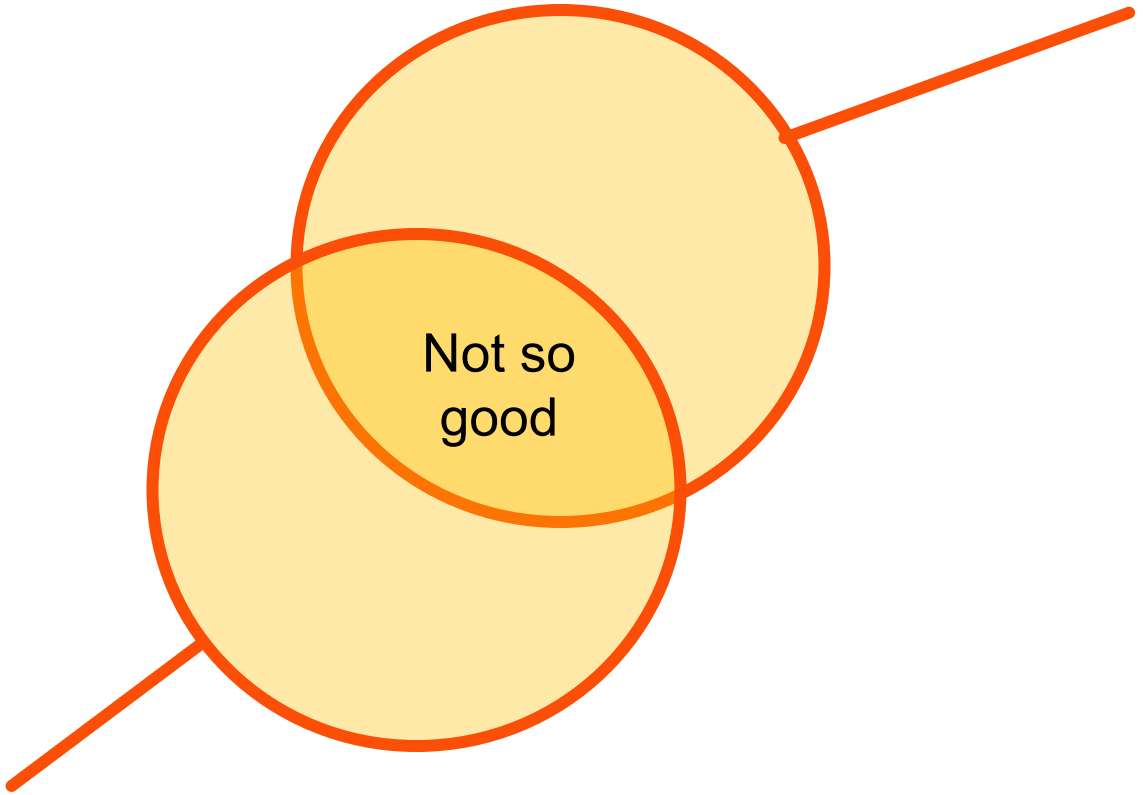
Exploits that bypass our defense layers (IPS, NGFW, Antivirus, ..)

