OWASP

The Open Web Application Security Project

App Sec FAQ

Μετάφραση: **Καζακώνης Αναστάσιος**
ΙΟΥΝΙΟΣ 2005

Εισαγωγή
Τι πραγματεύεται αυτό το εγχειρίδιο?

Το εγχειρίδιο αυτό απαντά σε ερωτήσεις προγραμματιστών σχετικά με την ασφάλεια των δικτυακών εφαρμογών. Δεν αναφέρεται σε κάποια συγκεκριμένη γλώσσα ή πλατφόρμα. Επισημαίνει κοινές κινδύνους που απειλούν δικτυακές εφαρμογές και ανταποκρίνονται σε οποιαδήποτε πλατφόρμα.

Ποιοι είναι αυτοί οι συνήθεις κίνδυνοι για τις δικτυακές εφαρμογές;

Κατά την ανάπτυξη μιας εφαρμογής οι περισσότεροι από εμάς εστιάζουμε περισσότερο στη λειτουργικότητα παρά στην ασφάλεια αυτής. Οι επιτιθέμενοι εκμεταλλεύονται το γεγονός αυτό χρησιμοποιώντας την εφαρμογή με πολλούς διαφορετικούς τρόπους. Κάποιοι κοινοί κίνδυνοι δικτυακών εφαρμογών είναι SQL Injection, Cross Site Scripting, Variable Manipulation και εκμετάλλευση σημαντικών στοιχείων, όπως εκτεθειμένα passwords. Υπάρχουν ξεχωριστά τμήματα στο εγχειρίδιο αυτό που απαντούν σε απλές ερωτήσεις σχετικές με τους παραπάνω κινδύνους.

Ποιος δημιούργησε το εγχειρίδιο αυτό??
Το κείμενο αυτό αποτελεί ένα ντοκουμέντο που συνεχώς εξελίσσεται και βελτιώνεται μέσω της προσφοράς της κοινότητας που ασχολείται με την ασφάλεια των υπολογιστικών συστημάτων. Η Sangita Pakala και η ομάδα της από το Paladion Networks δημιούργησε την πρώτη έκδοση του εγχειριδίου αυτού.

Πως μπορώ να προσφέρω στην προσπάθεια που γίνεται στο εγχειρίδιο αυτό;

Χρειαζόμαστε τη βοήθειά σας και τη συνδρομή σας για να βελτιώσουμε το κείμενο αυτό. Θα ήταν ευχάριστο για εμάς να μας στείλετε υλικό σχετικά με:
- Νέες ερωτήσεις που μπορούμε να προσθέσουμε σε αυτό.
- Καλύτερες απαντήσεις στις ήδη υπάρχουσες ερωτήσεις
- Καινούριες συνδέσεις προς έγγραφα, δοκίμια, εργαλεία
- Προτάσεις για βελτίωση του υπάρχοντος κειμένου

Μπορείτε να μας στείλετε τις προτάσεις σας στο appsecfaq@owasp.org

Ζητήματα «Πρόσβασης»
Ποιες είναι οι καλύτερες πρακτικές που πρέπει να έχουμε κατά νου σχεδιάζοντας σελίδες πρόσβασης; (login pages)

- Από τη σελίδα για το login, ο χρήστης πρέπει να παραπέμπεται σε μια σελίδα εξακρίβωσης των στοιχείων του. Από τη στιγμή που πραγματοποιείται η εξακρίβωση, ο χρήστης μεταφέρεται στην επόμενη σελίδα. Αυτό το εξηγούμε στην απάντηση της επόμενης ερώτησης.
- Το συνθηματικό δεν πρέπει σε καμία περίπτωση να αποστέλλεται σε απλό κείμενο text (μη κρυπτογραφημένο) καθώς μπορεί να υποκλαπεί. Η αποθήκευση των συνθηματικών σε clear text μορφή σε βάση του συστήματος θεωρείται επίσης επικίνδυνη. Η καλύτερη μέθοδος κωδικοποίησης και αποστολής συνθηματικών είναι ο αλγόριθμος κωδικοποίησης Salted MD5.

• Ο καλύτερος τρόπος διαχείρισης των συνδέσεων θα ήταν η χρήση ενός session token με δύο τιμές κατά τη διάρκεια της πιστοποίησης. Μια τιμή για την πιστοποίηση και μια για μετά.

Είναι πραγματικά αναγκαίο να μεταφέρεται ο χρήστης σε νέα σελίδα μετά την εξακρίβωση των στοιχείων του;

Ναι. Σκεφθείτε ότι η σελίδα που εισάγουμε όνομα χρήστη και συνθηματικό στέλνει τα στοιχεία αυτά ως ένα πακέτο με μια αίτηση στον server. Αν ο χρήστης πατήσει το κουμπί "ανανέωση" στην δεύτερη σελίδα (τη σελίδα μετά το login), το ίδιο αίτημα, περιλαμβανομένων των user name και password θα αποσταλούν και πάλι στον server. Υποθέστε τώρα ότι ένας χρήστης κάνει χρήση της εφαρμογής και μετά αποσυνδέεται, χωρίς όμως να κλείσει το παράθυρο της εφαρμογής. Οι επιτιθέμενοι έρχονται αμέσως μετά και κάνουν κλικ στο κουμπί back του browser μέχρι να φτάσουν στην $2^{η}$ σελίδα. Το μόνο που έχουν να κάνουν είναι ένα refresh και το όνομα μαζί με το συνθηματικό του προηγούμενου χρήστη ξανά υποβάλλονται στον server και ο επιτιθέμενος πλέον είναι συνδεδεμένος με τα στοιχεία του προηγούμενου. Ας υποθέσουμε τώρα ότι η εφαρμογή διαθέτει μια σελίδα για login που μεταφέρει τον χρήστη σε μια ενδιάμεση σελίδα για εξακρίβωση. Από τη στιγμή που εξακριβώνονται τα στοιχεία του, ο χρήστης μεταφέρεται στη δεύτερη σελίδα με ένα session token. Στην περίπτωση αυτή, ακόμη και αν οι επιτιθέμενοι φτάσουν στη δεύτερη σελίδα και κάνουν ανανέωση, τα στοιχεία του χρήστη δεν ξανά υποβάλλονται στον διακομιστή. Και αυτό διότι το αίτημα που υποβάλλεται αφορά την δεύτερη σελίδα που δεν περιλαμβάνει τα ευαίσθητα στοιχεία. Ωστόσο, είναι πάντα αποτελεσματικότερη η μέθοδος της μεταφοράς του χρήστη.

