



Malware: A Primer

Dimitris Gritzalis

November 2012
(updated May 2014)



Οικονομικό Πανεπιστήμιο Αθηνών

Τμήμα Πληροφορικής

Ιομορφικό Λογισμικό

Δημήτρης Γκρίτζαλης (dgrit@aub.gr)

Καθηγητής

Ασφάλειας στην Πληροφορική και τις Επικοινωνίες

Δομή περιεχομένων

Εισαγωγή - Βασικές έννοιες

Ιομορφικό Λογισμικό

Ιός, Αναπαραγωγός, Δούρειος Ιππος

Αλγοριθμική προσέγγιση

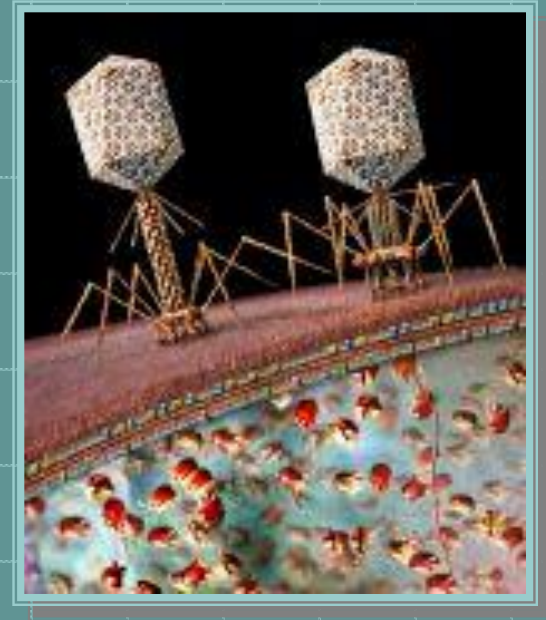
Μαθηματική θεμελίωση

Τεχνικές αντιμετώπισης προσβολών

Τεχνικές βασισμένες στην εμφάνιση

Τεχνικές βασισμένες στη συμπεριφορά

Βιβλιογραφία



Εισαγωγή – Βασικές έννοιες



Προσβολή

Προσβολή (intrusion) της ασφάλειας ενός πληροφοριακού συστήματος ονομάζεται η διαδικασία κατά την οποία ένα λογικό υποκείμενο εκτελεί συγκεκριμένες διαδικασίες, προκειμένου να αποκτήσει πρόσβαση σε πόρους του συστήματος, χωρίς να το δικαιούται.

Βασικά χαρακτηριστικά

- ✦ Αποτελούνται από σύνολα διαδικασιών, διαδοχικών ή διακοπτόμενων.
- ✦ Καθοδηγούνται απαραίτητα από κάποιο λογικό υποκείμενο.
- ✦ Δεν εξαρτώνται από κάποια συγκεκριμένη τεχνολογική πλατφόρμα.

Ιομορφικό λογισμικό

Ιομορφικό λογισμικό (viral software) ονομάζεται το λογικό αντικείμενο το οποίο, όταν ενεργοποιείται, προκαλεί την άμεση ή έμμεση εκτέλεση διαδικασιών οι οποίες δεν είναι γνωστές ούτε στο λογικό υποκείμενο που προκαλεί την ενεργοποίησή του, ούτε στο διαχειριστή του υπολογιστικού συστήματος.

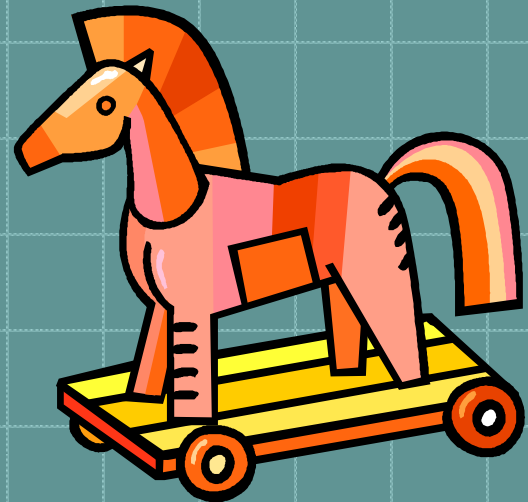
Βασικά χαρακτηριστικά

- ◆ Η δράση του δεν καθοδηγείται από κάποιο λογικό υποκείμενο (πιθανώς να ενεργοποιείται αρχικά από κάποιο λογικό υποκείμενο, πιθανώς να αυτοενεργοποιείται).
- ◆ Το ιομορφικό λογισμικό αποτελεί μορφή ενεργοποιήσιμου λογικού αντικειμένου, άρα πρέπει να είναι δομημένο κατάλληλα ώστε να μπορεί να ενεργοποιηθεί (συνεπώς πρέπει να είναι εκτελέσιμο αρχείο).
- ◆ Δεν εξαρτάται από κάποιο συγκεκριμένο τεχνολογικό περιβάλλον.

Μείζονες τύποι ιομορφικού λογισμικού

ΔΟΥΡΕΙΟΣ ΙΠΠΟΣ

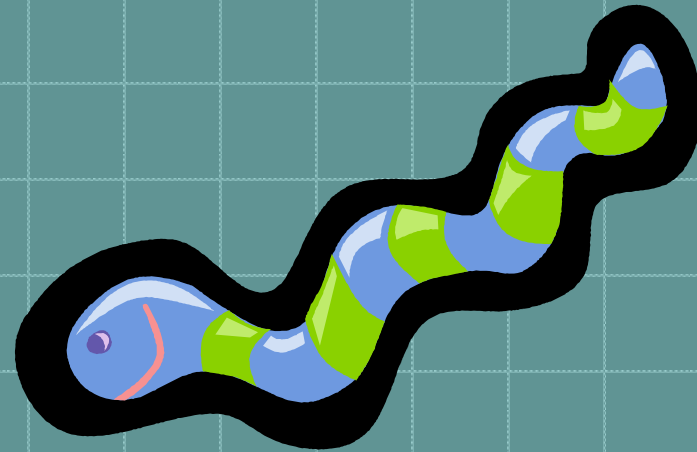
Δούρειος Ιππος (Trojan Horse) ονομάζεται κάθε μορφή ιομορφικού λογισμικού που έχει τη δυνατότητα να συνυπάρχει με κάποιο λογικό αντικείμενο (ξενιστής) και να προκαλεί συνέπειες άγνωστες στο χρήστη του αντικειμένου αυτού.



Μείζονες τύποι ιομορφικού λογισμικού

ΑΝΑΠΑΡΑΓΩΓΟΣ

Αναπαραγωγός (worm) ονομάζεται κάθε μορφή ιομορφικού λογισμικού που έχει τη δυνατότητα να διαδίδεται αυτοαναπαραγόμενο και να προκαλεί συνέπειες άγνωστες στους χρήστες του συστήματος.



Μείζονες τύποι ιομορφικού λογισμικού

ΠΡΟΓΡΑΜΜΑ ΙΟΣ

Πρόγραμμα ιός (virus) ονομάζεται το εκτελέσιμο λογισμικό που έχει τη δυνατότητα να προστίθεται και να συνυπάρχει με άλλο λογικό αντικείμενο (ξενιστής), να αναπαράγεται μέσω της ενεργοποίησης του αντικειμένου αυτού και να προκαλεί συνέπειες άγνωστες στο χρήστη του αντικειμένου.



Μηχανισμοί ιομορφικού λογισμικού

■ Διαδικασία πρόκλησης συνεπειών (Illicit action mechanism - **Act**)

Η διαδικασία αυτή προκαλεί στο σύστημα τις συγκεκριμένες συνέπειες που έχει καθορίσει ο δημιουργός του λογισμικού.