Level 2 – Protecting popular programs

Expectation that industry would focus on protecting highly prevalent program

Exploits targeting popular programs with large market share

Prevalent & vulnerable programs



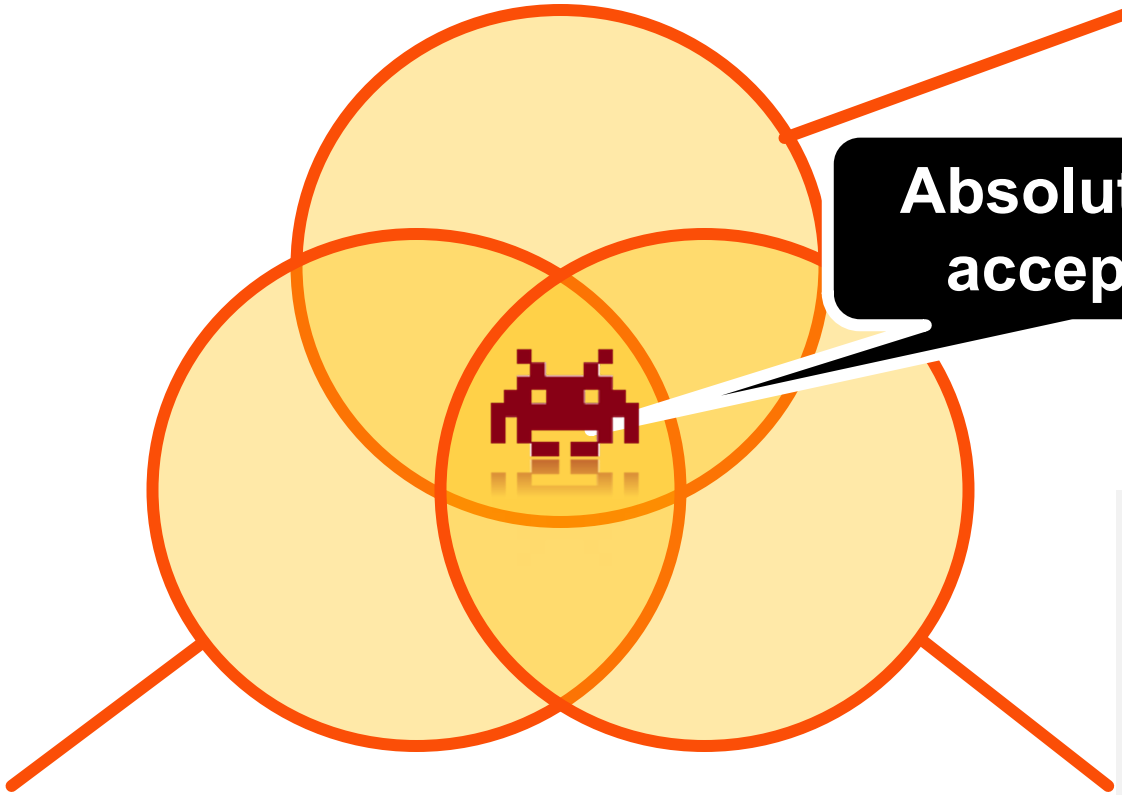
Exploits that bypass our defense layers (IPS, NGFW, Antivirus, ..)

Level 1 – Some attacks always get through

Old and known exploits successfully attacking popular programs !?

Exploits targeting popular programs with large market share

Prevalent & vulnerable programs



Exploits that bypass our defense layers (IPS, NGFW, ivirus, ..)

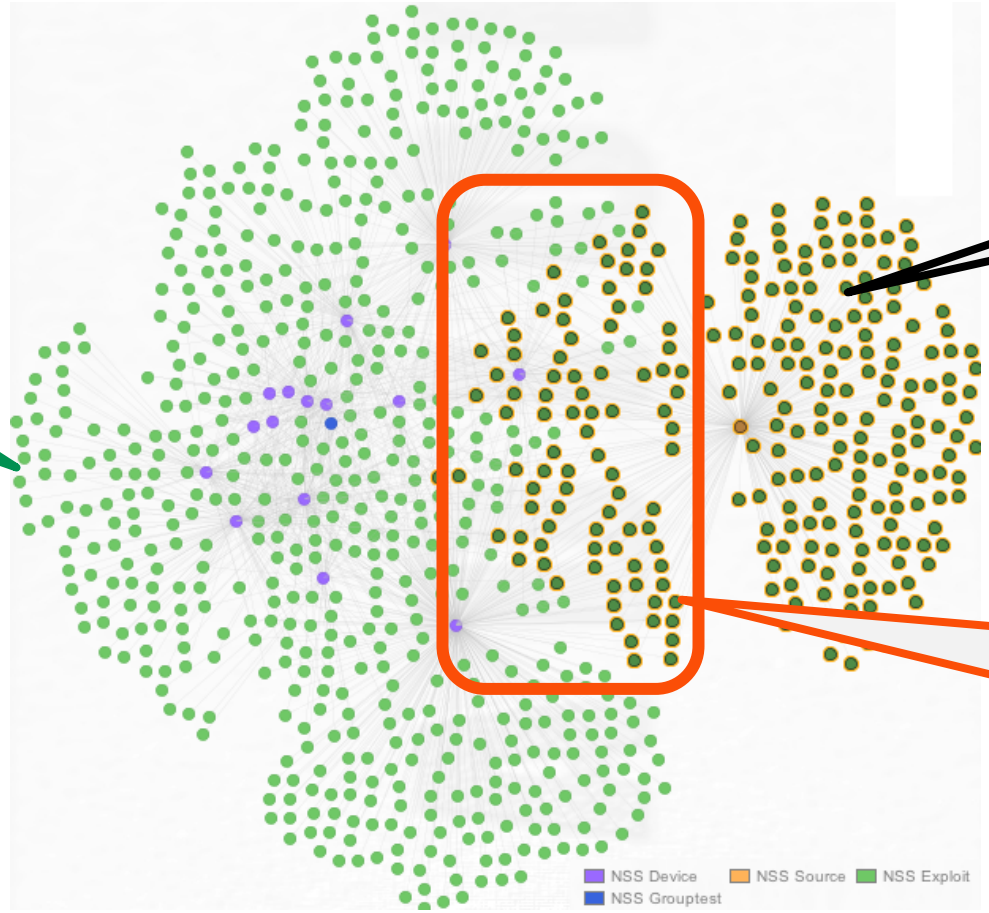
Absolutely not acceptable!