Πως λειτουργεί η τεχνική κωδικοποίησης salted MD5;;

Να πως λειτουργεί η τεχνική κωδικοποίησης salted MD5: Μια βάση αποθηκεύει μια MD5 κωδικοποίηση του συνθηματικού (MD5 hash είναι μια κρυπτογραφική μέθοδος την οποία η πραγματική τιμή δεν μπορεί ποτέ να επαναφερθεί). Όταν ένα τερματικό καλεί την σελίδα πρόσβασης, ο server δημιουργεί έναν τυχαίο αριθμό, το κλειδί, και τον στέλνει στο τερματικό μαζί με τη σελίδα πρόσβασης. Ένα κομμάτι κώδικα σε JavaScript στο τερματικό υπολογίζει την MD5 τιμή του συνθηματικού που εισάγει ο χρήστης. Τότε συνδέει το κλειδί με το MD5 κωδικοποιημένο συνθηματικό και ξανα υπολογίζει την MD5 κωδικοποίησή του. Το αποτέλεσμα αυτό αποστέλλεται στον διακομιστή. Ο διακομιστής αντλεί το κλειδί του συνθηματικού από τη βάση του, συνδέει το κλειδί και υπολογίζει την MD5 τιμή του. Αν ο χρήστης χρησιμοποίησε το σωστό συνθηματικό, τότε τα δύο κλειδιά θα πρέπει να ταιριάζουν. Ο διακομιστής τα συγκρίνει και αν όντως ταιριάζουν, ο χρήστης πιστοποιείται.

Πως μπορεί να χρησιμοποιηθεί κακόβουλα η δυνατότητα ανάκτησης του συνθηματικού μου;

Η δυνατότητα ανάκτησης ενός ξεχασμένου συνθηματικού υλοποιείται με διαφορετικούς τρόπους κάθε φορά. Ένας συνηθισμένος τρόπος είναι να ρωτάς τον χρήστη μια προσωπική ερώτηση την απάντηση της οποίας ο χρήστης έχει ήδη δώσει κατά την εγγραφή του στην υπηρεσία. Αυτές είναι ερωτήσεις του τύπου: Ποιο είναι το αγαπημένο σου χρώμα; ή Ποιο είναι το όνομα του κατοικίδιού σου; Αν η απάντηση είναι σωστή, είτε το password αποκαλύπτεται, είτε δίνεται ένα προσωρινό συνθηματικό που μπορεί να χρησιμοποιηθεί για να δοθεί πρόσβαση στον χρήστη. Με τη μέθοδο αυτή, ο επιτιθέμενος,

προσπαθώντας να υποκλέψει το συνθηματικό ενός χρήστη, μπορεί να μαντέψει τη σωστή απάντηση στην ερώτηση που τίθεται και να αλλάξει τελικά το κανονικό password.

Κατά την ανάκτηση ξεχασμένου συνθηματικού, είναι ασφαλές αυτό να προβάλλεται στην οθόνη και να δίνεται στον χρήστη;

Αν το παλιό συνθηματικό προβάλλεται στην οθόνη του χρήστη, μπορεί εύκολα να υποκλαπεί από ανθρώπους που περνούν από τον χώρο εργασίας τη στιγμή εκείνη και έχουν οπτική επαφή με την οθόνη. Είναι λοιπόν καλή ιδέα να μην φανερώνεται το συνθηματικό αλλά να δίνεται η δυνατότητα στο χρήστη να ορίσει ένα νέο. Επίσης, το να μπορείς να προβάλλεις το συνθηματικό συνεπάγεται ότι αυτό ήταν αποθηκευμένο στη βάση σε μια μορφή που μπορεί να επανέλθει από την κρυπτογραφημένη σε text, πρακτική που δεν είναι τόσο ασφαλής. Αν το συνθηματικό αποθηκεύεται με έναν μονόδρομο αλγόριθμο κωδικοποίησης στη βάση, ο μόνος τρόπος να επαναφέρουμε ένα "ξεχασμένο συνθηματικό" είναι να αφήσουμε τον χρήστη να δώσει ένα καινούριο. Είναι, λοιπόν, καλύτερο να επιβάλλουμε στον χρήστη να αλλάξει κωδικό χωρίς να του δίνουμε τον παλιό του, όταν τον έχει ξεχάσει. (Μια μονόδρομη κωδικοποίηση είναι το αποτέλεσμα που παίρνουμε όταν περνάμε μια συμβολοσειρά από μια μονόδρομη συνάρτηση. Το αποτέλεσμα αυτό είναι αδύνατον να μας δώσει την αρχική τιμή από την οποία προήλθε. Τα συνθηματικά είναι αποτελεσματικότερο να αποθηκεύονται ως μη ανακτήσιμα στη βάση.)

