IS ADVERSARY SIMULATION A NEW PERSPECTIVE ON CYBERSECURITY ?

F.Baiardi
f.baiardi@unipi.it

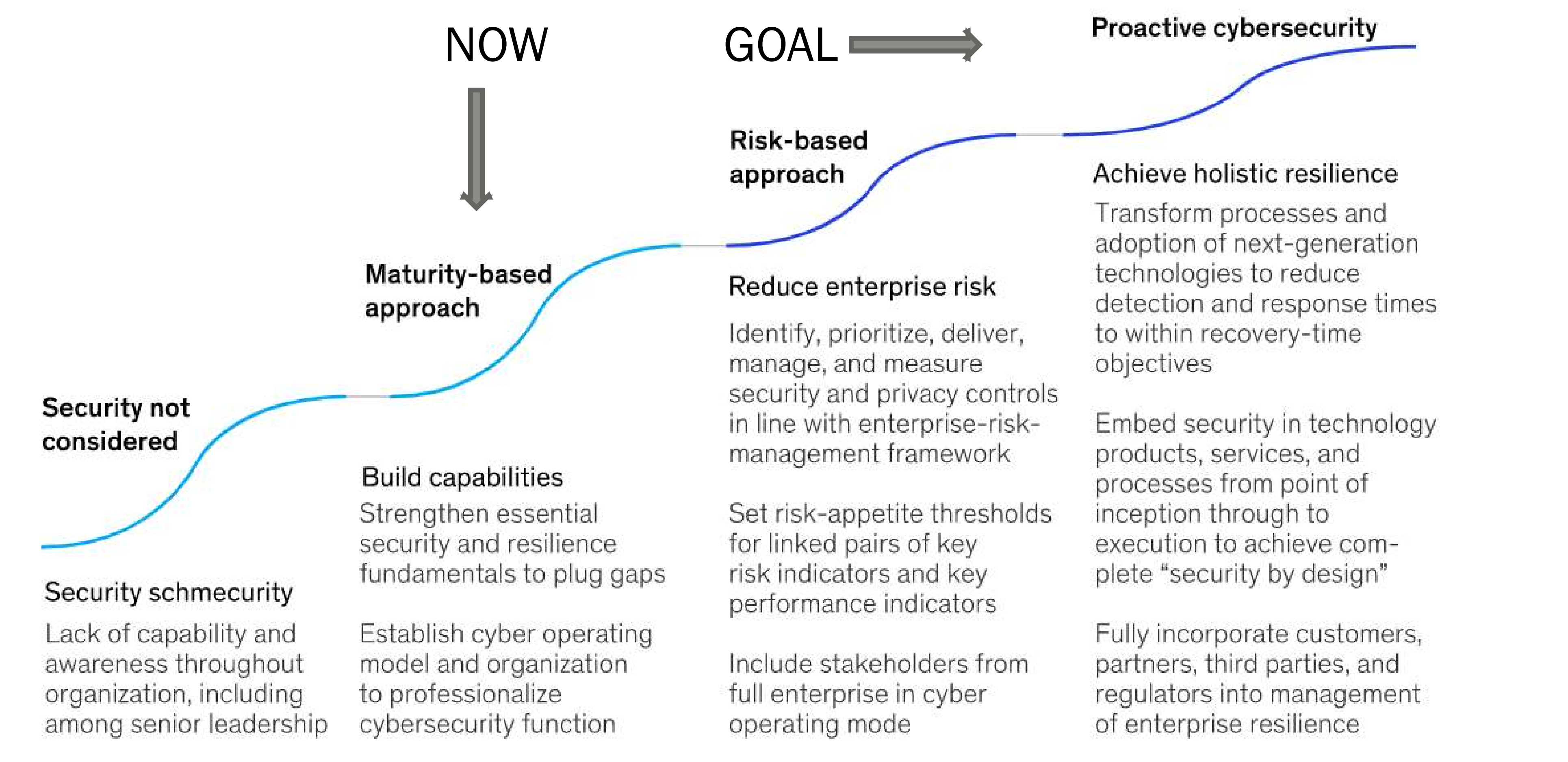
Security Engineering, Ross Anderson

One of the first things the security engineer needs to do when tackling a new problem is to identify the likely opponents. Although you can design some specific system components (such as cryptography) to resist all reasonable adversaries, the same is much less true for a complex real-world system.

You can't protect it against all possible threats and still expect it to do useful work at a reasonable cost. So ask yourself

- what sort of capabilities will the adversaries have?
- what motivation?
- how certain are you of this assessment?
- how might it change over the system's lifetime?

A new approach is required



NOW

GOAL

Security not considered

Security schmecurity

Lack of capability and awareness throughout organization, including among senior leadership

Maturity-based approach

Build capabilities

Strengthen essential security and resilience fundamentals to plug gaps

Establish cyber operating model and organization to professionalize cybersecurity function

Risk-based approach

Reduce enterprise risk

Identify, prioritize, deliver, manage, and measure security and privacy controls in line with enterprise-risk-management framework

Set risk-appetite thresholds for linked pairs of key risk indicators and key performance indicators

Include stakeholders from full enterprise in cyber operating mode

Proactive cybersecurity

Achieve holistic resilience

Transform processes and adoption of next-generation technologies to reduce detection and response times to within recovery-time objectives

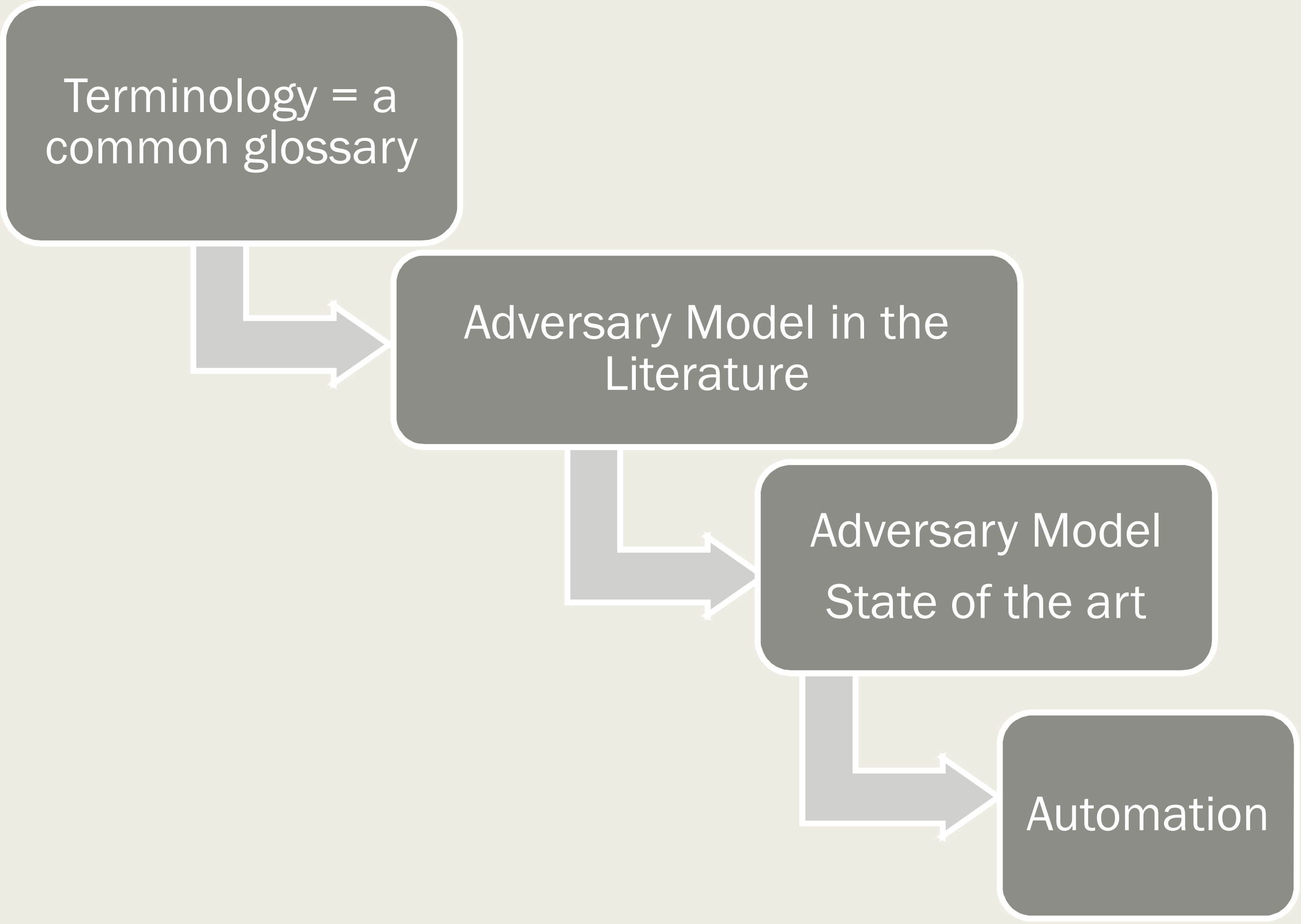
Embed security in technology products, services, and processes from point of inception through to execution to achieve complete “security by design”

Fully incorporate customers, partners, third parties, and regulators into management of enterprise resilience

Adversary simulation = simulation of an attack against the system of interest

- Understanding how a system can be attacked
- Understanding how to defeat the attack or to recover after the attack
- Evaluating robustness and resilience

Outline of this course subset



Terminology = a common glossary

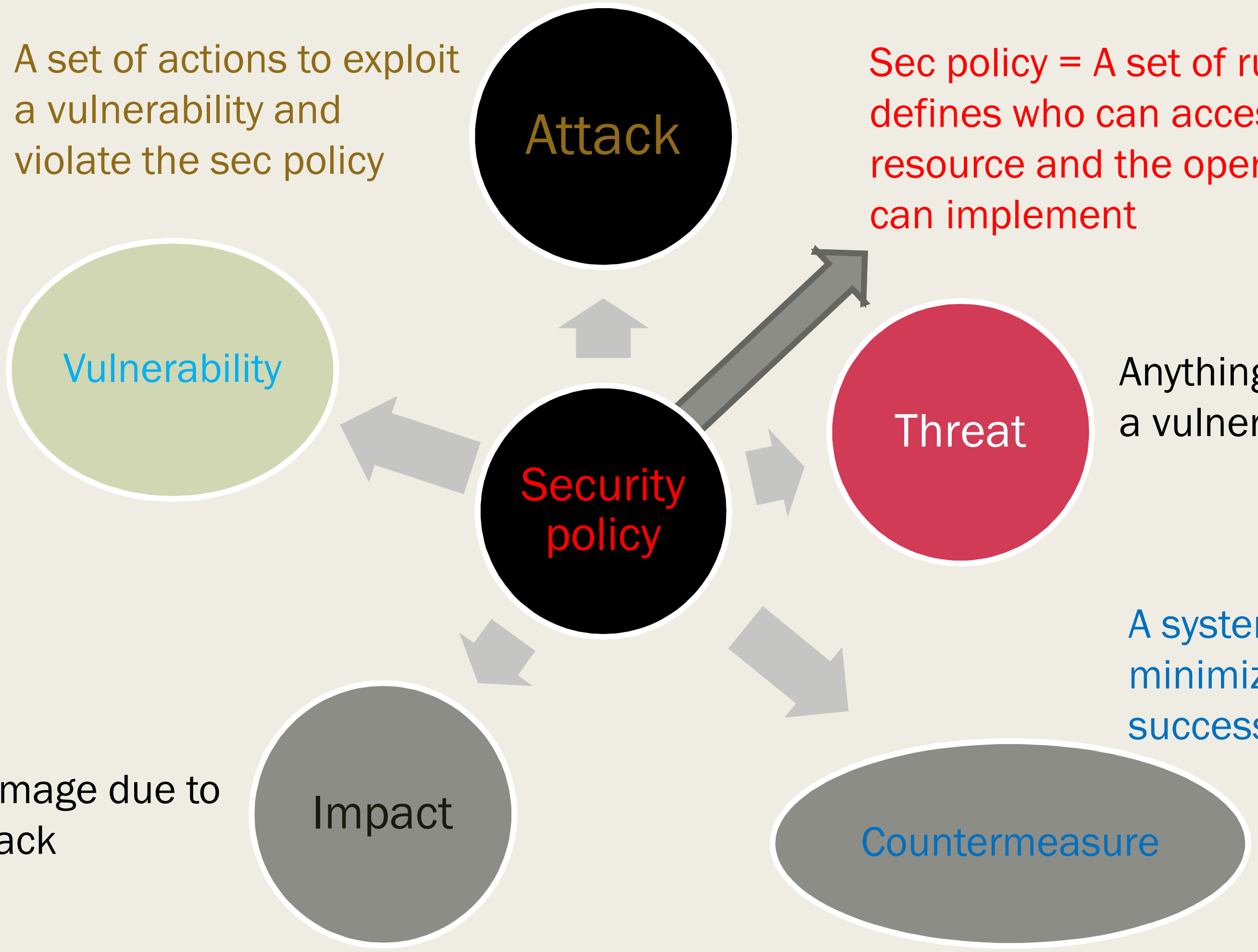
Adversary Model in the Literature

Adversary Model

State of the art

Automation

Terminology (basic concepts)



A set of actions to exploit a vulnerability and violate the sec policy

Sec policy = A set of rules that defines who can access a system resource and the operations he/she can implement

