ΕΘΝΙΚΟΝ & ΚΑΠΟΔΙΣΤΡΙΑΚΟΝ ΠΑΝΕΠΙΣΤΗΜΙΟΝ ΑΘΗΝΩΝ
NATIONAL & KAPODISTRIAN UNIVERSITY OF ATHENS

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
Πρόγραμμα Μεταπτυχιακών Σπουδών (Π.Μ.Σ.)

Ασφαλής Ανάπτυξη Λογισμικού

Δρ. Κωνσταντίνος Παπαπαναγιώτου
conpap@di.uoa.gr

```
while (*pwszTemp != L'\\')
        *pwszServerName++ = *pwszTemp++;
```

>1,500,000 μολυσμένα συστήματα

3,370,000 κλήσεις στην Υποστήριξη Πελατών το Σεπτέμβριο του 2003

(συνήθως όχι πάνω από 350,000)

Πολλά αρνητικά σχόλια:

" This [is] going to raise the level of frustration to the point where a lot of organizations will seriously contemplate alternatives to Microsoft" Gartner

"There's definitely caution warranted here. [Microsoft's security] efforts were sincere, but I am not sure if they were sincere enough" Forrester

Γιατί Ασφάλεια Λογισμικού;

Το λογισμικό αποτελεί το τσιμέντο της σύγχρονης κοινωνίας. Τα πάντα γύρω μας ελέγχονται από εφαρμογές, συνεπώς ο κώδικας από τον οποίο αποτελούνται είναι ένα κρίσιμο στοιχείο για την ίδια μας τη ζωή. Καθημερινά εξαρτόμαστε σημαντικά και χωρίς να το γνωρίζουμε από την ορθή λειτουργία του κώδικα εφαρμογών.

Ταυτόχρονα οι εφαρμογές βελτιώνουν τη ζωή μας, ενώ σωστά ανεπτυγμένες εφαρμογές βελτιώνουν την ικανοποίηση, ελαττώνουν τα λειτουργικά έξοδα κλπ.

Αποτελούν μία σημαντική επένδυση για τους οργανισμούς όσον αφορά την προμήθεια, ανάπτυξη και συντήρησή τους.

Πρακτικά όλα τα προβλήματα ασφαλείας οφείλονται σε λάθη στον κώδικα

Πολυπλοκότητα εφαρμογών, χιλιάδες γραμμές κώδικα => πιο εύκολο να βρεθούν και να εκμεταλλευθούν λάθη και αδυναμίες

Επίσης οι προγραμματιστές πιέζονται να παράγουν εφαρμογές σε σύντομο χρονικό διάστημα με αποτέλεσμα να προκύπτουν λάθη.

Οι εφαρμογές αποτελούν στόχο επιθέσεων καθώς μέσα από αυτές ο επιτιθέμενος μπορεί να αποκτήσει πρόσβαση σε πληροφορίες, προϊόντα, υποδομές και φυσικά **χρήματα**.

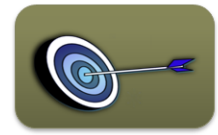
Ακόμα και αν υπάρχουν τα παραδοσιακά μέτρα δικτυακής προστασίας (π.χ. Firewalls) οι εφαρμογές μπορεί να παραβιαστούν.

Εάν παραβιαστούν τα μέτρα προστασίας γύρω από μία εφαρμογή ο επιτιθέμενος μπορεί να:

- Αλλοιώσει δεδομένα
- Αποκτήσει πρόσβαση σε εμπιστευτικά δεδομένα
- Αλλάξει οικονομικά δεδομένα

Σε ορισμένες περιπτώσεις μπορεί μέσα από μία τέτοια επίθεση να αποκτήσει πρόσβαση σε άλλα σημεία της εφαρμογής ή σε όλο το υπολογιστικό περιβάλλον.

Η εξέλιξη των επιθέσεων

1986-1995	1995-2003	2004+	2006+
			
• LANs • Πρώτοι ιοί • Κίνητρο: καταστροφή	• Η εποχή του Internet • Σκουλήκια • Κίνητρο: καταστροφή	• Επιθέσεις σε ΛΣ, ΒΔ • Spyware, Spam • Κίνητρο: οικονομικό	• Στοχευμένες επιθέσεις • Κοινωνική μηχανική • Κίνητρα: Οικονομικά και Πολιτικά

Οι επιθέσεις «ανεβαίνουν» στο επίπεδο εφαρμογής

Πηγή: © Microsoft Corporation

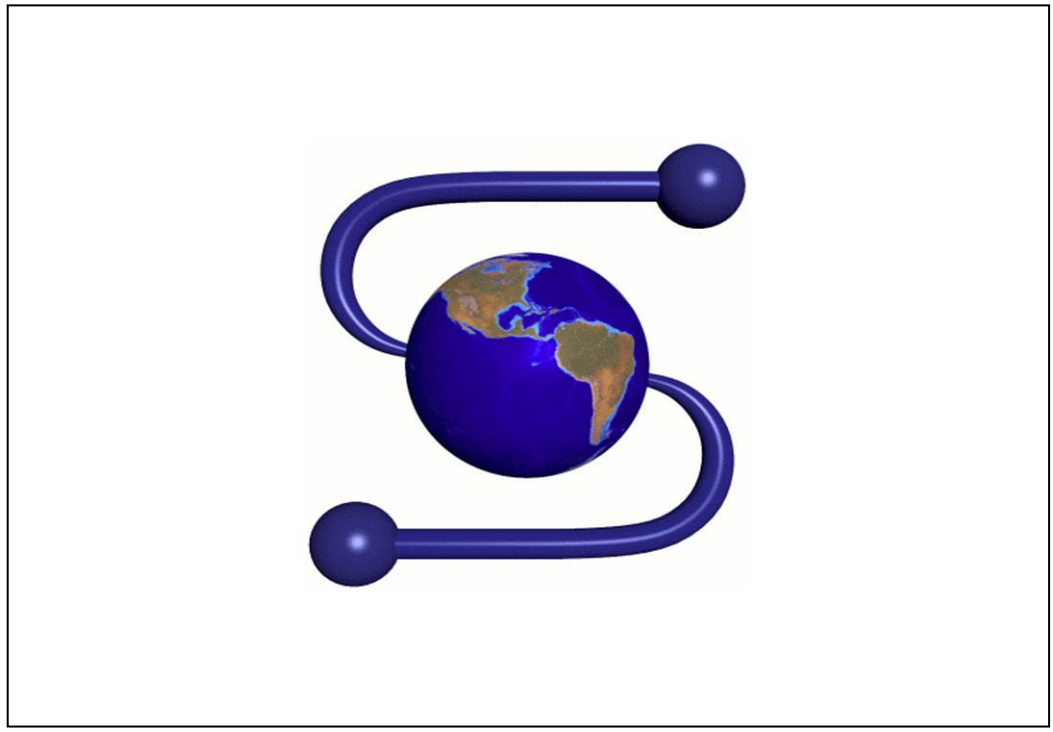
6

MS-DOS 6.22: λειτουργικό σύστημα σχεδιασμένο για ένα χρήστη, χωρίς δυνατότητες [δια]δικτύωσης.

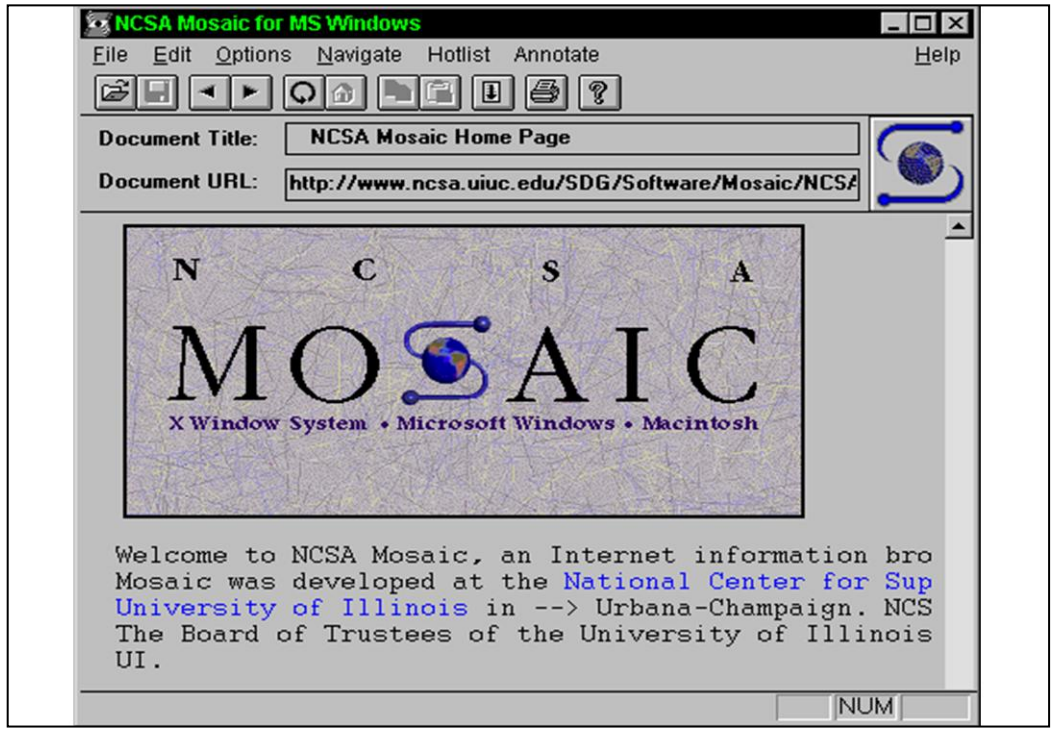
Οι απαιτήσεις ασφάλειας ήταν μικρές.

Windows 95: Δυνατότητα διασύνδεσης σε εταιρικά δίκτυα και το Internet

Νέο σημείο κατάρρευσης ασφάλειας: ο φυλλομετρητής (browser).



Ο NCSA Mosaic ήταν ο πρώτος υποτυπώδης browser.


NCSA Mosaic for MS Windows
File Edit Options Navigate Hotlist Annotate Help
Document Title: NCSA Mosaic Home Page
Document URL: http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSA
N C S A
MOSAIC
X Window System • Microsoft Windows • Macintosh
Welcome to NCSA Mosaic, an Internet information bro
Mosaic was developed at the National Center for Sup
University of Illinois in --> Urbana-Champaign. NCS
The Board of Trustees of the University of Illinois
UI.
NUM

Ακολούθησε ο Netscape Navigator που χρησιμοποιούσε την ίδια μηχανή εμφάνισης σελίδων με το Mosaic.

…και πολλοί άλλοι browsers (π.χ. Internet Explorer από τη Microsoft, Opera, Safari από την Apple και Chrome από τη Google)

Αυτό οδήγησε στο λεγόμενο πόλεμο των browser με απώτερο σκοπό τον «έλεγχο του Internet» και πολλαπλά αποτελέσματα.