Υπάρχει κίνδυνος εάν το νέο password αποσταλεί με mail στην εξουσιοδοτημένη e-mail διεύθυνση του χρήστη;

Η αποστολή του συνθηματικού μέσω e-mail σε μορφή text, αναγνώσιμη, δημιουργεί κινδύνους καθώς ο επιτιθέμενος μπορεί να το υποκλέψει. Ακόμη το ηλεκτρονικό μήνυμα που θα περιέχει το συνθηματικό μπορεί να αποθηκευτεί, χωρίς να καταστραφεί πράγμα που σημαίνει ότι μπορεί να πέσει στα χέρια του επιτιθέμενου μέσω του mailbox του χρήστη.

Ποιος είναι ο πιο ασφαλής τρόπος ασφαλής τρόπος να σχεδιάσουμε τη σελίδα επαναφοράς "ξεχασμένου συνθηματικού";;

Θα πρέπει πρώτα να ζητήσουμε από τον χρήστη να μας δώσει μερικές προσωπικές πληροφορίες ή να τον ρωτήσουμε μια προκαθορισμένη ερώτηση στην οποία έχει απαντήσει Μετά, πρέπει να σταλεί ένα mail στη δοσμένη κατά την εγγραφή από τον χρήστη ηλεκτρονική διεύθυνση που να περιέχει κάποιο link που θα τον παραπέμψει στη σελίδα στην οποία θα μπορέσει να κάνει reset το συνθηματικό του. Ο υπερσύνδεσμος που θα πηγαίνει τον χρήστη στη σελίδα αυτή, πρέπει να είναι ενεργό για μικρό χρονικό διάστημα και να τρέχει σε SSL πρωτόκολλο ασφαλείας, έτσι ώστε το συνθηματικό να μην φαίνεται στην οθόνη. Τα πλεονεκτήματα αυτής της μεθόδου ασφάλειας είναι τα ακόλουθα: 1. Το συνθηματικό δεν αποστέλλεται με ηλεκτρονικό ταχυδρομείο. Καθώς ο υπερσύνδεσμος είναι ενεργός για λίγο, δεν δημιουργείται πρόβλημα ακόμη και αν το ηλεκτρονικό μήνυμα παραμείνει στο mailbox του χρήστη και εκτεθεί καθ' οινδήποτε τρόπο.

Πως μπορώ να προστατευτώ από επιθέσεις με αυτόματα εργαλεία παραγωγής συνθηματικών;

Το να μαντεύονται συνθηματικά με τη χρήση αυτόματων εργαλείων είναι ένα σοβαρό πρόβλημα καθώς υπάρχουν πολλά προγράμματα που κάνουν τη δουλειά αυτή. Τα προγράμματα αυτά προσπαθούν συνεχώς να πετύχουν πρόσβαση σε λογαριασμούς κάνοντας χρήση τυχαίων διαφορετικών συνθηματικών, μέχρι κάποιο από αυτά να ταιριάζει με το σωστό. Μια καλή αμυντική μέθοδος εναντίον των προγραμμάτων αυτών είναι να κλειδώνεται ο λογαριασμός μετά από 5 αποτυχημένες προσπάθειες εισόδου σε αυτόν. Παρόλα αυτά, το σημαντικό εδώ είναι η χρονική διάρκεια για την οποία ο λογαριασμός παραμένει κλειδωμένος. Αν αυτό το διάστημα είναι μεγάλο, η υπηρεσία γίνεται μη διαθέσιμη ακόμη και στον πιστοποιημένο χρήστη αν κάποιος επιτιθέμενος αναλάβει δράση αν τακτά χρονικά διαστήματα. Αν, πάλι, το χρονικό αυτό διάστημα είναι μικρό (1 – 2 λεπτά για παράδειγμα) το εργαλείο παραγωγής συνθηματικών μπορεί να ξανά αρχίσει τις προσπάθειες μόλις αυτό παρέλθει. Βλέπουμε λοιπόν ότι η καλύτερη μέθοδος είναι να επιμείνουμε στην ανθρώπινη παρέμβαση μετά από μερικές αποτυχημένες προσπάθειες. Μια μέθοδος που χρησιμοποιείται από μερικούς δικτυακούς τόπους τον καιρό αυτό, είναι να ζητήσουν από τον χρήστη να διαβάσει και να πληκτρολογήσει μια τυχαία λέξη που εμφανίζεται σε μια εικόνα στη σελίδα. Καθώς τη δουλειά αυτή ΔΕΝ μπορεί να την κάνει κάποιο πρόγραμμα μπορούμε να αποκλείσουμε την υποκλοπή των συνθηματικών από τέτοια εργαλεία. Τα παρακάτω είναι μερικά προγράμματα που μαντεύουν συνθηματικά δικτυακών εφαρμογών:
Brutus - http://www.hoobie.net/brutus/
WebCracker - http://www.securityfocus.com/tools/706

Πως μπορώ να προστατευτώ από υποκλοπείς των πληκτρολογηθέντων στο τερματικό μου;

Υπάρχουν προγράμματα που ονομάζονται keystroke loggers και έχουν τη δυνατότητα να αποθηκεύουν οτιδήποτε πληκτρολογείται τοπικά σε κάποιο μηχάνημα. Προγράμματα σαν αυτά μπορούν να καταστρέψουν κάθε προσπάθεια μας ασφαλούς μετάδοσης των δεδομένων, καθώς δουλεύουν τοπικά σε κάθε μηχάνημα και υποκλέπτουν την αλληλουχία με την οποία πατιούνται τα πλήκτρα του πληκτρολογίου. Ο χρήστης είναι πολύ πιθανόν να μην γνωρίζει ότι κάτι τέτοιο έχει εγκατασταθεί στο μηχάνημά του και κρατάει αρχείο των πλήκτρων που πατάει ο χρήστης. Καθώς ο κίνδυνος της υποκλοπής στην περίπτωση αυτή πηγάζει από το πληκτρολόγιο, μπορούμε να δώσουμε στους χρήστες τη δυνατότητα να πιστοποιηθούν χωρίς να το χρησιμοποιήσουνε, κάνοντας δηλαδή χρήση άλλων μέσων. Οι διαφορετικοί αυτοί τρόποι μπορούν να είναι:

- Η χρήση ενός εικονικού πληκτρολογίου στην οθόνη, όπου ο χρήστης μπορεί να εισάγει χαρακτήρες κάνοντας κλικ πάνω στους χαρακτήρες του με το ποντίκι. Κάτι τέτοιο είναι κυρίως χρήσιμο για αριθμητικά συνθηματικά.
- Να ζητήσουμε από τον χρήστη να εισάγει μέρος του συνθηματικού του, και όχι ολόκληρο. Για παράδειγμα θα μπορούσαμε να ζητήσουμε να εισάγει τον $1^{ο}$, τον $2^{ο}$ και τον $6^{ο}$ χαρακτήρα του συνθηματικού του. Ο κανόνας αυτός θα ζητάει κάθε φορά τυχαίο αριθμό και σειρά χαρακτήρων από το συνθηματικό.

Ο δικτυακός μου τόπος χρησιμοποιείται από τερματικά διαθέσιμα σε ευρύ κοινό. Τι προφυλάξεις πρέπει να πάρω;

Αν η εφαρμογή που διαχειριζόμαστε εισάγει στο σύστημα χρήστες που χρησιμοποιούν τερματικά διαθέσιμα σε ευρύ κοινό, όπως τερματικά σε βιβλιοθήκες, μπορούμε να πάρουμε τις ακόλουθες προφυλάξεις:

- Μπορούμε να εξασφαλίσουμε ότι οι σελίδες δεν αποθηκεύονται στην cache μνήμη του συστήματος, κάνοντας τις ανάλογες ρυθμίσεις στις οδηγίες ελέγχου της μνήμης cache **.**
- Μπορούμε να εξασφαλίσουμε ότι καμία ευαίσθητη πληροφορία δεν περιλαμβάνεται στο URL καθώς το "Αρχείο Επισκέψεων" του browser τα αποθηκεύει μετά από κάθε επίσκεψη.
- Μπορούμε να χρησιμοποιήσουμε ένα εικονικό πληκτρολόγια στην οθόνη για την εισαγωγή του συνθηματικού από τον χρήστη ή να του ζητήσουμε να εισάγει συγκεκριμένους χαρακτήρες από το συνθηματικό του. Τα μέτρα αυτά μας προστατεύουν από υποκλοπείς των πληκτρολογηθέντων που περιγράψαμε παραπάνω.
- Για να αποτρέψουμε υποκλοπές των συνθηματικών μέσω του δικτύου κατά την μεταφορά τους μέσα σε αυτό, και την επανάληψη των επιθέσεων κάνοντας επανειλημμένη χρήση αυτών, πρέπει να χρησιμοποιούμε SSL πρωτόκολλο ασφαλείας ή κωδικοποίηση MD5 για τα συνθηματικά. Το συνθηματικό που πληκτρολογείται σε μορφή text πρέπει να διαγράφεται από τη μνήμη μετά τον υπολογισμό της MD5 μορφής του.

SQL Injection

Τι είναι η SQL Injection?

Η SQL Injection είναι μια τεχνική με την οποία οι επιτιθέμενοι μπορούν να εκτελέσουν ερωτήσεις της επιλογής τους στην SQL για τη βάση, υπολογίζοντας την είσοδο για την εφαρμογή. Ας εξηγήσουμε την SQL Injection μέσα από ένα παράδειγμα μιας σελίδα πιστοποίησης χρηστών μιας δικτυακής εφαρμογής όπου η βάση δεδομένων που τρέχει στο background είναι ο SQL server. Ο χρήστης καλείται να εισάγει το "Όνομα Χρήστη" και το συνθηματικό του στα αντίστοιχα πεδία της σελίδας login.asp. Υποθέτουμε εισάγει τα στοιχεία: Username: *"obelix"* και Password: *"dogmatix"*. Αυτή η εισαγωγή στοιχείων δημιουργεί δυναμικά μια SQL ερώτηση της μορφής: *SELECT * FROM Users WHERE username= 'Obelix' and password='Dogmatix'* Το ερώτημα αυτό θα επιστρέψει στην εφαρμογή μια πλειάδα από τη βάση που ανταποκρίνεται στις δοσμένες τιμές. Ο χρήστης θεωρείται πιστοποιημένος αν η βάση επιστρέψει μια ή περισσότερες πλειάδες στην εφαρμογή. Υποθέστε τώρα ότι ο χρήστης εισάγει στα αντίστοιχα πεδία τα εξής στοιχεία: Username: ' ή 1=1--. Η SQL ερώτηση που θα δημιουργηθεί θα είναι κάπως έτσι: *SELECT * FROM Users WHERE username='' ή 1=1--* και *password=''* Οι χαρακτήρες – στον SQL Server χρησιμοποιούνται για να δηλώσουν το υπόλοιπο της γραμμής. Το ερώτημα λοιπόν γίνεται: SELECT * FROM Users WHERE username='' or 1=1 Το ερώτημα αυτό ψάχνει μέσα στη βάση για πλειάδες όπου το όνομα είτε είναι κενό, είτε ικανοποιεί τη συνθήκη 1 = 1. Καθώς το τελευταίο είναι πάντα αληθές, το ερώτημα θα επιστρέψει όλες τις πλειάδες του πίνακα και ο χρήστης θα πιστοποιηθεί κανονικά. Ο επιτιθέμενος λοιπόν καταφέρνει να κάνει login στην εφαρμογή χωρίς να χρησιμοποιήσεις κάποιο username και password. Μπορείτε να διαβάσετε περισσότερα σχετικά με το θέμα αυτό στον δικτυακό τόπο της Securiteam και στο URL:
http://www.securiteam.com/securityreviews/5DP0N1P76E.html