Exploits readily available in crimeware kits or penetration testing tools

Old exploits available in crimeware kits

Detection of known exploits

Known exploits freely available in the Metasploit testing tool bypass up-to-date, enterprise grade security engines.



Exploits not detected

Exploits in Metasploit

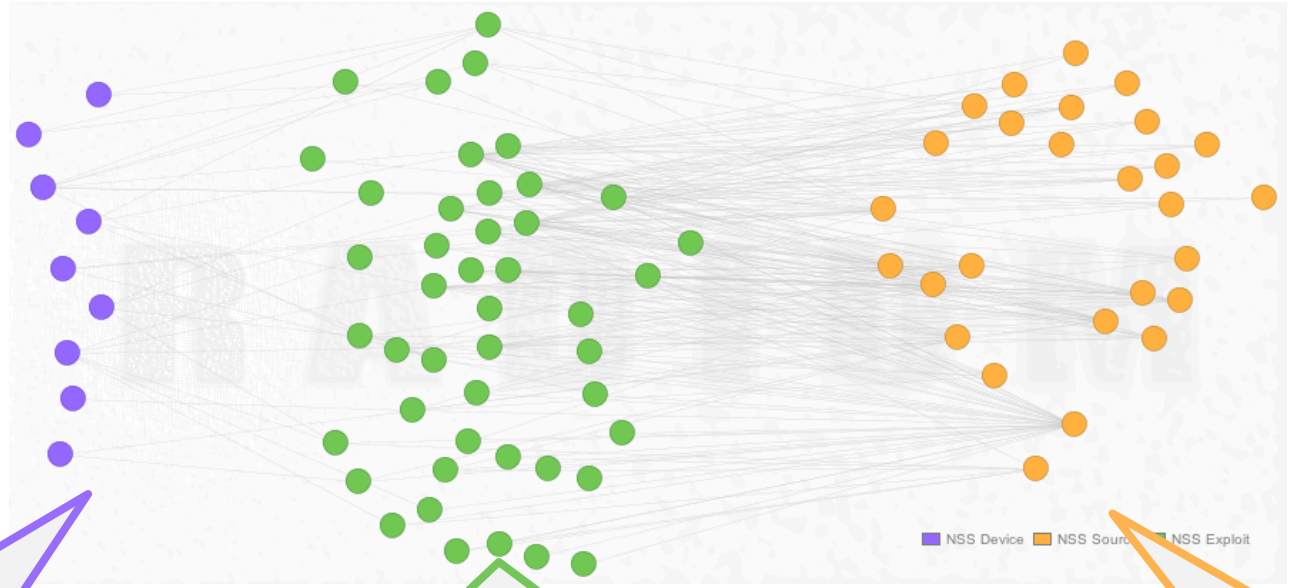
Exploits from Metasploit not detected

26% of 866 exploits from Metasploit not detected by at least one Intrusion Detection System (IPS) or Next Generation Firewall (NGFW)

Source: https://techzoom.net/whitepaper/correlation-of-detection-failures/

Exploits in Crimeware Kits

Detection of exploits available in well known and easily available crimeware kits bypass enterprise grade security engines.



IPS/NGFW devices that missed exploits (9/23)

Exploits of crimeware kits that are not detected (43)

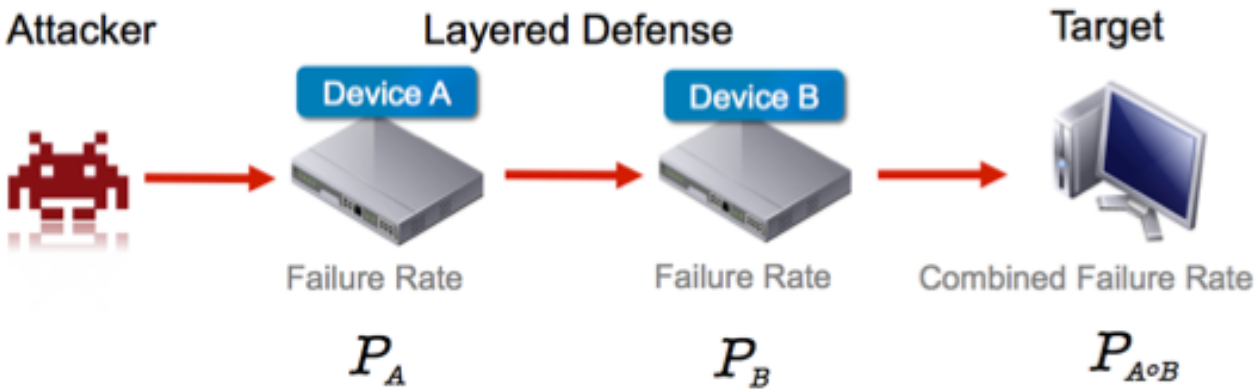
Crimeware kits (27)

Of 23 detection engines 9 were unable to detect 43 of 117 exploits attributed to popular crimeware kits.

