Attack analysis

Characterizing an attack - I

- Any attack can be described through six elements

1. Precondition
   - rights on system objects
   - resources
   - competences and info
2. Post condition
   - rights on system objects

3. enabling vulns
4. actions to be executed
5. success probability
6. noise

Characterizing an attack - II

- The attack post condition is the set of rights of the the attacker if the attack is successful
- The postcondition always include the preconditon (monotone right acquisition)
- The actions to be executed include
  - Human actions
  - Program execution
- Fully automated attack = no human action is required
- Noise = events that enable the detection of the attack

Example -I

- To implement a buffer overflow, one needs
  - To be able to invoke a procedure (rights)
  - To be able to write a parameter that includes the program to be executed (know how)
  - To know the memory map to determine the size of the parameter to overflow the stack (info)
  - Fully automated attack
  - Success probability = depends on controls in the attacked system

Example -II

- If the attack is successful, the injected program is executed as root and it can access any system resource
- The noise of the attack is a function of the security controls the attacked system executes and that make it possible to detect the attack
- The controls influence both the success probability and the noise as they can only discover (log) or also prevent (type -canary) the attack

Attack taxonomies

- Several alternative taxonomies do exist that are focused on distinct features
  - 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 may be focused on other properties or on several of these features

Elementary vs complex attacks

- An elementary attack is the one previoulsy described and characterized by the previous elements
- In a complex system a threat cannot achieve one goal (set of rights) through just one elementary attack
- Elementary attacks have to be composed into a complex one (attack plan, privilege escalation) to increase the rights of the attacker till reaching one of the goals of interest
- Intelligent attackers with a plan of action
- The precondition of each attack in the plan has to be included in the rights the attacker acquires through the previous attacks in the plan (the union of the postconditions of these attack plus any initial rights)

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
  - In planning or programming we are interested in one program/strategy (optimal or suboptimal) to reach a given goal (consider one robot moving in a space)
  - Several attacks can be selected (several robots simultaneously)
  - An assessment is interested in discovering all the programs/strategies an attacker can implement to achieve a given goal (we have to stop all the robots)

Complex attacks - II

- Elementary attacks are composed to increase the rights of the attackers
- Elementary attacks targeting the same system = increase the attacker rights on the system resources
- Elementary attacks targeting another system = increase the attacker rights by exploiting the trust relation among systems

Complex attack: An example

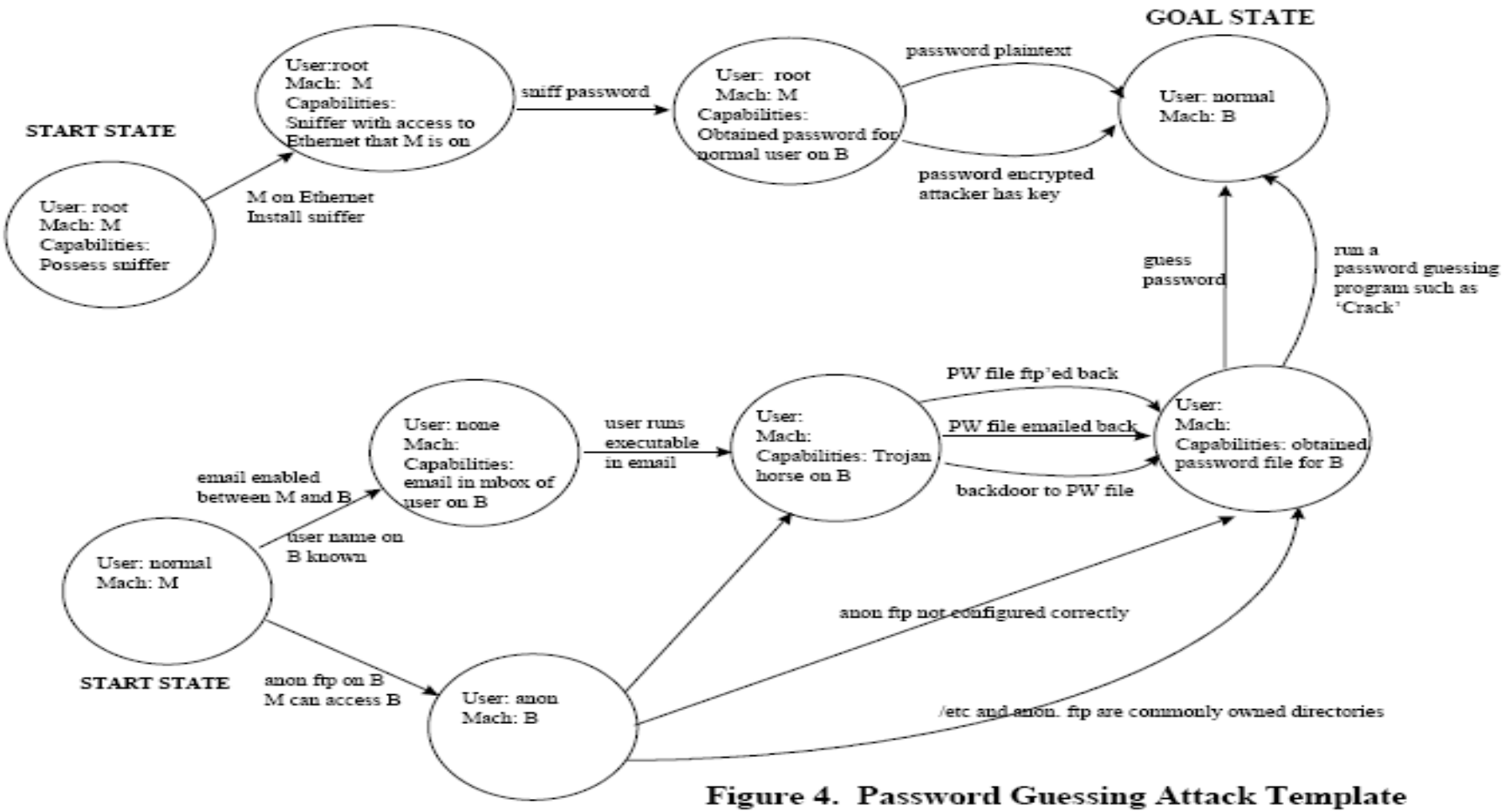


Figure 4. Password Guessing Attack Template

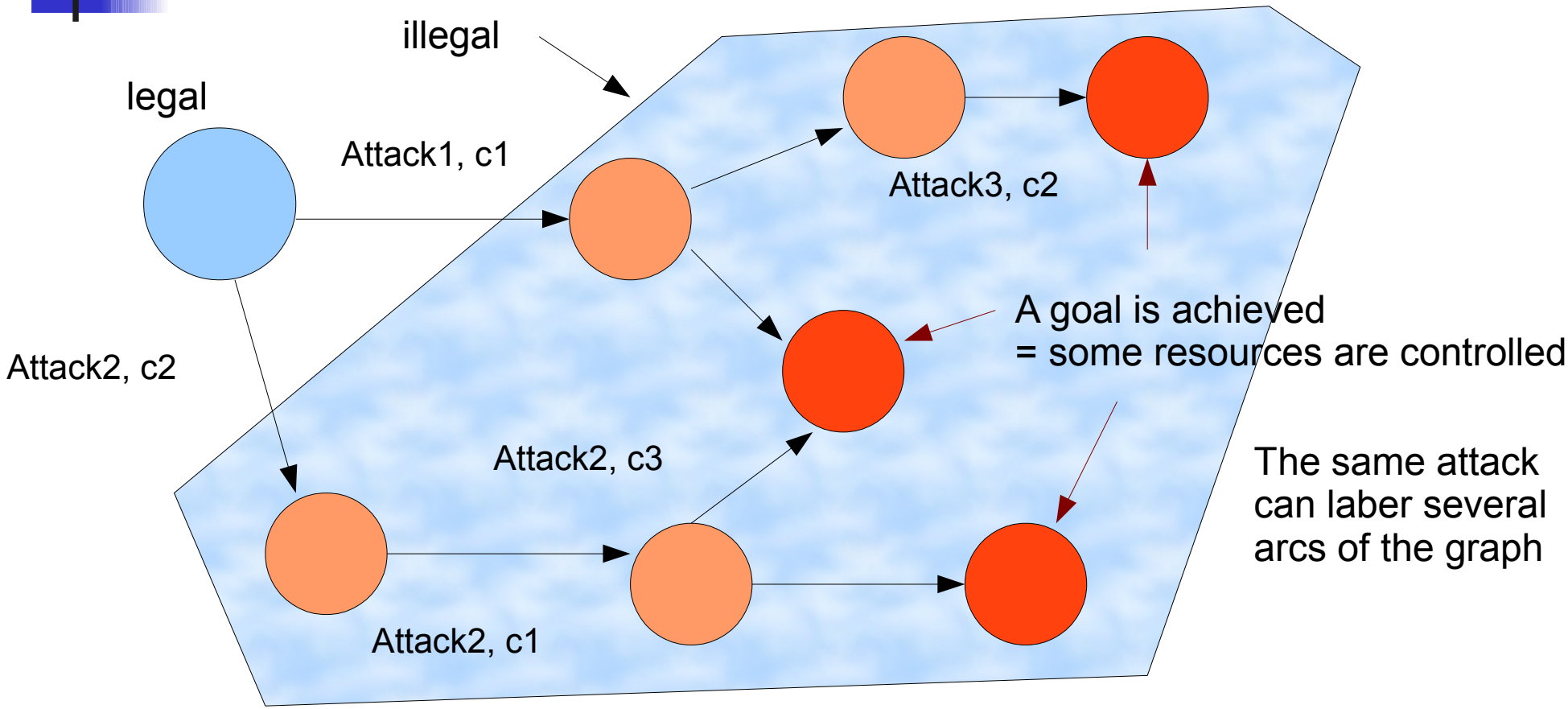
Some other example

aiardi\Documents\Magic Briefcase\cloud\riferimenti\BHUSA09-Kortchinsky-Cloudburst-SLIDES.pdf

Attack graph

- It shows how a threat can compose elementary attacks to achieve a given goal
- It is a function of current vulns and of the goals of the attackers
- The graph is acyclic because of the monotone right acquisition process
- It consider the worst case where attacks are successful
- In each state the threat can execute all the attacks that are possible in the previous states

Evolution of a user state



Some states are useful only to reach a final state

State= set of rights

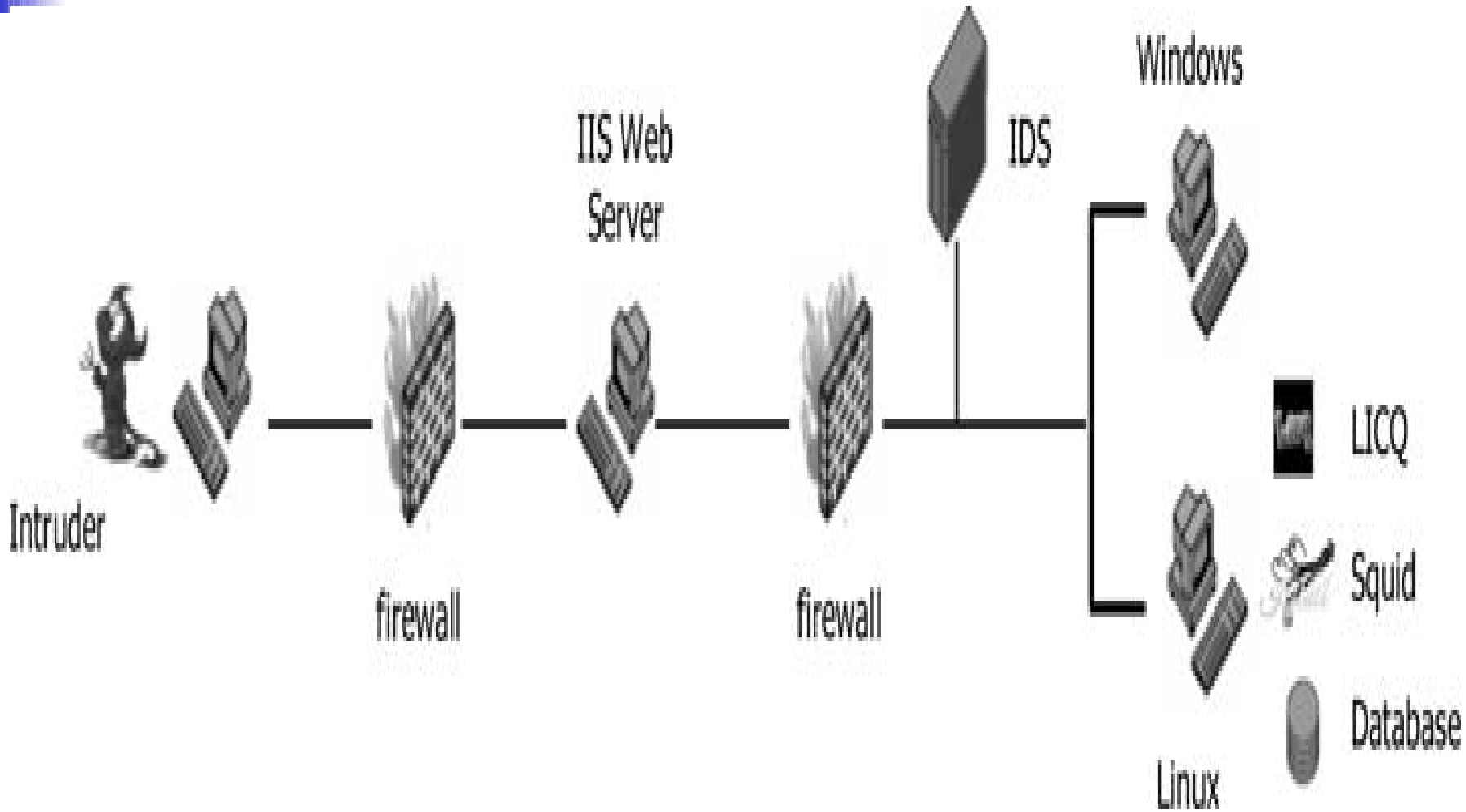
System evolution

- 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)
- The graph that describes the system evolution may not be acyclic because if a threat implements a DOS this may reduce the rights of other threats