A defect in a system component or user that enables a sec policy violation

Anything that can expoit a vulnerability

A system change to minimize the attack success probability

The damage due to the attack

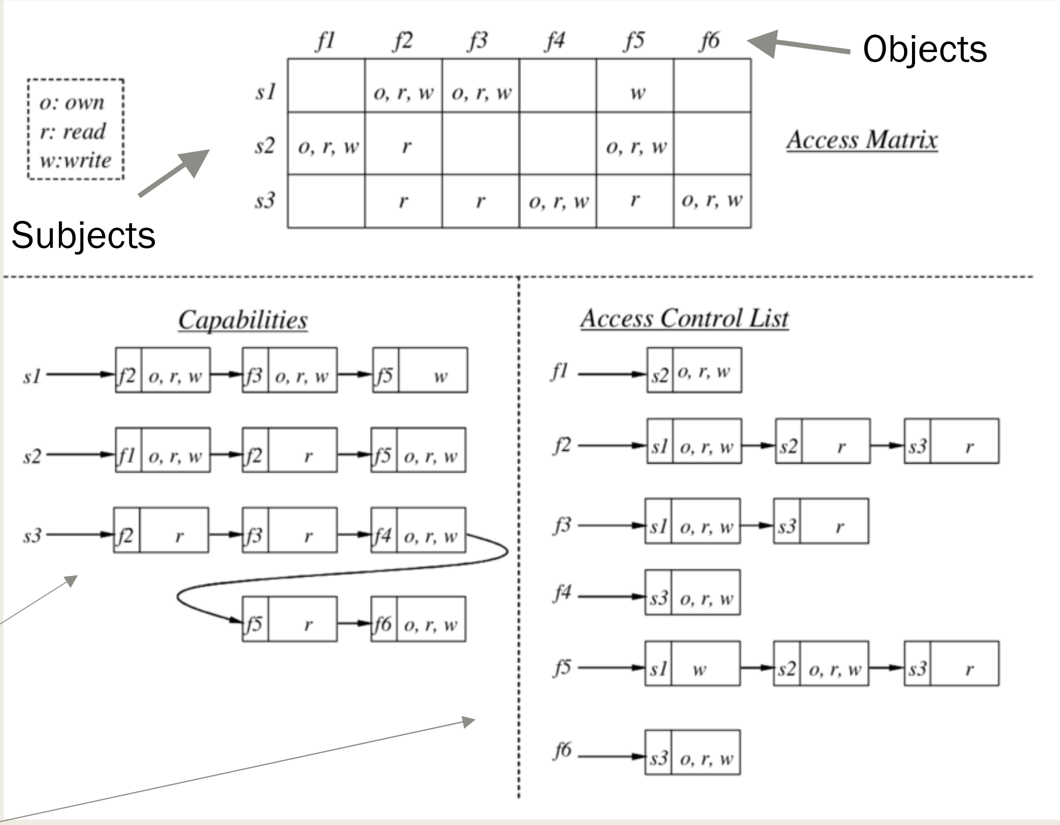
Security Policy

- Its goal is to preserve some basic properties against threats
  - *Confidentiality of information*
  - *Integrity of information and resources*
  - *Availability of information and resources*
- There are several other properties of interest
  - *Accountability*
  - *Privacy*
  - *....*
- But the CIA triad is the basis of all the other properties, any other property can be expressed as a combination of CIA for some resources
  - *Accountability requires the Integrity and the Availability of those resources that record how many other resources a user has accessed, how long it has accessed them, when …*
  - *Forensics requires the integrity of some log files*

Security Policy

- Alternative ways of expressing a security policy
  - *Role based*
  - *Attribute based*
  - *Mandatory access control*
  - *Discretionary access control*

  but they all define an access control matrix where each position defines the access rights of a user or subject (what the subject can do)



Objects
o: own
r: read
w:write
Subjects
Access Matrix

	f1	f2	f3	f4	f5	f6
s1		o, r, w	o, r, w		w	
s2	o, r, w	r			o, r, w	
s3		r	r	o, r, w	r	o, r, w

Capabilities
s1 → f2 o, r, w → f3 o, r, w → f5 w
s2 → f1 o, r, w → f2 r → f5 o, r, w
s3 → f2 r → f3 r → f4 o, r, w → f5 r → f6 o, r, w
Access Control List
f1 → s2 o, r, w
f2 → s1 o, r, w → s2 r → s3 r
f3 → s1 o, r, w → s3 r
f4 → s3 o, r, w
f5 → s1 w → s2 o, r, w → s3 r
f6 → s3 o, r, w

- The matrix can be stored as
  - *an array of rows or*
  - *an array of columns*
- A distinct matrix for each execution level (assembly, os, application …)
- Subject = users, accounts, processes, threats, instruction

Fundamental relation

- A security problem requires
  - *A system with some assets and a security policy (an acm)*
  - *Some vulnerabilities*
  - *Some one that is interested in exploiting the vulnerability (threat actors, threat agents, attackers)*
- If any of the three is missing, we do not have a security problem, we may have a safety problem but not a security one
- Those aiming to exploit the vulnerability are malicious ie intelligent and willing of attacking the system = of forcing a system behavior that violates the adopted security policy

Safety

- The ability of a system to resist to erroneous behaviors (random threats) of components
  - *An erroneous communication link*
  - *An erroneous memory module*
  - *…*
- The adversary chooses at random where the fault occurs
- A huge number of system have a
  - *very high safety, because of their natural redundancy*
  - *a very low security because a malicious threat actor can target a critical components*
- Network with a scale free topology (the rich will become richer and the poor will become poorer) are a good example because
  - *Most nodes have a single connection to some backbone nodes*
  - *there is an overwhelming probability that a fault will affect one of these nodes rather than the few critical ones with a large number of connections*

A malicious agent will target exactly a node with a large number of connections

The least privilege principle

- Minimize the number of not empty position of the matrix
- The most invoked and most neglected security principle
- It implies that the matrix is updated with a high frequency

  ⟺ access rights have to be dynamically managed
- A potential problem when a blockchain implements the acm and smart contracts add and remove access rights

Vulnerability

- A defect in
  - *A system module*
  - *A system user*
  - *A system protocol*
- Vulnerability = missing control
  - *On an input parameter*
  - *On a system resource*
  - *On a user identity*

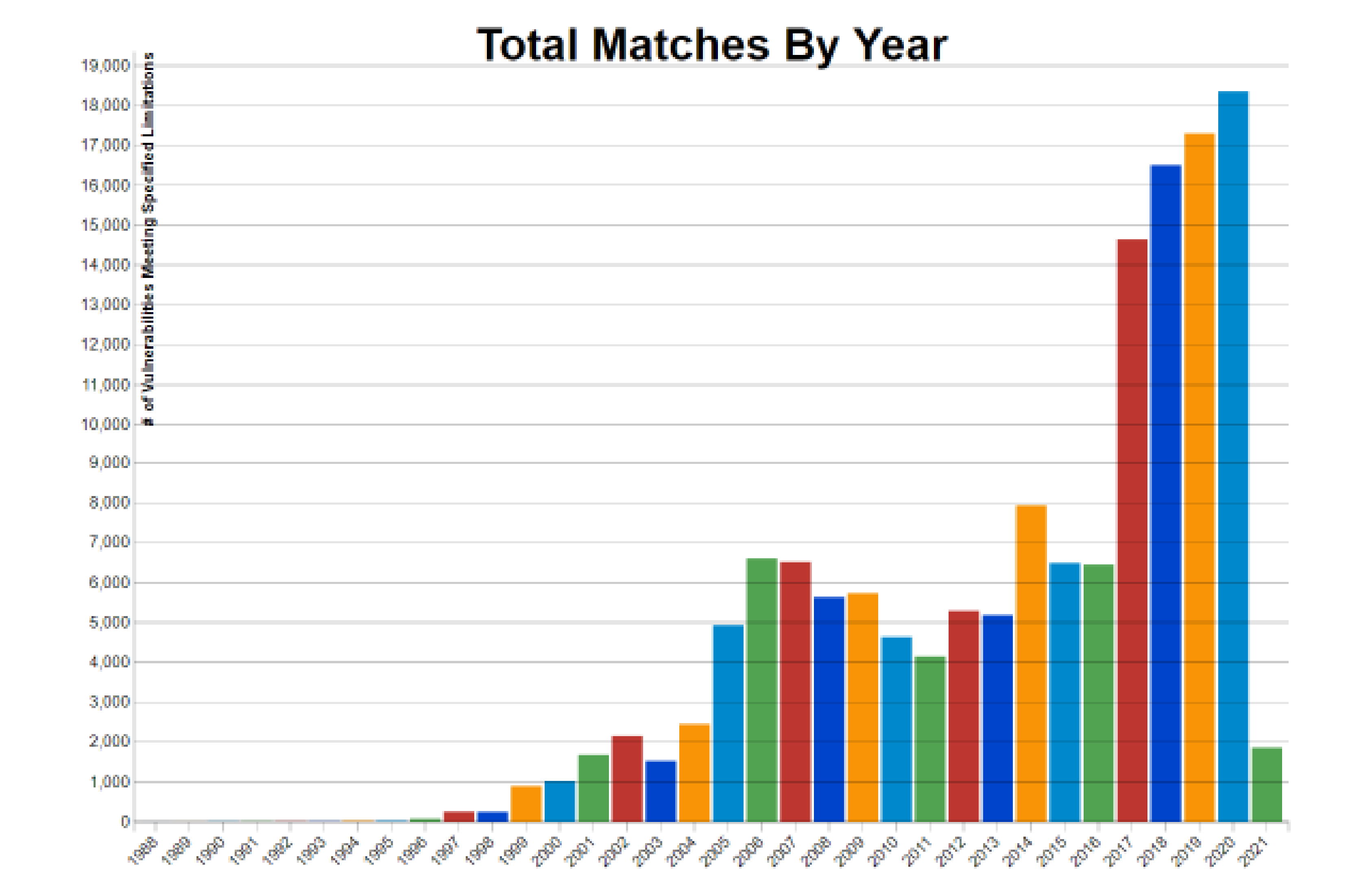
Vulnerability

MITRE and SANS classify vulnerabilities into three main classes

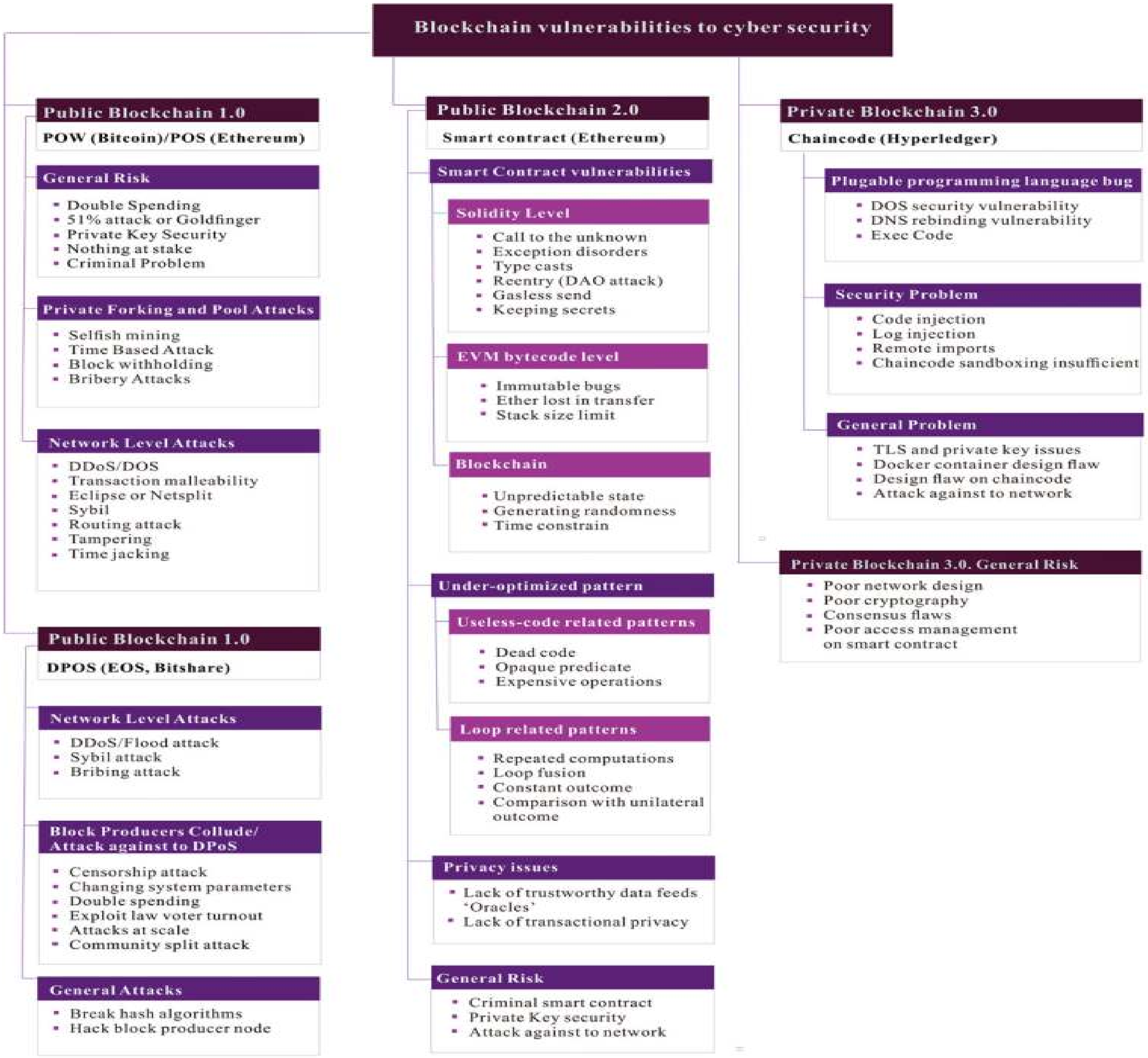
- Porous defenses
  - The misuse, abuse, bad use of defensive techniques such as encryption, authentication, and authorizationare.
- Risky resource management
  - Resource management creates, uses, transfers, and destroys system resources such as memory. These vulnerabilities range from Stack Overflow to Path Traversal
  - Two defense strategies
    - know what inputs you are using and whether they come from known “good” sources.
    - use those inputs properly for their intended purposes.
- Insecure interaction between components
  - SQL injection
  - Cross Site Scripting or Request Forging