Αποτελέσματα του «πολέμου των browser»

- Νέες δυνατότητες και γρήγορα!
  - Από κείμενο που αναβοσβήνει μέχρι Javascript
- Ασυμβατότητες
  - Οι χρήστες προτιμούν browsers που εμφανίζουν σωστά όλες (τις περισσότερες) σελίδες
- Προσπάθειες για προτυποποίηση

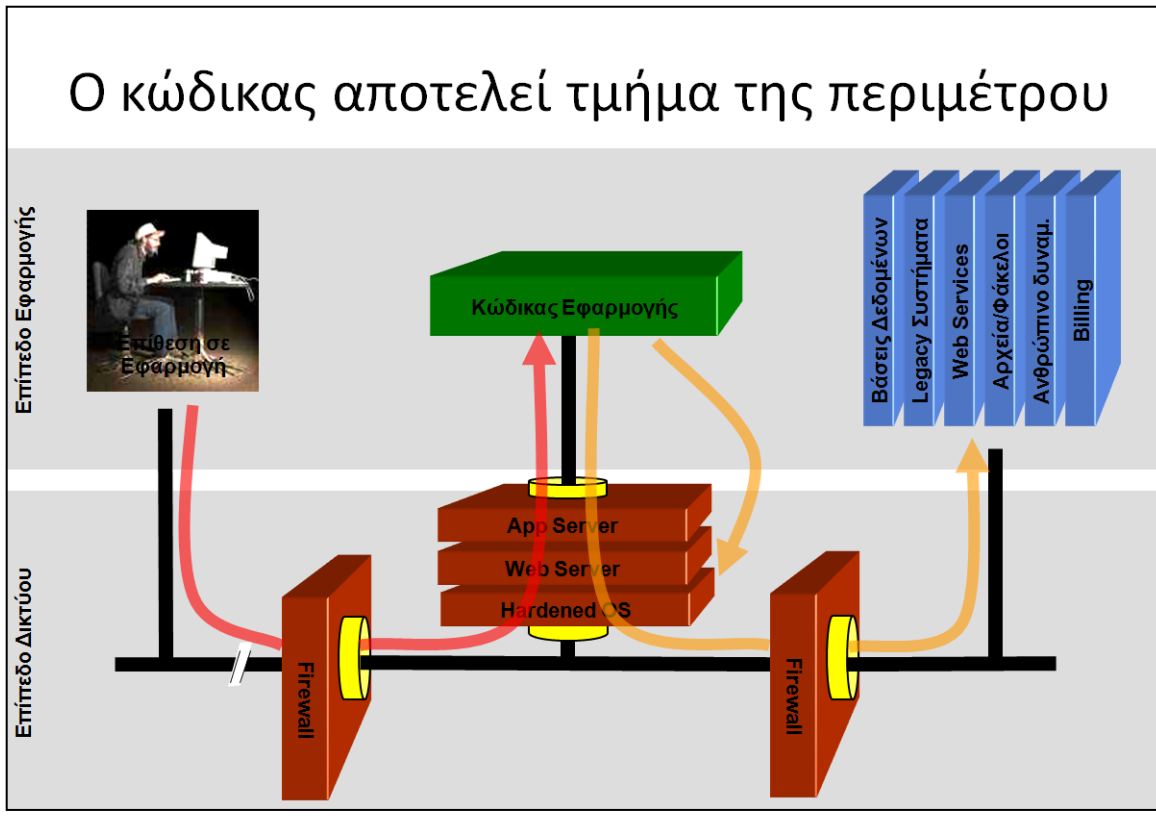
Ένα από τα αποτελέσματα του «πολέμου των browsers» ήταν η συνεχής υλοποίηση νέων δυνατοτήτων στους browsers (και συνεπώς νέων δυνατοτήτων για όσους αναπτύσσουν σελίδες) από τις εταιρίες κατασκευής τους, με απώτερο σκοπό την επίτευξη ανταγωνιστικού πλεονεκτήματος. Οι αλλαγές αυτές γίνονταν πολύ γρήγορα και περιλάμβαναν από απλές αισθητικές βελτιώσεις (π.χ. κείμενο που αναβοσβήνει), μέχρι την υιοθέτηση ενσωματωμένων γλωσσών προγραμματισμού. (π.χ. Javascript)

Αποτέλεσμα ήταν η έλλειψη συμβατότητας μεταξύ των browsers. Για παράδειγμα σελίδες που φαίνονταν κανονικά στον Internet Explorer, δεν εμφανίζονταν με τον ίδιο τρόπο στον Netscape Navigator.

Αναπόφευκτα, οι χρήστες προτιμούν browsers που εμφανίζουν σωστά τις περισσότερες σελίδες, τουλάχιστον από αυτές που επισκέπτονται συχνότερα.

Στη σημερινή εποχή του Web 2.0 η κατάσταση έχει βελτιωθεί, κυρίως μετά από προσπάθειες προτυποποίησης από το World Wide Web Consortium (W3C) αλλά τα προβλήματα παραμένουν, ειδικά σε ότι αφορά την ασφάλεια.

Δεν υπάρχουν πλέον διακριτοί ρόλοι μεταξύ server και client (browser) αφού κώδικας για μια εφαρμογή μπορεί να εκτελείται και στους δύο!



Η προστασία των εφαρμογών διαφέρει από την προστασία υποδομών και δικτύων. Το λογισμικό είναι πολύ πιο σύνθετο και περιλαμβάνει δυνητικά περισσότερες ευπάθειες.

Υλοποιώντας παραδοσιακές τεχνικές προστασίας δικτύων (π.χ. Firewalls, Intrusion Detection Systems, κλπ.) θεωρούμε ότι έχουμε προστατεύσει επαρκώς την υποδομή μας.

Όμως, κανένα firewall δεν περιορίζει την πρόσβαση στην πόρτα 80 (http). Έτσι οι επιθέσεις που στοχεύουν καθαρά στις εφαρμογές και εκμεταλλεύονται ευπάθειες στον κώδικά τους εκδηλώνονται ανεμπόδιστα.

Με τον τρόπο αυτό μπορεί ο επιτιθέμενος να παραβιάσει τα συστήματα ασφάλειας για να:

•Τροποποιήσει δεδομένα

•Υποκλέψει ευαίσθητες πληροφορίες

•Αποκτήσει πληροφορίες οικονομικού χαρακτήρα (π.χ. αριθμούς πιστωτικών καρτών).

Η προστασία των εφαρμογών απαιτεί διαφορετική προσέγγιση σε σχέση με την ασφάλεια δικτύων και αυτό γιατί οι επιθέσεις εμφανίζονται σαν κανονικά δεδομένα εισόδου ή αιτήσεις.

Συχνά υπάρχει η θεώρηση ότι τα προβλήματα σε επίπεδα κώδικα είναι αμιγώς τεχνικά. Όμως τα τεχνικά αυτά προβλήματα μπορεί να επηρεάσουν συνολικά τη λειτουργία του οργανισμού και τελικά αφορούν και τη διοίκηση.

Οι κανονιστικές διατάξεις επιβάλλουν και στην περίπτωση αυτή τη λήψη μέτρων ασφάλειας.

Τα περισσότερα πρότυπα ασφάλειας (π.χ. ISO 27001) αναφέρονται γενικά σε διαδικασίες ασφαλούς ανάπτυξης/προμήθειας/συντήρησης λογισμικού.

Το πρότυπο PCI DSS αναφέρει ρητά διαδικασίες που πρέπει να ακολουθούνται και τύπους/κατηγορίες ευπαθειών εφαρμογών που πρέπει να ελέγχονται και να προστατεύονται

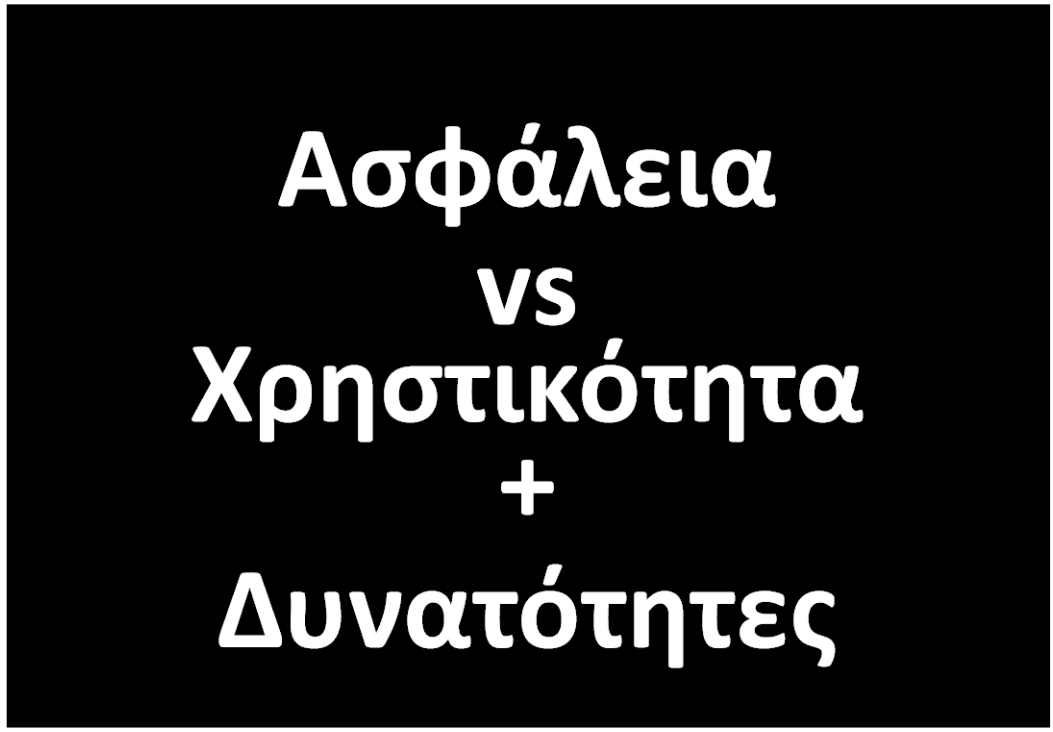
Ποιούς αφορά;

Η ασφάλεια λογισμικού αφορά κυρίως τους developers και γενικά όσους ασχολούνται με την ανάπτυξή του.

Αποτελεί μέρος της ποιότητας λογισμικού και άρα αφορά όσους θέλουν να αναπτύσσουν σωστά λογισμικό, το οποίο δεν αποτυγχάνει και επιτελεί τις λειτουργίες για τις οποίες σχεδιάστηκε και μόνο αυτές.

Οι Managers χρειάζεται να υποστηρίζουν ενεργά την ασφαλή ανάπτυξη λογισμικού (π.χ. Bill Gates)

Οι χρήστες περιμένουν το λογισμικό που χρησιμοποιούν να συμπεριφέρεται όπως πρέπει και προ πάντων με ασφάλεια.



Η Microsoft προσπάθησε να δημιουργήσει απόλυτα ασφαλές λογισμικό και ήταν ήδη ξεπερασμένο όταν τελικά κυκλοφόρησε.

Το πιο ασφαλές λογισμικό είναι αυτό που δεν έχει καμία δυνατότητα/feature.