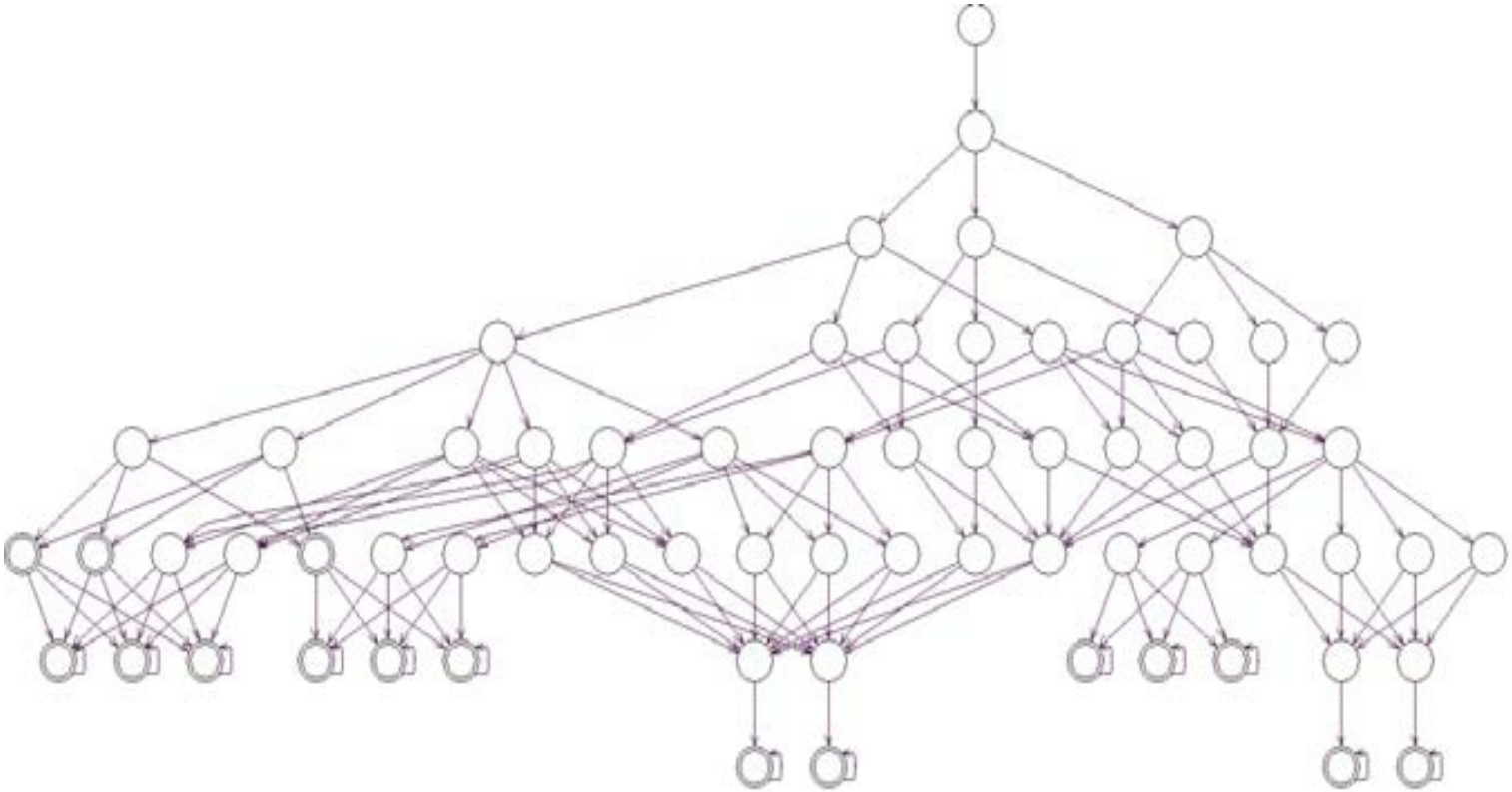
State explosion

- There is a huge number of states that strongly increases the complexity of any analysis
- It is not practical to assume the knowledge of this graph
- Two main reasons for the explosion
  - Several attacks in a plan may commute
  - Distinct attackers can implement their attacks
    - Sequentially
    - In parallel

System architecture



Attack Graph



One goal of one user

Elementary vs complex attacks

- The problem of discovering elementary attacks is rather different from the discovering of how the attacks can be composed to reach a goal
- The discovery of elementary attacks depends upon both system vulns and on the components of the system that is available
- The composition of elementary attacks may be considered as an instance of a well known optimization problem = how to reach some nodes of a graph

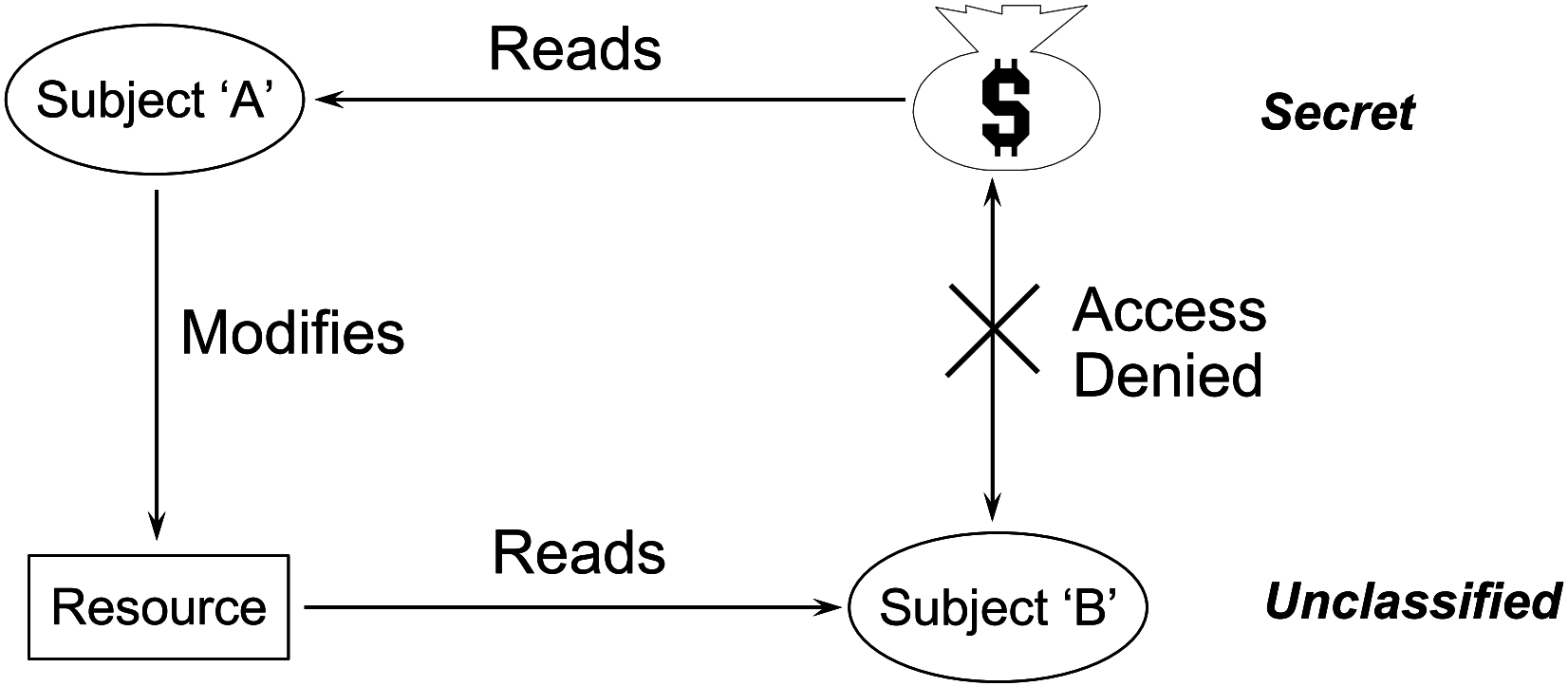
Attack surface

- The attack surface of a system includes all those elementary attacks that are the starting points of complex attacks, the elementary attacks to begin a complex one
- An elementary attack that is not in the surface can be stopped by preventing the execution of some attacks in the surface
- The ratio r between the number of attacks in the surface and the overall number of attacks in attack plans may be seen as an approximated evaluation on the system security
  - $r \rightarrow 1 \Leftrightarrow$ there are several ways to compose the attacks into plans, so the overall security is low
  - $r \rightarrow 0 \Leftrightarrow$ if a few attacks in the surface are stopped all the plans are stopped

A simple taxonomy of elementary attacks

1. Buffer/stack/heap overflow
2. Exchanged information is illegaly read (sniffing)
3. Some of the legal messages of a legal user are repeated (replay attack)
4. Interface operations are 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
10. Impersonating
    - A user
    - A machine (IP spoofing, DNS spoofing, Cache poisoning)
    - A connection (connection stealing/insertion)

Covert Channel



Cryptographic 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

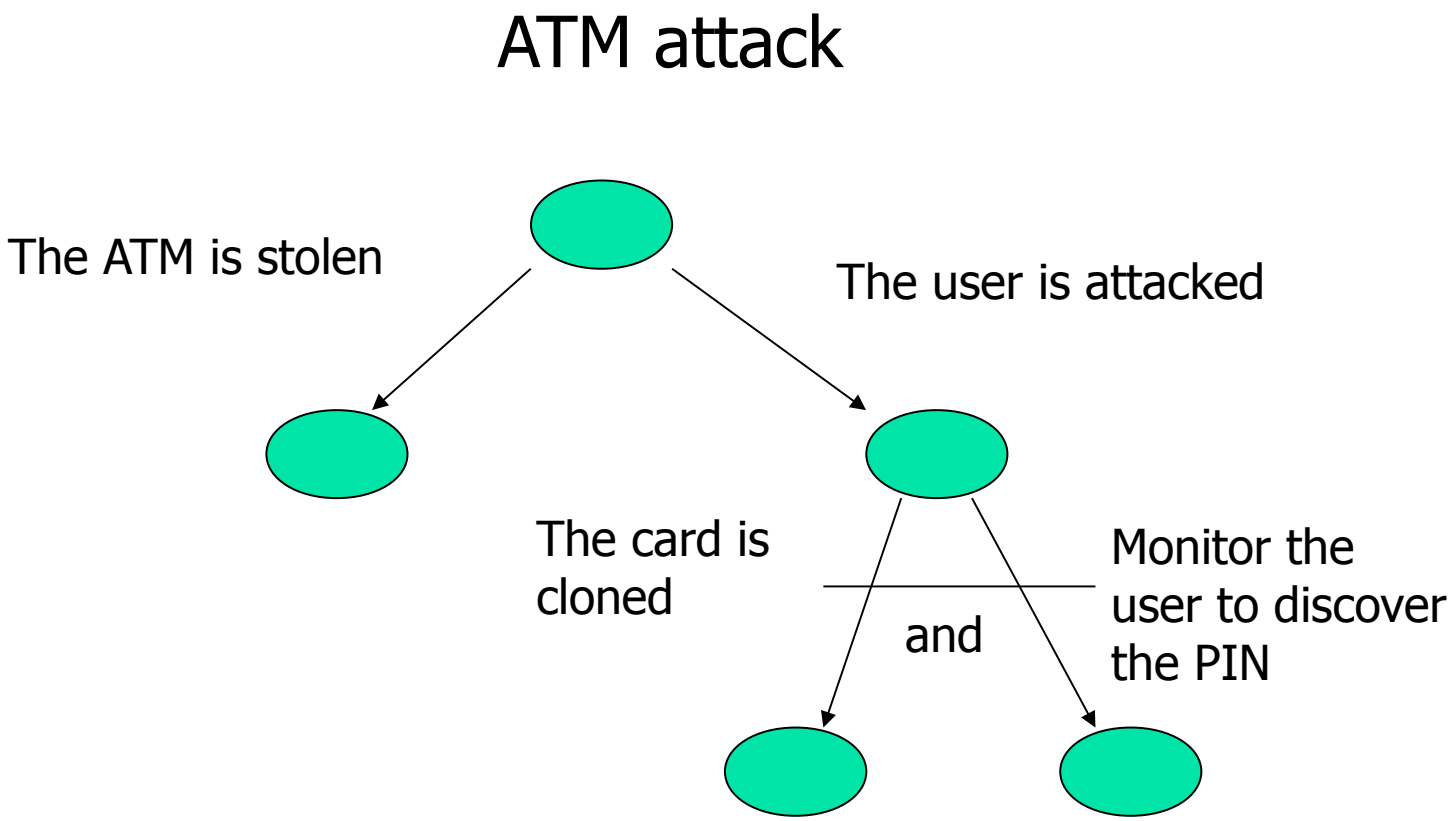
Attacks against the TCB

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

Attack Tree Analysis – I

- A top down approach to discover how a complex attacks can be implemented
- A complex attack is decomposed into simpler attacks
- The top down procedure stops when one of the elementary attacks is matched
- Two decompositions
  - AND = all the attacks are required
  - OR = just one of the attacks is required