All these vulnerabilities are related to how data is exchanged between separate components, modules, programs, processes, threads, or systems

Vulnerability



Blockchain Vulnerability



Blockchain vulnerabilities to cyber security

Public Blockchain 1.0

POW (Bitcoin)/POS (Ethereum)

General Risk

- Double Spending
- 51% attack or Goldfinger
- Private Key Security
- Nothing at stake
- Criminal Problem

Private Forking and Pool Attacks

- Selfish mining
- Time Based Attack
- Block withholding
- Bribery Attacks

Network Level Attacks

- DDoS/DOS
- Transaction malleability
- Eclipse or Netsplit
- Sybil
- Routing attack
- Tampering
- Time jacking

Public Blockchain 1.0

DPOS (EOS, Bitshare)

Network Level Attacks

- DDoS/Flood attack
- Sybil attack
- Bribing attack

Block Producers Collude/ Attack against to DPoS

- Censorship attack
- Changing system parameters
- Double spending
- Exploit law voter turnout
- Attacks at scale
- Community split attack

General Attacks

- Break hash algorithms
- Hack block producer node

Public Blockchain 2.0

Smart contract (Ethereum)

Smart Contract vulnerabilities

Solidity Level

- Call to the unknown
- Exception disorders
- Type casts
- Reentry (DAO attack)
- Gasless send
- Keeping secrets

EVM bytecode level

- Immutable bugs
- Ether lost in transfer
- Stack size limit

Blockchain

- Unpredictable state
- Generating randomness
- Time constrain

Under-optimized pattern

Useless-code related patterns

- Dead code
- Opaque predicate
- Expensive operations

Loop related patterns

- Repeated computations
- Loop fusion
- Constant outcome
- Comparison with unilateral outcome

Privacy issues

- Lack of trustworthy data feeds 'Oracles'
- Lack of transactional privacy

General Risk

- Criminal smart contract
- Private Key security
- Attack against to network

Private Blockchain 3.0

Chaincode (Hyperledger)

Plugable programming language bug

- DOS security vulnerability
- DNS rebinding vulnerability
- Exec Code

Security Problem

- Code injection
- Log injection
- Remote imports
- Chaincode sandboxing insufficient

General Problem

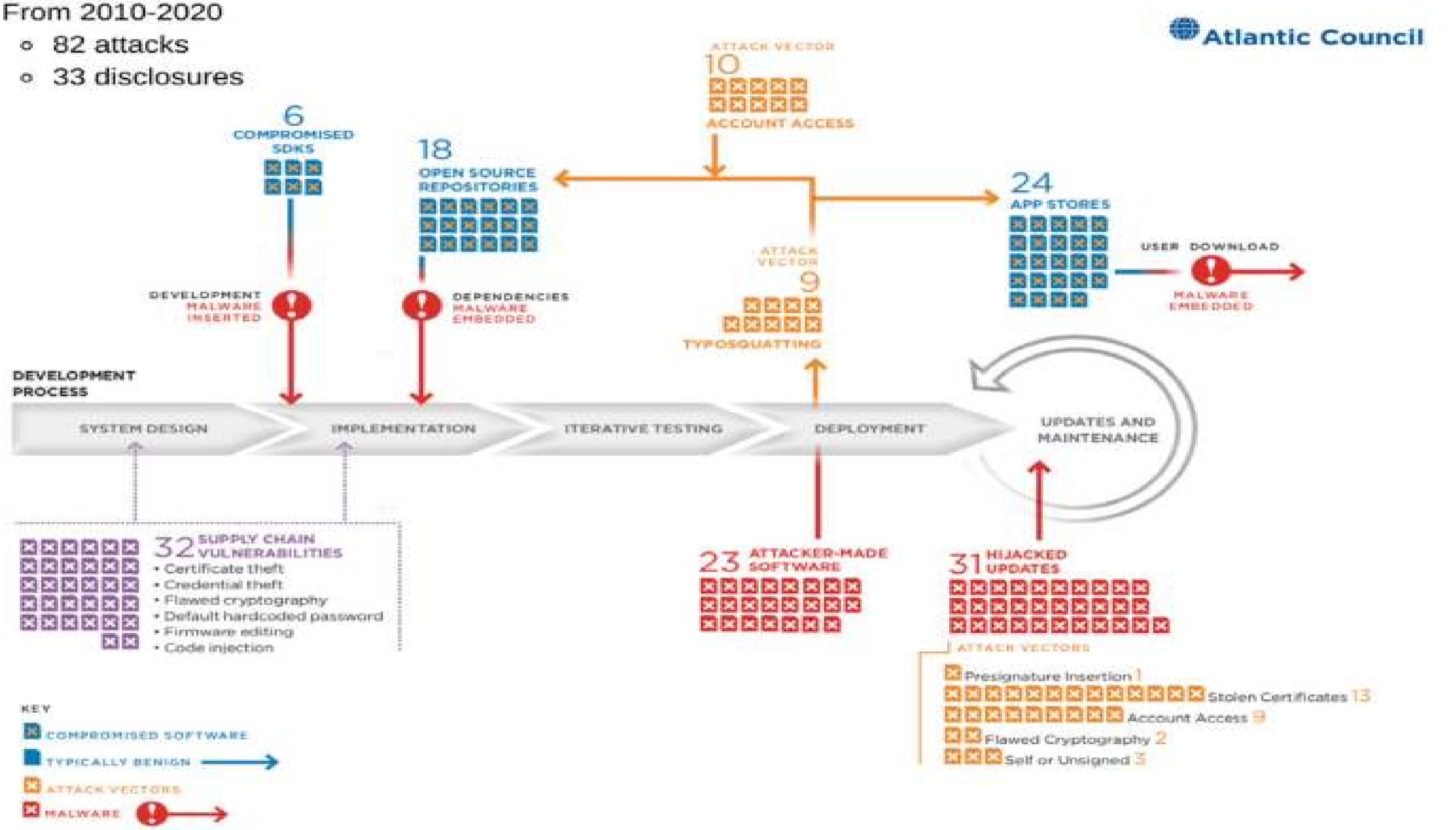
- TLS and private key issues
- Docker container design flaw
- Design flaw on chaincode
- Attack against to network

Private Blockchain 3.0. General Risk

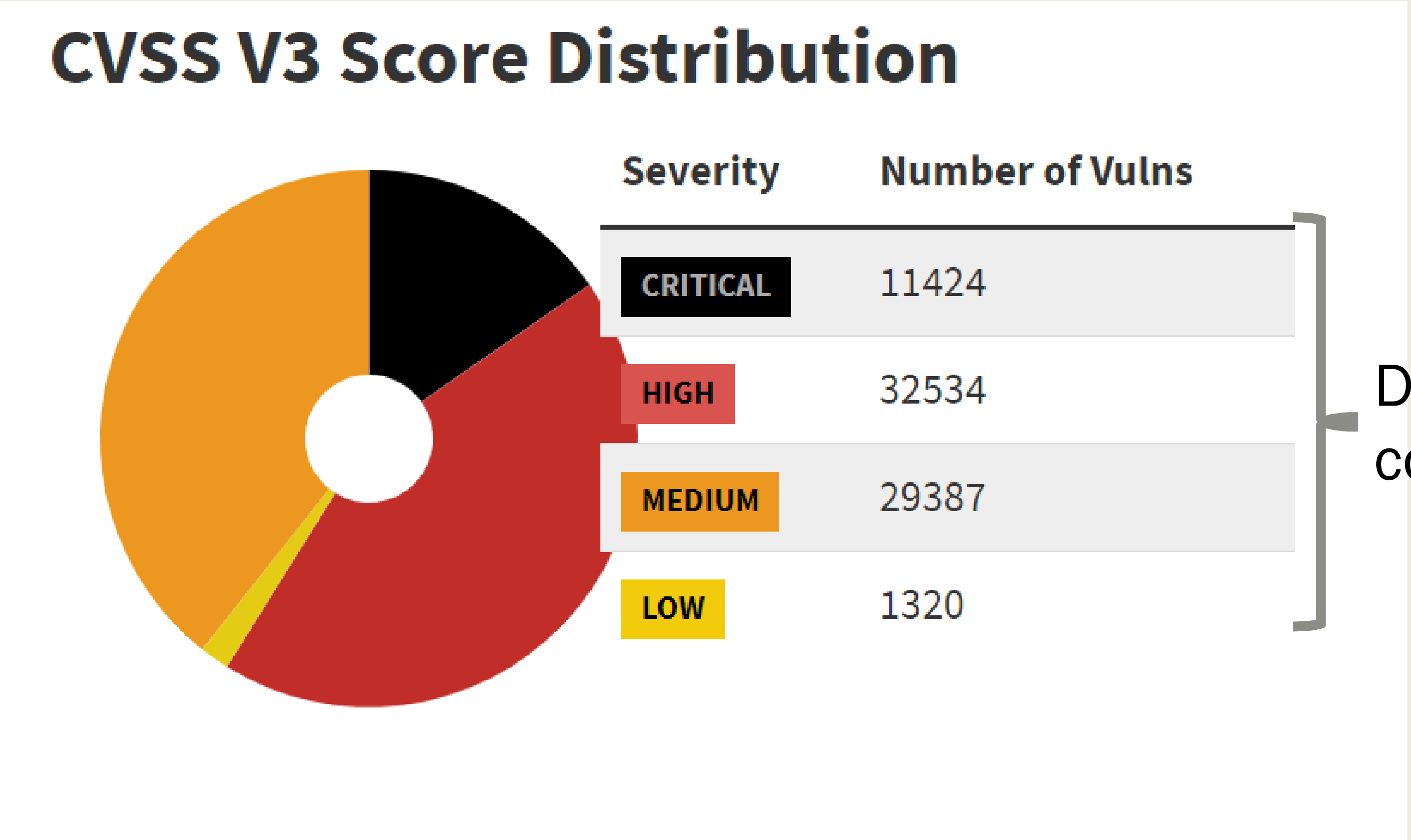
- Poor network design
- Poor cryptography
- Consensus flaws
- Poor access management on smart contract

Supply Chain Vulnerabilities

- From 2010-2020
  - 82 attacks
  - 33 disclosures



Vulnerability ranking



CVSS V3 Score Distribution

Severity	Number of Vulns
CRITICAL	11424
HIGH	32534
MEDIUM	29387
LOW	1320

Distribution not normal
consistency problems

All the versions of Common Vulnerability Scoring Sytem are contex independent

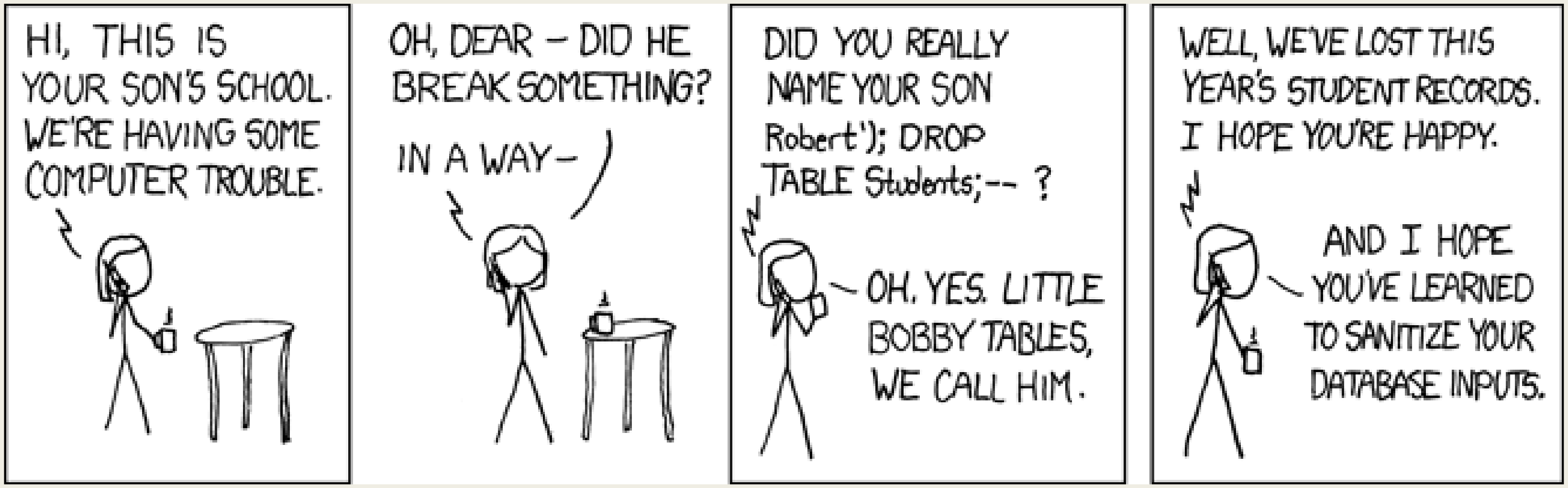
Attack (an elementary attack)

Enabled by a vulnerability

- Stack overflow. It transmits to a procedure a parameter that is so large that
  - *It overwrites (overflow) values on the system stack*
  - *It includes a program (code injection)*
  - *The stack overflow updates the return address on the stack so that it points to the program in the parameter stored in the stack*
  - *The injected program is executed with the rights of the invoked procedure*
- This attack shares some important features with a lot of other attacks
  - *A missing control*
  - *Code injection = the program is expecting a data but receives some instructions*
  - *Writing the parameters may be complex but once it has been written all the os that runs that procedure can be attacked*

Attack (an elementary attack)

- SQL injection


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

- Again
  - *A missing control*
  - *Code injection = the program is expecting a data but receives some instructions*
  - *Writing the parameters may be complex but once it has been written all the web server that runs that procedure can be attacked*

Attack (an elementary attack)

- Enabled by a vulnerability
- Important properties of an attack for adversary emulations are
  - *the resources, the information, the abilities it requires*
  - *the access right it requires*
    - Remote attack : worms and ransomware (SQL injection)
    - Local attack: (stack overflow)
  - *The access rights it grants if successful*
  - *The success probability*
  - *The execution time*
  - *The noise it produces*
- These properties determine who can execute the attack, when, why …
- Attack automation = a piece of code (= exploit) that can implement the attack
- Attack automation is strongly related to attack platform = an ICT platform that can replace the attacker

Countermeasure

A system update to remove or mitigate a vulnerability.