■ Διαδικασία ενεργοποίησης (Threshold mechanism - **Thr**)

Η διαδικασία αυτή καθορίζει πότε θα ενεργοποιηθεί η διαδικασία της πρόκλησης των συνεπειών.

■ Διαδικασία αναπαραγωγής (Replication mechanism - **Rep**)

Η διαδικασία αυτή προκαλεί την αυτοαναπαραγωγή του ιομορφικού λογισμικού.

■ Διαδικασία ένταξης (Persistence-in-host mechanism - **Per**)

Η διαδικασία αυτή επιτρέπει στον κώδικα του ιομορφικού λογισμικού να προστίθεται και να συνυπάρχει με κάποιο άλλο λογικό αντικείμενο, με τη μορφή ενός ενιαίου λογικού αντικειμένου.

Ειδοποιοί μηχανισμοί κάθε τύπου ιομορφής

Τύπος ιομορφικού λογισμικού	Μηχανισμός			
	Act	Thr	Rep	Per
Πρόγραμμα Ιός (virus)	NAI	NAI	NAI	NAI
Αναπαραγωγός (worm)	NAI	NAI	NAI	OXI
Δούρειος Ιππος (trojan horse)	NAI	NAI	OXI	OXI

Δευτερεύοντες τύποι ιομορφών

Τύπος	Βασικό χαρακτηριστικό
Ιός τομέα εκκίνησης (Boot Sector virus)	Τοποθετείται στον τομέα εκκίνησης ενός σκληρού δίσκου.
Παρασιτικός ιός (Parasitic virus)	Μολύνει εκτελέσιμα προγράμματα και εισάγει τον κώδικα του πριν ή μετά τον κώδικα του προγράμματος.
Πολυμερής ιός (Multipartite virus)	Μολύνει τομείς εκκίνησης ή εφαρμογές.
Ιός παραμένων στην κύρια μνήμη (Resident virus)	Παραμένει στην κύρια μνήμη μετά την εκτέλεση του προγράμματος ή την εκκίνηση του δίσκου που έχει μολύνει.
Αποκρυπτόμενος ιός (Stealth virus)	Αποκρύπτει τη μόλυνση αρχείων.
Κρυπτογραφημένος ιός (Encrypted virus)	Κρυπτογραφεί τον κώδικά του, εκτός από μια ρουτίνα κρυπτογράφησης.
Πολυμορφικός ιός (Polymorphic virus)	Αλλάζει μορφή κάθε φορά που μολύνει ένα πρόγραμμα.
Μακροϊός (Macro virus)	Αποτελείται από μια ακολουθία εντολών η οποία διερμηνεύεται (interpreted), αντί να εκτελείται.

Άλλες μορφές ιομορφικού λογισμικού

ΚΕΡΚΟΠΟΡΤΕΣ

Οι κερκόπορτες (**trapdoors** ή **backdoors**) είναι σημεία εισόδου που επιτρέπουν την πρόσβαση σε ένα σύστημα, παρακάμπτοντας τη συνήθη διαδικασία πρόσβασης ασφαλείας. Τοποθετούνται από προγραμματιστές, για νομότυπους σκοπούς, ή από εισβολείς.

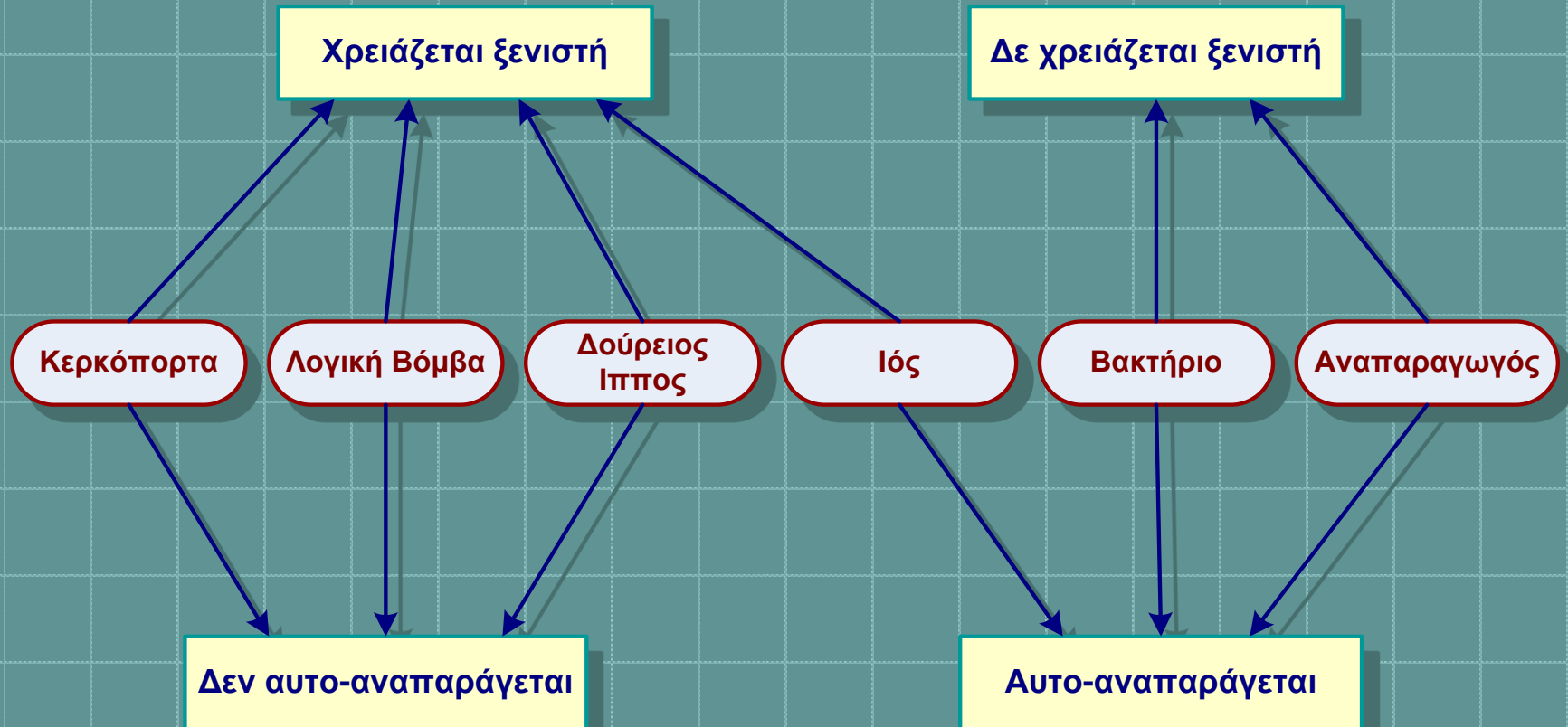
ΛΟΓΙΚΕΣ ΒΟΜΒΕΣ

Μια λογική βόμβα (**logic bomb**) είναι λογισμικό που εκτελεί μια ενέργεια που αντίκειται στην πολιτική ασφάλειας ενός συστήματος, όταν πληρείται μια λογική συνθήκη στο σύστημα ή σε συγκεκριμένη χρονική στιγμή (**time bomb**).

ΒΑΚΤΗΡΙΑ

Τα **βακτήρια** (**bacteria**) είναι λογισμικό το οποίο αναπαράγεται όπως και οι ιοί, αλλά δεν απαιτεί την ύπαρξη ξενιστή. Παρόλο που δεν προκαλεί κάποια επιζήμια ενέργεια, καταναλώνει πόρους του συστήματος.