Attack Tree Analysis - II



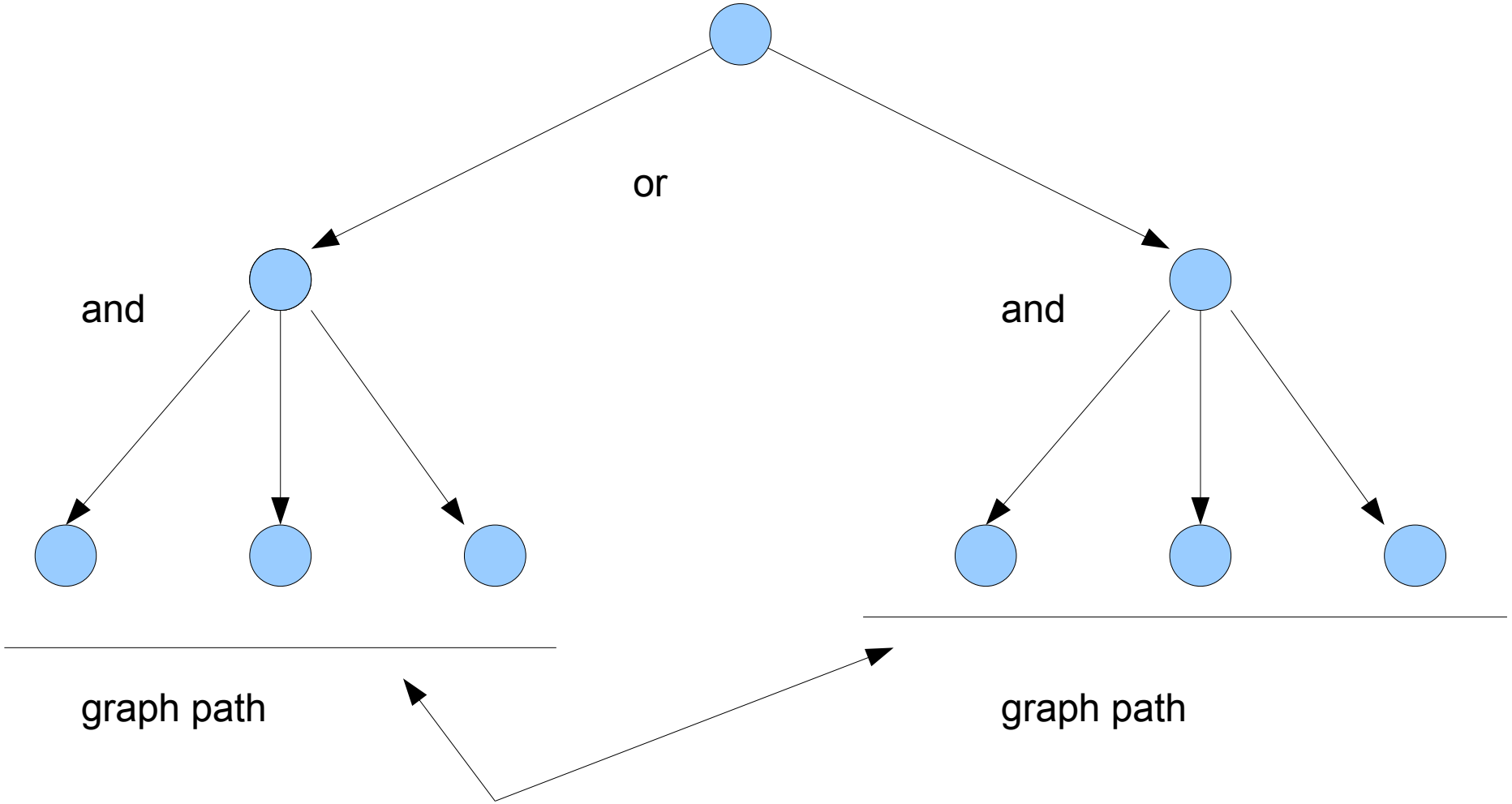
Attack Tree Analysis -III

- 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 and, hence, of common problems
- States = set of access rights that have been acquired
- Automata = attack graph

Attack tree vs graph (automata)

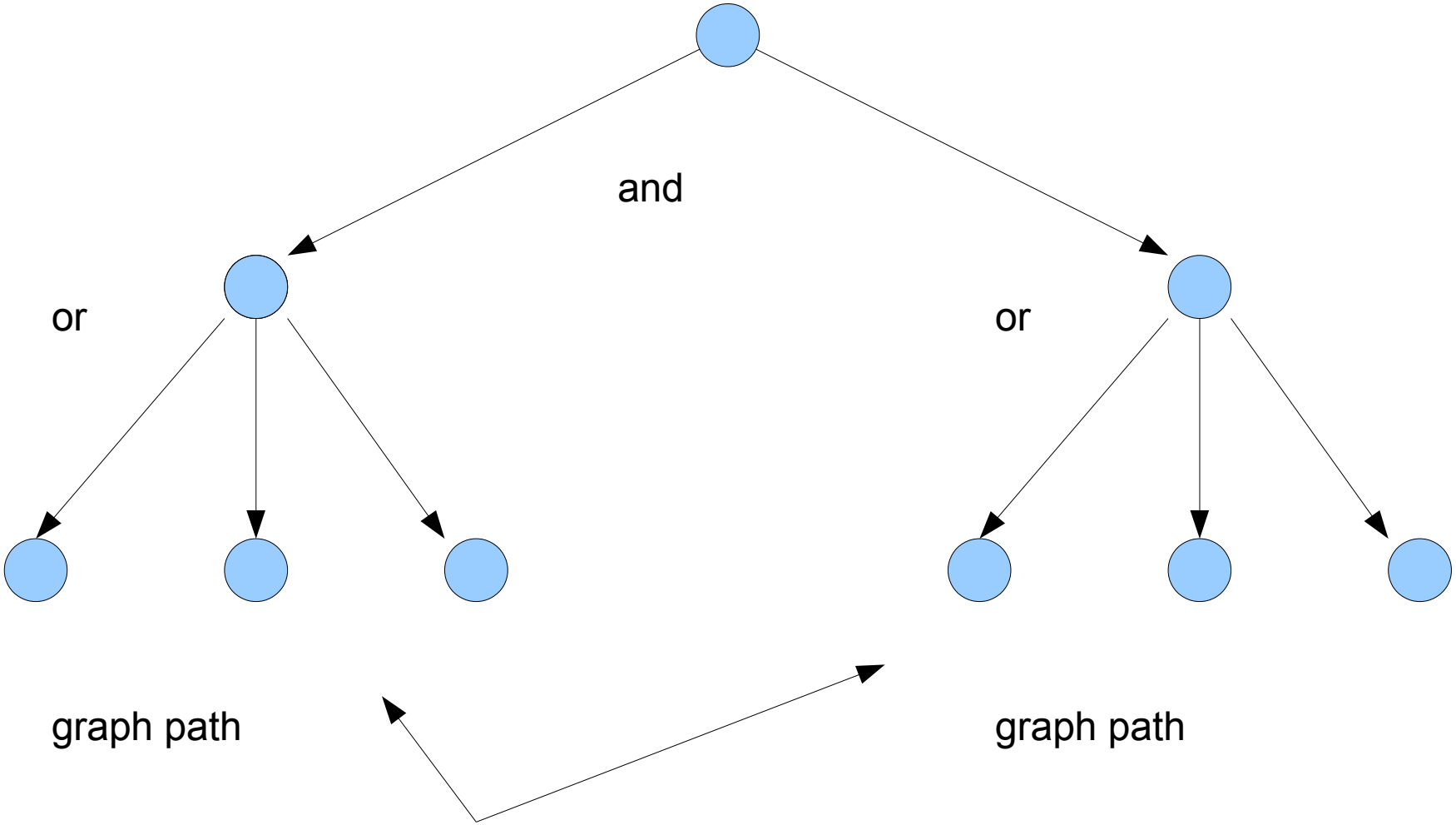
- The attacks in an AND relation in the tree belongs to the same path of the graph
- An OR nodes implies that several paths can be defined and do exist in the graph
- A tree represents one or more complex attacks
  - Consider the subtree rooted in the root of the tree
  - This subtree includes all the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

Complex attacks and countermeasures

- A complex attack is stopped if any of its elementary attacks is stopped
- If an elementary attack is shared among several complex ones, all are stopped.
- 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 the attacks are stopped)
- A cut set includes at least one elementary attack for each complex one that enables a threat to reach one goal
- 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 = how many paths to a goal shares an arc that corresponds to the same attack

How dangerous is an attack?

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

How dangerous?

- The model assumes that the 5 coordinates are orthogonal, eg independent
- In this way, an attack is represented as a point in a 5 dimension space
  - Technology competence
  - Info on the target system
  - Attack experience
  - Probability of opportunity
  - Devices

Automating an attack

- Original features of ICT security are
  - Fully automated attacks = fully programmable attacks
  - Automatic tools to implement attacks (execute the program)
- The existence of tools that implement the attacks
  - Simplify the implementation of attacks
  - Strongly enlarge the pool of potential attackers

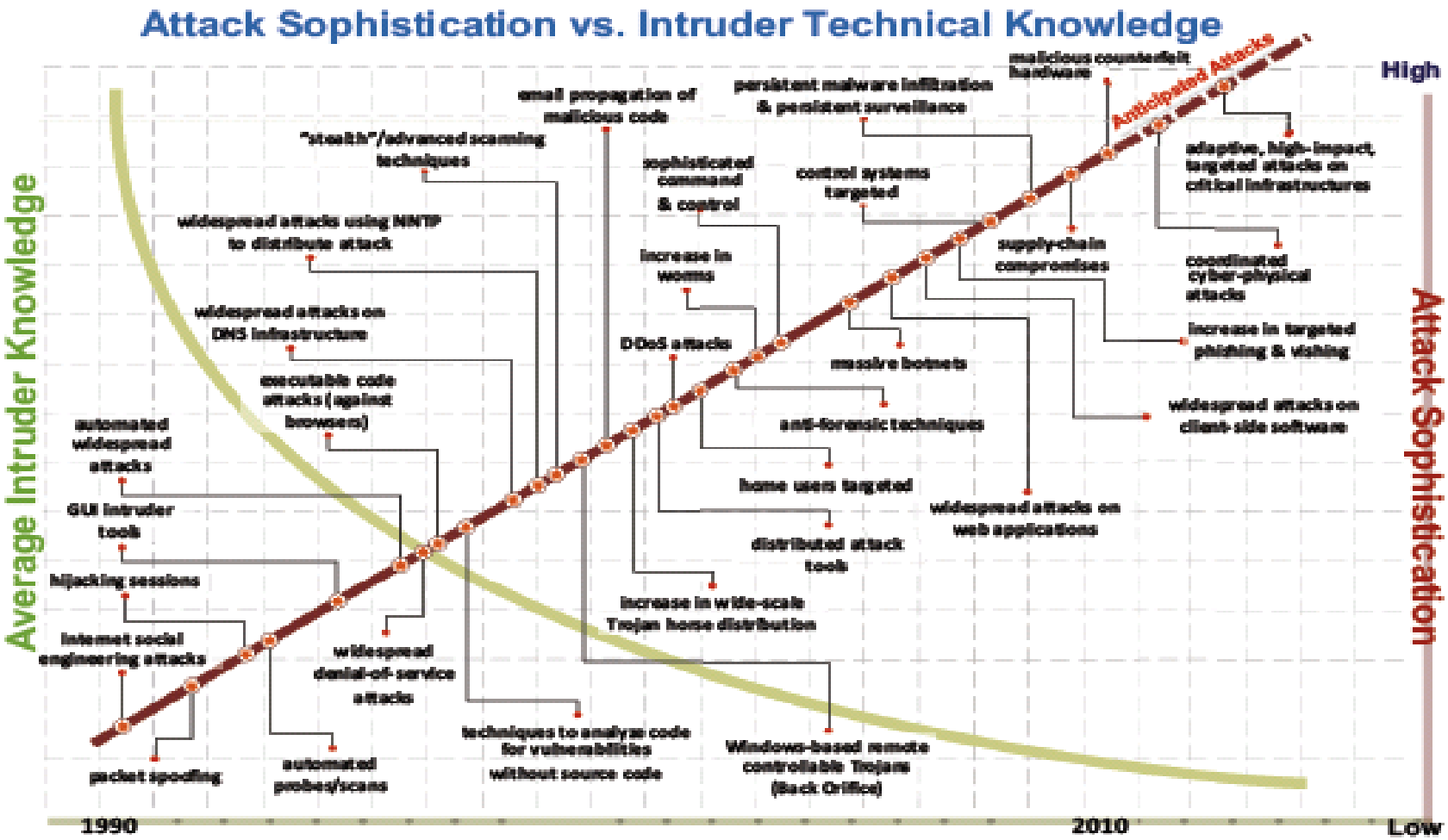
⇒ The potential impact of a vulnerability and the probability an attack is implemented strongly increase