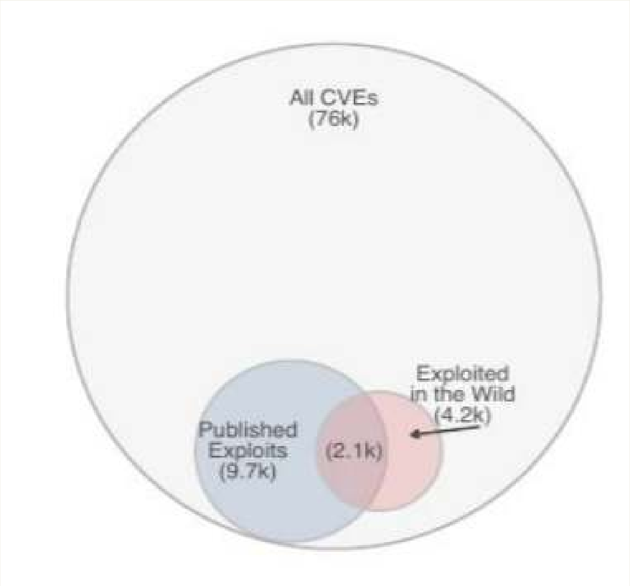
- Static countermeasures are permanent updates to the system
- Dynamic countermeasures are temporary changes while the system is under attack

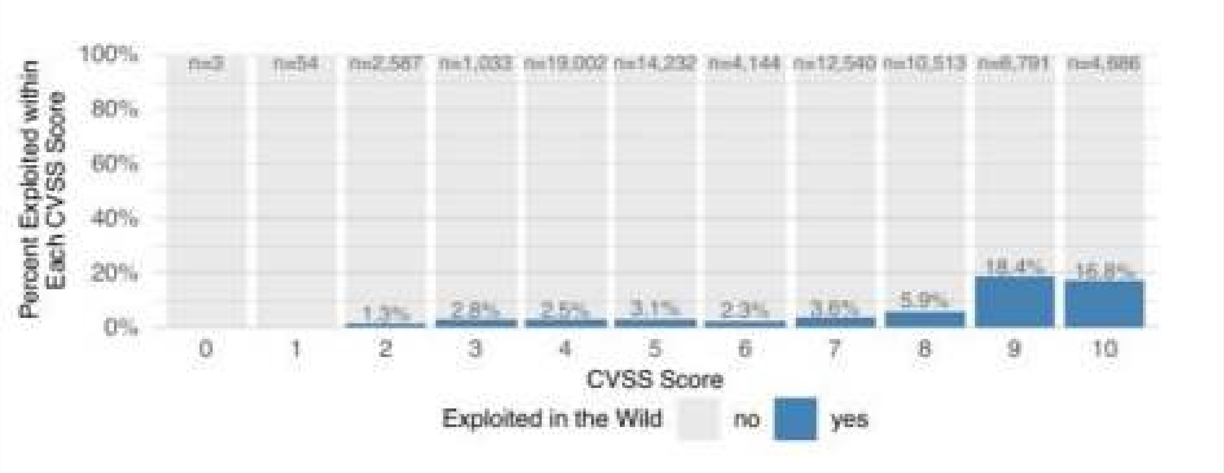
Possible countermeasures :

- Control on some input parameters (static)
- Filtering out messages to prevent the attacker from reaching a vulnerable component (static/dynamic)
- Shutting down a connection or a component (dynamic)
- Segmentation = Partition a network into subnetworks + filtering (static)
- Micro segmentation = Map applications onto distinct VMs on the same node (static)
- Sandboxing an application = Run an application in a protected environment (static)
- Hardening = remove useless functionalities from an environment (static/dynamic)
- Moving target defense = VM migration to hide their presence (dynamic)

Vulns and countermeasures: key numbers

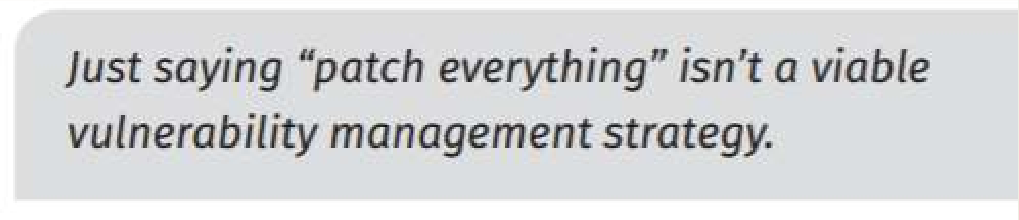
Improving Vulnerability Remediation Through Better Exploit Prediction, WEIS 2019





Exploited vulns as a function of dangers

Exploited vulns



Just saying "patch everything" isn't a viable vulnerability management strategy.

Do you remember Ross Anderson? You cannot protect ....

How to find vulnerabilities ?

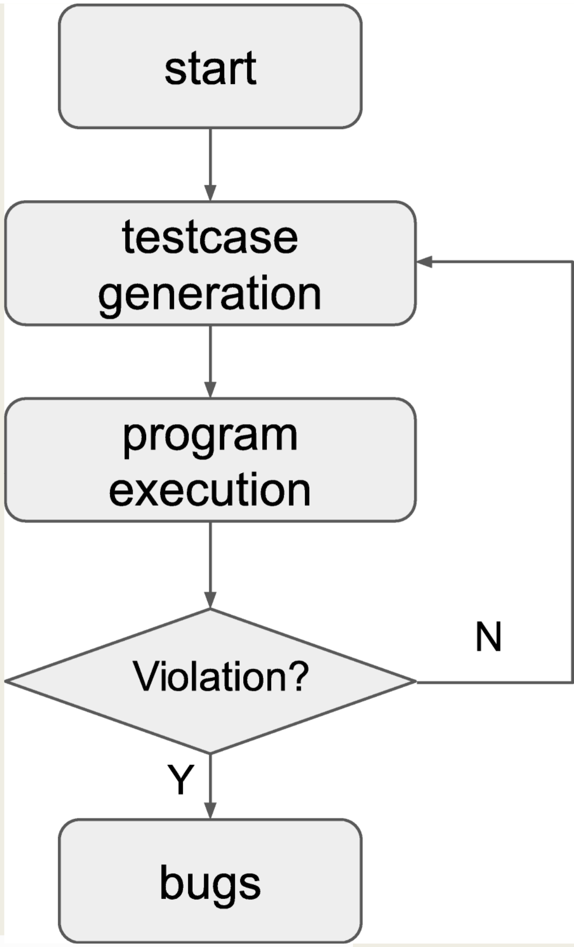
- According to Google and Microsoft, more than 80% of vulnerabilities can be find by fuzzing, an automated method first proposed by Barton in 1990s,
- A fuzzing test that generates massive normal and abnormal inputs to target applications and detects exceptions by feeding the generated inputs to the target applications and monitoring the execution states
- Fuzzing
  - is easy to deploy, of good extensibility and applicability
  - can be performed with or without the source code.
  - gains a high accuracy because is performed in the real execution
  - requires few knowledge of target applications and could be easily scaled up to large scale applications.
  - low efficiency and low code coverage
  - the most effective and efficient state-of-the-art discovery technique.

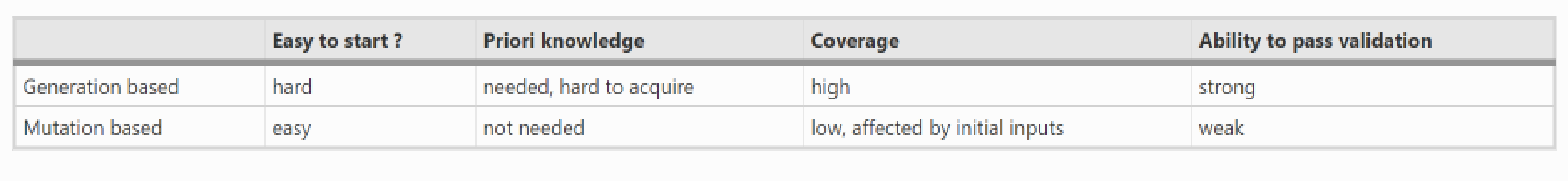
Fuzzing

A fuzzer can be

- generation-based if inputs are generated from scratch according to the specs
- mutation based by modifying existing inputs.
- dumb or smart depending on whether it is aware of input structure
- white-, grey-, or black-box, depending on whether it is aware of program structure.

Usually integrated with a taint analysis (which instructions receive an input)





	Easy to start ?	Priori knowledge	Coverage	Ability to pass validation
Generation based	hard	needed, hard to acquire	high	strong
Mutation based	easy	not needed	low, affected by initial inputs	weak

Threat and threat actors

- An event that may result in the violation of the system security policy
- The threat event is due to the action of an agent, the threat actors
- The threat actor exploits some system vulnerabilities
- In security the threat actor is aware of its actions and willing to violate the security policy to reach a goal
- Each intelligent threat actor
  - *Has a goal*
  - *Needs a sequence of actions (=an intrusion) to reach the goal,*
  - *Some but not all its actions in an intrusion are attacks*
  - *The attacks in an intrusion exploit distinct vulnerabilities*

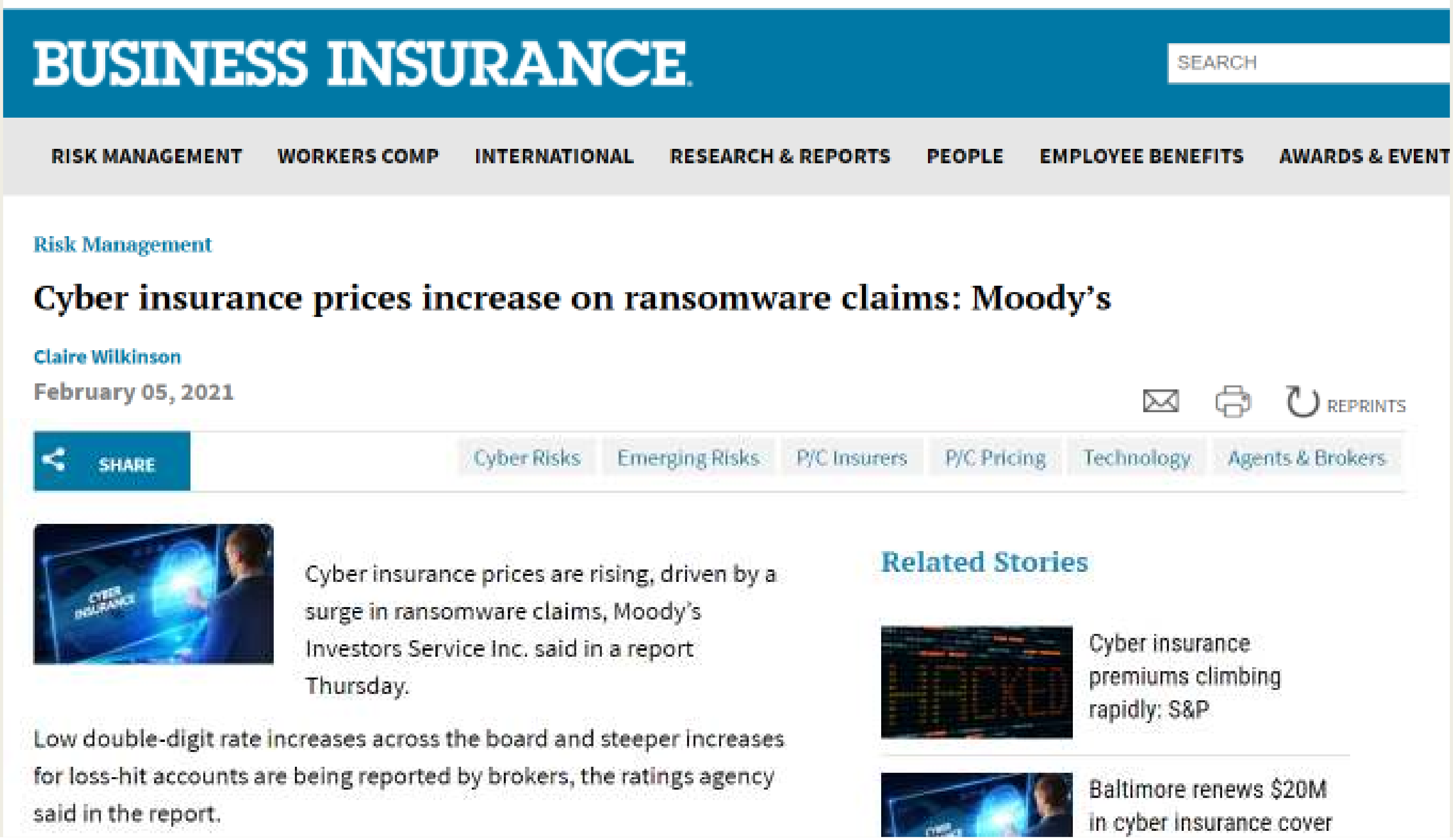
Threat actor – Properties

- The goal of an attacker = which are the access rights and the resources it aims to acquire
- An elementary attack is a sequence of actions exploiting a single vulnerabilitiy
- An intrusion (complex attack, an attack chain) is a sequence of elementary attacks to escalate the actor privileges to acquire the access rights in its goal. The first attack in an intrusion targets the attack surface
- Attack surface = the component affected by the first vulnerability the actor can exploit in an intrusion. It depends upon
  - *Attacker access rights*
  - *The system modules the attacker can access*
  - *Insiders vs outsiders*
- An intrusion includes not only several attacks but even other actions that are not attacks but make it possible to execute the attacks (more in the following)

Some threat actors

- Organized crime = Criminal environment, a supply chain with specialized groups
  - *Discovering vulnerability*
  - *Developing attack platform and exploit*
  - *Developing ransomware*
  - *Deploying ransomware and collecting ransom*
- Competitor
- Hacktivism
- State actor
  - *PI exfiltration = espionage*
  - *Infrastructure exploitation = hiding malware for future attacks*
- Ransomware
  - *automated software to attack a target using a predefined set of exploits for remote attacks*
  - *Previously not targeted attack (attack worm with a random spreading with the goal of attacking the largest number of nodes)*
  - *Currently shifting toward targeted attacks where information is exfiltrated before encryption*

Threat actor



Risk Management

Cyber insurance prices increase on ransomware claims: Moody's

Claire Wilkinson

February 05, 2021

Cyber insurance prices are rising, driven by a surge in ransomware claims, Moody's Investors Service Inc. said in a report Thursday.