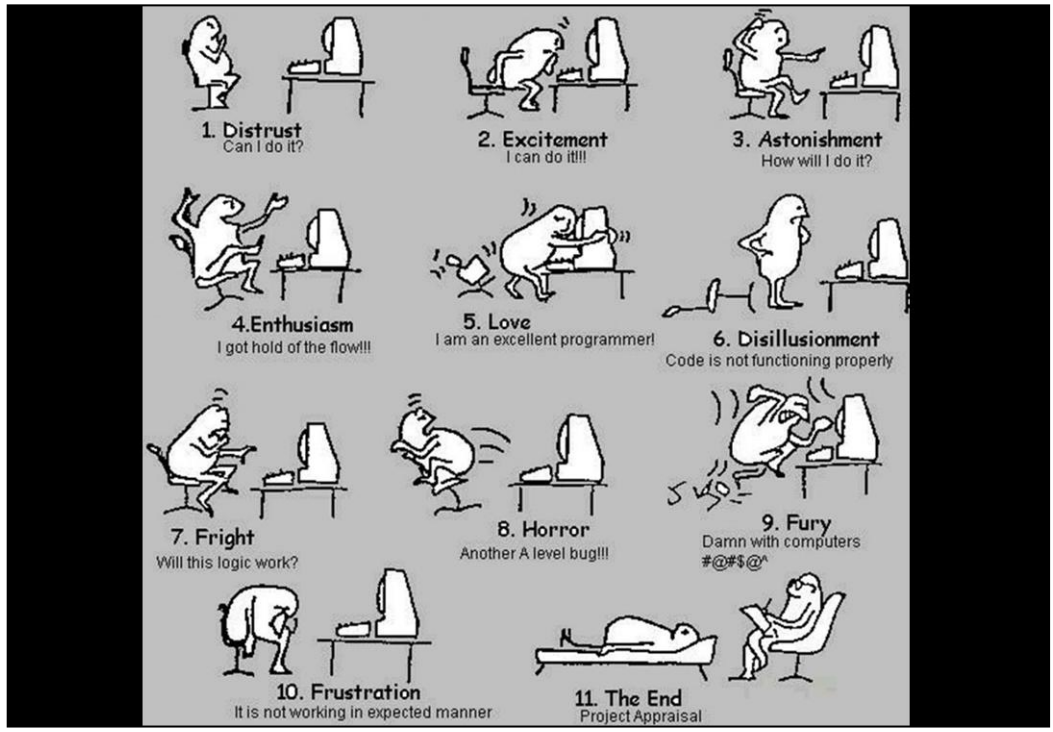
Σαν χρήστες ανεχόμαστε τα λάθη-έλλειψη ασφάλειας με αντίτιμο τη σύγχρονη τεχνολογία και τις επιπλέον δυνατότητες.

Η πραγματικότητα…

- Όσο και αν προσπαθήσουμε δε θα έχουμε ποτέ 100% σωστό κώδικα
  - «Ασύμμετρη απειλή»
    - **Εμείς** πρέπει να είμαστε 100% σωστοί στο 100% των περιπτώσεων, με αυστηρές προθεσμίες, γνωρίζοντας μόνο τα σημερινά δεδομένα.
      - Και φυσικά το τελικό προϊόν πρέπει να είναι αξιόπιστο, φθηνό, εύκολα διαχειρίσιμο, συμβατό με πολλές πλατφόρμες και πρότυπα, εύκολο στη χρήση, κλπ.
    - **Αυτοί** μπορούν να αφιερώσουν όσο χρόνο θέλουν για να εντοπίσουν μία ευπάθεια, με δεδομένα που μπορεί να γίνουν γνωστά από μελλοντική έρευνα.
- Είμαστε άνθρωποι και άρα όχι τέλειοι.
- Οι υπολογιστές είναι χαζοί.
- Υπάρχει περιθώριο βελτίωσης της ασφάλειας.

Η [ακόμα πιο σκληρή] πραγματικότητα

- Υπάρχουν εργαλεία που καθιστούν πανεύκολη τη δημιουργία exploits
- Εργαλεία αντίστροφης μηχανικής (reverse engineering)
- Exploit payloads: www.metasploit.com



Υπάρχουν διάφορες μεθοδολογίες ανάπτυξης και κύκλου ζωής λογισμικού (π.χ. καταρράκτη, Agile, κλπ.) που συνήθως περιλαμβάνουν στάδια όπως:

- Ανάλυση επιχειρηματικών απαιτήσεων
- Σχεδιασμός αρχιτεκτονικής
- Υλοποίηση κώδικα
- Δοκιμές
- Ενεργοποίηση στην Παραγωγή
- Διαχείριση αλλαγών και αντιμετώπιση προβλημάτων

Τα μοντέλα αυτά όμως δεν περιλαμβάνουν στοιχεία ασφάλειας σε κανένα βήμα του κύκλου ζωής.

Η αντιμετώπιση των προβλημάτων ασφάλειας συνήθως γίνεται εκ των υστέρων, μετά την ανάπτυξη της εφαρμογής.

Τότε καλούνται ειδικοί για να διεξάγουν δοκιμές παρείσδυσης (penetration tests). Στις δοκιμές αυτές δρουν σαν κακόβουλοι χρήστες και εντοπίζουν αδυναμίες και ευπάθειες.

Τα αποτελέσματα των δοκιμών αυτών είναι συχνά αόριστα, χωρίς να δίνουν σαφή εικόνα των βημάτων που πρέπει να ακολουθήσουν οι προγραμματιστές για να τα επιλύσουν.

Έτσι, δεν καλύπτουν τελικά όλες τις ευπάθειες και τα προβλήματα παραμένουν, ενώ απαιτείται μεγάλη προσπάθεια για επιδιόρθωσή τους.

Αποτέλεσμα είναι να εκδηλώνονται επιθέσεις οι οποίες εκμεταλλεύονται κενά ασφάλειας. Τα κενά και οι αδυναμίες αυτές στις περισσότερες περιπτώσεις θα ήταν πολύ εύκολο να διορθωθούν αν είχαν εντοπιστεί κατά τη διάρκεια του σχεδιασμού και της ανάπτυξης και όχι μετά το τέλος της ανάπτυξης της εφαρμογής.

Ασφάλεια στον Κύκλο Ζωής Ανάπτυξης Λογισμικού

Software Assurance

SSA - Software Security Assurance

SDL – Security Development Lifecycle

SDLC – για μεγαλύτερο μπέρδεμα

sSDLC – secure Software Development Lifecycle

SPLC – Secure Project Lifecycle

CLASP - Comprehensive, Lightweight Application Security Process

7 Touchpoints

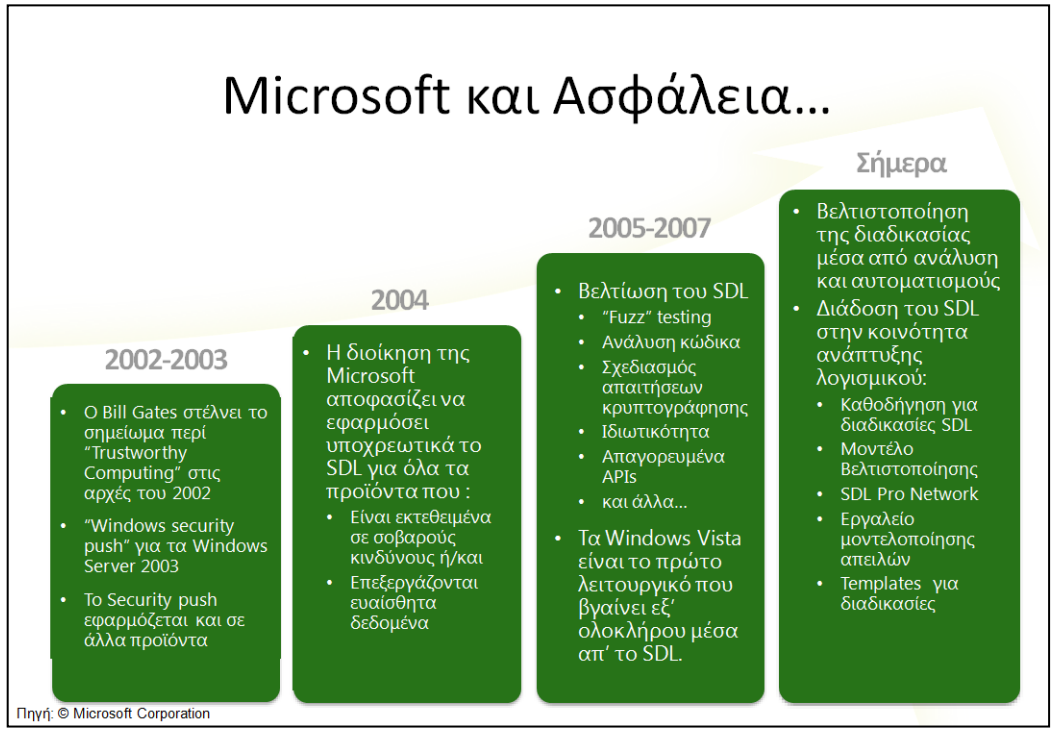
SSF – System Security Framework

…

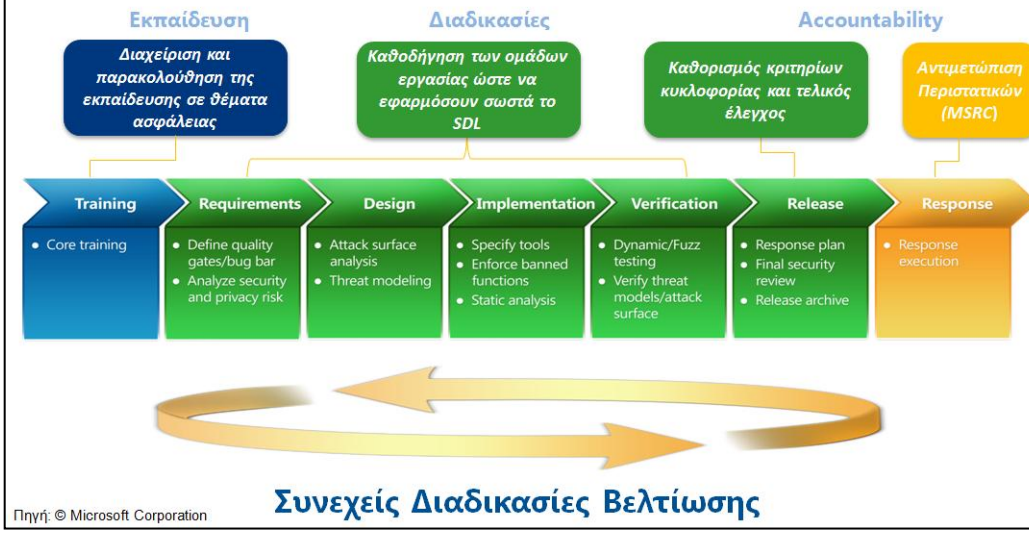
Έχουν αναπτυχθεί διάφορα πρότυπα για ασφάλεια στον κύκλο ζωής λογισμικού. Τα περισσότερα από αυτά είναι δύσκολα στην κατανόηση για ανθρώπους που δεν έχουν γνώσεις ασφάλειας ή απαιτούν αυξημένους πόρους για την υλοποίησή τους.

Πρακτικά δεν υπάρχει μία εδραιωμένη μεθοδολογία. Αντίθετα κάθε οργανισμός προσπαθεί να προσαρμόσει ένα πρότυπο στις δικές του ανάγκες και να εισάγει διαδικασίες ασφάλειας σε υπάρχουσες διαδικασίες κύκλου ζωής λογισμικού.

Η ασφάλεια όμως πρέπει να υπάρχει εγγενώς μέσα στο κύκλο ζωής λογισμικού και όχι να προστίθεται εκ των υστέρων.