Fully depends upon the feasibility of automating an attack

Fully automated attacks

- Exploit = the program that exploit the vulnerability to implement the attack to control some components
  = an instance for each distinct system
- All the instances of a standard component
  - Are affected by the same vulns
  - Can be attacked by the same exploit
- Fully automated attack= no further actions, information, abilities are required besides the ability of running the exploit
- In the previous evaluation, the first 3 dimensions are equal to zero and the fifth one is outside the control of the defender
- Currently, several exploit databases are available that store exploit that can be tested against a system

Fully automated attacks



Fully automated attacks

- The functions show how really dangerous attacks are implemented through tools that are distributed and accessed through the web
- It is more and more critical the window of exposure = the time interval between
  - The time an exploit is pubblicly available
  - The vuln is removed from the system

$\Leftrightarrow$ even a complex organization has to apply the patches to remove a vuln in a very short time

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 /□HHHH

The ICT zoo (malware)

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

Most important problem

In the future

Virus

- A program that
  - Hides itself in other program or data
  - It is transmitted together with such a program or such 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

Fully automated and mobile attacks

- Worms and virus implements automated attacks and can replicate on system nodes
- A worm is an autonomous program that after successfully attacking another node, creates on the node
  - An instance of the code to attack (infect) other nodes
  - Some payload (send spam, steal/update/ modify the info of the node …)
- The program attacks any node that can be reached from an infected one
- Genetic diversity is important because a windows worm will not attack a linux node and the other way around, but multiple versions in the same worm may exist

Sapphire/Slammer worm

- 376 byte in one UDP packet
- It exploits a vuln in the SQL server
- An infect node can infect from 100 to 10000 further node in one second
  - The number of infected nodes doubles in 8.5 seconds
  - $\approx$ 100 times faster than previous worms
- In 10 minutes it has infected 90% of nodes that may have been inf
- More than 75.000 infected nodes

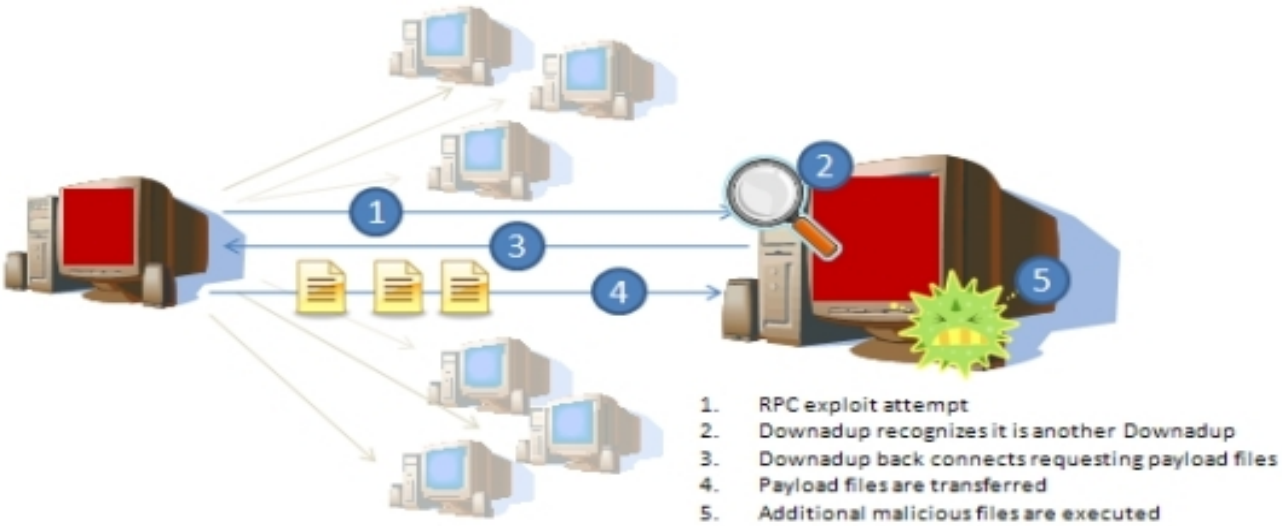
Conficker: an hybrid

- 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 and 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



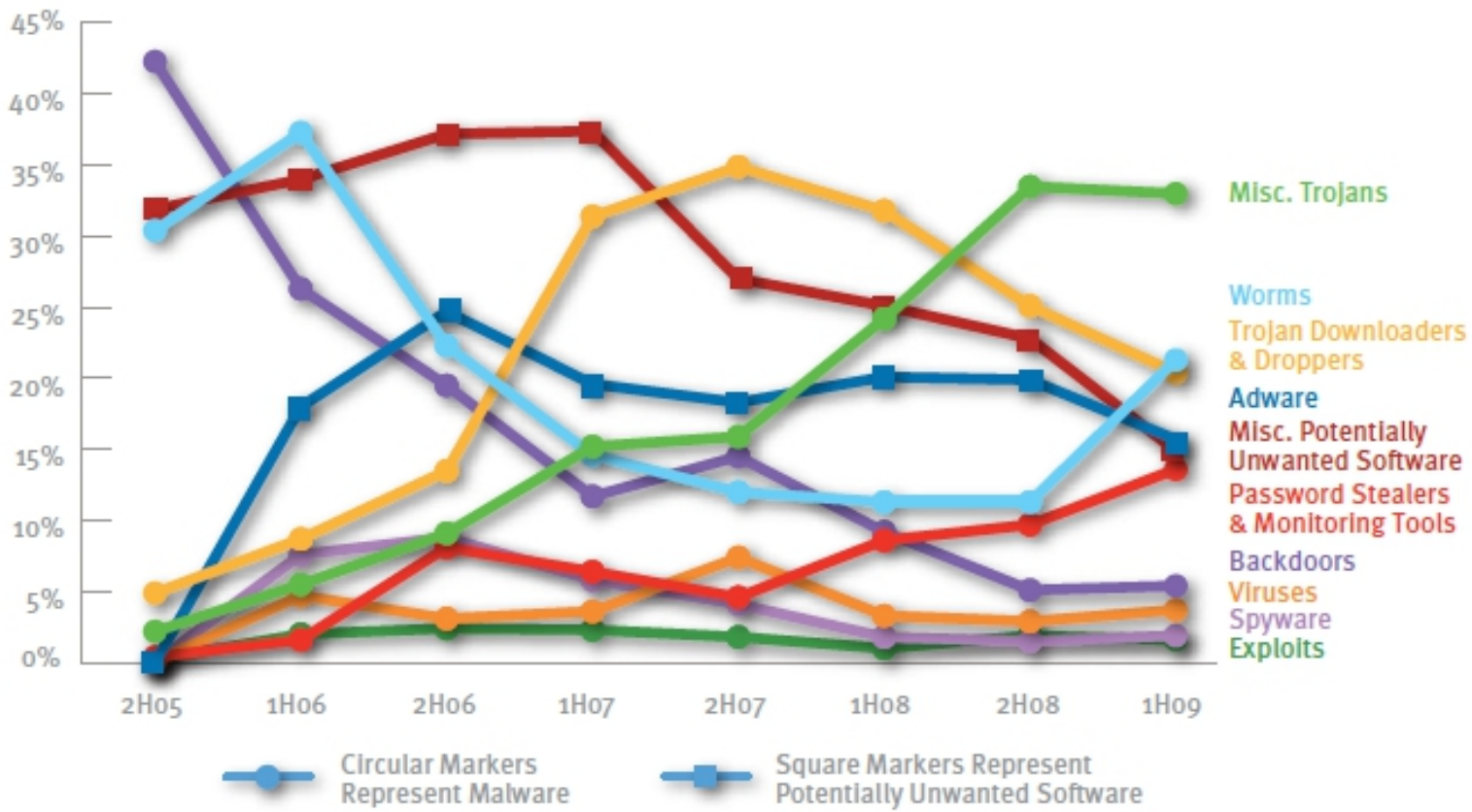
- Let us assume that an infected node is attacked
- The infected node
  - understands that the attacker is a peer (is infected)
  - connects to the attacker and downloads any update

Conficker

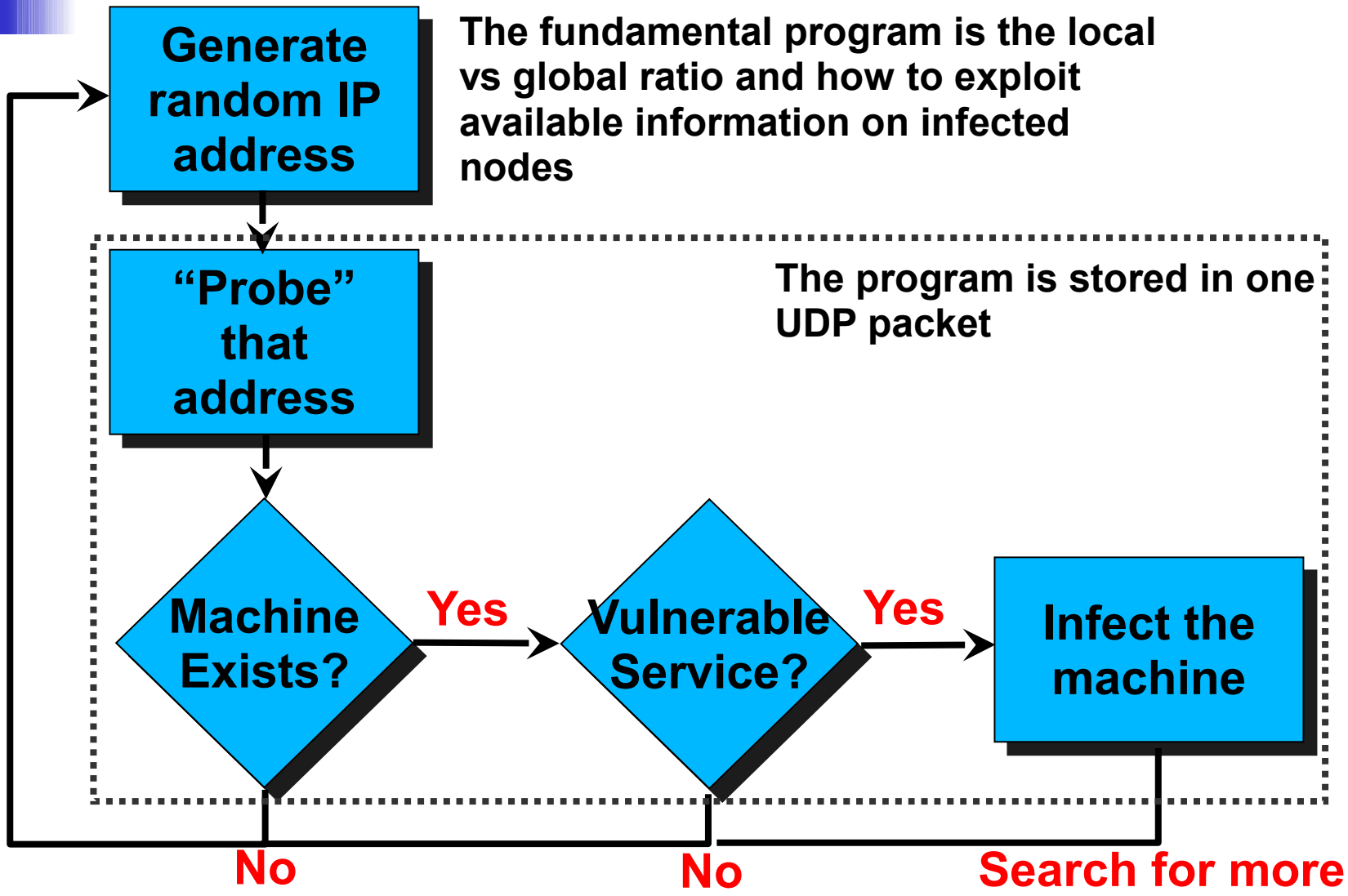
- Domain flux = generates alternative domains and nodes in these domains to download the updates
- Input/output connessions are encrypted
- Payload = information collection + creation of a botnet
- Botnet= overlay network including the nodes that have been attacked. It is controlled by the worm creator rather than by the legal owner