Ταξινόμηση ιομορφικού λογισμικού



Αλγοριθμική προσέγγιση



Αλγόριθμος τυπικού ιού

Program Test-virus:=

{Identification("virus");

 Main-program:=

 {Infect-executable-file;

 If (threshold-process) then Do-illicit-
 action;

 Goto Next }

*

Subroutine Infect-executable-file:=

{Loop: file=random-executable;

 If (first-line-of-file=Identification("virus"))
 then Goto Loop;

 else Prepend Test-virus to file }

* ...

...

Subroutine Do-illicit-action:=

{ whatever action the virus designer
 desires }

*

Subroutine Threshold-process

{ whatever trigger the virus designer has
 chosen }

*

Subroutine Prepend

{ actions required to prepend a file to
 another }

Next }

Execute executable-file

Αλγόριθμος εναλλασσόμενου ιού

Program Alternative-virus:=

{Identification (Select at random a pattern "pati" from the set PAT={pat1,...,patk});

Main-program:=

{ Infect-executable-file;

If (threshold-process) then Do-illicit-action;

Goto Next}

*

Subroutine Infect-executable-file:=

{Select at random an infection strategy infi, from the set IS={inf1,...,infn}

{Loop: file=random-executable;

If (first-line-of-file=Identification("pati"))

Then Goto Loop;

Else Prepend Alternative-virus to file}

*

Subroutine Do-illicit-action:=

{Select at random an illicit action ili from the set IL={il1,...,ilk}}

*

Subroutine Threshold-process

{Select at random a trigger tri from the set

TR={tr1,...,trm}}

*

Subroutine Prepend

{actions required to prepend a file to another}

Next}

Execute executable-file

Αλγόριθμος μεταλλασσόμενου ιού

Program Evolutionary-virus:=

Main-program:=

```
{ Copy-virus-with-insertions
  Infect-executable-file;
  If (threshold-process) then Do-
  illicit-action;
  Goto Next }
```

*

Subroutine Print-random-
statement:=

```
{ Print (random-variable-name,
  "=", random-variable-name);
  Loop: If random-bit=1 then
  { Print (random-operator,
  random variable-name);
  Goto Loop; }
  Print (semicolon) }
```

* ...

...Subroutine copy-virus-with-insertions:=

```
{ Loop: Copy evolutionary-virus to virus, till semicolon;
  If random-bit=1 then print-random-statement;
  If NOT(end-of-input-file) then Goto Loop }
```

*

Subroutine Infect-executable-file:=

```
{ Loop: file=random-executable;
  Prepend Evolutionary-virus to file }
```

*

Subroutine Do-illicit-action:=

```
{ whatever action the virus designer desires }
```

*

Subroutine Threshold-process

```
{ whatever trigger the virus designer has chosen }
```

*

Subroutine Prepend

```
{ actions required to prepend a file to another }
```

Next }

Execute executable-file

Αλγόριθμος ελεγχόμενου αναπαραγωγού

Program Test-worm

{Main-program:=

{If (threshold-process) then Do-illicit-action;

else Replicate-itself;

Goto Next}

*

Subroutine Replicate-itself:=

{Count:=1;

Do until count=N;

system=random-system;

Copy Test-worm to system, with random
name;

N:=N+1;

End until}

*...

...Subroutine Do-illicit-action:=

{whatever action the worm designer
desires}

*

Subroutine Threshold-process

{whatever trigger the worm designer
has chosen}

*

Next}

Αλγόριθμος μη ελεγχόμενου αναπαραγωγού

Program Overloading-worm

{Main-program:=

{If (threshold-process) then Do-illicit-action;
else Replicate-itself;

Goto Next}

*

Subroutine Replicate-itself:=

{Loop: system=random-system;
Copy overloading-worm to system, with
random name}

*

Subroutine Do-illicit-action:=

{whatever action the worm designer desires}

*...

...

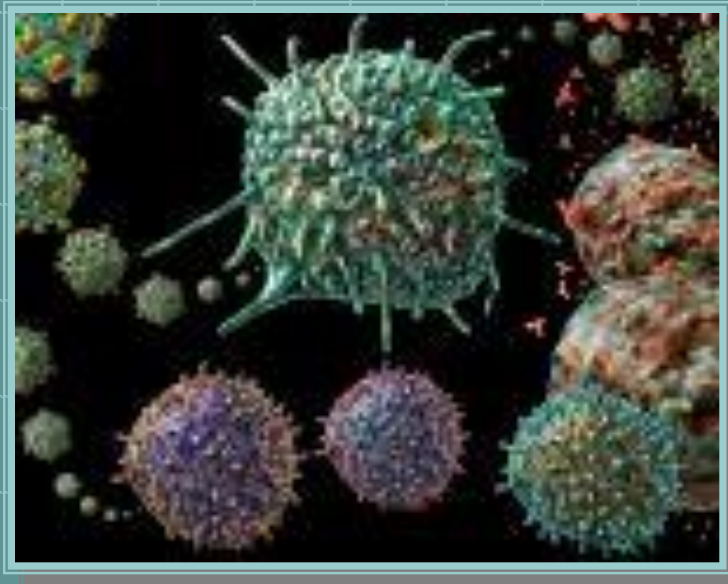
Subroutine Threshold-process

{whatever trigger the worm designer
has chosen}

*

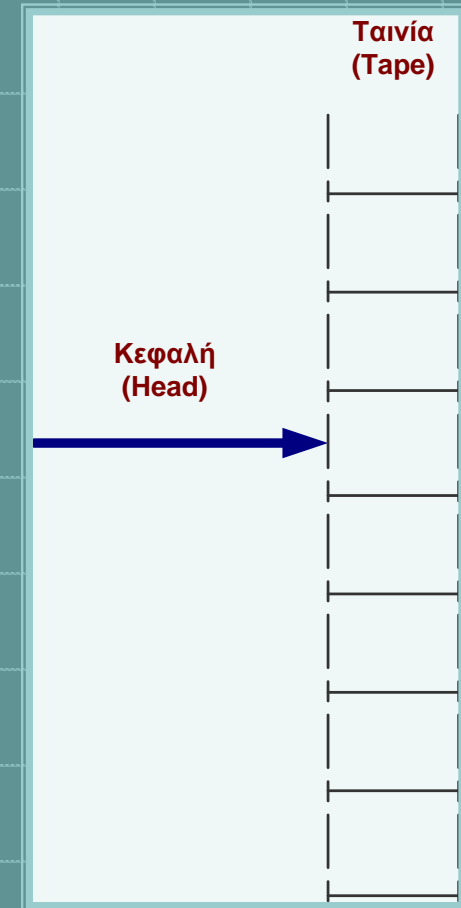
Next}

Μαθηματική Θεμελίωση



Η έννοια της μηχανής Turing

- **S** το σύνολο των καταστάσεων μιας μηχανής, δηλαδή $S = \{s_0, \dots, s_v\}$, $v \in \mathbb{N}$.
- **I** το σύνολο των στοιχείων εισόδου της μηχανής (των επιτρεπτών συμβόλων προς επεξεργασία), δηλαδή $I = \{i_0, \dots, i_k\}$, $k \in \mathbb{N}$.