Microsoft και Ασφάλεια…
2002-2003
• Ο Bill Gates στέλνει το σημείωμα περί "Trustworthy Computing" στις αρχές του 2002
• "Windows security push" για τα Windows Server 2003
• Το Security push εφαρμόζεται και σε άλλα προϊόντα
2004
• Η διοίκηση της Microsoft αποφασίζει να εφαρμόσει υποχρεωτικά το SDL για όλα τα προϊόντα που :
• Είναι εκτεθειμένα σε σοβαρούς κινδύνους ή/και
• Επεξεργάζονται ευαίσθητα δεδομένα
2005-2007
• Βελτίωση του SDL
• "Fuzz" testing
• Ανάλυση κώδικα
• Σχεδιασμός απαιτήσεων κρυπτογράφησης
• Ιδιωτικότητα
• Απαγορευμένα APIs
• και άλλα…
• Τα Windows Vista είναι το πρώτο λειτουργικό που βγαίνει εξ' ολοκλήρου μέσα απ' το SDL.
Σήμερα
• Βελτιστοποίηση της διαδικασίας μέσα από ανάλυση και αυτοματισμούς
• Διάδοση του SDL στην κοινότητα ανάπτυξης λογισμικού:
• Καθοδήγηση για διαδικασίες SDL
• Μοντέλο Βελτιστοποίησης
• SDL Pro Network
• Εργαλείο μοντελοποίησης απειλών
• Templates για διαδικασίες
Πηγή: © Microsoft Corporation

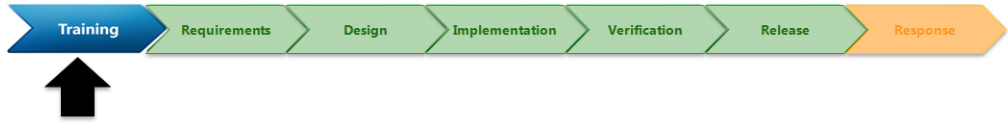
Microsoft SDL



Πηγή: © Microsoft Corporation

Συνεχείς Διαδικασίες Βελτίωσης

Προαπαιτούμενα: Εκπαίδευση



Αξιολόγηση της πραγματικής γνώσης σε ασφάλεια και ιδιωτικότητα

Δημιουργία προγραμμάτων εκπαίδευσης αν είναι απαραίτητο

- Καθορισμός κριτηρίων εκπαίδευσης
  - Το περιεχόμενο καλύπτει θέματα ασφαλούς σχεδιασμού, υλοποίησης, ελέγχου και ιδιωτικότητας.
- Καθορισμός ελάχιστης συχνότητας εκπαίδευσης
  - Οι εργαζόμενοι πρέπει να παρακολουθούν *n* μαθήματα το χρόνο
- Καθορισμός ελάχιστων αποδεκτών ορίων εκπαίδευσης
  - Επιχειρησιακοί στόχοι εκπαίδευσης (π.χ. το 80% του τεχνικού προσωπικού θα εκπαιδεύεται πριν την τελική έκδοση του προϊόντος.)

Φάση 1η: Απαιτήσεις

Ευκαιρία για ένταξη της ασφάλειας στο λογισμικό από την αρχή του έργου

- Η ομάδα ανάπτυξης εντοπίζει απαιτήσεις ασφάλειας και ιδιωτικότητας.
- Καθορίζονται υπεύθυνοι επικοινωνίας για θέματα ασφάλειας και ιδιωτικότητας
- Επιλογή και διορισμός Σύμβουλου Ασφάλειας (Security Advisor)
- Ο Σύμβουλος Ασφάλειας επιθεωρεί το σχέδιο απαιτήσεων και υλοποίησης, προχωρεί σε συστάσεις και μπορεί να θέσει επιπλέον απαιτήσεις.
- Καθιέρωση της χρήσης συστήματος εντοπισμού σφαλμάτων και ανάθεσης αντίστοιχων εργασιών (service desk)
- Ορισμός και καταγραφή επιτρεπόμενων ορίων για bugs που αφορούν ασφάλεια και ιδιωτικότητα.

Πηγή: © Microsoft Corporation

Φάση 2η: Σχεδιασμός

Καθορισμός και καταγραφή αρχιτεκτονικής ασφάλειας, εντοπισμός κρίσιμων στοιχείων

- Καθορισμός τεχνικών σχεδιασμού (managed κώδικας, ελάχιστα δικαιώματα, ελαχιστοποίηση επιφάνειας επιθέσεων, ανάπτυξη σε στρώματα, κλπ.)
- Καταγραφή επιφάνειας επιθέσεων (attack surface) και αρχικός περιορισμός με ρυθμίσεις
- Ορισμός επιπλέον κριτηρίων για την παράδοση εξαιτίας ιδιαίτερων θεμάτων για συγκεκριμένα προϊόντα.
  - Έλεγχοι για cross-site scripting
  - Ενίσχυση των αδύναμων αλγορίθμων κρυπτογράφησης
- Μοντελοποίηση Απειλών (Threat Modeling)
  - Συστηματική επισκόπηση των δυνατοτήτων και της αρχιτεκτονικής από τη σκοπιά της ασδφάλειας.
  - Εντοπισμός απειλών και αντιμετώπισή τους.
- Εξειδικευμένες απαιτήσεις για online υπηρεσίες.

Πηγή: © Microsoft Corporation

Φάση 3η: Υλοποίηση

Πλήρης επισκόπηση – χρησιμοποιείται για τον προσδιορισμό διεργασιών, την τεκμηρίωση και την υιοθέτηση των απαραίτητων εργαλείων για την ασφαλή υλοποίηση και λειτουργία.

- Καθορισμός εγκεκριμένων εργαλείων ανάπτυξης και σχετικών επιλογών
- Στατική ανάλυση (χρήση κατάλληλων εργαλείων)
- Επισφαλή APIs
- Χρήση των μέτρων προστασίας του λειτουργικού συστήματος.
- Εξειδικευμένες απαιτήσεις για online υπηρεσίες (π.χ., Cross-site scripting, SQL Injection κλπ.)
- Άλλες απαιτήσεις

Πηγή: © Microsoft Corporation

Επίσης: επαναχρησιμοποίηση κώδικα, είτε από εφαρμογές που έχουν αναπτυχθεί εσωτερικά, είτε από εφαρμογές τρίτων. Χρειάζεται ιδιαίτερη προσοχή για την επαναχρησιμοποίηση κώδικα ώστε αφενός να γίνει σωστά και αφετέρου να μην εισαχθεί κακόβουλος κώδικας ή κώδικας που δεν έχει ελεγχθεί επαρκώς.

Φάση 4η: Επαλήθευση

Ξεκινάει όσο το δυνατό νωρίτερα – υλοποιείται πλήρως αφού ολοκληρωθεί ο κώδικας

- Αρχικός σχεδιασμός για αντιμετώπιση περιστατικών ασφάλειας – πλάνων αντιμετώπισης αναφορών για ευπάθειες.
- Επανεκτίμηση της επιφάνειας επιθέσεων
- Fuzz testing – αρχεία, installable controls and network facing code
- Διεξαγωγή "security push" (αν και όποτε χρειάζεται)
  - Δεν αντικαθιστά τη δουλειά που γίνεται για ασφάλεια κατά την ανάπτυξη της εφαρμογής.
  - Επισκόπηση κώδικα (Code review)
  - Δοκιμές παρείσδυσης (penetration testing) και άλλοι έλεγχοι ασφάλειας
  - Επισκόπηση σχεδιασμού και αρχιτεκτονικής με βάση νέες απειλές
- Εξειδικευμένες απαιτήσεις για online υπηρεσίες.

Πηγή: © Microsoft Corporation

Η επαλήθευση είναι πολύ σημαντική και δύσκολη διαδικασία. Είναι εξαιρετικά δύσκολο να εντοπίσουμε σφάλματα και κενά ασφάλειας σε κώδικα που έχει αναπτυχθεί με καλές προθέσεις, πόσο μάλλον σε κώδικα που έχει αναπτυχθεί κακόβουλα και έχει γίνει προσπάθεια απόκρυψής του (obfuscation).

Φάση 5η: Πλάνο Αντιμετώπισης Περιστατικών

Δημιουργία μιας ξεκάθαρης πολιτικής υποστήριξης

- Πλάνο αντιμετώπισης περιστατικών ασφάλειας λογισμικού
  - Προσδιορισμός υπευθύνων για την ανταπόκριση σε περιστατικά
  - Πληροφορίες επικοινωνίας 24x7x365 με μηχανικούς, marketing και διεύθυνση
- Εξασφάλιση της υποστήριξης για όλο τον κώδικα, συμπεριλαμβανομένων «ανεπίσημων» εκδόσεων αλλά και κώδικα τρίτων.

Πηγή: © Microsoft Corporation

Φάση 5η: Τελική Επισκόπηση Ασφάλειας

Τελική επαλήθευση ότι ικανοποιήθηκαν οι απαιτήσεις του SDL και δεν υπάρχουν γνωστές ευπάθειες.

- Ανεξάρτητος τελικός έλεγχος ασφάλειας που πιστοποιεί ότι το προϊόν είναι έτοιμο.
- ΔΕΝ είναι:
  - Δοκιμή παρείσδυσης – ακόμα και αν γίνει τέτοια δοκιμή δεν ακολουθείται η συνηθισμένη διαδικασία επιδιόρθωσης
  - Συνολική επισκόπηση ασφάλειας
  - Δίνει το τελικό οκ (ή δεν το δίνει) για την κυκλοφορία
  - Στόχος δεν είναι ο εντοπισμός όσων ευπαθειών δεν εντοπίστηκαν μέχρι αυτό το σημείο, αλλά η επιβεβαίωση ότι τα βήματα του SDL ακολουθήθηκαν κανονικά και ήταν επαρκή.

Πηγή: © Microsoft Corporation

Επίσης γίνεται έλεγχος να για να διαπιστωθεί αν οι πρακτικές και οι διαδικασίες του SDL ήταν επαρκείς.

Φάση 5η: Έκδοση και Αρχειοθέτηση

Ολοκλήρωση πλάνου αντιμετώπισης περιστατικών

- Ενημέρωση τεκμηρίωσης
- Αρχειοθέτηση τελικού κώδικα, συμβόλων, μοντέλων απειλών.
- Τελική επικύρωση πολιτικών ασφάλειας, ιδιωτικότητας και συμμόρφωσης.

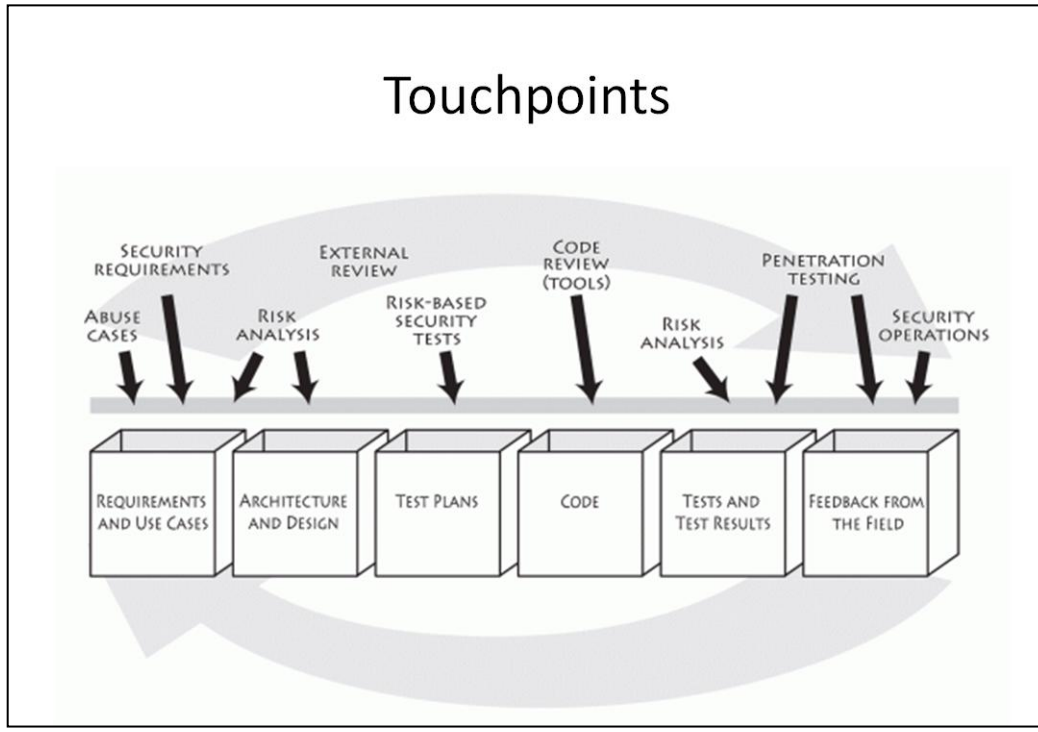
Πηγή: © Microsoft Corporation

Μετά το SDL: Αντιμετώπιση Περιστατικών

"Plan the work, work the plan…"

- Εκτέλεση πλάνων αντιμετώπισης περιστατικών όπως προδιαγράφηκαν στις προηγούμενες φάσεις.