Είναι μόνο η ASP και ο SQL Server ευάλωτοι σε επιθέσεις ή όλες οι πλατφόρμες είναι τρωτές;

Οι περισσότερες πλατφόρμες είναι τρωτές σε επιθέσεις SQL Injection. Ανεπαρκής έλεγχος της εισόδου που δίνει ο χρήστης και η χρήση δυναμικών SQL ερωτήσεων είναι

τα δύο στοιχεία που κάνουν ευάλωτες τις εφαρμογές σε τέτοιου είδους επιθέσεις. Η σύνταξη της εισόδου που δίνει ο χρήστης για επίθεση με SQL Injection εξαρτάται από τη βάση που χρησιμοποιείται. Κατά τη διάρκεια που γράφαμε το δοκίμιο αυτό, βρήκαμε πολλές εφαρμογές που έκαναν χρήση άλλων βάσεων και ήταν τρτές σε τέτοιου είδους επιθέσεις. Το παραπάνω παράδειγμα θα μπορούσε να δουλέψει σε SQL Server, Oracle και MySQL. Αυτό δείχνει ότι το πρόβλημα εντοπίζεται στην ανυπαρξία ελέγχου της εισόδου που δίνει ο χρήστης και της χρήσης δυναμικών ερωτήσεων σε SQL και όχι στην ίδια τη βάση που τρέχει από πίσω.

Εκτός από το "Όνομα χρήστη" και το "Συνθηματικό" ποιες άλλες παράμετροι είναι μπορούν να προσβληθούν από SQL Injection?

Οποιοδήποτε πεδίο εισαγωγής στοιχείων που δημιουργεί το WHERE τμήμα της SQL ερώτησης σε μια βάση είναι υποψήφιο να προσβληθεί από SQL Injection, πχ. Αριθμοί λογαριασμών και αριθμοί πιστωτικών καρτών, στην περίπτωση μιας online τραπεζικής εφαρμογής. Εκτός των πεδίων εισαγωγής στοιχείων μιας φόρμας, ο επιτιθέμενος μπορεί να χρησιμοποιήσει και κρυμμένα πεδία και συμβολοσειρές ερωτήσεων επιπροσθέτως για να εξάγει εντολές ή ερωτήσεις προς τη βάση.

Πως μπορούμε να αποτρέψουμε SQL Injection επιθέσεις στις εφαρμογές μας;

Είναι αρκετά απλό να εμποδίσουμε επιθέσεις με SQL injection κατά την ανάπτυξη της εφαρμογής μας. Χρειάζεται να ελέγχουμε όλες τις εισόδους που προέρχονται από τον client πριν δημιουργήσουμε μέσω αυτών μια SQL ερώτηση. Η καλύτερη μέθοδος είναι να απομακρύνουμε όλες τις ανεπιθύμητες εισόδους και να κάνουμε δεκτές μόνο τις αναμενόμενες. Καθώς η επικύρωση της ορθότητας της εισόδου από τον διακομιστή είναι η πιο αποτελεσματική μέθοδος αποτροπής SQL Injection επιθέσεων, η άλλη μέθοδος αποτροπής είναι η μη χρήση δυναμικών ερωτήσεων σε SQL. Αυτό μπορεί να επιτευχθεί με τη χρήση αποθηκευμένων διεργασιών ή συγκεκριμένων μεταβλητών σε βάσεις δεδομένων που υποστηρίζουν αυτά τα χαρακτηριστικά. Για εφαρμογές γραμμένες σε JAVA, μπορούν να χρησιμοποιηθούν CallableStatements και PreparedStatements. Για εφαρμογές σε ASP, μπορούν να χρησιμοποιηθούν ADO Command Objects. Μπορείτε να δείτε το παρακάτω άρθρο για περισσότερες πληροφορίες σχετικά με την SQL Injection στην Oracle
http://www.integrigy.com/info/IntegrigyIntrotoSQLInjectionAttacks.pdf

Χρησιμοποιώ αποθηκευμένες διεργασίες για πιστοποίηση των χρηστών. Είμαι ευάλωτος;

Όχι. Κάνοντας χρήση αποθηκευμένων διεργασιών προστατευόμαστε από SQL Injection γιατί η είσοδος που δίνει ο χρήστης δεν χρησιμοποιείται πλέον για την δυναμική κατασκευή ερωτήσεων. Από τη στιγμή που οι αποθηκευμένες διεργασίες είναι μια ομάδα μεταγλωττισμένων SQL δηλώσεων και η διαδικασία δέχεται την είσοδο σαν παράμετρο, η δυναμική ερώτηση αποφεύγεται. Παρότι η είσοδος εισάγεται στην μεταγλωττισμένη ερώτηση ως έχει, καθώς η ερώτηση από μόνη της βρίσκεται σε διαφορετικό format, δεν υπόκειται στις συνέπειες των αλλαγών της ερώτησης όπως θα περιμέναμε. Με τη χρήση αποθηκευμένων διεργασιών, επιτρέπουμε στη βάση να χειριστεί την εκτέλεση της ερώτησης αντί να ζητήσει την εκτέλεση της ερώτησης που εμείς φτιάξαμε.