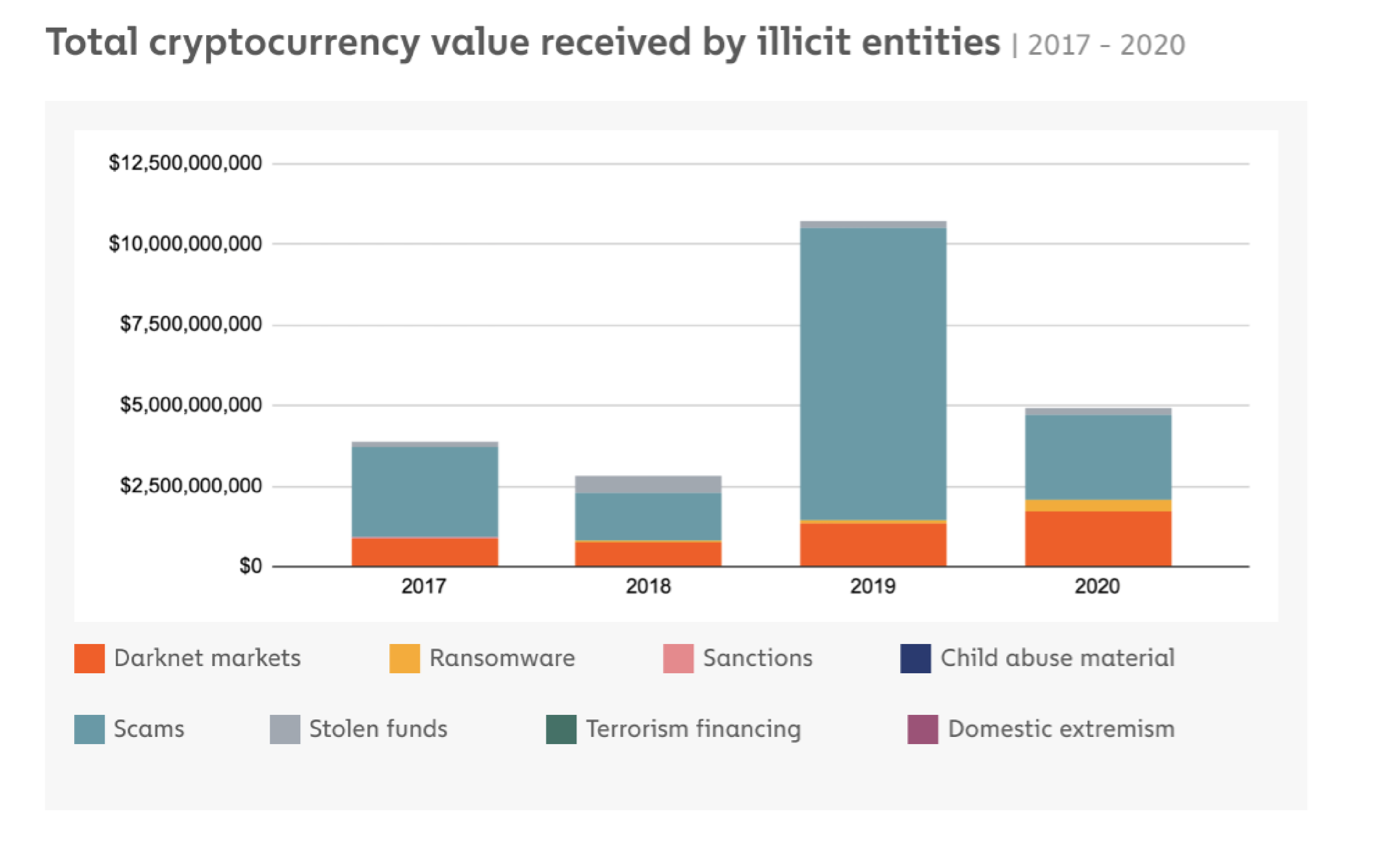
Low double-digit rate increases across the board and steeper increases for loss-hit accounts are being reported by brokers, the ratings agency said in the report.

Related Stories

Cyber insurance premiums climbing rapidly: S&P

Baltimore renews $20M in cyber insurance cover

Threat actor – Ransomware



Total cryptocurrency value received by illicit entities | 2017 - 2020

Threat actor – Ransomware

However, the big story for cryptocurrency-based crime in 2020 is ransomware. That may sound counterintuitive, as ransomware accounted for just 7% of all funds received by criminal addresses at just under $350 million worth of cryptocurrency. But that figure represents a 311% increase over 2019.

No other category of cryptocurrency-based crime rose so dramatically in 2020, as Covid-prompted work-from-home measures opened up new vulnerabilities for many organizations.

The 2021 Crypto Crime Report

Everything you need to know about ransomware, darknet markets, and more

February 2021

Threat actor – Ransomware

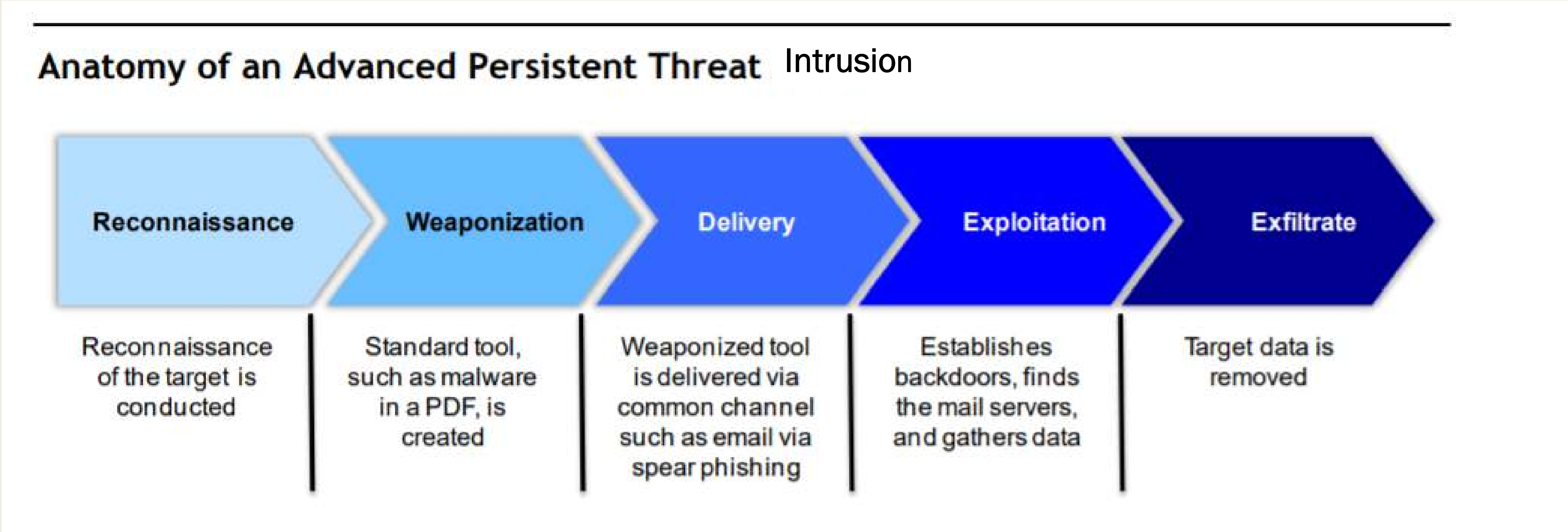
Ransomware estimates should always be considered lower bounds due to underreporting. The 2020 figure for total ransomware payments will likely grow as we identify more addresses associated with different strains, particularly in the later months of the year. Looking beyond the numbers, we also must note that ransomware is uniquely destructive in that attacks can cripple local governments and businesses for weeks, including several hospitals, last year in the midst of the pandemic. When we consider the total economic losses not just from payments, but from businesses and governments being taken offline in attacks, that ransomware cost $20 billion in economic losses in 2020.

The 2021 Crypto Crime Report

Everything you need to know about ransomware, darknet markets, and more

February 2021

Emulating a threat actor –
All the actions in an intrusion



Anatomy of an Advanced Persistent Threat Intrusion

Reconnaissance

Reconnaissance of the target is conducted

Weaponization

Standard tool, such as malware in a PDF, is created

Delivery

Weaponized tool is delivered via common channel such as email via spear phishing

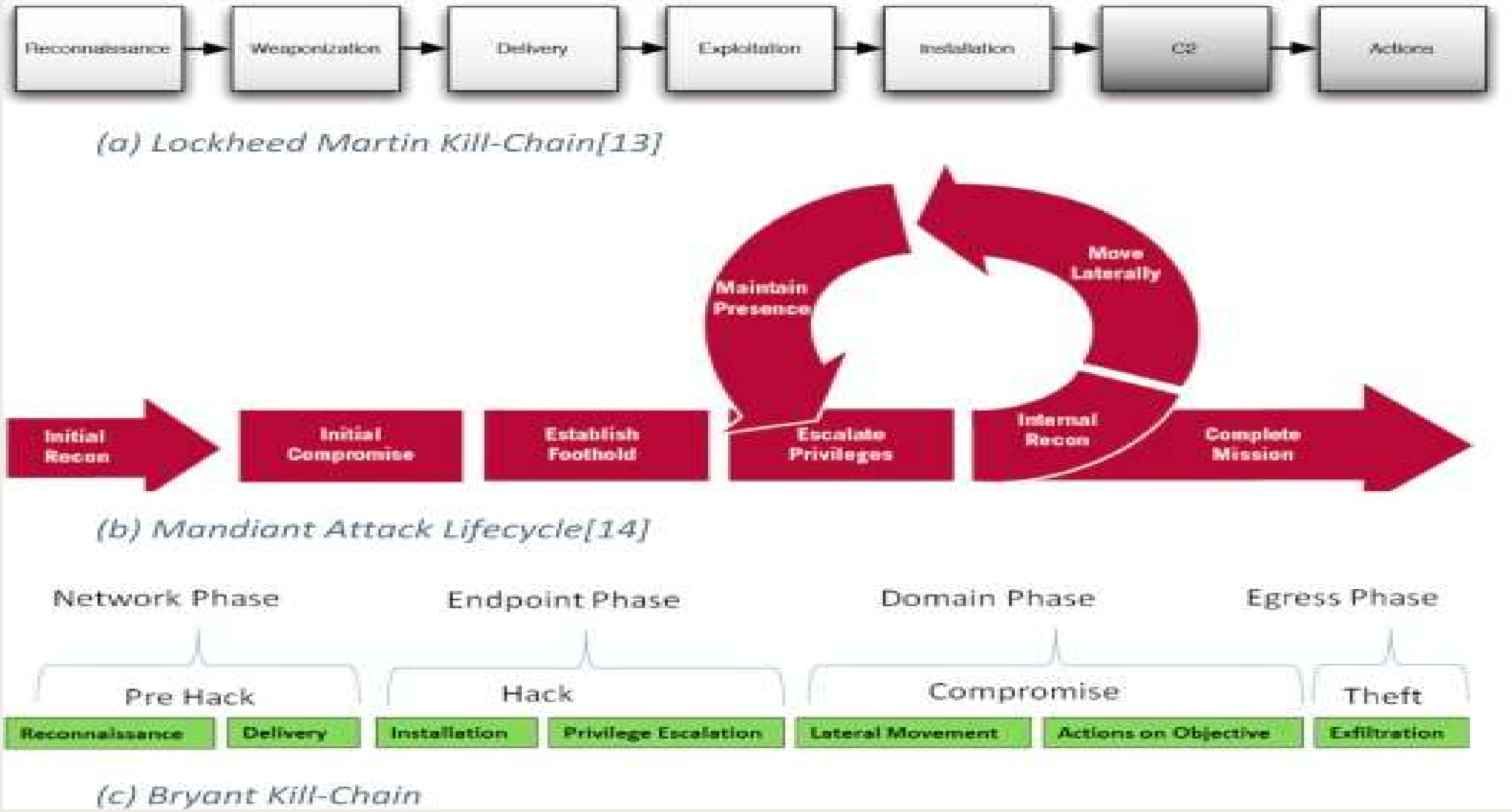
Exploitation

Establishes backdoors, finds the mail servers, and gathers data

Exfiltrate

Target data is removed

Threat actor – Emulation – All the actions



(a) Lockheed Martin Kill-Chain[13]

(b) Mandiant Attack Lifecycle[14]

(c) Bryant Kill-Chain

The actions in an intrusion

- The previous descripitions are too abstract, they focus on the abilities
- Enter a system (initial attack to the attack surface)
- Lateral moves and attacks in the system to escalate the attacker privileges
- Lateral movements
  - *they may require information the attacker has not available*
  - *the collection of information requires vulnerability scanning and attacks*
  - *a blind interleaving of collection and exploitation*
- No total planning is possible before the intrusion = constrained visibility
- The attacker may not know a priori if its current action is useful or useless
- Useful/useless strongly related with the target system and the attacker strategy = which action to execute at each step