Source: https://techzoom.net/whitepaper/correlation-of-detection-failures/

Multiple Detectors

Combining two independent detectors, do we get a better detector?



What is the combined failure rate (false negatives)?

$$P_{A \circ B} = P_A \cdot P_B \quad ?$$

Multiple Detectors

Combining two independent detectors, do we get a better detector?

Is the detection performance of difference security devices independent?

MEASURED

Measured Data

Group Test	Exploits total	Devices total	Devices max	Number of exploits bypassing X or more devices (measured) x=0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
EPP 2012	43	13	12	43	39	37	34	33	31	24	16	15	10	9	4	2	0			
IPS 2012	1,486	16	11	1,486	716	289	127	76	47	32	21	9	4	2	1	0	0	0	0	0
NGFW 2012	1,486	8	6	1,486	752	216	58	23	8	2	0	0								
NGFW 2013	1,711	9	9	1,711	392	102	41	23	18	9	8	8	8							

102 exploits not detected by two devices in series

MODEL

Model (Ignoring Correlation)

Group Test	Exploits total	Devices total	Devices max	Number of exploits bypassing X or more devices (model) 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
EPP 2012	43	13	10	43	43	43	42	39	34	25	16	8	3	1	0	0	0			
IPS 2012	1,486	16	5	1,486	892	331	83	15	2	0	0	0	0	0	0	0	0	0	0	0
NGFW 2012	1,486	8	4	1,486	782	230	42	5	0	0	0	0								
NGFW 2013	1,711	9	7	1,711	521	80	7	0	0	0	0	0	0							

model predicted 80 exploits not detected

Source: https://techzoom.net/whitepaper/correlation-of-detection-failures/

Cross Correlation of Detection Capability

Not a single combination of two Next Generation Firewall (NGFW) devices provided 100% protection.

- The best product missed **31 exploits**, the worst missed **155 exploits**
- The best combination of two products **missed 9 exploits**.

No combination of two products detected all exploits

Diagonal: Exploits missed by specific product

Exploits missed by both products

n1	product	Check Point 12600	Sonicwall SuperMassive e10800	Fortigate 3600c	Juniper SRX3600	Palo Alto PA5020	Sourcefire 8250	Sourcefire 8290	Stonesoft 3202	WatchGuard XTM 2050
1	Check Point 12600	**31**	10	13	11	15	9	9	9	16
2	Sonicwall SuperMassive e10800		**51**	16	12	16	11	11	17	13
3	Fortigate 3600c			**55**	22	18	15	15	13	17
4	Juniper SRX3600				**93**	17	13	13	16	16
5	Palo Alto PA5020					**67**	20	20	12	14
6	Sourcefire 8250						**45**	**45**	14	12
7	Sourcefire 8290							**45**	14	12
8	Stonesoft 3202								**67**	12
9	WatchGuard XTM 2050									**155**

Conclusion

Common Terms

Given the similarity between these systems there has been some convergence over time

FW	A **Firewall (FW)** is a device or application that analyzes network packet headers and enforces policy based on protocol type, source address, destination address, source port, and/or destination port. Packets that do not match policy are rejected.
IDS	An **Intrusion Detection System (IDS)** is a device or application that analyzes whole packets, both header and payload, looking for known events. When a known event is detected a log message is generated detailing the event. The IDS operates passively and does not block attacks.
IPS	An **Intrusion Protection System (IPS)** is a device or application that analyzes whole packets, both header and payload, looking for known events. When a known event is detected the packet is rejected. It is an inline IPS that can block attacks.
NGFW	A **Next-Generation Firewall (NGFW)** is a device that enforces a policy unilaterally across more than just network packet header information. It combines a traditional firewall with other network device filtering functions, such as application protocol awareness using in-line deep packet inspection (DPI) or virtual private networking (VPN). It acts as both a traditional Firewall and IPS,
WAF	A **Web Application Firewall (WAF)** filters, monitors, and blocks HTTP traffic to and from a web application. A WAF is differentiated from a regular firewall in that it is able to filter the content of specific web applications.

Take Home Message

- 100% protection is an illusion: **Assume you're already compromised**, and get compromised over again.
- A successful **breach should be planned for**, and handled in a **controlled process** (rather than being treated as an exception)
- Deploy tools and processes to **quickly detect and remediate successful breaches**.
- We need anti-virus, anti-malware, intrusion detection et al, but we also **need to know their limitations**.
- Most devices can achieve a high detection rate – at the **price of an unacceptably high rate of false positives**.
- An installed security patch/update is better than millions of detection signatures.