Χρησιμοποιώ client side JavaScript κώδικα για τον έλεγχο των δεδομένων που εισάγει ο χρήστης. Δεν είναι αυτό αρκετό;

Όχι. Παρότι ο έλεγχος από τη μεριά του client αποτρέπει τον επιτιθέμενο από το να εισάγει κακόβουλα δεδομένα κατ' ευθείαν στα πεδία εισόδου, αυτό από μόνο του δεν είναι αρκετό για να αποτρέψει SQL Injection επιθέσεις. Τα client side scripts απλά ελέγχουν την είσοδο στον browser. Αλλά αυτό δεν εγγυάται ότι οι πληροφορίες θα παραμείνουν ίδιες μέχρι να φτάσουν στον διακομιστή. Υπάρχουν εργαλεία που μπορούν να αντλήσουν το αίτημα στην πορεία του από τον client στον server και να το αλλάξουν πριν σταλεί σε αυτόν. Ο επιτιθέμενος μπορεί επίσης να στείλει εντολές στο πεδίο εισαγωγής ερωτήσεων μέσω μεταβλητών, που δεν είναι ελεγμένες από τα scripts που τρέχουν στον client.

Τα Java servlets είναι τρωτά σε SQL injection επιθέσεις;

Ναι, είναι αν η είσοδος που δώσει ο χρήστης δεν ελεγχθεί σωστά και οι SQL ερωτήσεις δημιουργούνται δυναμικά. Αλλά τα Java servlets έχουν επιπλέον δυνατότητες που εμποδίζουν επιθέσεις SQL Injection όπως CallableStatements και PreparedStatements. Όπως οι αποθηκευμένες διεργασίες και συγκεκριμένες μεταβλητές υπεκφεύγουν της ανάγκης δυναμικής δημιουργίας SQL δηλώσεων.

Διαχείριση των μεταβλητών
Γιατί δεν μπορώ να εμπιστευτώ την πληροφορία που έρχεται από τον browser;

Υπάρχει η πιθανότητα να έχουν μορφοποιηθεί πριν φτάσουν στον διακομιστή. Ο επιτιθέμενος καθώς πλοηγείται στον δικτυακό τόπο μπορεί να διαχειριστεί κακόβουλα την πληροφορία με ένα GET ή POST αίτημα. Υπάρχουν κάποια εργαλεία όπως το Achilles που μπορούν να παρεμβληθούν της μεταβιβαζόμενης πληροφορίας και να επιτρέψουν στον επιτιθέμενο που τρέχει το εργαλείο αυτό να την τροποποιήσει. Επίσης, η πληροφορία που ο χρήστης βλέπει ή παρέχει σε μια σελίδα του διαδικτύου πρέπει να ταξιδέψει μέσα στο internet πριν φτάσει στον server. Παρότι ο client και ο server μπορεί να είναι ασφαλείς και να τους εμπιστευόμαστε, δεν μπορούμε να είμαστε σίγουροι ότι η πληροφορία αυτή δεν έχει τροποποιηθεί αφού φύγει από τον browser. Ο επιτιθέμενος μπορεί να "συλλάβει" την πληροφορία στη διαδρομή και να την τροποποιήσει κακόβουλα.

Τι είδους πληροφορία μπορεί να διαχειριστεί κακόβουλα ο επιτιθέμενος;

Η διαχείριση των μεταβλητών σε κάποιο URL είναι απλή υπόθεση. Αλλά ο επιτιθέμενος μπορεί επίσης να διαχειριστεί σχεδόν όλη την πληροφορία από τον client στον server όπως τα πεδία μιας φόρμας εισαγωγής, και κρυμμένα πεδία.

Πως μπορεί ο επιτιθέμενος να διαχειριστεί κακόβουλα την πληροφορία; Τι εργαλεία θα μπορούσε να χρησιμοποιήσει;

Για την διαχείριση και παραποίηση οποιασδήποτε πληροφορίας, συμπεριλαμβανομένων πεδίων εισαγωγής κρυμμένων μεταβλητών και cookies, ο επιτιθέμενος χρησιμοποιεί εργαλεία γνωστά ως HTTP proxy tools. Από τη στιγμή που τα proxy settings του browser έχουν ρυθμιστεί να "βγαίνει έξω" μέσω του web proxy, τα εργαλεία αυτά μπορούν να δουν όλη την πληροφορία που διακινείται ανάμεσα στον client και τον server. Επιτρέπει

ακόμη στον επιτιθέμενο να μεταβάλει οποιοδήποτε μέρος του αιτήματος ή της απάντησης πριν αυτό σταλεί. Μερικά εργαλεία σας αυτά είναι: WebScarab που μπορείτε να το βρείτε στον δικτυακό τόπο του OWASP www.owasp.org και το Odysseus που μπορείτε να βρείτε εδώ:
http://www.wastelands.gen.nz/odysseus/index.php

Χρησιμοποιώ SSL. Μπορεί ακόμη και έτσι κάποιος επιτιθέμενος να τροποποιήσει την διακινούμενη πληροφορία;