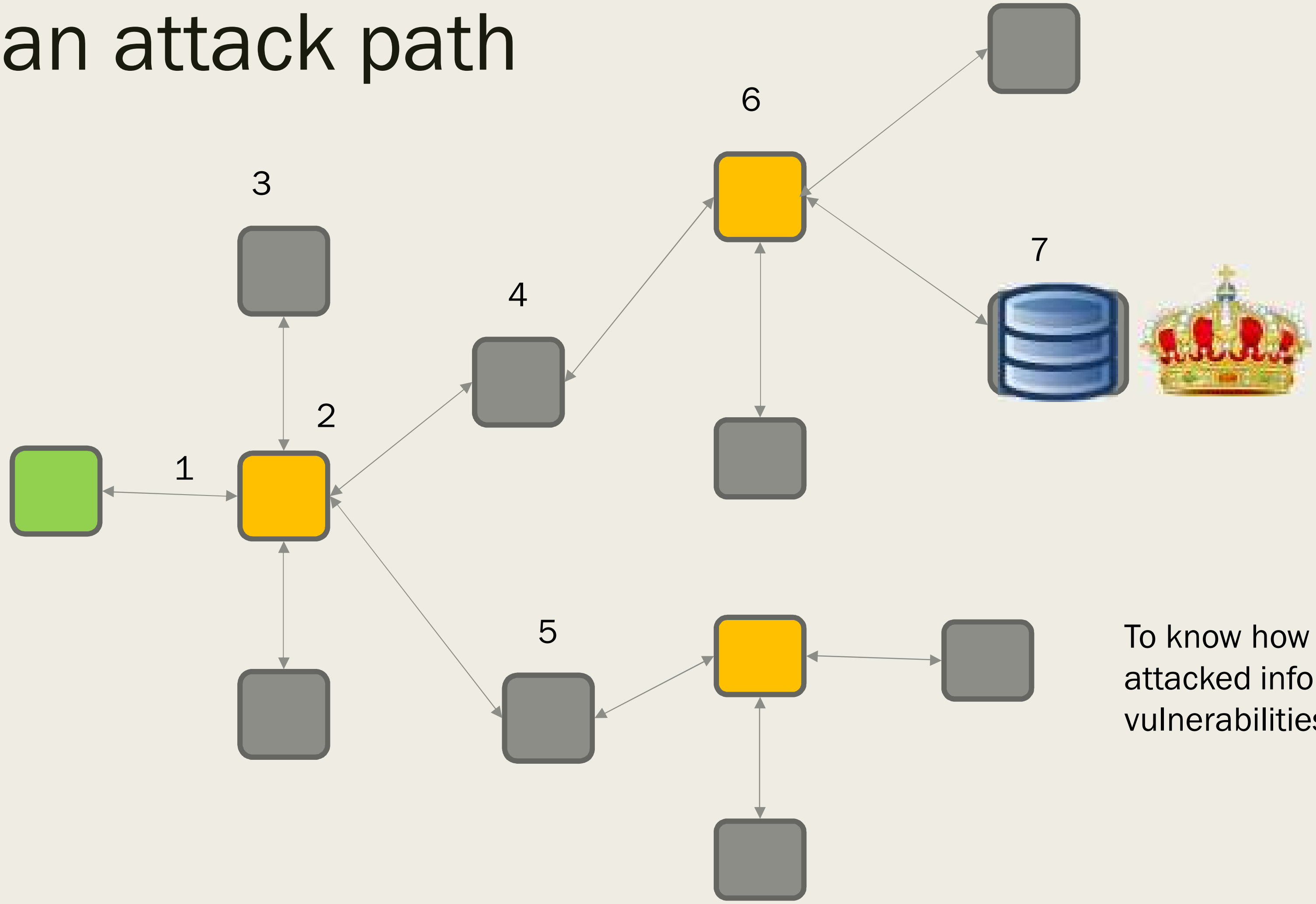
Everything depends upon the strategy

Can an attacker learn how to attack a system before attacking it??

Vulnerability scanning

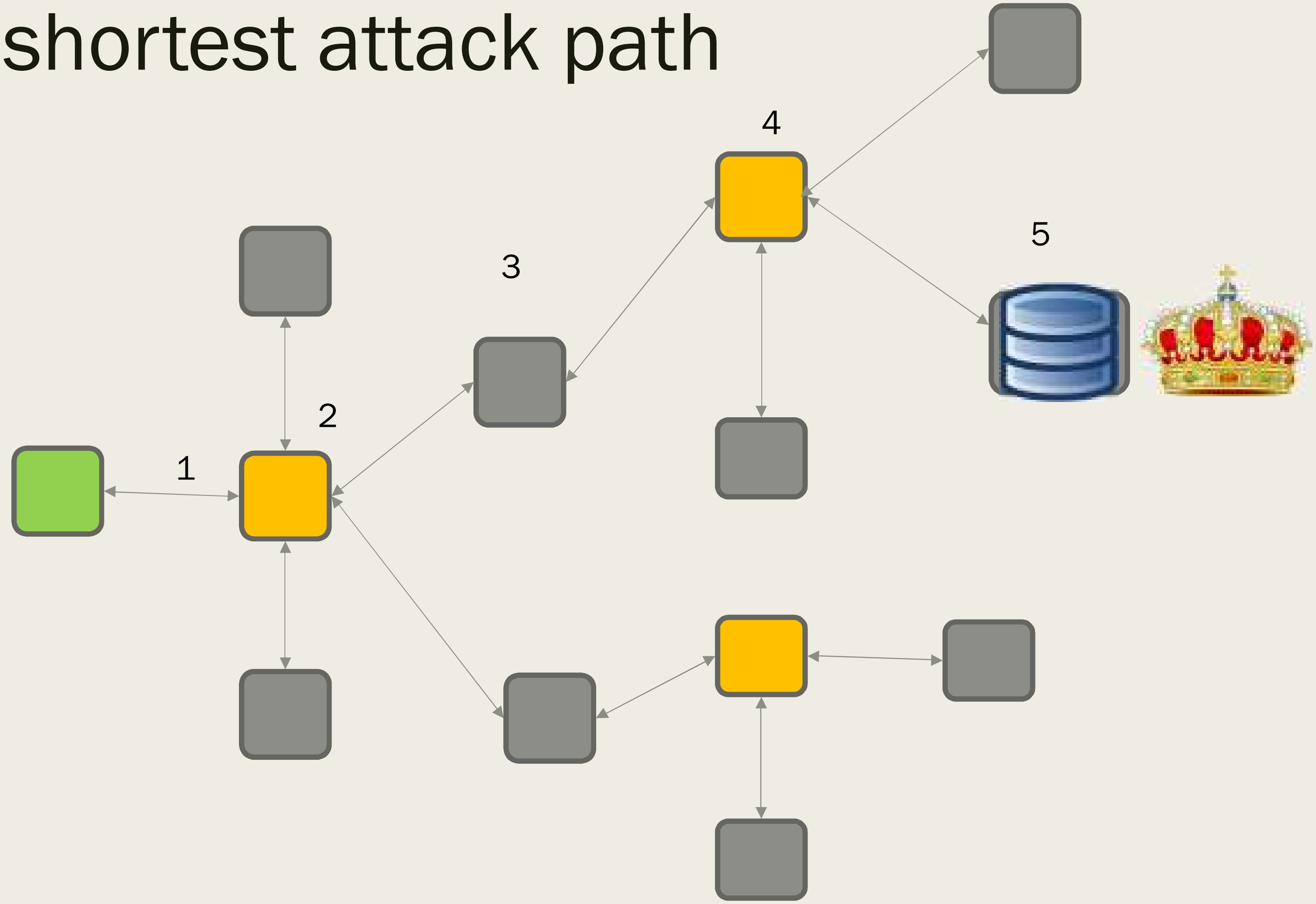
- A vulnerability scanner is a tool that receives an IP range and returns the vulnerabilities of the nodes in the range
- It uses a technique called fingerprinting based upon the transmission of malformed packets because the answer to bad malformed packet identifies the OS and the applications on a node. The scanners has a database with the distinct answers to each packet. To solve ambiguities several packets may be sent. 3-4 packets suffice to identify an OS
- After discovering the OS and the applications, they are mapped into a second database that returns their vulnerabilities

Learning to attack a system an attack path

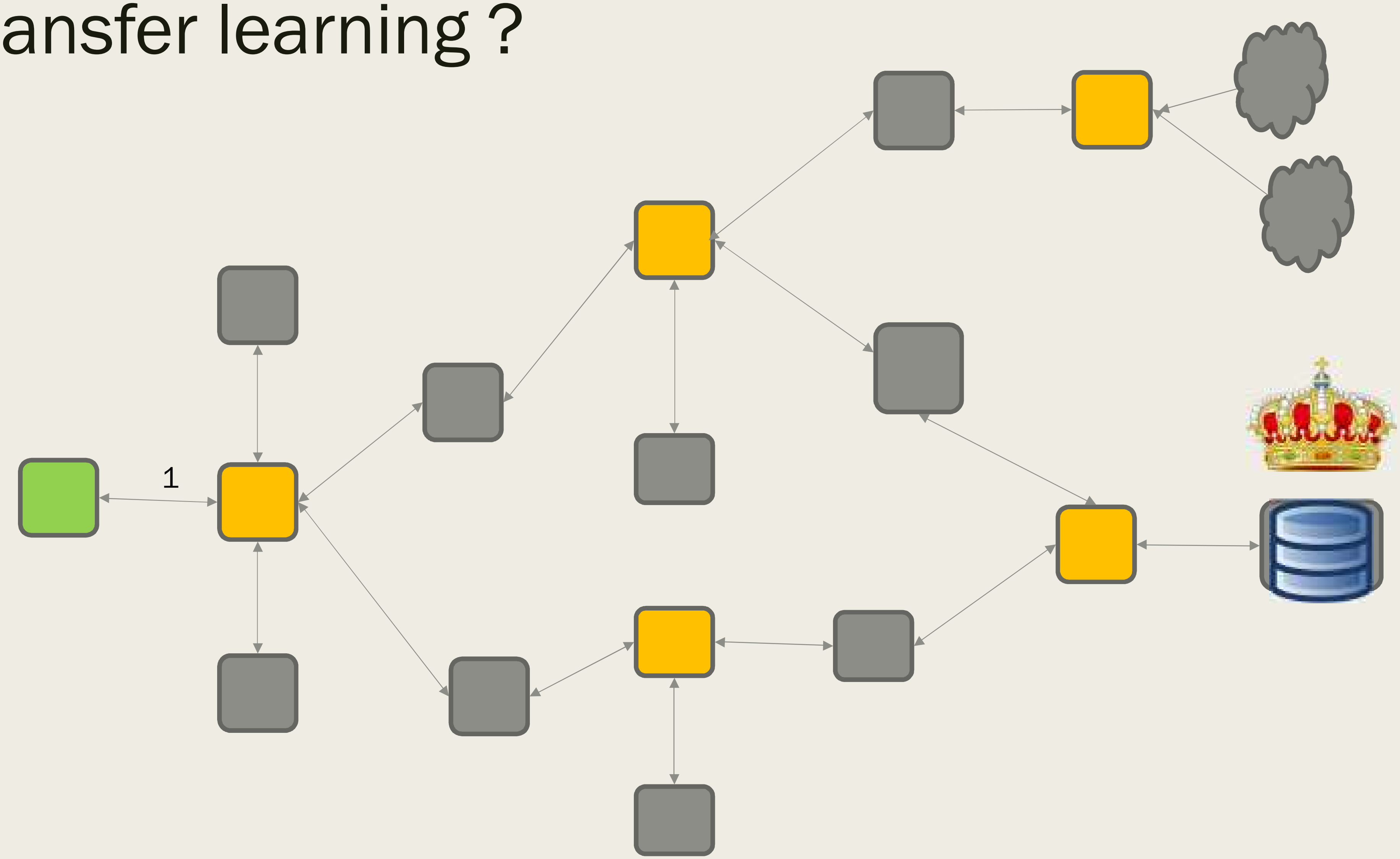


To know how a node can be attacked information on its vulnerabilities has to be acquired