Βασικές συναρτήσεις

- **Συνάρτηση εξόδου (Output function) $O(s,i)$** μιας μηχανής, ονομάζεται μια συνάρτηση $O:S \times I \Rightarrow I$ που αντιστοιχεί μια συγκεκριμένη κατάσταση s μιας μηχανής και ένα στοιχείο εισόδου i της μηχανής αυτής, σ' ένα παραγόμενο στοιχείο εξόδου της μηχανής.
- **Συνάρτηση επόμενης κατάστασης (Next state function) $N(s,i)$** μιας μηχανής ονομάζεται μια συνάρτηση $N:S \times I \Rightarrow S$ που αντιστοιχεί μια συγκεκριμένη κατάσταση s μιας μηχανής και ένα στοιχείο εισόδου i της μηχανής αυτής, στη νέα κατάσταση που προκύπτει μετά την είσοδο του i .
- **Συνάρτηση κατεύθυνσης (Direction) $D(s,i)$** μιας μηχανής ονομάζεται μια συνάρτηση $D:S \times I \Rightarrow k$, $k \in \{-1, 0, 1\}$, που αντιστοιχεί μια συγκεκριμένη κατάσταση s μιας μηχανής και ένα στοιχείο εισόδου i της μηχανής αυτής, στην κίνηση της ταινίας της μηχανής κατά μία θέση δεξιά (-1), αριστερά (+1) ή στην ακινησία της ταινίας (0).

Μηχανή Turing

- Υπολογιστική μηχανή ή Μηχανή Turing (computing machine) ονομάζεται κάθε μηχανή M εφοδιασμένη με το σύνολο $T = \{S, I, O, N, D\}$, όπου S το σύνολο των καταστάσεών της, I το σύνολο των στοιχείων εισόδου της, O η συνάρτηση εξόδου της, N η συνάρτηση της επόμενης κατάστασής της και D η συνάρτηση κατεύθυνσης της ταινίας της.
- Ένα σύνολο μηχανών TM ονομάζεται **σύνολο υπολογιστικών μηχανών ή μηχανών Turing** αν: $\forall M [M \in TM] \Leftrightarrow M: (S_M, I_M,$
 $O_M: S_M \times I_M \Rightarrow I_M,$
 $N_M: S_M \times I_M \Rightarrow I_M,$
 $D_M: S_M \times I_M \Rightarrow k_M)$

Ιστορία μιας μηχανής Turing

- ⊙ Η (αναδρομική) συνάρτηση $S(t)$ της *κατάστασης* (state) της υπολογιστικής μηχανής, μετά παρέλευση χρόνου t , που ορίζεται ως: $S_M: N \Rightarrow S_M$
- ⊙ Η (αναδρομική) συνάρτηση $P(t)$ της *θέσης* (position) της υπολογιστικής μηχανής, μετά παρέλευση χρόνου t , που ορίζεται ως: $P_M: N \Rightarrow N$
- ⊙ Η (αναδρομική) συνάρτηση $C(c, \tau)$ των *περιεχομένων* (content) ενός στοιχείου της ταινίας, μετά παρέλευση χρόνου t και δεδομένου του αύξοντα αριθμού τ του στοιχείου αυτού, που ορίζεται ως $C_M: N \times N \Rightarrow I_M$
- ⊙ **Ιστορία** H_M , μιας υπολογιστικής μηχανής M , ονομάζεται η διατεταγμένη τριάδα (S_M, C_M, P_M)

Πρόγραμμα Turing

- Ένα διατεταγμένο σύνολο συμβόλων v ανήκει στα **προγράμματα Turing** (Turing programs) TP_M μιας υπολογιστικής μηχανής, αν ισχύει:

$$\forall M \in TM [\forall v [\forall i \in I, v \in TP_M \Leftrightarrow v \in LS_M^i]]$$

όπου LS_M^i είναι επιτρεπτές καταστάσεις της μηχανής M για κάθε δεδομένο εισόδου i .

- **Σύνολο προγραμμάτων Turing** (Turing machine program set) TS_M , των προγραμμάτων μιας υπολογιστικής μηχανής M , ορίζεται το σύνολο των συμβολοσειρών V , για τις οποίες ισχύει:

$$\forall M \in TM [\forall V \in TS_M \Leftrightarrow [\exists v \in V \text{ και } v \in V: v \in TP_M]]$$

Ιομορφή (κατά Cohen)

Ενα διατεταγμένο ζεύγος (M, V) ορίζεται ως **Ιομορφή** (viral set) **VS** αν ισχύει:

$$\forall M \forall V, (M, V) \in VS \Leftrightarrow$$

$$[V \in TS] \text{ και } [M \in TM] \text{ και}$$

$$\forall v \in V [\forall H_M \forall t, j \in \mathbb{N} [$$

$$[P_M(t)=j \text{ και } S_M(t)=S_{M0} \text{ και } (C_M(t,j), \dots, C_M(t, j+|v|-1))=v] \Rightarrow$$

$$[\exists v' \in V [\exists t', t'', j' \in \mathbb{N}, \text{ όπου } t < t'' < t':$$

$$(1) [(j'+|v'|) \leq j \text{ ή } (j+|v|) \leq j'] \text{ και}$$

$$(2) C_M(t', j'), \dots, C_M(t', j'+|v'|-1))=v' \text{ και}$$

$$(3) P_M(t'') \in \{j', \dots, j'+|v'|-1\}]]]]$$



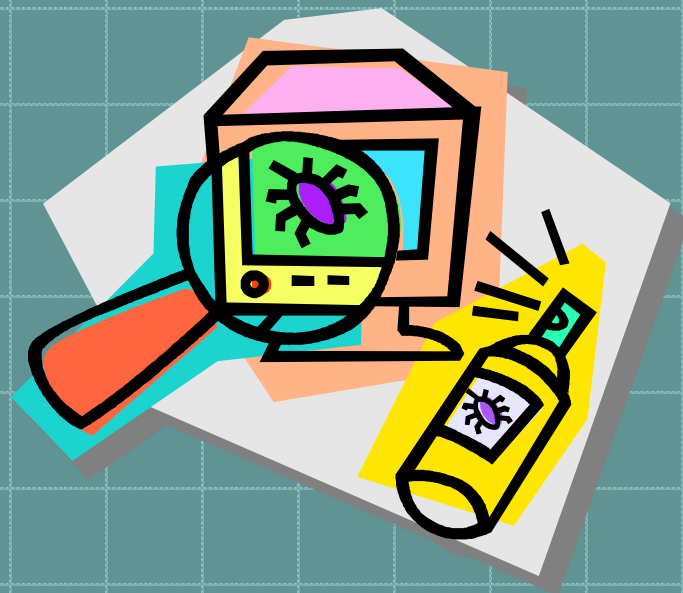
Θεωρήματα

1. Η ένωση οποιουδήποτε συνόλου ιών είναι ιός.
2. Το πλήθος των διαφορετικών ιών μιας υπολογιστικής μηχανής είναι άπειρο.
3. Το πλήθος των προγραμμάτων που δεν είναι ιοί, για μια συγκεκριμένη υπολογιστική μηχανή, είναι άπειρο.
4. Δεν είναι δυνατό να σχεδιασθεί υπολογιστική μηχανή ικανή να αποφανθεί, σε πεπερασμένο χρόνο, αν μια ακολουθία συμβόλων είναι ιός ή όχι.
5. Δεν υπάρχει πρόγραμμα το οποίο να μπορεί να ανιχνεύσει όλους τους ιούς μιας συγκεκριμένης υπολογιστικής μηχανής.