Παρότι το SSL παρέχει μεγάλη ασφάλεια, από μόνο του δεν είναι αρκετό για να αποτρέψει επιθέσεις μέσω υποκλοπής των διακινούμενων δεδομένων και κακόβουλης διαχείρισης της πληροφορίας. Το SSL είναι χρήσιμο για να αποτρέψει επιθέσεις από ανθρώπινο χέρι στο ενδιάμεσο της διαδικασίας, όπου ο επιτιθέμενος παρεμβάλλεται στην ενότητα που δουλεύει ένας χρήστης βλέπει τα περιεχόμενά της και μεταβάλει τα δεδομένα της. Αλλά δεν μπορεί από μόνο του να εμποδίσει έναν επιτιθέμενο από το να παρεμβάλει τη δική του σύνδεση και να μεταβάλει τις μεταβλητές και τα δεδομένα που διακινούνται. Ας δούμε πως το Achilles λειτουργεί πάνω στο SSL για να πάρει πρόσβαση και να μεταβάλει τα δεδομένα του: Το Achilles έχει ένα ψεύτικο πιστοποιητικό με ένα ζευγάρι κλειδιών που δημιούργησε από μόνο του. Όταν ο client ζητάει την προστατευμένη από SSL σελίδα, το Achilles το στέλνει όπως είναι στον server. Ο server τότε, στέλνει το δικό του πιστοποιητικό με το δημόσιο κλειδί ως απάντηση. Το Achilles τώρα παρεμβάλλεται σε αυτό, δημιουργεί ένα νέο κλειδί και το στέλνει στον server κρυπτογραφημένο μαζί με το δημόσιο κλειδί του server. Καταφέρνει λοιπόν να δημιουργήσει επιτυχώς μια SSL σύνδεση με τον server. Τώρα, από τη μεριά του client, το Achilles στέλνει το δικό του πιστοποιητικό και δημόσιο κλειδί σε αυτόν. Ο browser του client θα δείξει ένα μήνυμα που θα λέει ότι το πιστοποιητικό δεν το εμπιστεύεται και θα ρωτήσει αν πρέπει να γίνει αποδεκτό ή όχι. Καθώς όμως μιλάμε για τον browser του επιτιθέμενου και ο επιτιθέμενος έχει βάλει το Achilles, θα δεχθεί το πιστοποιητικό. Τώρα ο client δημιουργεί ένα session key, το κωδικοποιεί με το δημόσιο κλειδί του Achilles και το στέλνει. Τώρα λοιπόν το Achilles έχει δημιουργήσει δύο SSL συνδέσεις. Μια με τον server και άλλη μια με τον client. Αποκωδικοποιεί τις πληροφορίες που έρχονται από τον server, τις προβάλει σε μορφή απλού κειμένου στον επιτιθέμενο, και μετά τις κωδικοποιεί ξανά με το κλειδί του client και τις στέλνει. Μια παρόμοια μέθοδος ακολουθεί για την κίνηση από την αντίθετη κατεύθυνση.

Υπάρχει κάποιος τρόπος να εμποδίσω αυτά τα proxy tools από το να επεξεργαστούν τα δεδομένα μου;

Ο κυριότερος κίνδυνος που πηγάζει από αυτά τα εργαλεία είναι η πρόσβαση και η επεξεργασία των δεδομένων που αποστέλλονται από τον client στον server. Ένας τρόπος για να εμποδίσουμε τη διαδικασία αυτή είναι να πιστοποιήσουμε το μήνυμα που αποστέλλεται από τον client με ένα Java Applet που θα κατεβάσουμε στο τερματικό. Καθώς το applet που θα δημιουργήσουμε θα είναι αυτό που θα πιστοποιεί την εγκυρότητα του πιστοποιητικού, και όχι ο browser, ένα proxy tool δεν θα μπορεί να παρέμβει ανάμεσα στον client και τον server με ένα ψεύτικο πιστοποιητικό. Το applet θα απορρίψει το ψεύτικο πιστοποιητικό. Το δημόσιο κλειδί του πιστοποιητικού αυτού, μπορεί τότε να χρησιμοποιηθεί για να "υπογράψει" ψηφιακά κάθε μήνυμα που αποστέλλεται ανάμεσα στον client και τον server. Ένας επιτιθέμενος θα πρέπει τότε να αντικαταστήσει το εμφωλευμένο στο applet πιστοποιητικό με ένα ψεύτικο πιστοποιητικό

για να επιτύχει. Το γεγονός αυτό ανεβάζει τον πήχη όσον αφορά τον βαθμό δυσκολίας για τον επιτιθέμενο.

Η προσωρινή μνήμη (cache) του browser
Πως μπορεί η cache μνήμη του browser να χρησιμοποιηθεί σε επιθέσεις;

Ο browser έχει την ικανότητα να αποθηκεύει προσωρινά κάποιες σελίδες που επισκέφθηκε πρόσφατα. Αυτές οι σελίδες αποθηκεύονται (cacháρονται) σε έναν κατάλογο όπως ο Temporary Internet Files στην περίπτωση του Internet Explorer. Όταν ξαναζητήσουμε τις σελίδες αυτές, ο browser τις αντλεί από την cache. Η διαδικασία αυτή είναι πολύ πιο γρήγορη από το να ξανακατεβαίνουν οι σελίδες από τον server τους. Ας σκεφτούμε ένα σενάριο όπου ο χρήστης έχει κάνει login σε κάποια εφαρμογή χρησιμοποιώντας ένα username και ένα password. Ο χρήστης πλοηγείται ανάμεσα σε διαφορετικές σελίδες που περιέχουν ευαίσθητες πληροφορίες. Ας υποθέσουμε ότι μια σελίδα περιέχει τον αριθμό της πιστωτικής κάρτας του χρήστη. Και εκείνη cacháρεται στον browser. Ο χρήστης κάνει logout από την εφαρμογή που χρησιμοποίησε. Υποθέστε τώρα ότι ο επιτιθέμενος παίρνει πρόσβαση στο ίδιο μηχάνημα και ψάχνει μέσα στα Temporary Internet Files. Τότε μπορεί εύκολα να βρει λεπτομέρειες σχετικά με την πιστωτική κάρτα του χρήστη. Ο επιτιθέμενος δεν χρειάζεται να ξέρει το user name και το password του χρήστη για να υποκλέψει τις πληροφορίες αυτές.