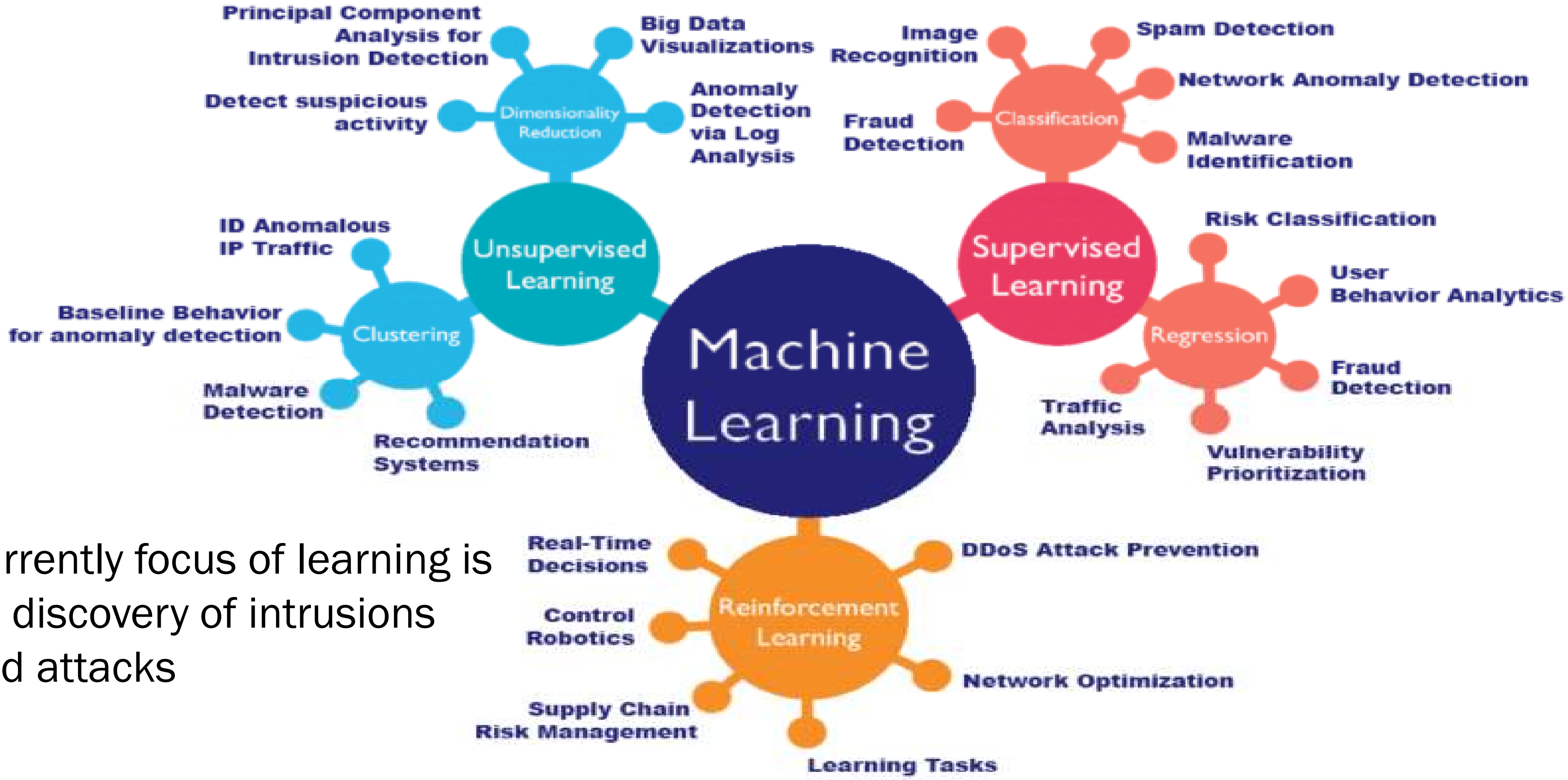
Learning to attack a system shortest attack path



Transfer learning ?



Learning to attack a system ???



Currently focus of learning is on discovery of intrusions and attacks

Attack Path

- A sequence of actions that enables the attacker to reach its goals
- There may be alternative attack paths to the same goal
- We can map an attack path into some other paths
  - *The attack plan = the sequence of useful actions in the path to reach a goal*
  - *The logical path = the sequence of the nodes that are attacked in the path*
  - *Privilege escalation = the sequence with the sets of access rights the attacker acquires*
- The number and the length of attack paths is a rough indication of the overall system robustness
- In some systems there are more than 9000 attack paths
- Natural honeypot = a system with so many attack paths that confuse the attacker
- Finding all these paths requires some automation

Command & Control of an Intrusion

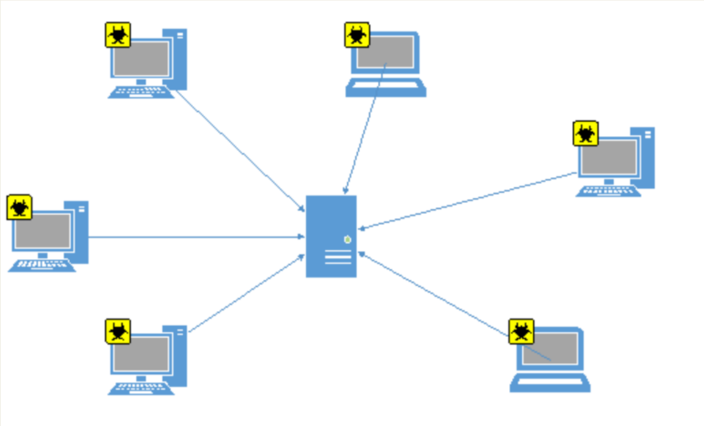
- Actions such as exfiltrate or mantain persistence in an intrusion require interactions between some modules in the target system and the attacker
- This is mediated a command and control infrastructure implemented by a botnet
- Botnet
  - *an overlay network including computers (bots) infected by some malware and under the control of threat actor, the "bot-herder"*
  - *from one central point, the attacking party can command modules on every computer on its botnet to simultaneously carry out a coordinated action*
  - *the size of a botnet enable the attacker to perform large-scale actions*
  - *infected machines of a botnet can receive updates and change their behavior on the fly to allow bot-herders to rent access to segments of their botnet on the black market to threat actors for financial gain*
- Renting of a botnet shows a first example of the division of roles in cyber crime

Botnet

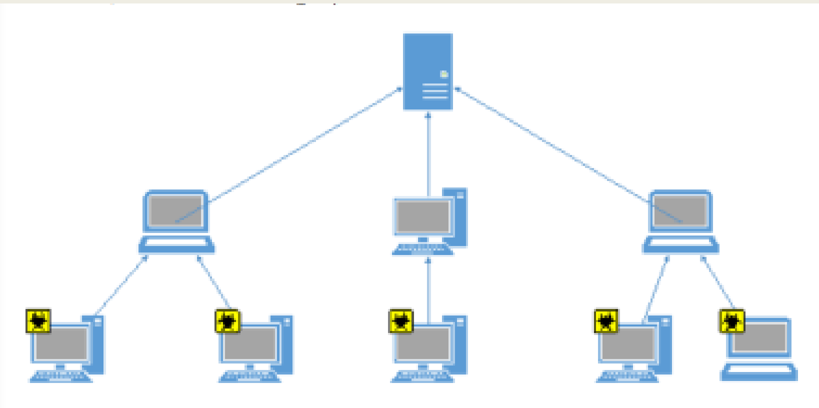
- In 2020, we find many botnets with a few tens of thousands of machines that are too small for most defenders to care about, plus some large ones that tend to be multilayer – typically with peer-to-peer mechanisms at the bottom that enable the footsoldier bots to communicate with a few control nodes, which in turn use a domain generation algorithm to find the botmaster.
- Fragmenting the footsoldiers into a number of small botnets makes it hard for defenders to infiltrate all of them, while the control nodes may be located in places that are hard for defenders to get at. The big money for such botnets in 2020 appears to be in clickfraud.

Security Engineering, Ross Anderson

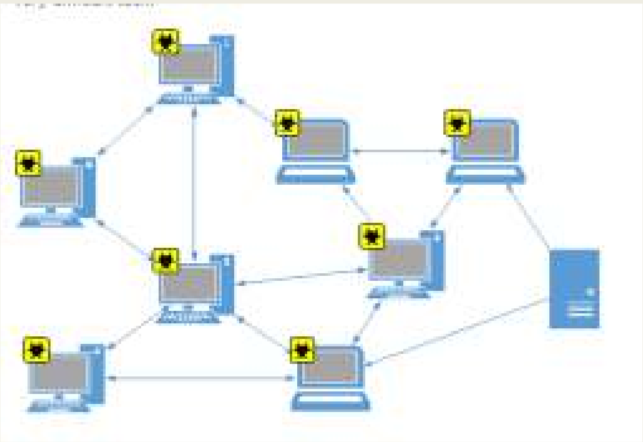
Botnet



Centralized



Proxied



Peer to peer

Botnet and Blockchains

- A recent piece of malware from a known crypto mining botnet campaign has started leveraging Bitcoin blockchain transactions in order to hide its backup C2 IP address. It's a simple, yet effective, way to defeat takedown attempts.
- Recent infection attempts against Akamai SIRT's custom honeypots uncovered an interesting means of obfuscating command and control (C2) infrastructure information. The operators of a long-running crypto-mining botnet campaign began disguising their backup C2 IP address on the Bitcoin blockchain.
- The campaign uses this data to help distribute malware, ensure persistence, and likely serve as an uncensorable defense against take-down efforts
- To convert the Bitcoin transactions into an IP address, the script needs to know the transactions the wallet that has sent and received. This requires a simple HTTP request to a blockchain explorer API for the last two transactions for the wallet address, then converting the transaction Satoshi values into the backup C2 IP address.

Attack Signature

- In computer security, a signature is a typical footprint or pattern associated with a known malicious attack on a computer network or system
- Signature-based attack detection uses a known list of indicators of compromise (IOCs). These may include specific network attack behaviors, known byte sequences and malicious domains as well as email subject lines and file hashes.
- Identifying malicious threats and adding their signatures to a repository is the critical pillar of antivirus and antimalware
- It's been used since the first antivirus solutions appeared. So, there is a degree of consistency of results and demonstrable success associated with it. The approach isn't very complex, is fast, easy to run and manage.
- As malware became more sophisticated, malware authors began using new techniques, like polymorphism to change the pattern each time the malware spread from one system to the next.

Attack Signature Evasion

- Encrypting string in the program
- Encoding API names using their hashing values.
  - With a low probability of collision on name hashes, the API call addresses can easily be retrieved by generating the hash of each API name in the import table and retrieving the API call address when a match is found.
  - This method avoids revealing the API name strings. Besides, with all other critical string information encrypted, the analyst can only predict the function of the routines by looking at the numeric values and call follows, thus, static analysis is nearly impossible
- Deeply nested if/case with computed value
- Bogus variables
- Permutation of useless instruction
- Code fragmentation in overlapping fragment

Anomaly

- In security an anomaly is an action by a system component that has been previously observed with a very low probability
- The use of anomalies implies that historical data are available and that they have been mapped into the probability that each action occurs
- The occurrence of an action paired with a very low probability may denote that an intrusion has occurred or is occurring
- The threshold to determine very low plays a fundamental role
- While a signature based detection can detect only attacks that have occurred at least once in the past, an anomaly based detection can signal even new attacks (it does not signal attacks but their consequences)
- As a counterpart, anomaly based solution suffers because of a large number of false positives that may reach values close to 50% (a random detection)

Botnet detection

- Early day solution
  - *definition of signatures of the botnet traffic based on honeypots to analyze node interactions and communication*
  - *traffic analysis to build a graph to analyse interactions Focus on IRC and HTTP*
- Current detection focuses on anomaly detection as well because a botnet works continuously even when nodes are unattended. Further anomalies are due to
  - *the frequent change of the botmaster domain, domain flux, resulting in anomalous DNS traffic due to DNS cache misses*
  - *anomalous interactions among nodes (traffic analysis and anomalies) but with large number of false positives*
- Detection improves with deep packet inspection to check signatures of known botnet protocol
- Botnet detection stresses the importance of egress filtering = the analysis of the traffic that leaves a network and not only of the one that enters a network
  - *Analysis of input traffic = discover and stop attacks*
  - *Egress filtering = discover if our network has been attacked and we do not known*

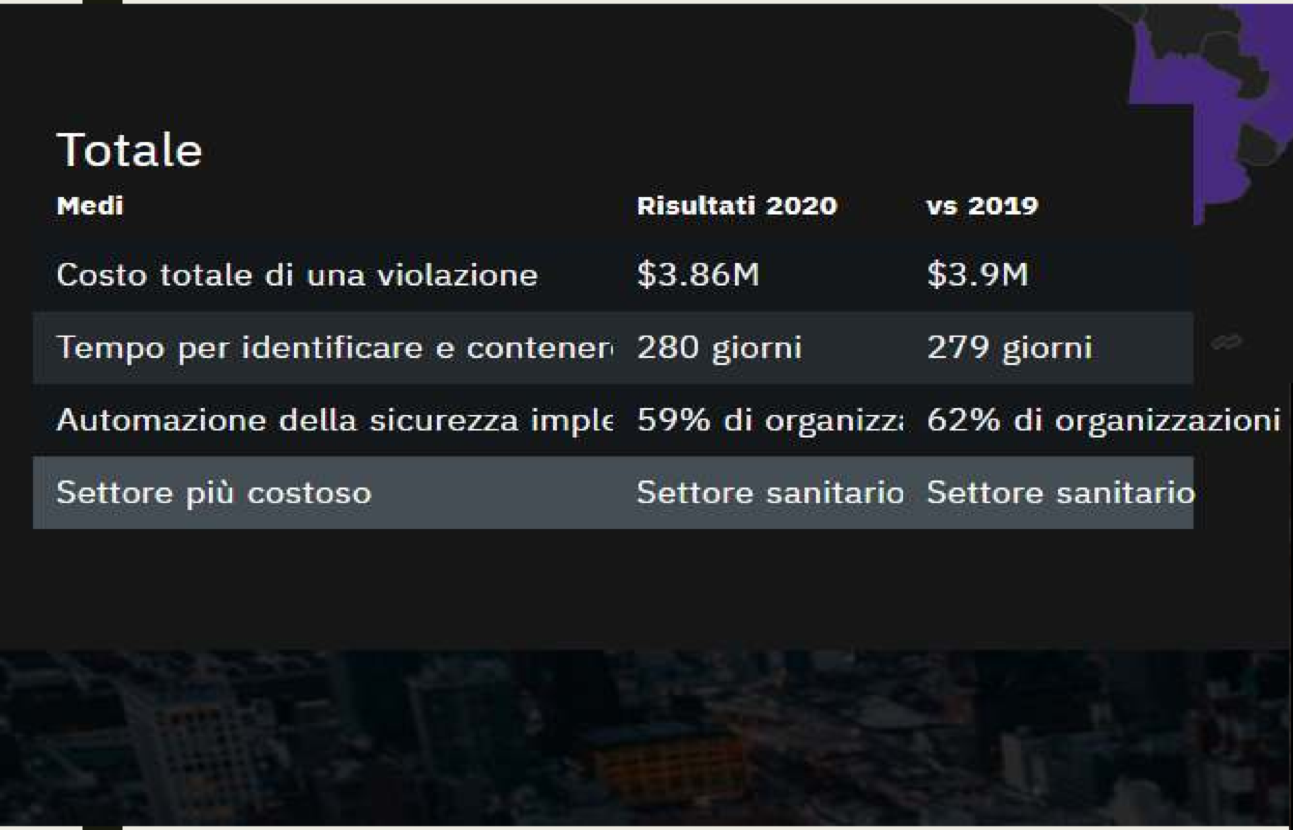
Automating Attacks and Intrusions

- Worms:
  - *an automated intrusion that spreads in a network with a payload*
  - *payload = ransomware, malware, botnet*
  - *without a predefined target, the goal is to replicate in the largest number of nodes*
  - *Implementation of remote attacks + a law to generate the address of the target nodes + a payload*
- Attack platform
  - *A software platform that can implement an intrusion and reach a predefined target*
  - *It can implement several attacks (exploits, bag of tricks) and actions*
  - *It is driven by an intelligent strategy to implement one of a predefined set of strategies*
  - *It replaces a human attacker*
- Breach and simulation tool
  - *A tool to discover the vulnerabilities in a network not the possible intrusions*
  - *Installed in a node it implements elementary attacks against other network nodes*
  - *It signal successful attacks to the administrators to adopte countermeasures*
  - *Main goal = discover unpatched vulnerabilities*