Θεωρήματα

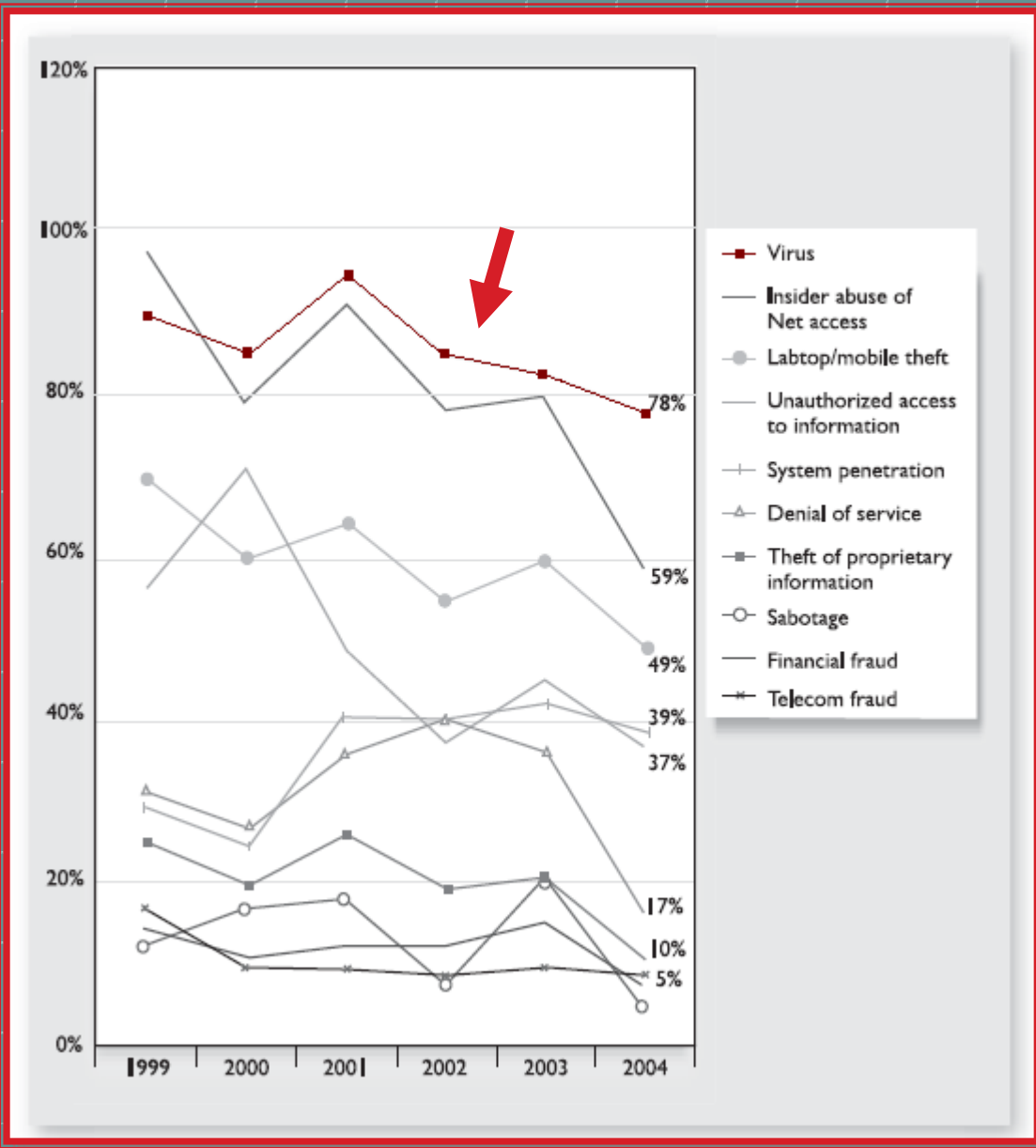
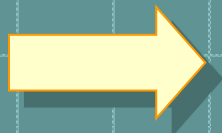
6. Δεν είναι δυνατή η σχεδίαση ενός προγράμματος, το οποίο να αποφαινεται, σε πεπερασμένο χρόνο, αν μια δεδομένη ακολουθία συμβόλων, που αποτελούν ένα πρόγραμμα για μια συγκεκριμένη μηχανή, είναι ιός ή όχι.
7. Η διαπίστωση αν μια δεδομένη ακολουθία συμβόλων, που είναι ιός, μπορεί να παραχθεί από μια άλλη δεδομένη ακολουθία συμβόλων, που επίσης είναι ιός, είναι μη επιλύσιμο (undecidable) πρόβλημα.
8. Δεν υπάρχει πρόγραμμα που μπορεί να εντοπίσει από ποιο πρόγραμμα-φορέα προκλήθηκε η προσβολή ενός προγράμματος.

Αντιμετώπιση προσβολών



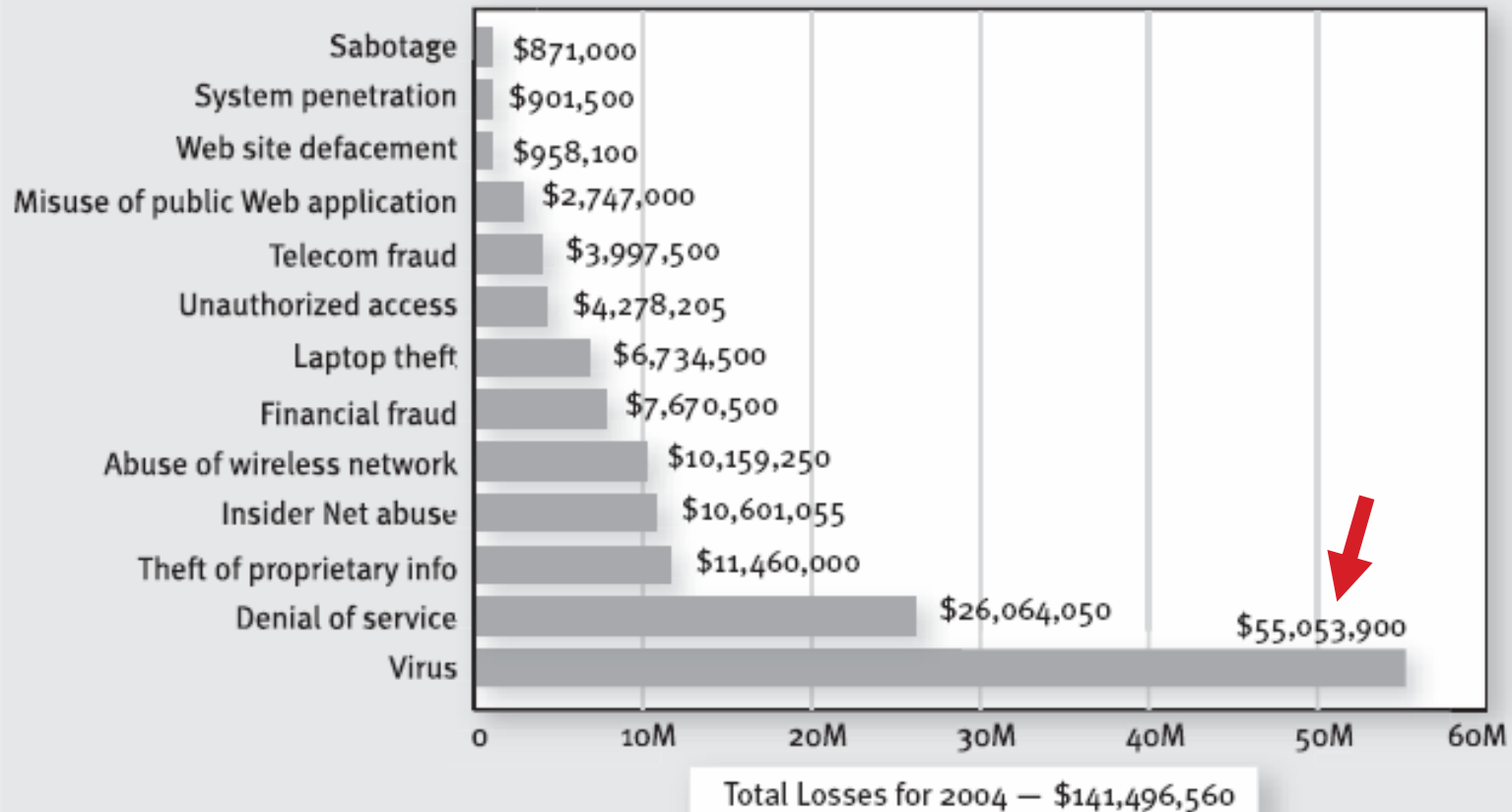
Συχνότητα εμφάνισης

Κατηγορίες εντοπισμένων επιθέσεων (1999-2004)