Some statistics

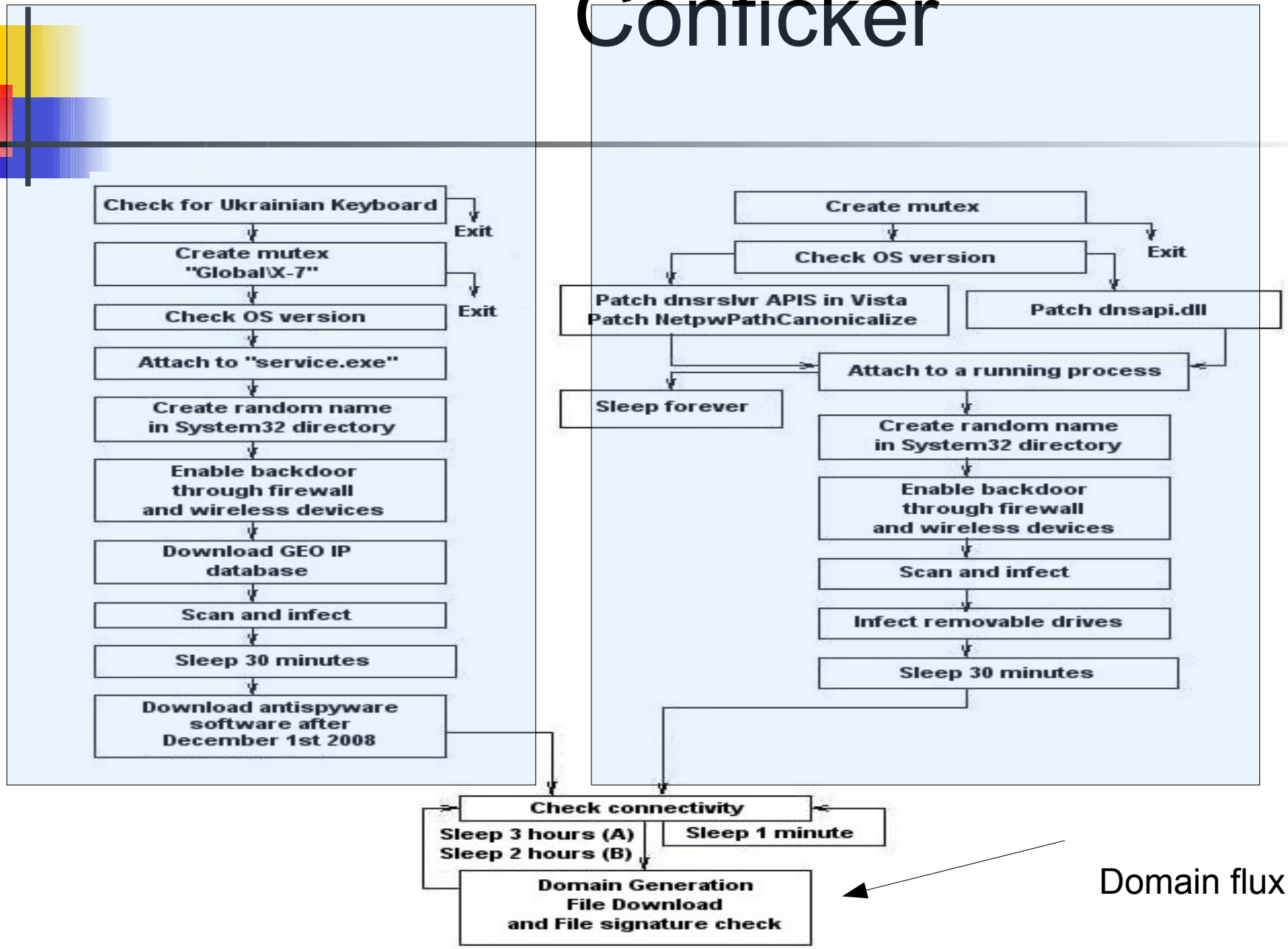
FIGURE 10. Computers cleaned by threat category, in percentages, 2H05–1H09



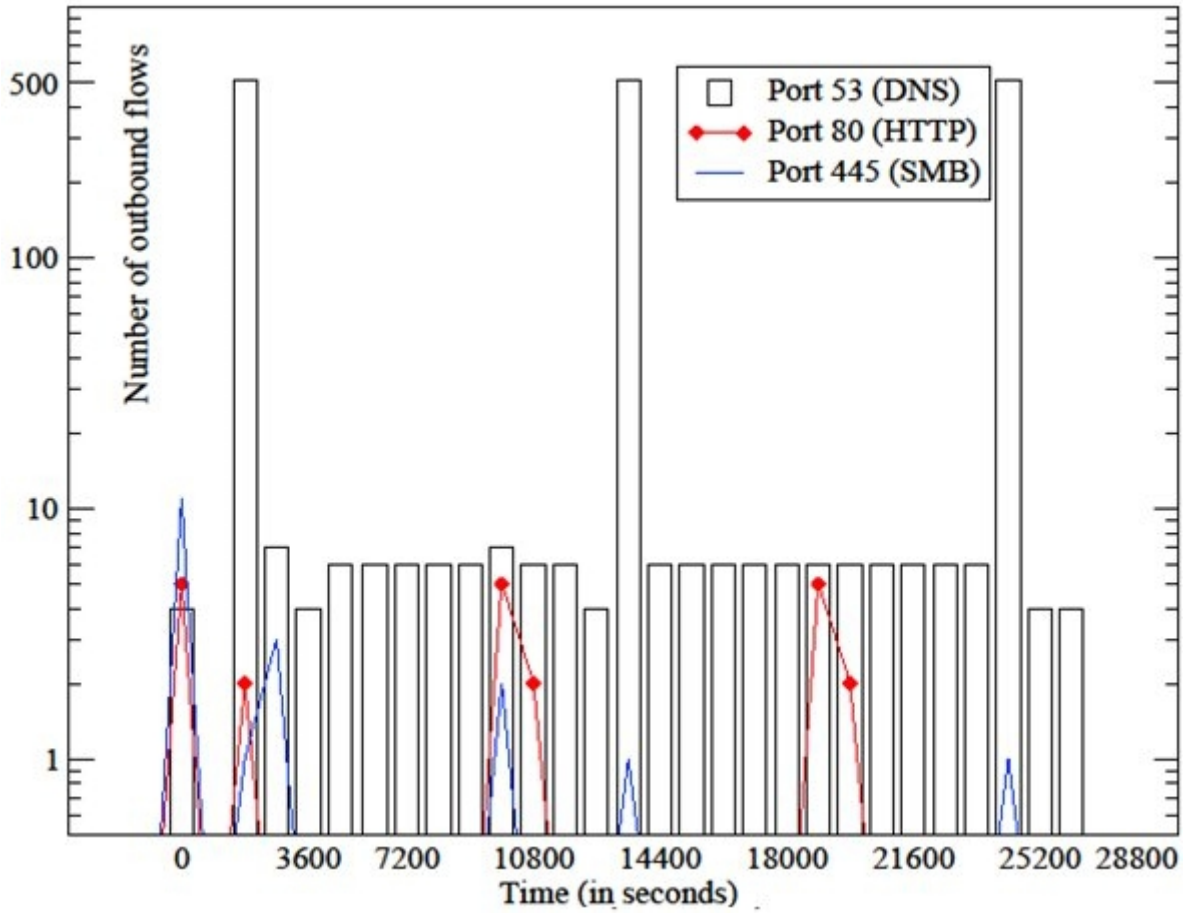
The general structure of a worm



Conficker


Version A
Version B
Check for Ukrainian Keyboard
Exit
Create mutex "Global\X-7"
Exit
Check OS version
Attach to "service.exe"
Create random name in System32 directory
Enable backdoor through firewall and wireless devices
Download GEO IP database
Scan and infect
Sleep 30 minutes
Download antispyware software after December 1st 2008
Create mutex
Exit
Check OS version
Patch dnsrslvr APIS in Vista
Patch NetpwPathCanonicalize
Patch dnsapi.dll
Attach to a running process
Sleep forever
Create random name in System32 directory
Enable backdoor through firewall and wireless devices
Scan and infect
Infect removable drives
Sleep 30 minutes
Check connectivity
Sleep 3 hours (A)
Sleep 2 hours (B)
Sleep 1 minute
Domain Generation
File Download
and File signature check
Domain flux

Conficker

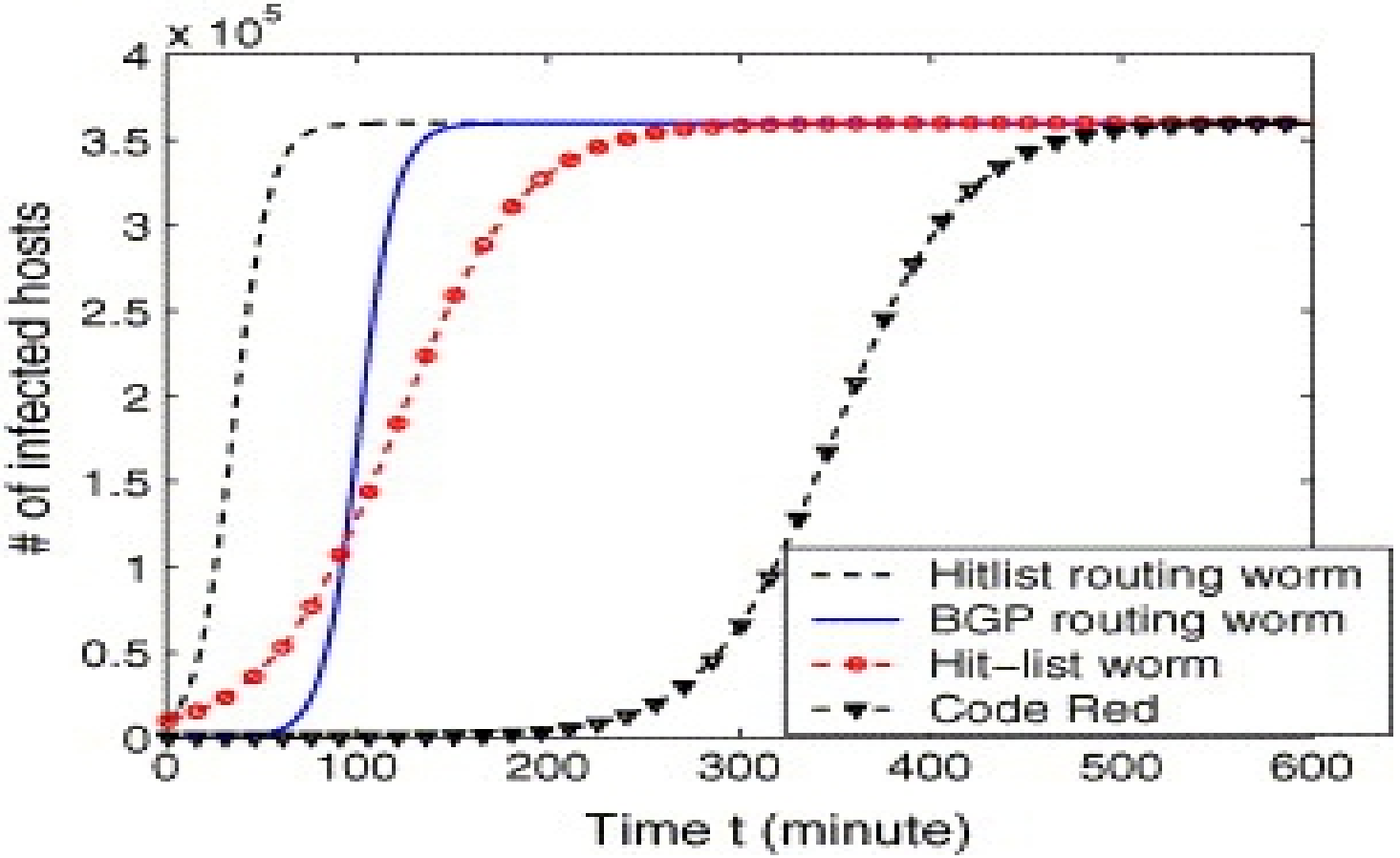


Generation of IP addresses in an infected nodes

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 percentage 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