Πηγή: © Microsoft Corporation



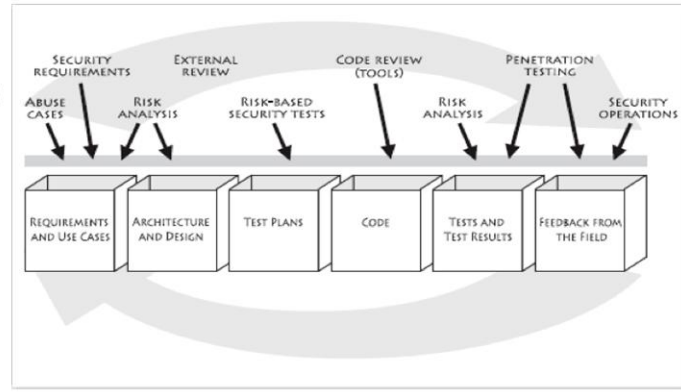
7 Touchpoints: Πρότυπο που δημιούργησε ο Gary McGraw και βασίζεται σε 7 σημεία προσοχής στον κύκλο ζωής λογισμικού.

OWASP CLASP

Comprehensive, Lightweight Application Security Process

Touchpoints

(Gary McGraw - Cigital)



OWASP Comprehensive, Lightweight Application Security Process

Κινείται γύρω από 7 Βέλτιστες Πρακτικές Ασφάλειας Λογισμικού

Καλύπτει όλο τον κύκλο ζωής λογισμικού όχι μόνο την ανάπτυξή του

Μπορεί να προσαρμοστεί σε όλες τις διαδικασίες ανάπτυξης

Ορίζει ρόλους

24 διαδικασίες βάσει ρόλων

Μπορεί να εφαρμοστεί και σε μικρές εταιρίες αφού προσαρμόζεται εύκολα σε οποιεσδήποτε ανάγκες

Microsoft SDL

«Βαρύ» πρότυπο, κατάλληλο μόνο για μεγάλους οργανισμούς

Touchpoints

Υψηλού επιπέδου, δε δίνει αρκετές λεπτομέρειες που να βοηθούν στην υλοποίηση.

CLASP

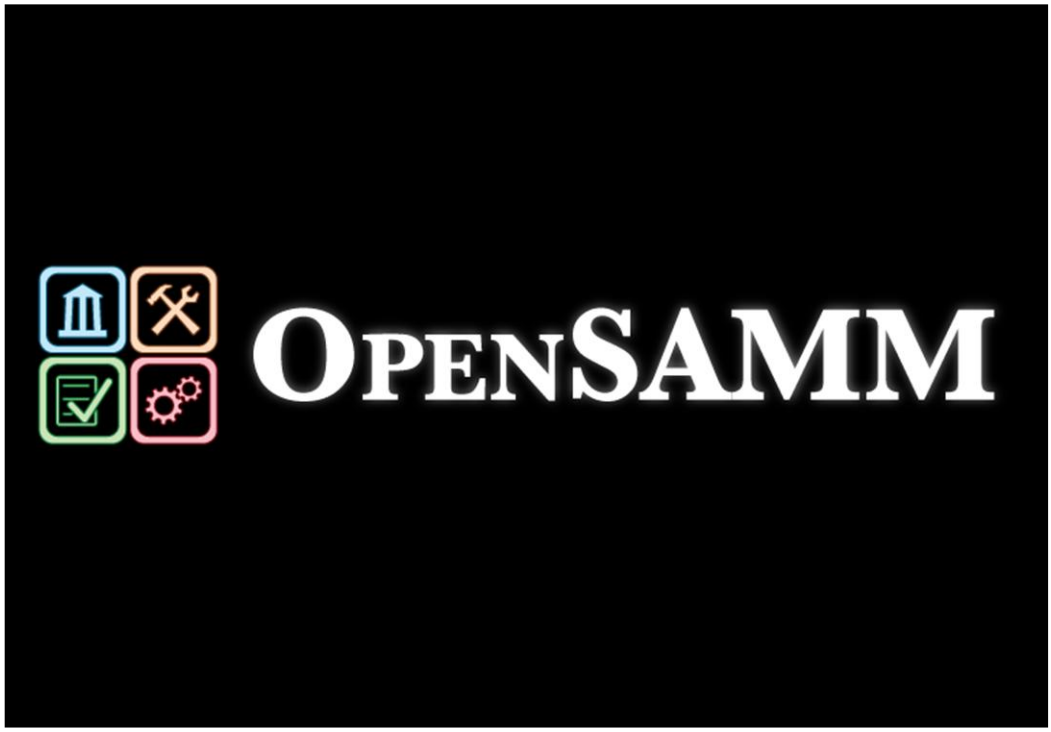
Μεγάλη συλλογή από δραστηριότητες-πρακτικές αλλά χωρίς να δίνει προτεραιότητες

Συνολικά τα πρότυπα αυτά απευθύνονται περισσότερο σε ειδικούς ασφάλειας.

Διασφάλιση (Assurance)

«η εμπιστοσύνη ότι ένα σύστημα ικανοποιεί τις απαιτήσεις ασφάλειας που το διέπουν, βάσει συγκεκριμένων στοιχείων που προκύπτουν από την εφαρμογή τεχνικών διασφάλισης»

Matt Bishop, Computer Security, Addison-Wesley, 2003.

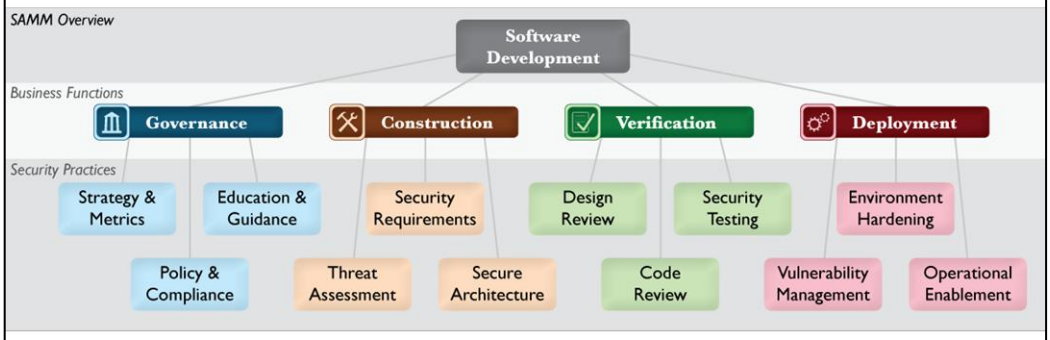

OpenSAMM

Στόχοι

- Ορισμός βασικών στοιχείων για την ανάπτυξη ενός προγράμματος διασφάλισης
  - Σκιαγράφηση όλων των λειτουργιών που μπορούν **σταδιακά** να βελτιωθούν σε έναν οργανισμό
- Δυνατότητα στους οργανισμούς να καθορίσουν το δικό τους πρόγραμμα διασφάλισης
  - Κάθε οργανισμός μπορεί να **επιλέξει** τη σειρά και το εύρος βελτιστοποίησης κάθε πρακτικής
- Ενδεικτικά προγράμματα υλοποίησης για συνηθισμένες κατηγορίες οργανισμών
  - Κάθε πρόγραμμα αποτελεί μία **βάση** που κάθε οργανισμός μπορεί να επεξεργαστεί ανάλογα με τις ανάγκες και τις ανησυχίες του.