2004 CSI/FBI Computer Crime and Security Survey

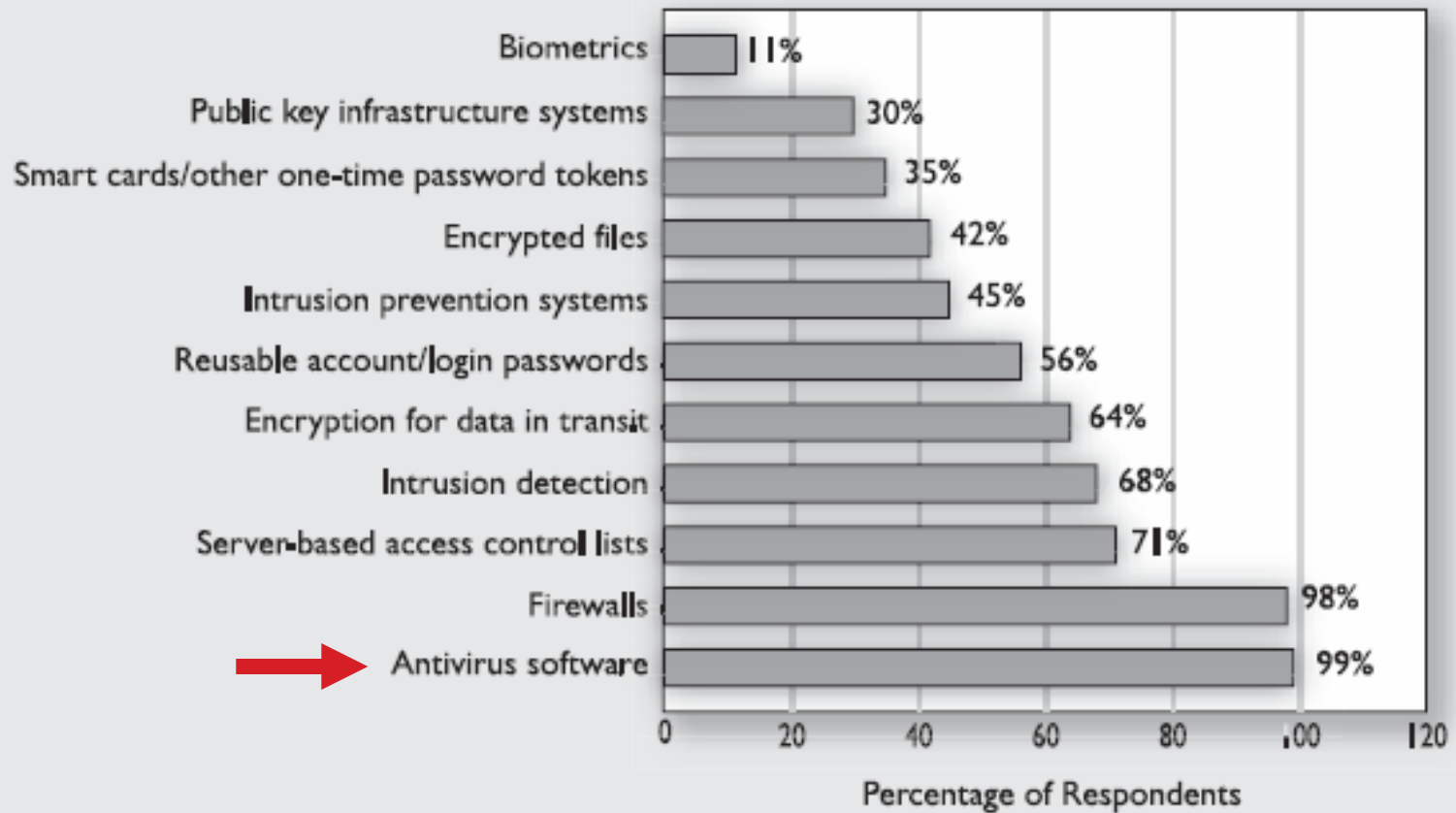
Οικονομικές απώλειες



2004 CSI/FBI Computer Crime and Security Survey

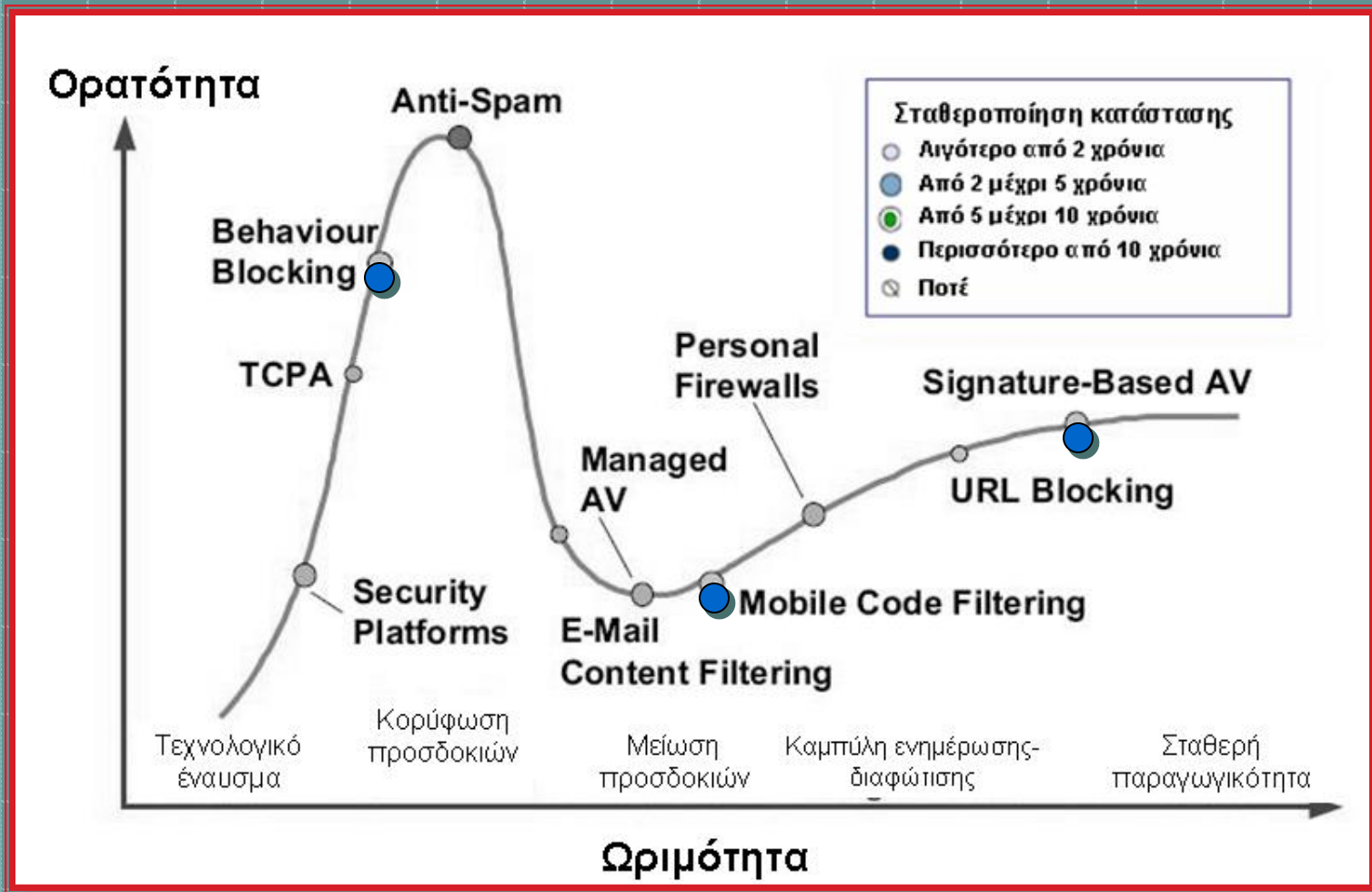
Malware: A Primer

Τεχνολογίες αντιμετώπισης

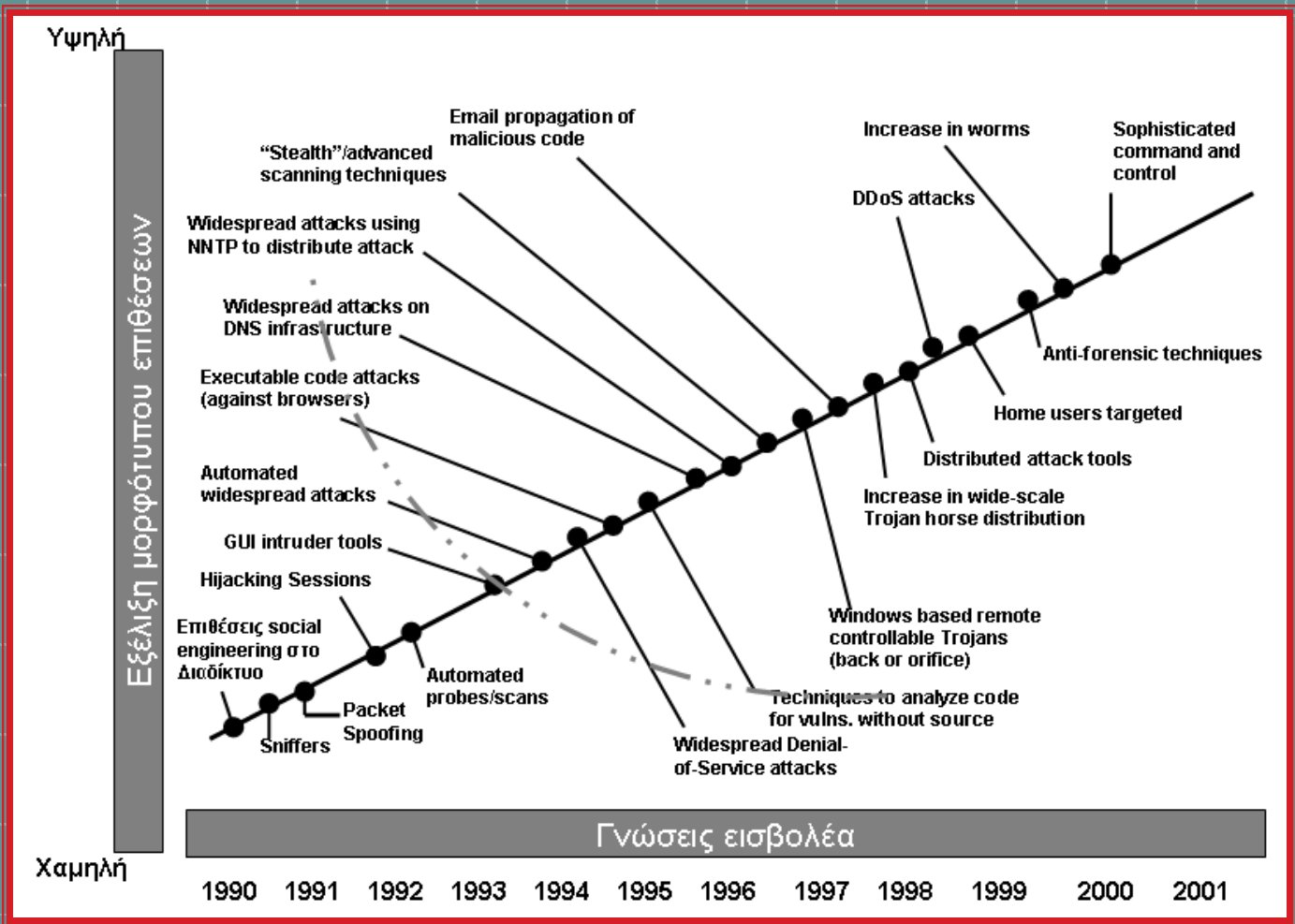


2004 CSI/FBI Computer Crime and Security Survey

Τεχνολογίες αντιμετώπισης ιομορφών



Τεχνολογίες αντιμετώπισης ιομορφών



Στρατηγικές αντιμετώπισης ιομορφών

- **Πρόληψη:** Προλαμβάνει τη μόλυνση από ιομορφικό λογισμικό (πχ. διαχειριστικά μέτρα, ενημέρωση χρηστών).
- **Ανίχνευση:** Ανιχνεύει τη μόλυνση από ιομορφικό λογισμικό (πχ. ανιχνευτές ιών).
- **Αντιμετώπιση:** Επαναφέρει το σύστημα στην αρχική του κατάσταση (πχ. χρήση αντιγράφων ασφαλείας).

Τεχνικές ανίχνευσης ιομορφών

Τεχνικές που βασίζονται στην εμφάνιση

- Ανιχνευτές ιών (scanners)
- Αντίδοτα (antidotes)
- Αθροίσματα ελέγχου (checksums)