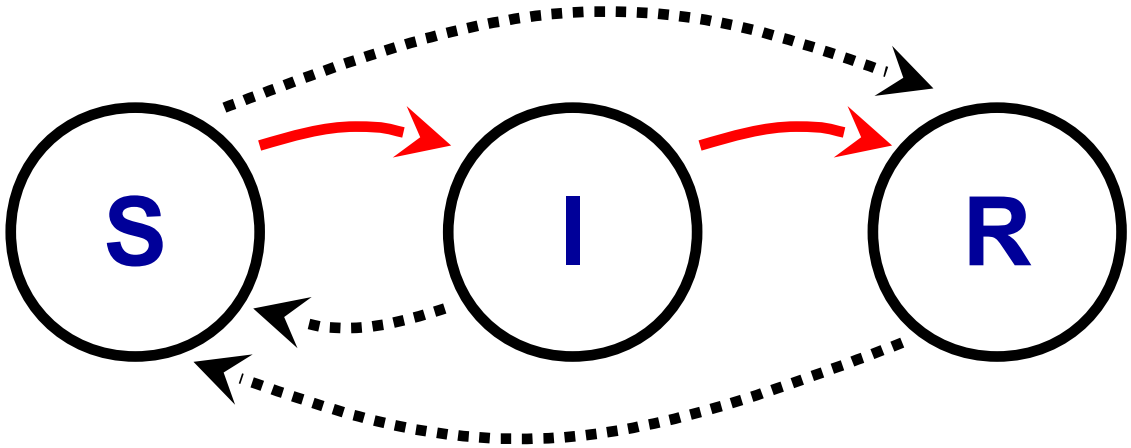
The influence of the ratio



A theoretical model

- Let us discuss a theoretical model to study the spreading of a worm
- The model is epidemiological = it has been defined to evaluate the number of people infected overtime
  - because of a contagious illness
  - in a closed population

A finite state model of individual to study the spreading



Model states

- **susceptible** = Host that may be infected
- **Infected** = Infected host
- **Recovered** = Host that cannot be infected

Typical transition sequences (red arrows)

- The host runs the software that is vulnerable (potential).
- The worm has exploited the vuln and successfully attacked the node (infected).
- The infection is detected and the system reconfigured (recovered).

A set of diff equations

Classic epidemiology

- [Kermack and McKendrick, 1927]
- Alll the nodes follows the red paths in the automata (P tlo I, I to R)

$$\frac{\mathrm{d}s}{\mathrm{d}t} = -\beta si$$

$$\frac{\mathrm{d}i}{\mathrm{d}t} = \beta si - \gamma i$$

$$\frac{\mathrm{d}r}{\mathrm{d}t} = \gamma i$$

s = potentially infected

i = infected

r = recovered

Beta = *infection rate*

Gamma = recovery rate

Gamma may be neglected in the case of worms because the time to spread is very litte

Kermack and McKendrick model

- β is a function of
  - The function to generate the IP addresses
  - The number of the system affected by the vulns
  - It increase with the virulence
  - The model assume that a node can infected any other node eg a fully connected system is assumed (no filter, no protection)
- γ should not be neglected anytime
  - The spreading is rather slow
  - There are some automatic components to detect and remove the infected nodes

Epidemiological threshold

$R_0 = \beta s / \gamma$

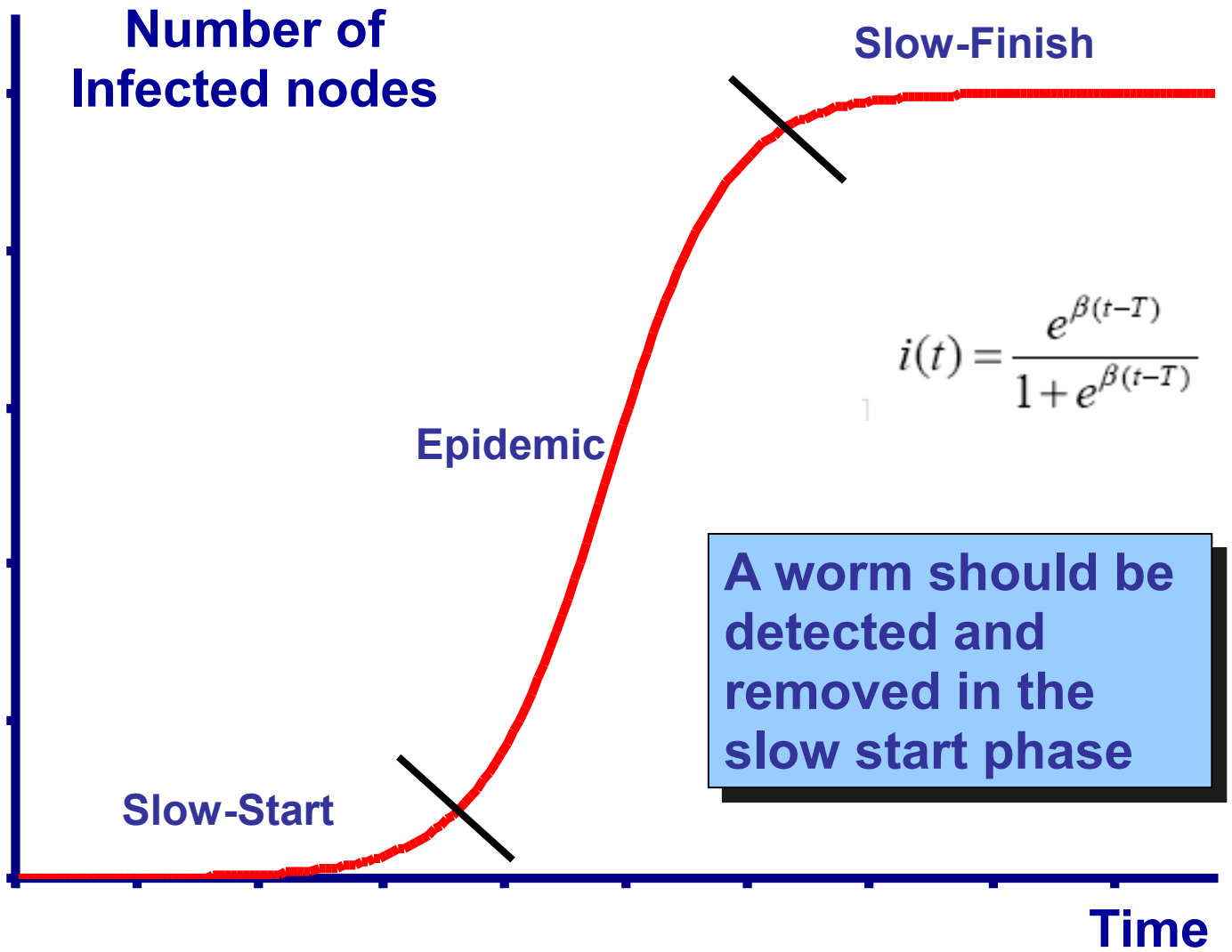
- s= percentage of nodes that may be infected
- It is the average number of nodes infected by an infected node
- If $R_0 > 1$ the worm spreads, otherwise it will be defeated

Solution of the system of diff equations

- No exact solution can be computed
- Anytime the initial number of infected may be neglected ($I(0) \cong 0$) then

$$I(t+\Delta t) = I(t) + \rho c S(t)\frac{I(t)}{N}\Delta t - I(t)\gamma\Delta t + o(\Delta t).$$

Solution = logistic function



$$i(t) = \frac{e^{\beta(t-T)}}{1+e^{\beta(t-T)}}$$

A worm should be detected and removed in the slow start phase

A model that consider patching

$$dS(t)/dt = -\beta S(t)I(t) - dQ(t)/dt$$

$$dR(t)/dt = \gamma I(t)$$

$$dQ(t)/dt = \mu S(t)J(t) \quad \text{patched}$$

$$dJ(t)/dt = I(t)+R(t)$$

$$S(t) + I(t) + R(t) + Q(t) = N$$

There are two reasons why a node is no longer susceptible

1. It has been infected
2. It has been patched

The number of patched nodes is proportional to the susceptible and of infected ones

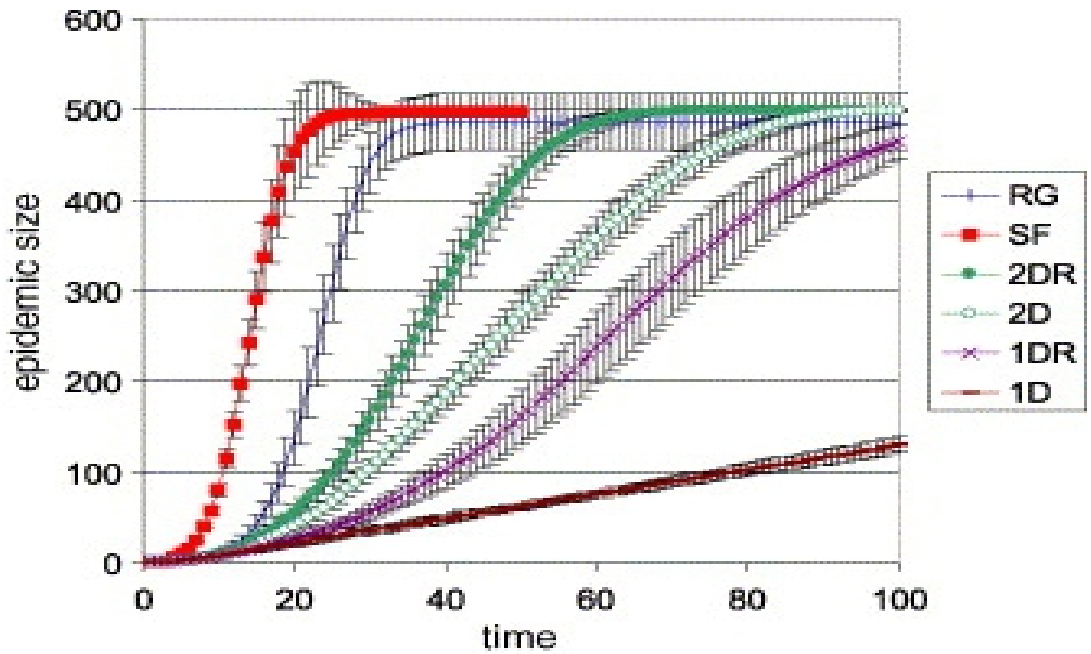
Further interesting models

- Let suppose that there is a partial connection among nodes (scale free, small world, …)
- Initially some nodes are infected
- We would like to know
  - How the connection structure influences the spreading and the parameter R_0
  - How patching (=vaccination) influences the spreading
  - Alternative vaccination strategy
- Several topologies may be be considered to discover how they influence the spreading

Scale free

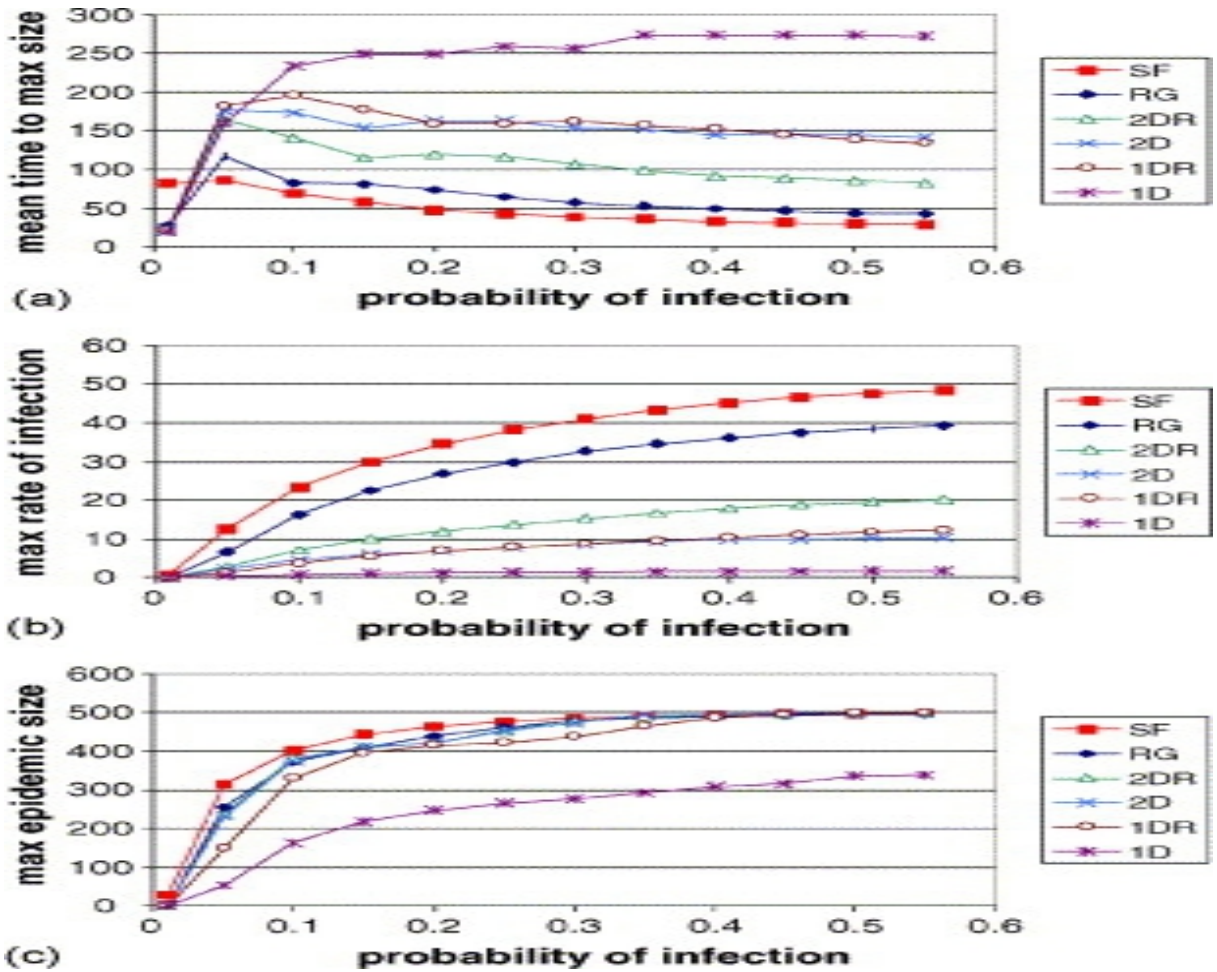
- Scale free
  - When a connection is created, nodes with a larger number of connections are preferred
  - The rich becomes richer
  - There are some network hubs with an exponential increase in the number of their connections
- Very robust with respect to random node attacks, highly fragile with respect to intelligent attacks i