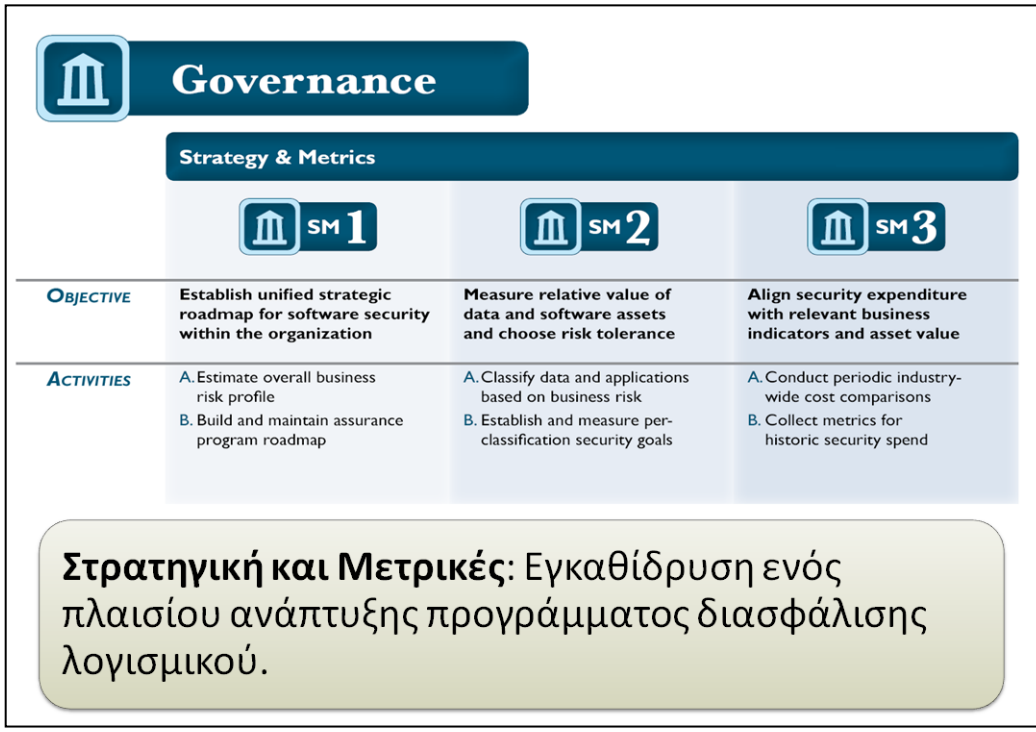
Επιχειρησιακές Συνιστώσες

Πρακτικές Ασφάλειας

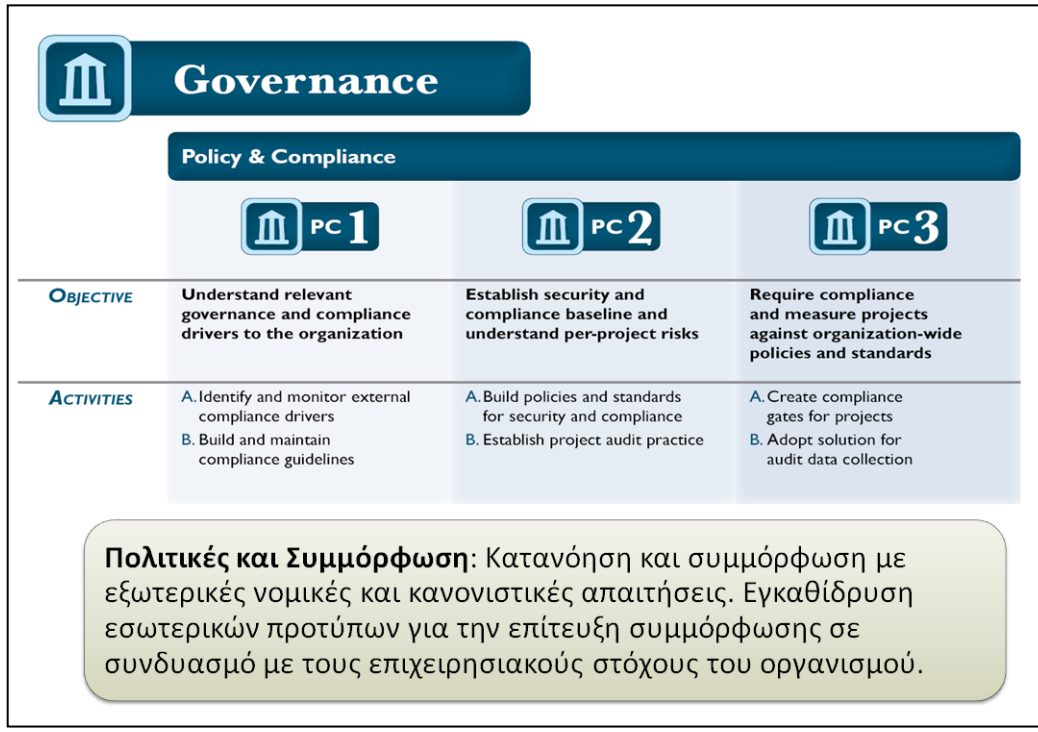


Για κάθε Πρακτική Ασφάλειας…

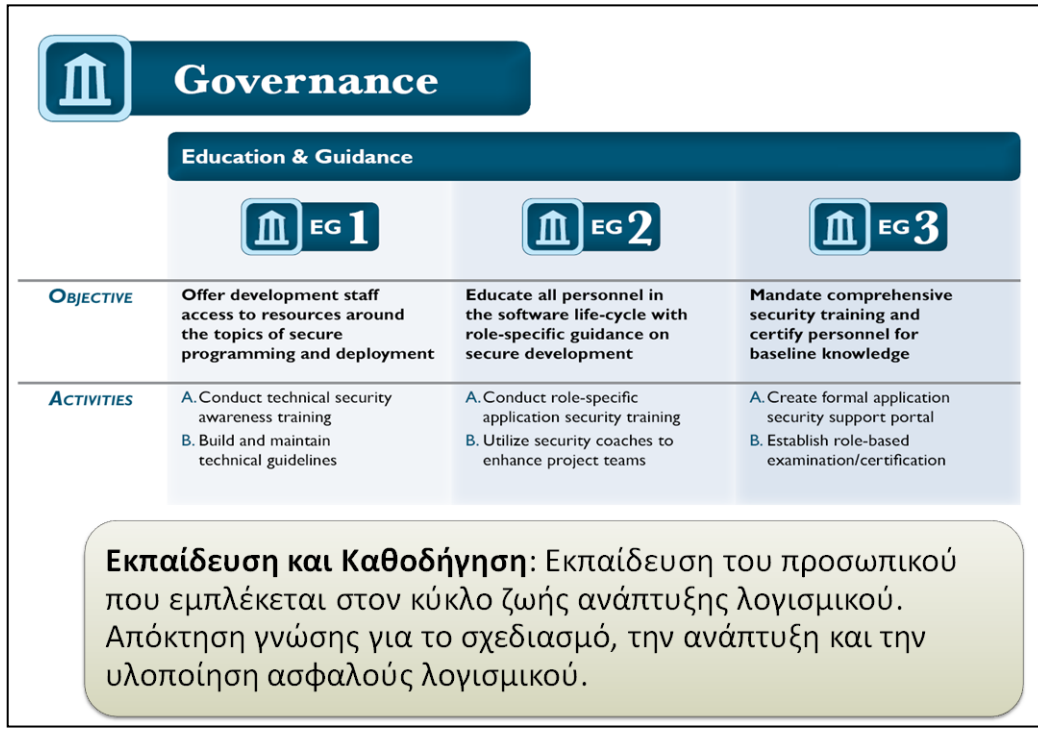
- Τρεις διαδοχικοί στόχοι για κάθε **Πρακτική Ασφάλειας** ορίζουν πώς μπορεί να βελτιωθεί με το χρόνο
  - Ορίζεται έτσι ένα **Επίπεδο** στο οποίο ένας οργανισμός ικανοποιεί τη συγκεκριμένη Πρακτική
- **Τρία Επίπεδα** αντιστοιχούν σε κάθε Πρακτική:
  - (0: Σημείο εκκίνησης. Δεν έχει γίνει τίποτα για την Πρακτική)
  - 1: Αρχική κατανόηση και υιοθέτηση της Πρακτικής κατά περίπτωση
  - 2: Αύξηση της αποδοτικότητας της Πρακτικής
  - 3: Συνολική, εις βάθος υιοθέτηση της πρακτικής



Στρατηγική και Μετρικές: Εγκαθίδρυση ενός πλαισίου ανάπτυξης προγράμματος διασφάλισης λογισμικού.

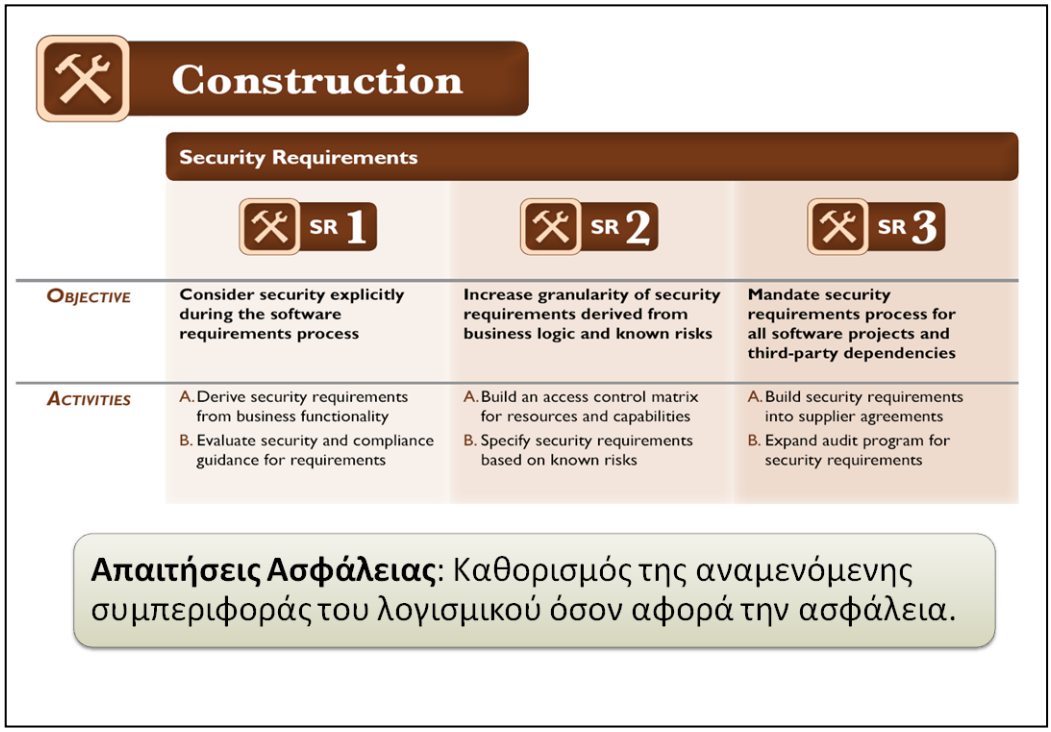


Πολιτικές και Συμμόρφωση: Κατανόηση και συμμόρφωση με εξωτερικές νομικές και κανονιστικές απαιτήσεις. Εγκαθίδρυση εσωτερικών προτύπων για την επίτευξη συμμόρφωσης σε συνδυασμό με τους επιχειρησιακούς στόχους του οργανισμού.



Εκπαίδευση και Καθοδήγηση: Εκπαίδευση του προσωπικού που εμπλέκεται στον κύκλο ζωής ανάπτυξης λογισμικού. Απόκτηση γνώσης για το σχεδιασμό, την ανάπτυξη και την υλοποίηση ασφαλούς λογισμικού.

Αξιολόγηση Απειλών: Αναγνώριση και κατανόηση των κινδύνων σε επίπεδο έργου και σε συνάρτηση με τη λειτουργικότητα του υπό ανάπτυξη λογισμικού και των χαρακτηριστικών του περιβάλλοντος εκτέλεσης.

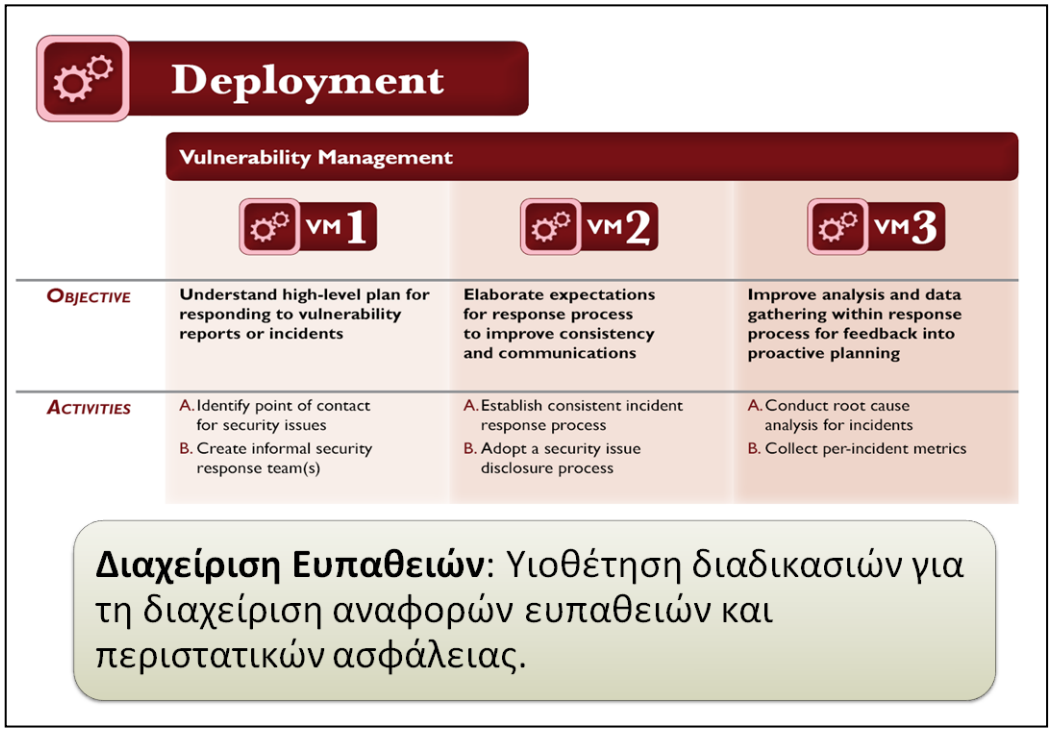

Construction
Security Requirements
SR 1
SR 2
SR 3
OBJECTIVE
Consider security explicitly during the software requirements process
Increase granularity of security requirements derived from business logic and known risks
Mandate security requirements process for all software projects and third-party dependencies
ACTIVITIES
A. Derive security requirements from business functionality
B. Evaluate security and compliance guidance for requirements
A. Build an access control matrix for resources and capabilities
B. Specify security requirements based on known risks
A. Build security requirements into supplier agreements
B. Expand audit program for security requirements
Απαιτήσεις Ασφάλειας: Καθορισμός της αναμενόμενης συμπεριφοράς του λογισμικού όσον αφορά την ασφάλεια.

Αρχιτεκτονική Ασφάλειας: Βήματα για το σχεδιασμό και την ανάπτυξη ασφαλούς λογισμικού.