Impact

- The hardest parameter to evaluate
- It depends upon the goal hence upon the attacker
- For a commercial organization it includes
  - *Loss of IP*
  - *Loss of share value (very little and for a short time usually)*
  - *Loss of customer*
  - *Cost of recovering*
- For a state we have to add
  - *Potential social turmoil (elections, government, …)*
  - *Future damage to infrastructures (hidden malware)*
  - *…*

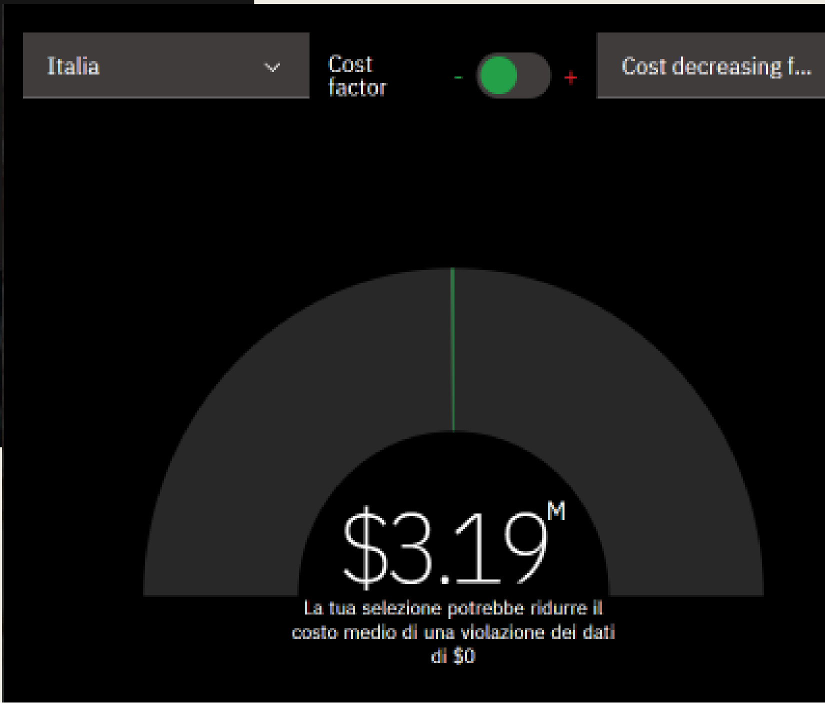
Impact



Totale

Medi	Risultati 2020	vs 2019
Costo totale di una violazione	$3.86M	$3.9M
Tempo per identificare e contener	280 giorni	279 giorni
Automazione della sicurezza imple	59% di organizz	62% di organizzazioni
Settore più costoso	Settore sanitario	Settore sanitario

IBM report, February 2021



Italia

Cost factor - + Cost decreasing f...

$3.19^M

La tua selezione potrebbe ridurre il costo medio di una violazione dei dati di $0

Risk

- The average impact of an intrusion that occurs and is successful
- It depends upon two main parameters
  - *I, the impact of a successul intrusion*
  - *P, the probability of a successful intrusion that depends upon*
    - The probability an intrusion is attempted = there is at least one threat
    - The probability the attempted intrusion is successful

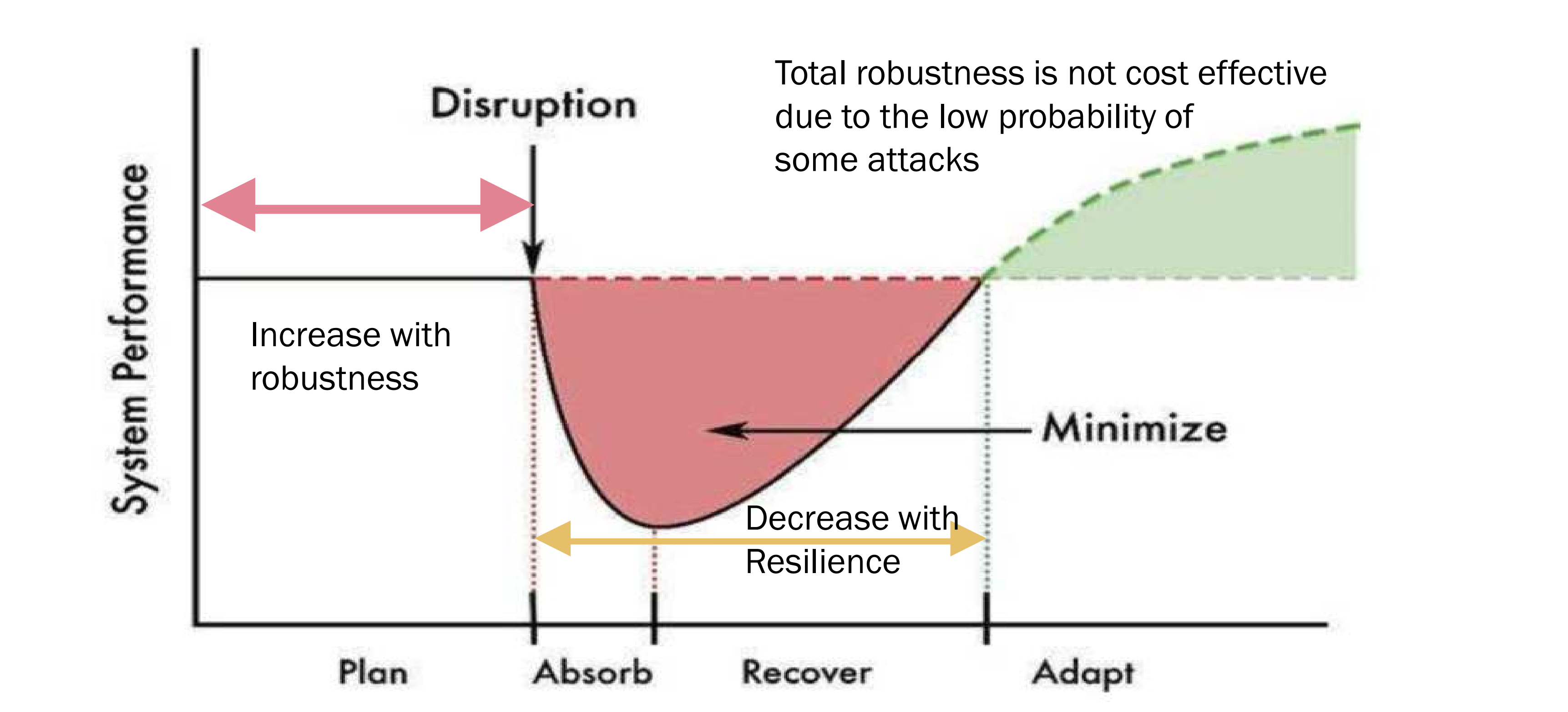
  The simplest definition is R = I *P

- Other definitions have been adopted, the most general definition

  R = R(I, P) an increasing function of I and P

Robustness vs Resilience

Robustness= the ability of resisting to intrusion
Resilience = the ability of returning to normal after an intrusion

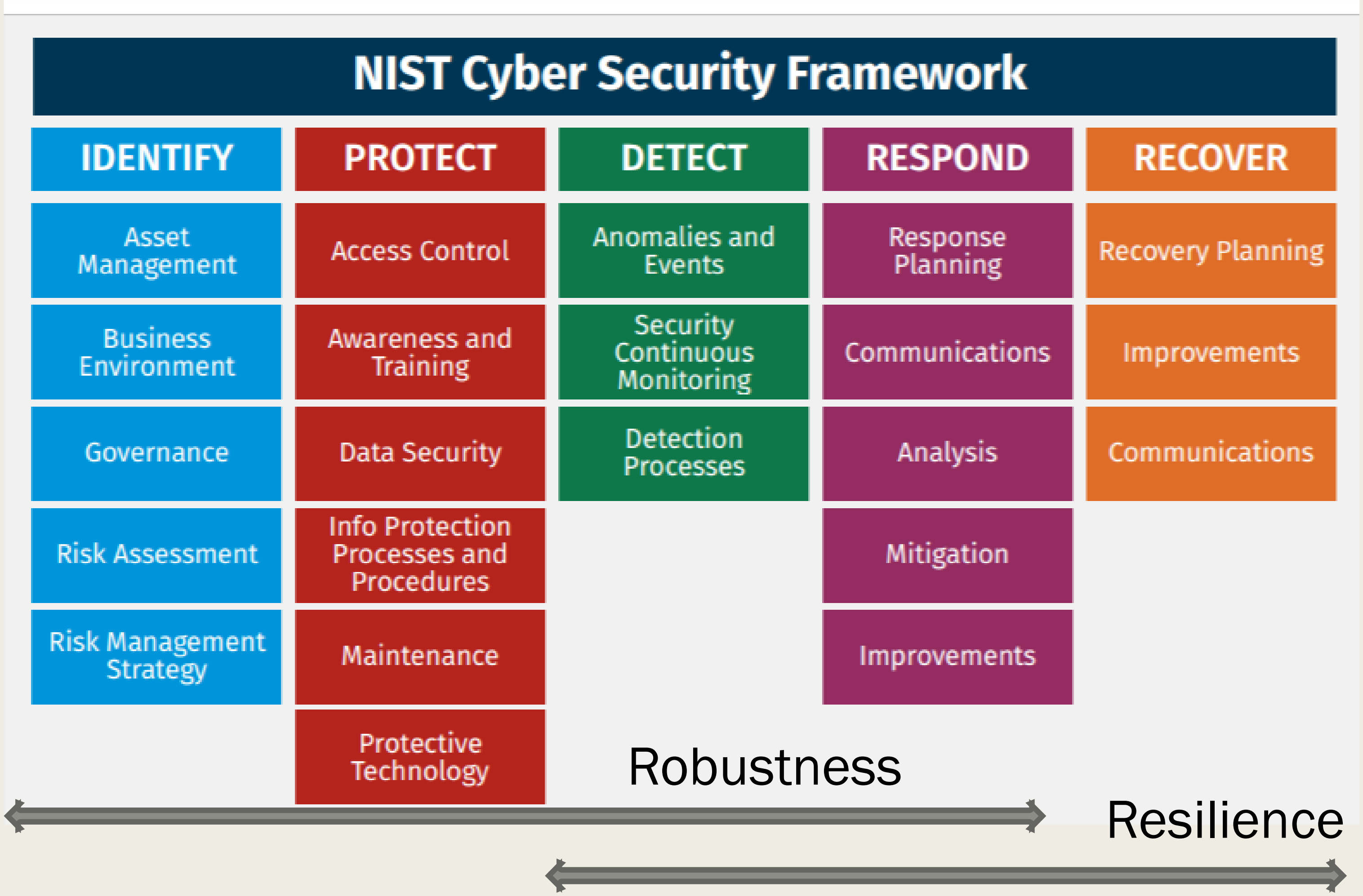


Total robustness is not cost effective due to the low probability of some attacks

Increase with robustness

Decrease with Resilience

Robustness & Resilience



NIST Cyber Security Framework

IDENTIFY	PROTECT	DETECT	RESPOND	RECOVER
Asset Management	Access Control	Anomalies and Events	Response Planning	Recovery Planning
Business Environment	Awareness and Training	Security Continuous Monitoring	Communications	Improvements
Governance	Data Security	Detection Processes	Analysis	Communications
Risk Assessment	Info Protection Processes and Procedures		Mitigation	
Risk Management Strategy	Maintenance		Improvements	
	Protective Technology			

Robustness vs Resilience

Resilience analysis preserves most philosophical background as traditional risk assessment (robustness), but resilience analysis additionally delves into the unknown. Resilience thinking requires its practitioners to ponder potential future threats to stability and develop countermeasures or safeguards to prevent longstanding losses, not just direct losses from historical (known) threats.

…

By and large, risk assessment exercises require the prevention and mitigation of the most consequential and likely risk firsts, where more minor externalities and very low-probability, high consequence events are, respectively, given less emphasis for risk preparation and mitigation.

…

While predicting what will happen tomorrow is already an inaccurate science, accounting for risk over the course of years or decades can quickly become an impractical task without some mechanism for decision support.

The science of resilience, I.Linkov and B.Trump

Resilience adds a time dimension to robustness to handle both known unknowns and unknown unknowns

Robustness vs Resilience

- An alternative perspective is the one of highly optimized robustness
- We can optimize robustness by very detailed models of the attackers

  but these detailed models requires an accurate prediction of the future

We have two alternatives (but security and safety)

a) *use very detailed attacker models to build a system that is highly optimized with respect to modelled threats but, simultaneously, very fragile if something unexpected happens*

a) *add some noise, ie uncertainty, to our attacker models to build a system that is less optimized (less cost effective) but can tolerate some unexpected event*

Resilience = Give up optimization and embrace some redundancy