Interconnection Topology



RG=random, SF=scale free, 2D= two dimensions lattice, 1D= one dimension lattice 2DR= two dimensions lattice rewired , 1DR= one dimension rewired

Other interesting values



Average time to max infected

Max infection rate

Number of infected

Computing a worm β

$$i(t) = \frac{e^{\beta(t-T)}}{1+e^{\beta(t-T)}}$$

$$\beta = \frac{C}{N} \times \frac{\alpha}{\tau}$$

Alpha

Tau

C = 1 (a random machine is selected)

C= N (an infected machine is always selected)

N = 2^{32} (size of IP address)

Alpha = number of nodes tested in parallel

Tau =average time for testing a machine

Code red

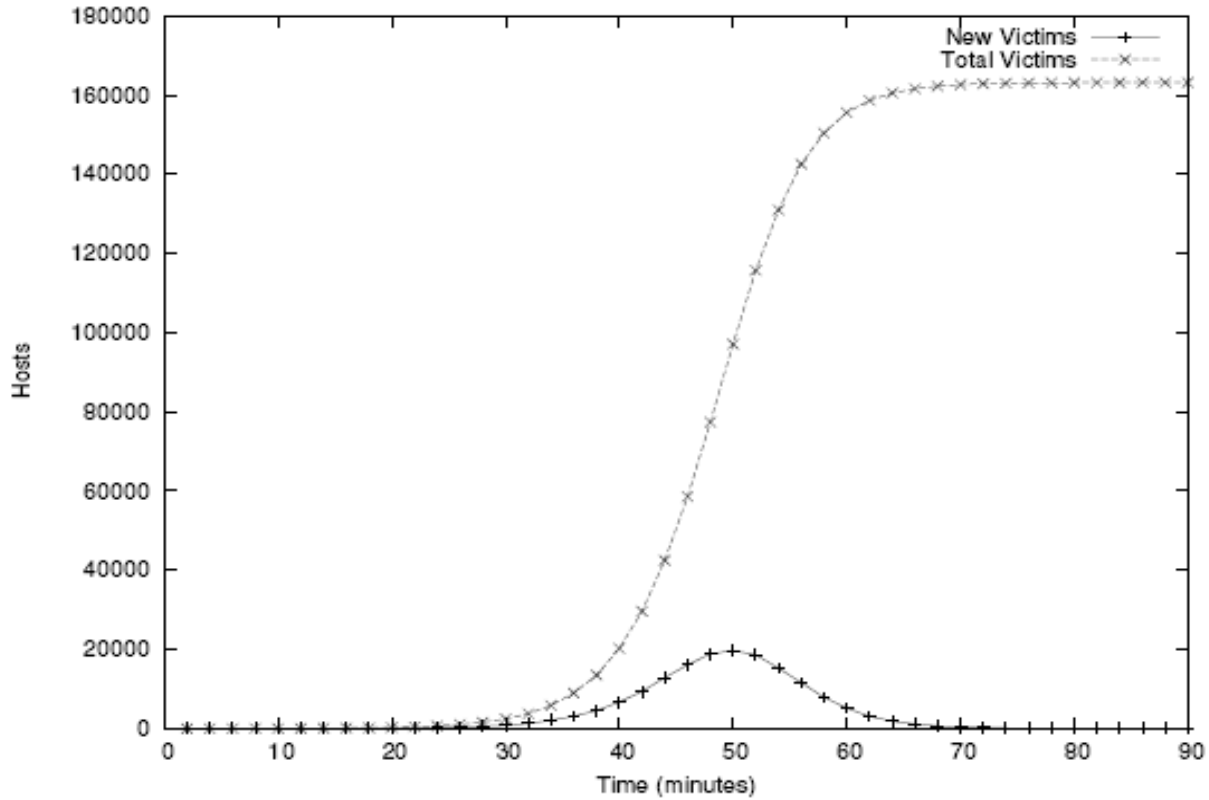
***Tau* = 19 seconds**

***Alpha* = 100**

$$\beta = \frac{1}{2^{32}} \times \frac{100}{19} = 1.23 \times 10^{-9}$$

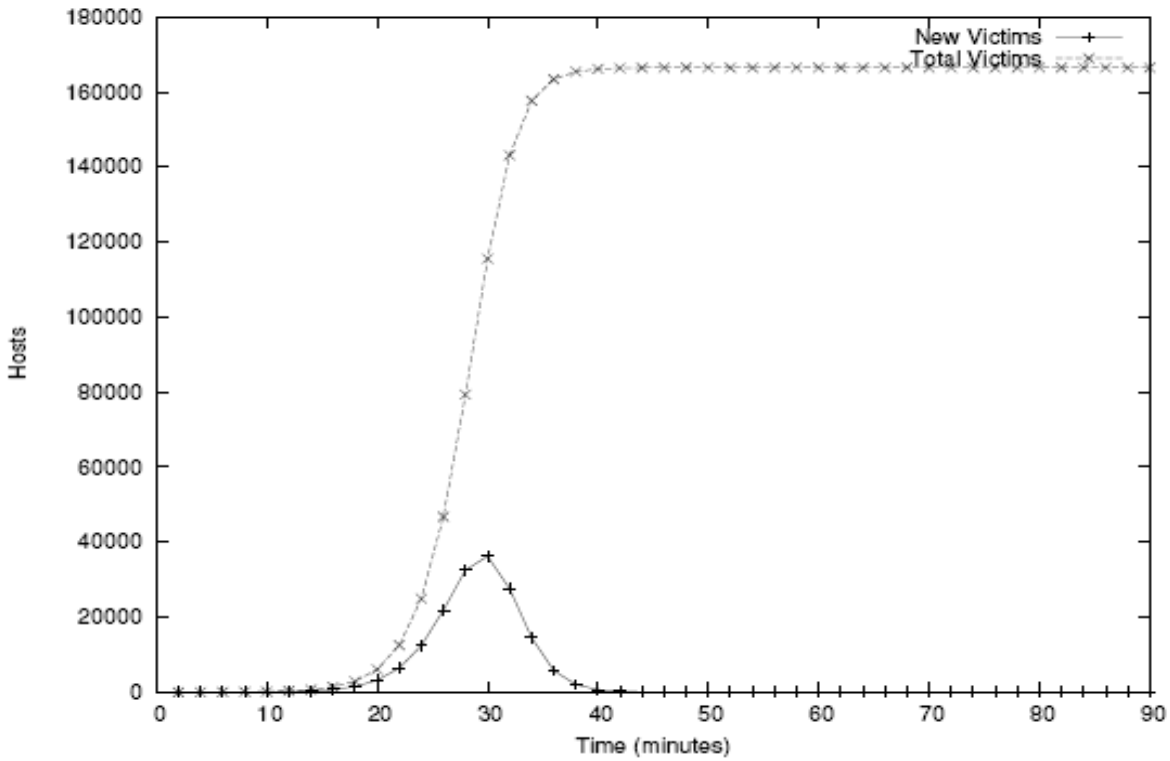
Good approximation

Spreading - I



10 thread in parallels and conflicts on nodes to be infected are neglected

Spreading - II



Optimization of the time out to detect that no node has the IP address that has been generated