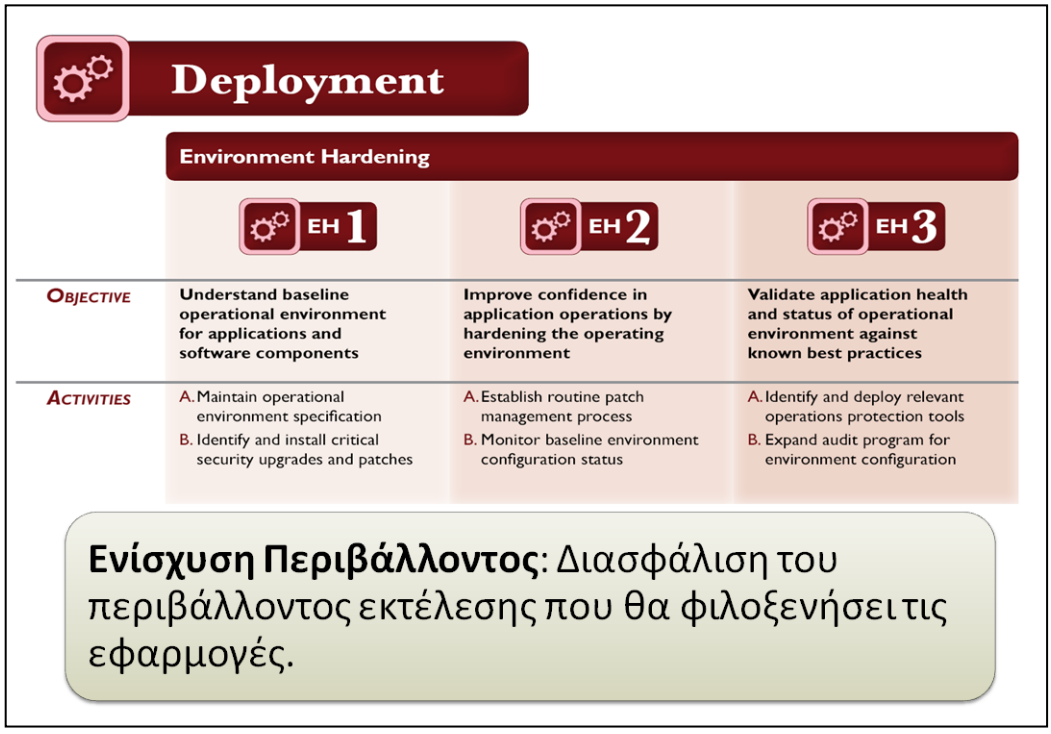
Επισκόπηση Σχεδιασμού: Αξιολόγηση αρχιτεκτονικής και σχεδιασμού για προβλήματα ασφάλειας.

Επισκόπηση Κώδικα: Αναζήτηση ευπαθειών σε επίπεδο πηγαίου κώδικα.

Έλεγχος Ασφάλειας: Αξιολόγηση του λογισμικού σε περιβάλλον εκτέλεσης για τον εντοπισμό προβλημάτων ασφάλειας.



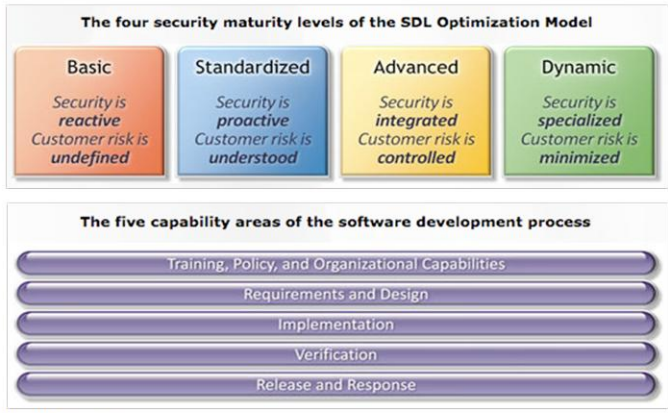
Διαχείριση Ευπαθειών: Υιοθέτηση διαδικασιών για τη διαχείριση αναφορών ευπαθειών και περιστατικών ασφάλειας.


Deployment
Environment Hardening
EH 1
EH 2
EH 3
OBJECTIVE
Understand baseline operational environment for applications and software components
Improve confidence in application operations by hardening the operating environment
Validate application health and status of operational environment against known best practices
ACTIVITIES
A. Maintain operational environment specification
B. Identify and install critical security upgrades and patches
A. Establish routine patch management process
B. Monitor baseline environment configuration status
A. Identify and deploy relevant operations protection tools
B. Expand audit program for environment configuration
Ενίσχυση Περιβάλλοντος: Διασφάλιση του περιβάλλοντος εκτέλεσης που θα φιλοξενήσει τις εφαρμογές.

Επιχειρησιακή Ενεργοποίηση: Συλλογή κρίσιμων πληροφοριών ασφάλειας από ομάδες έργου που αναπτύσσουν λογισμικό και μεταφορά τους στους χρήστες του λογισμικού.

Microsoft SDL Optimization Model

- Δημιουργήθηκε από τη Microsoft για να κάνει την υιοθέτηση του μοντέλου πιο εύκολη.



Πηγή: © Microsoft Corporation

BSIMM

- Ξεκίνησε μαζί με το SAMM
- Μοντέλο «ωριμότητας»
- Αρχικά βασίστηκε αρχικά σε πραγματικά δεδομένα από 9 μεγάλους οργανισμούς
- BSIMM2: Δεδομένα από 30 (σε σύνολο 60) οργανισμών

BSIMM2

- 4 τομείς – 12 πρακτικές
- Επιχειρησιακοί στόχοι ανά τομέα και πρακτική
- 109 δραστηριότητες για την επίτευξη των στόχων
- 3 επίπεδα ωριμότητας

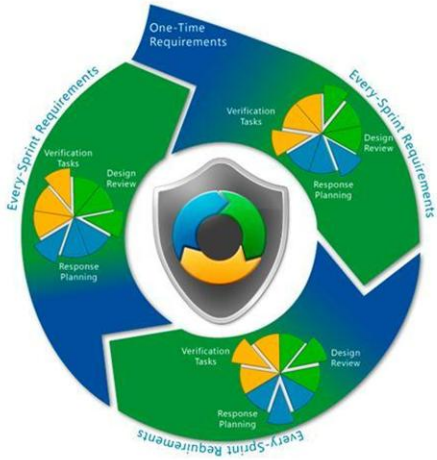
Governance	Intelligence	SSDL Touchpoints	Deployment
Strategy and Metrics	Attack Models	Architecture Analysis	Penetration Testing
Compliance and Policy	Security Features and Design	Code Review	Software Environment
Training	Standards and Requirements	Security Testing	Configuration Management and Vulnerability Management


Στην πράξη...
REALITY
CHECK
AHEAD

Agile Development

- "*Manifesto for Agile Software Development*" http://agilemanifesto.org/
- Βασίζεται σε μικρές, επαναλήψιμες διαδικασίες ανάπτυξης κώδικα (sprints)
  - Διάρκεια: 2 μέρες, 2 εβδομάδες, 2 μήνες
- Οι απαιτήσεις καθορίζονται δυναμικά στην πορεία από ετερογενείς ομάδες που δημιουργούνται κατά περίπτωση
- Δεν υπάρχει αρχικός σχεδιασμός-τεκμηρίωση

Microsoft SDL For Agile



Κατηγορίες απαιτήσεων για SDL:

- Σε κάθε Sprint
- Bucket
  - Επαλήθευση
  - Επισκόπηση σχεδιασμού
  - Σχεδιασμός αντιμετώπισης περιστατικών
- One-Time

Πηγή: © Microsoft Corporation

Απαιτήσεις για κάθε Sprint

- Ενημέρωση του μοντέλου απειλών
- Ενημέρωση του υπεύθυνου για την ιδιωτικότητα όταν προκύπτουν σχετικές αλλαγές στο σχεδιασμό.
- Επιδιόρθωση όλων των ευπαθειών που εντοπίζονται από εργαλεία ανάλυσης κώδικα
- Πιστή εφαρμογή των οδηγιών για επαλήθευση δεδομένων εισόδου και κωδικοποίηση δεδομένων εξόδου

Απαιτήσεις τύπου Bucket

- Ανάθεση προτεραιοτήτων σε δραστηριότητες που αφορούν πολλά sprints
- Για κάθε sprint, τουλάχιστον μια δραστηριότητα από κάθε «κουβά» πρέπει να ολοκληρώνεται.
- Κάθε δραστηριότητα πρέπει να ολοκληρώνεται τουλάχιστον μια φορά κάθε 6 μήνες.

Παραδείγματα:

- Επαλήθευση Ασφάλειας
- Εργαλεία fuzzing
- Επισκόπηση κώδικα
- Επισκόπηση σχεδιασμού
- Επισκόπηση απαιτήσεων ιδιωτικότητας
- Μοντελοποίηση απειλών σε βάθος
- Σχεδιασμός αντιμετώπισης περιστατικών
- Τεκμηρίωση

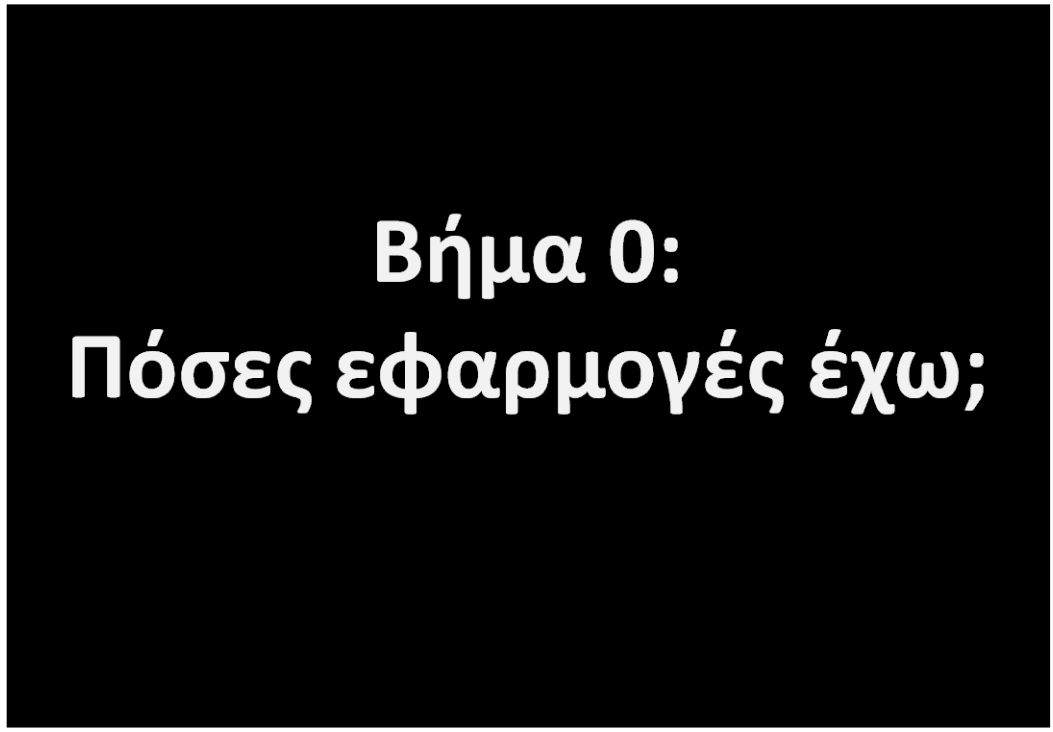
Απαιτήσεις τύπου One-Time

Γιατί;