- Αυτοάμυνα λογισμικού (software self-defense)
- Απρόσβλητο λογισμικό (fault tolerance)
- Ελεγχος μεταβολών (change control)
- Κελύφη προστασίας (integrity shells)

Τεχνικές που βασίζονται στη συμπεριφορά

- Εμπειρα Συστήματα (Expert systems)
- Νευρωνικά Δίκτυα (Neural networks)
- Γλώσσες Προσδιορισμού Προδιαθέσεων (Intent Specification Languages)

Τεχνικές βασισμένες στην εμφάνιση (detection-by-appearance techniques)

Μετρική	Ανιχνευτές ιών	Αντίδοτα	Αθροίσματα ελέγχου	Αυτοάμυνα λογισμικού	Απρόσβλητο λογισμικό	Ελεγχος μεταβολών	Κελύφη προστασίας
Ταχύτητα	Χαμηλή	Μέση	Μέση	Μέση	Μέση	Χαμηλή	Μέση
Κόστος	Χαμηλό	Χαμηλό	Χαμηλό	Υψηλό	Υψηλό	Υψηλό	Μέσο
Δυνατότητα πρόληψης	OXI	NAI	NAI/OXI	NAI	NAI	NAI	NAI
Δυνατότητα ανίχνευσης	NAI	NAI	NAI	NAI	NAI	NAI	NAI
Αντιμετώπιση γνωστών ιομορφών	NAI	NAI	NAI	NAI	NAI	NAI	NAI
Αντιμετώπιση άγνωστων ιομορφών	OXI	OXI	NAI	NAI	NAI	NAI	NAI