Spreading - III

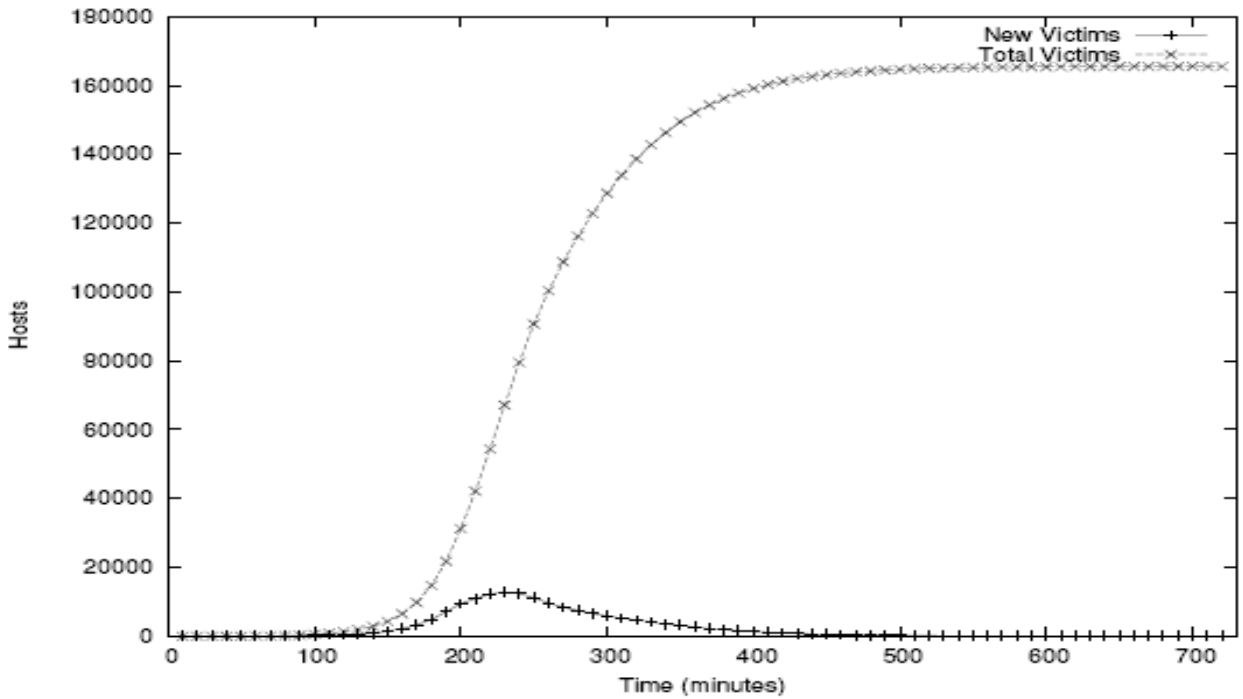


Figure 7: Local preference propagation

Local bias in the generation

Spreading - IV

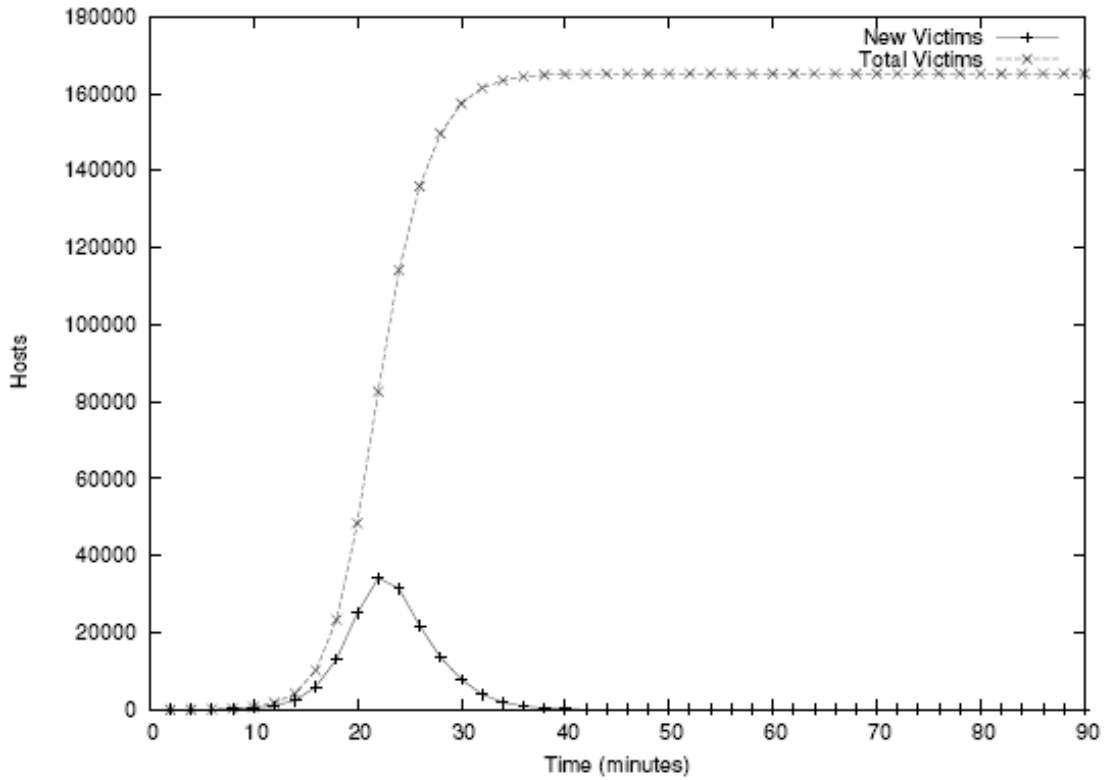
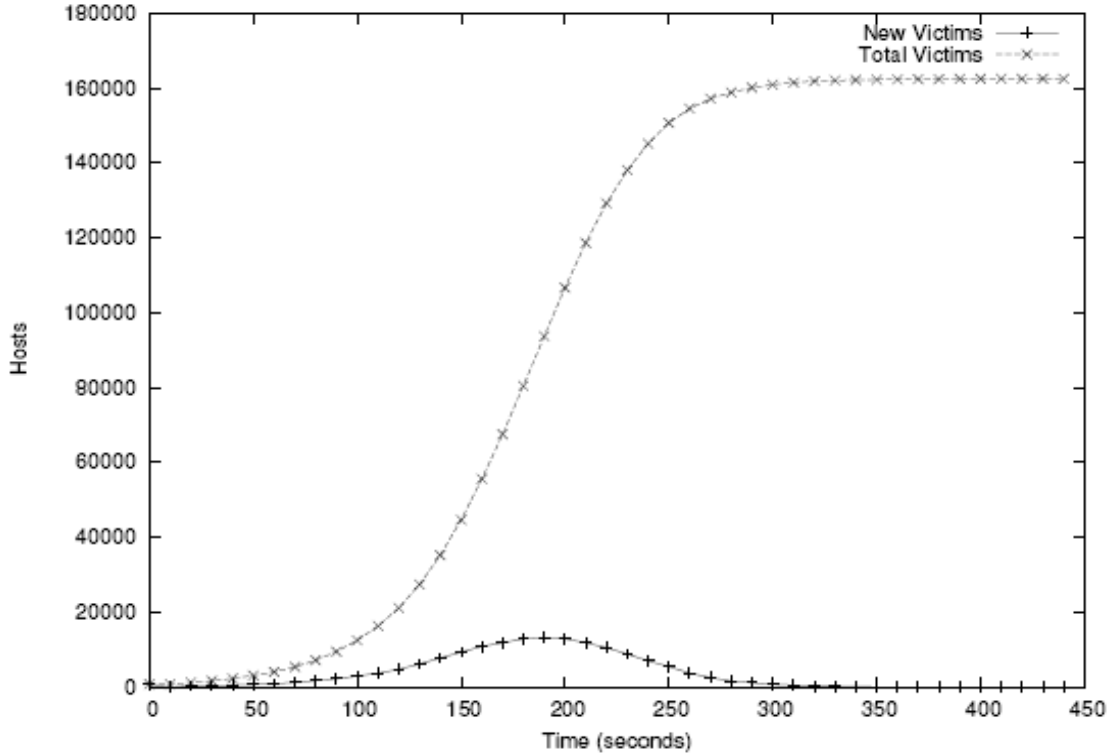


Figure 8: Local preference with multi-threading and short timeouts

Spreading - V



- prescan to find better subspaces to generate IP addresses and with a large number of susceptible nodes
- Infected nodes are remembered and neglected
- multithread

Local vs global

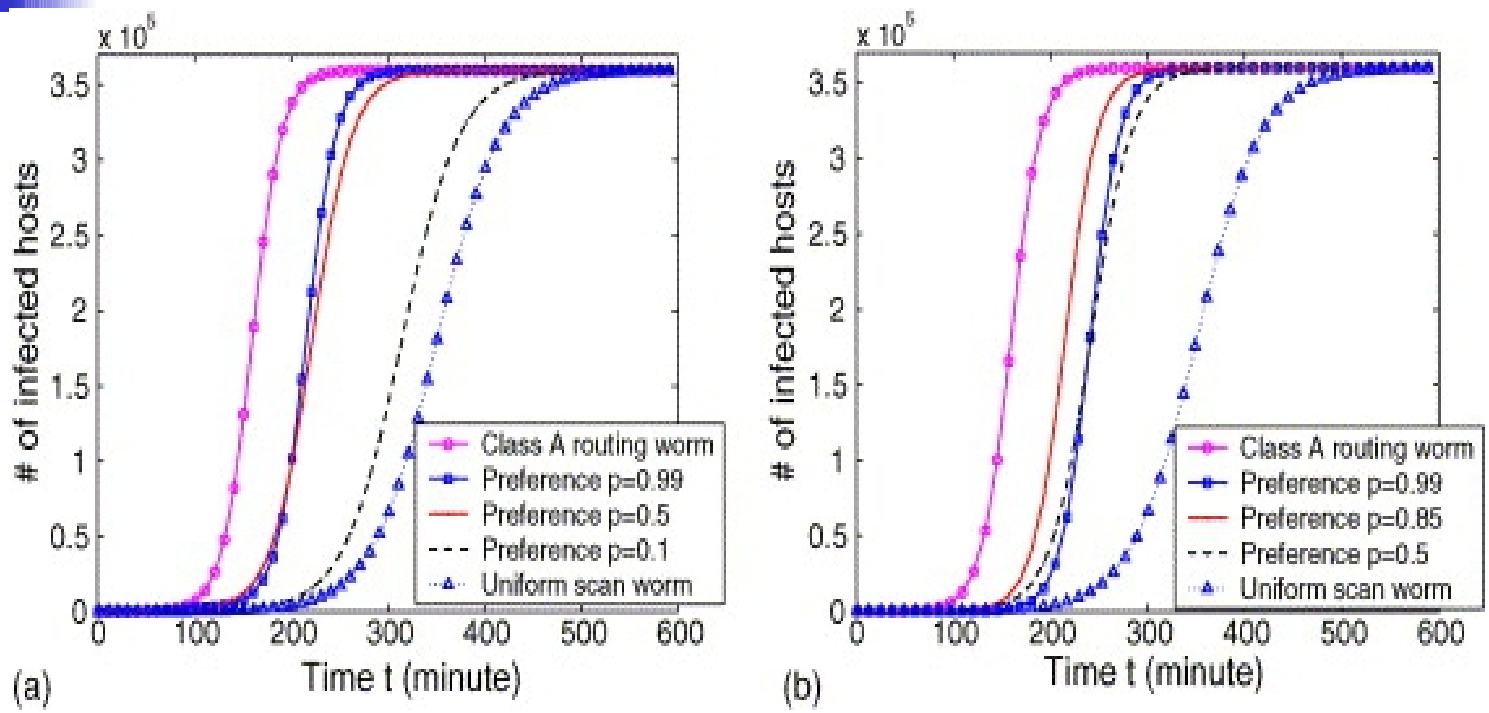
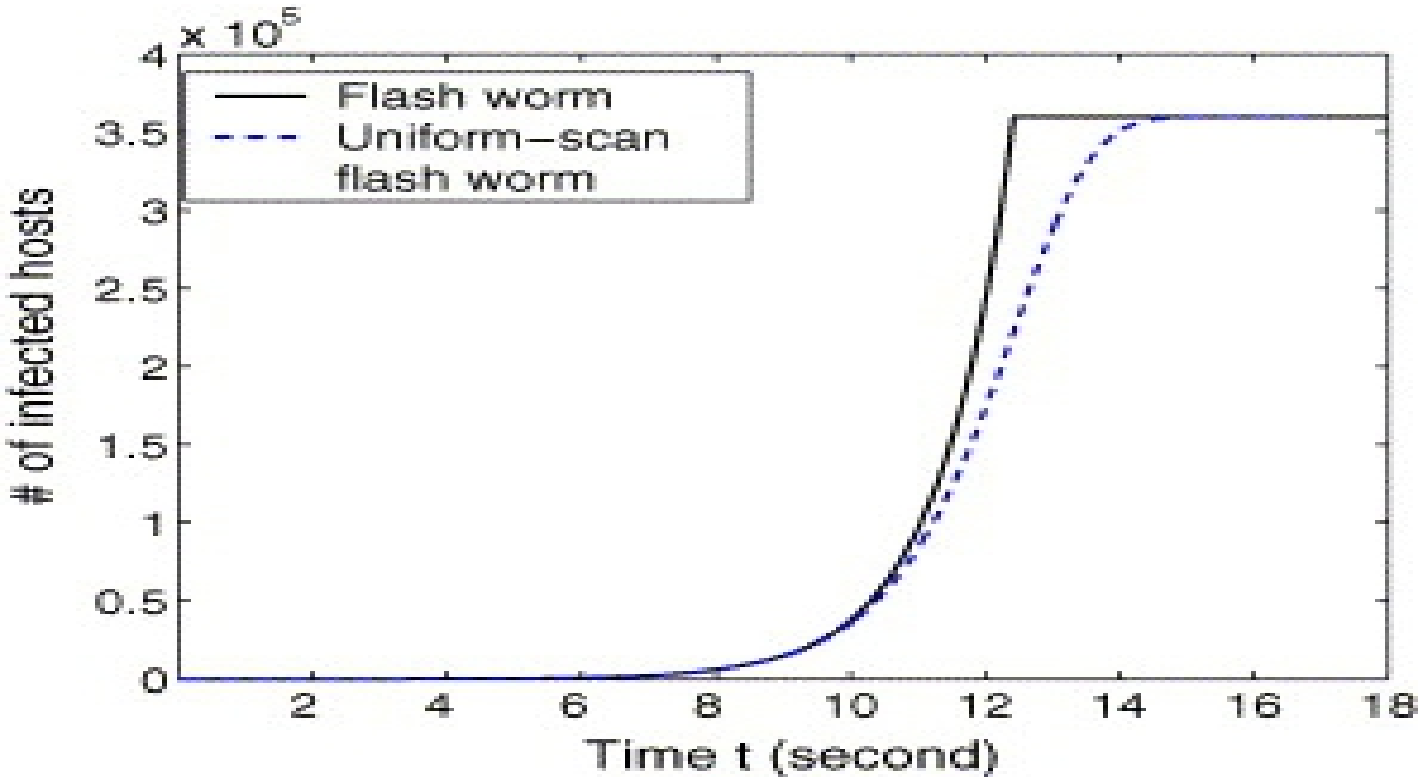


Fig. 5. Comparison of Code Red, a /8 routing worm, a local preference worm with different preference probabilities p.
(a) Local preference scan on “/8” network level (K=256, m=116).
(b) Local preference scan on “/16” network level (K=65,536, m=29,696).

Extreme optimization



The time scale has changed

Which address space?

- Some worms consider IP addresses
  - ⇒ Any node can infect any other nodes
  - ⇒ The addresses that are generated depend upon the adopted function and not upon the interconnection
- Some worms consider logical addresses, ie the email addresses
  - ⇒ A node can infect only nodes it already knowns
  - ⇒ The interconnection structure that has to be considered is the logical one

Trojan horse

- A program that has a different goal from the expected one
- Its main goal is to implement a backdoor to enable illegal accesses to the system
- Malware

Hybrid

- Most malware current integrates all the previous behavior
- Software with an opportunistic approach to spread to other nodes
  - Usb
  - Share
  - Mail
  - Attack
  - ....

Autonomous Hybrid

- They can transmit themselves to other nodes without exploiting the node resources
- Even if the node does not exchange email, it can
  - Trasmit email from the node
  - Hide in the mail

Symantec Global Internet Security Threat Report 2009

in 2008, there were six trojans in the top 10 new malicious code families detected. ***Three of the six trojans include a back door component and one includes a virus component.***
The remaining four families consist of worms,
one has a back door component and one has a virus component.

The previous edition of the Report noted that the prevalence of trojans is indicative of multistage attacks.
A multistage attack typically involves an initial compromise,followed by the installation of an additional piece of malicious code, such as a trojan that downloads and installs adware. As was the case in 2007, during this reporting period, five of the top 10 new malicious code families that were identified download additional threats.