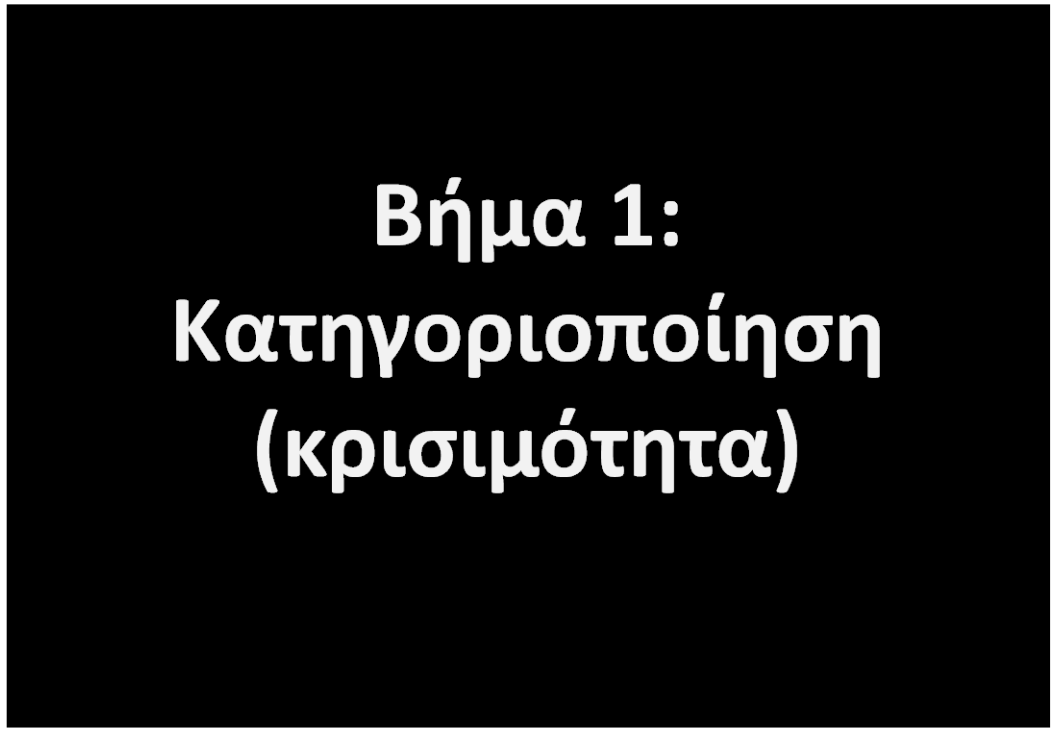
- Δεν απαιτείται επανάληψη
- Πρέπει να διενεργούνται στην αρχή του έργου
- Αδύνατο να εκτελεστούν από την αρχή του έργου

Παραδείγματα:

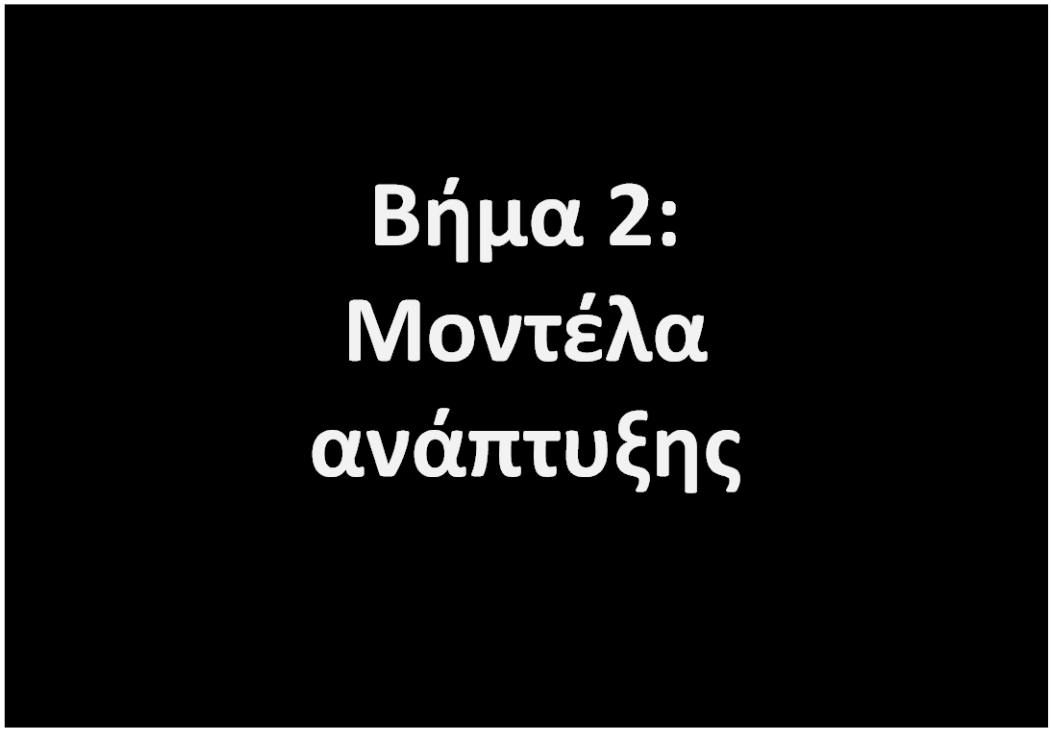
- Δημιουργία συστήματος παρακολούθησης σφαλμάτων (3 μήνες)
- Ορισμός ειδικών σε ασφάλεια και ιδιωτικότητα (1 μήνας)
- Μοντελοποίηση απειλών
- Καθορισμός σχεδίου αντιμετώπισης περιστατικών (6 μήνες)



Δημιουργία inventory εφαρμογών



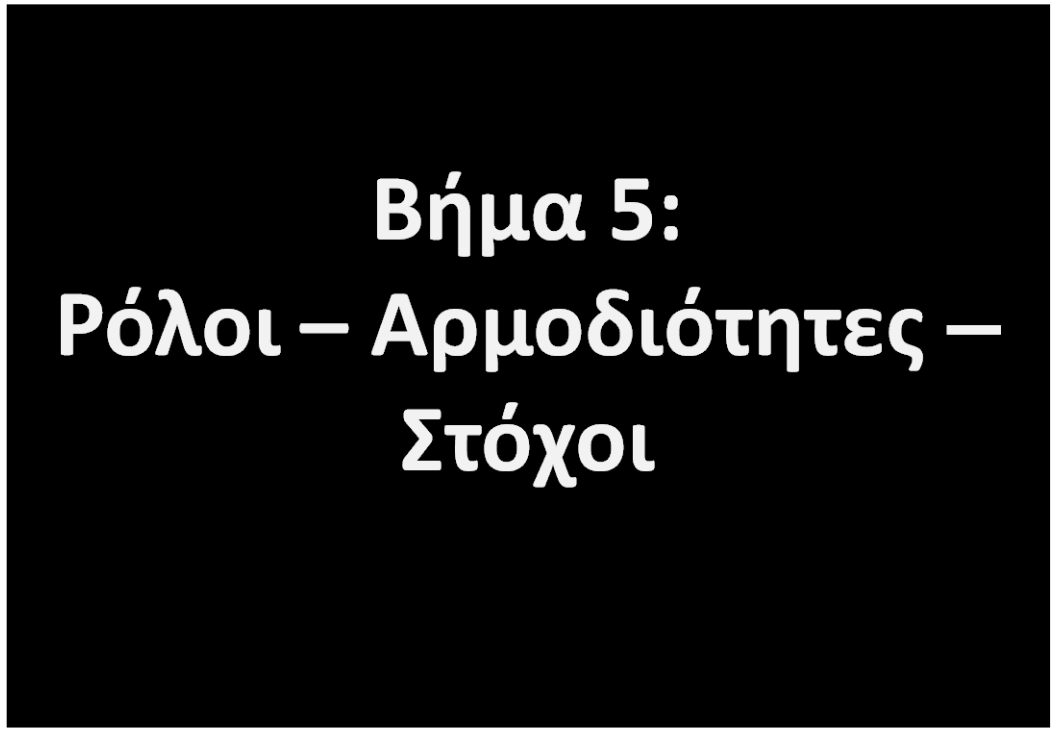
Κατηγοριοποίηση των assets (στην προκειμένη περίπτωση εφαρμογών) ανάλογα με την κρισιμότητά τους ώστε να γνωρίζουμε πού πρέπει να δοθεί περισσότερη προσοχή.



Απόφαση για υιοθέτηση συγκεκριμένων μοντέλων ανάπτυξης και κύκλου ζωής λογισμικού

Απόφαση για την υιοθέτηση συγκεκριμένου προγράμματος και διαδικασιών ασφάλειας.

Καθορισμός διαδικασιών ασφάλειας και μηχανισμών ελέγχων και αντιστοίχησή τους με τους στόχους ασφάλειας και την κρισιμότητα της εφαρμογής.



Καθορισμός συγκεκριμένων ρόλων, αρμοδιοτήτων και στόχων, ώστε κάθε μέλος της ομάδας ανάπτυξης να γνωρίζει με σαφήνεια τα καθήκοντά ως προς την ανάπτυξη λογισμικού.

Ορισμός συγκεκριμένων μετρικών ώστε να έχουμε μετρήσιμα και απτά αποτελέσματα αλλά και να επιτύχουμε βελτιώσεις στις διαδικασίες.

Επιδιορθώσεις ασφάλειας

- Αναγκαίο «κακό»
- «Φθηνές» για τις εταιρίες ανάπτυξης λογισμικού
- «Ακριβές» για τους χρήστες

Οι εταιρίες ανάπτυξης λογισμικού έχουν μικρό κόστος υλοποίησης διορθώσεων ασφαλείας (patches).

Απ' την άλλοι οι χρήστες επωμίζονται όλο το κόστος της εγκατάστασης των patches, τα πιθανά προβλήματα και τις ασυμβατότητες που μπορεί να προκύψουν καθώς και το κόστος παρακολούθησης και ενημέρωσης για νέες διορθώσεις.

Συχνά υλοποιούνται από μεγάλους οργανισμούς τεστ συστήματα στα οποία εφαρμόζονται τα patches για να ελεγχθεί η ορθή λειτουργία τους πριν περαστούν στα συστήματα παραγωγής.

Στατιστικά

- BSIMM: ~60 οργανισμοί ακολουθούν κάποιο πρότυπο ασφαλούς ανάπτυξης λογισμικού
  - Διάρκεια ωρίμανσης: 4,5 χρόνια (Μ.Ο.) - 14 χρόνια το μέγιστο
  - Ομάδα ασφαλούς ανάπτυξης: 21,9 (1% των προγραμματιστών)
  - Μέγεθος ομάδας προγραμματιστών: 5061
- OpenSAMM: ~50 οργανισμοί
- Microsoft SDL: ;;;

"So now, when we face a choice between adding features and resolving security issues, we need to choose security."

-Bill Gates

Βιβλιογραφία

- Gary McGraw, Brian Chess, and Sammy Migues, *Building Security In Maturity Model 2*, http://bsimm.com 2010.
- OWASP, *Security Assurance and Maturity Model*, http://www.opensamm.org/ 2009.
- Microsoft, *Secure Development Lifecycle,* http://www.microsoft.com/security/sdl/
- Microsoft, *SDL Optimization Model,* http://www.microsoft.com/security/sdl/getstarted/assess.aspx
- OWASP, *Comprehensive Lightweight Application Security Process,* http://www.owasp.org/index.php/Category:OWASP_CLASP_Project
- Matt Bishop, *Computer Security*, Addison-Wesley, 2003.
- Michael Howard and Steve Lipner, *The Security Development Lifecycle*, Microsoft Press, 2006.
- Gary McGraw, *Software Security: Building Security In*, Addison-Wesley, 2006.