Τεχνικές βασισμένες στην εμφάνιση (detection-by-appearance techniques)

Μετρική	Ανιχνευτές ιών	Αντίδοτα	Αθροίσματα ελέγχου	Αυτοάμυνα λογισμικού	Απρόσβλητο λογισμικό	Έλεγχος μεταβολών	Κελύφη προστασίας
Λάθος ανίχνευση	NAI	NAI	OXI	OXI	OXI	OXI	OXI
Πολυπλοκότητα σχεδίασης	Χαμηλή	Χαμηλή	Μέση	Υψηλή	Υψηλή	Υψηλή	Μέση
Αναγκαιότητα ασφαλούς διαύλου	OXI	OXI	NAI	OXI	OXI	OXI	OXI
Προσδιορισμός πρόθεσης χρήστη	OXI	OXI	OXI	OXI	OXI	OXI	OXI
Αλλοιώσιμο	NAI	NAI	NAI	NAI	NAI	OXI	NAI
Αντιμετώπιση ειδικών ιομορφών	OXI	NAI	NAI	NAI	NAI	NAI	NAI
Παρενέργειες	OXI	NAI	OXI	NAI	OXI	NAI	NAI

Τεχνικές βασισμένες στη συμπεριφορά (detection-by-behavior techniques)

Μετρική	Γλώσσες προσδιορισμού προδιαθέσεων (ISL)	Εμπειρα Συστήματα	Νευρωνικά Δίκτυα
Αντιμετώπιση γνωστών ιομορφών	NAI	NAI	NAI
Αντιμετώπιση άγνωστων ιομορφών	NAI	NAI	NAI
Ταχύτητα	Χαμηλή/Μέση	Χαμηλή	Υψηλή
Κόστος	Άγνωστο	Υψηλό	Μέσο
Δυνατότητα πρόληψης	NAI	NAI	NAI
Δυνατότητα ανίχνευσης	NAI	NAI	NAI
Λάνθασμένη ανίχνευση	NAI	NAI	NAI
Πολυπλοκότητα σχεδίασης	Πολύ υψηλή	Μέση	Μέση
Παρενέργειες	NAI	NAI	OXI

Τεχνικές βασισμένες στη συμπεριφορά (detection-by-behavior techniques)

Μετρική	Γλώσσες προσδιορισμού προδιαθέσεων (ISL)	Εμπειρα Συστήματα	Νευρωνικά δίκτυα
Επεκτασιμότητα	NAI	NAI	?
Αντιμετώπιση ειδικών ιομορφών	Αγνωστο	NAI	NAI
Προσδιορισμός πρόθεσης χρήστη	NAI	NAI/OXI	OXI
Γενικευσιμότητα	NAI	NAI	NAI
Αναγκαιότητα ασφαλούς διαύλου	OXI	OXI	OXI
Αλλοιώσιμο	NAI	NAI	OXI
Τεχνικές εξαρτήσεις	Αγνωστο	Database	Database

Συμπεράσματα

- ◆ Δεν υπάρχει τεχνική αντιμετώπισης ιομορφικού λογισμικού, η οποία να παρέχει πλήρη και αποδοτική, προληπτική και κατασταλτική εξασφάλιση, απέναντι σε γνωστούς και άγνωστους τύπους ιομορφών.
- ◆ Η κατασταλτική αντιμετώπιση των ιομορφών μπορεί να επιτευχθεί με πολλές τεχνικές, αποδοτικότερες των οποίων είναι οι συγκριτές αρχείων και τα κρυπτογραφημένα αθροίσματα ελέγχου.
- ◆ Ενώ η προληπτική αντιμετώπιση γνωστών τύπων ιομορφών είναι δυνατή με χρήση πολλών τεχνικών, η προληπτική αντιμετώπιση άγνωστων τύπων ιομορφών είναι δυνατή μόνο με χρήση τεχνικών που βασίζονται στη συμπεριφορά.

Συμπεράσματα

- ◆ Τα στοιχεία κόστους και πολυπλοκότητας των τεχνικών καθιστούν απαγορευτική την εφαρμογή ορισμένων από αυτές σε περιβάλλοντα μικρής ευπάθειας και περιορισμένης υπολογιστικής ισχύος.
- ◆ Πολλές τεχνικές αντιμετώπισης των ιομορφών είναι - και οι ίδιες - ευπρόσβλητες από ιομορφές, με εξαίρεση τον έλεγχο των αλλαγών του λογισμικού και τα νευρωνικά δίκτυα που χρησιμοποιούν υλικό (hard-ware-based), που είναι απρόσβλητα από γνωστές ιομορφές.
- ◆ Η ανταπόκριση των τεχνικών σε πραγματικό χρόνο (real time) είναι κρίσιμο ζητούμενο σε περιβάλλοντα που διαχειρίζονται δεδομένα μεγάλου όγκου ή υψηλής ευπάθειας.

References

1. Adleman L., "An abstract theory of computer viruses", in Hoffman L. (Ed.), *Rogue Programmes: Viruses, Worms and Trojan Horses*, pp. 307-323, Van Nostrand, USA, 1990.
2. Cohen F., "Computer viruses: Theory and experiments", *Computers & Security*, Vol. 6, No. 1, pp. 22-35, 1987.
3. Cohen F., "Computational aspects of computer viruses", *Computers & Security*, Vol. 8, No. 4, pp. 325-344, 1989.
4. Denault M., Gritzalis D., Karagiannis D., Spirakis P., "Intrusion detection: Evaluation and performance issues of the SECURENET system", *Computers & Security*, Vol. 13, No. 6, pp. 495-508, October 1994.
5. Denning P. (Ed.), *Computers under attack: Intruders, Worms and Viruses*, Addison-Wesley, USA, 1990.
6. Katsikas S., Spyrou T., Gritzalis D., Darzentas J., "Model for network behaviour under viral attack", *Computer Communications*, Vol. 19, No. 2, pp. 124-132, 1996.
7. Kephart J., White S., "Directed graph epidemiological models of computer viruses", in *Proc. of the 1991 IEEE Symposium on Research in Security and Privacy*, pp. 343-359, IEEE, USA, 1991.
8. Mylonas A., Dritsas S, Tsoumas V., Gritzalis D., "Smartphone Security Evaluation - The Malware Attack Case", in *Proc. of the 9th International Conference on Security and Cryptography*, pp. 25-36, Spain, 2011.
9. Spinellis D., Gritzalis D., "A domain-specific language for intrusion detection", in *Proc. of the 1st ACM Workshop on Intrusion Detection and Prevention Systems*, ACM, Athens, 2000.
10. Theoharidou M., Kotzanikolaou P., Gritzalis D., "Risk-based Criticality Analysis", in *Proc. of the 3rd IFIP International Conference on Critical Infrastructure Protection*, Springer, USA, 2009
11. Virvilis N., Gritzalis D., "The Big Four - What we did wrong in Advanced Persistent Threat detection?", in *Proc. of the 8th International Conference on Availability, Reliability and Security*, pp. 248-254, IEEE, Germany, 2013.
12. Virvilis N., Gritzalis D., "Trusted Computing vs. Advanced Persistent Threats: Can a defender win this game?", in *Proc. of 10th IEEE International Conference on Autonomic and Trusted Computing*, pp. 396-403, IEEE, Italy, 2013