Πως μπορώ να σιγουρέψω ότι οι σελίδες με ευαίσθητες πληροφορίες δεν cacháρονται;

Ο header της σελίδας που έρχεται από τον server περιέχει κάποιες οδηγίες σχετικά με τον τρόπο που θα χειριστεί η μνήμη cache τα περιεχόμενα της σελίδας αυτής. Οι οδηγίες αυτές έχουν τη μορφή *cache-control : no-cache* ή *cache-control : no-store*.

Ποια η διαφορά ανάμεσα στις οδηγίες για την cache μνήμη: *no-cache* και *no-store*;

Η οδηγία no-cache σε κάποια σελίδα που αποστέλλεται στον client σημαίνει ότι η απάντηση αυτή δεν πρέπει να χρησιμοποιηθεί για να δοθεί απάντηση σε κάποιο μεταγενέστερο παρόμοιο ερώτημα. Ειδικότερα, η μνήμη cache δεν πρέπει να δίνει κάποια απάντηση όταν η σελίδα περιέχει την οδηγία αυτή στον header της αλλά πρέπει να επιτρέπει στον server να εξυπηρετήσει το αίτημα αυτό. Η οδηγία no-cache μπορεί να περιέχει κάποια ονόματα πεδίων. Στην περίπτωση αυτή οι απαντήσεις μπορούν να δοθούν από την cache, όχι όμως αυτές που περιέχονται στα πεδία εκείνα που στον header ορίζεται ότι πρέπει να εξυπηρετηθούν από τον server.

Οι no-store οδηγίες αναφέρονται σε ολόκληρο το μεταφερόμενο μήνυμα και δείχνει ότι η προσωρινή μνήμη δεν πρέπει να αποθηκεύει κανένα μέρος από την δοσμένη απάντηση ή από την ερώτηση που ζήτησε την απάντηση αυτή.

Είμαι απολύτως ασφαλής με τη χρήση των οδηγιών αυτών;

Οι οδηγίες αυτές επιλύουν το πρόβλημα της αποθήκευσης στην προσωρινή μνήμη σε κάποιες περιπτώσεις αλλά όχι εντελώς, καθώς οι οδηγίες no-cache και no-store δεν υποστηρίζονται από τις μνήμες HTTP 1.0. Έχουμε, επίσης, παρατηρήσεις ότι οι σελίδες που δεν είναι σε html μορφή, όπως τα PDF και τα λογιστικά φύλλα του Excel,

cachάρονται στον browser ακόμη και αν έχουν δοθεί αυτές οι οδηγίες στο header της σελίδας.

Πως μπορώ να μάθω περισσότερα για τη διαδικασία αποθήκευσης στην προσωρινή μνήμη;

Μερικά χρήσιμα links που πραγματεύονται το ζήτημα του caching είναι: Caching Tutorial for Web Authors and Webmasters του Mark Nottingham στο
http://www.mnot.net/cache_docs/
και το HTTP RFC στο
http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.9.1

Cross Site Scripting
Τι είναι το Cross Site Scripting?
Το Cross Site scripting (XSS) είναι ένα είδος επίθεσης που μπορεί να χρησιμοποιηθεί για να υποκλαπούν ευαίσθητες πληροφορίες που ανήκουν στους χρήστες ενός δικτυακού τόπου. Αυτό στηρίζεται στην αντίδραση του server όταν δέχεται input από τον χρήστη χωρίς να ελέγχει την ύπαρξη ενός εμφωλευμένου JavaScript. Τέτοιες μέθοδοι μπορούν να χρησιμοποιηθούν για να υποκλαπούν cookies και session IDs. Ας δούμε πως λειτουργεί. Όλοι μπορεί να έχουμε συναντήσει κάποια φορά την εξής περίπτωση: - Πληκτρολογούμε κάποιο URL στον browser, ας πούμε www.abcd.com/mypage.asp και λαμβάνουμε ένα μήνυμα λάθους που λέει "Sorry www.abcd.com/mypage.asp does not exist" ή μια σελίδα με κάποιο παρόμοιο μήνυμα. Με άλλα λόγια, σελίδες που περιέχουν το input που έδωσε ο χρήστης στον browser. Σελίδες αυτές, μπορούν να αποδειχθεί ότι χρησιμοποιούν XSS. Αντί για ένα κανονικό – αναμενόμενο input, σκεφθείτε τι θα μπορούσε να συμβεί αν το input περιείχε κάποιο script. Κατά την επιστροφή της εισόδου, αντί να προβληθεί μια κανονική HTML σελίδα ως έξοδος, ο browser το χειρίζεται σαν να είναι κάποιο script και το εκτελεί. Το script αυτό θια μπορούσε να περιέχει κώδικα που να παραβιάζει την ασφάλεια του συστήματος. Ο επιτιθέμενος μπορεί τότε να στείλει ένα link που να περιέχει ένα script ως μέρος του URL σε έναν χρήστη. Όταν ο χρήστης κάνει κλικ πάνω σε αυτό, το script εκτελείται στον υπολογιστή του χρήστη. Το script αυτό μπορεί να έχει γραφτεί για να συλλέξει σημαντικές πληροφορίες σχετικές με τον χρήστη και να τις στείλει στον επιτιθέμενο. Το δοκίμιο του Kevin Spett's Cross Site Scripting, Are your web applications vulnerable? είναι μια καλή πηγή πληροφοριών πάνω σε αυτό το θέμα και είναι διαθέσιμο στο
http://www.spidynamics.com/whitepapers/SPIcross-sitescripting.pdf
To Cross Site Scripting FAQ στο CGI Security που είναι ακόμη μια καλή πηγή για να μάθετε περισσότερα σχετικά με το XSS.

Τι πληροφορίες μπορεί να υποκλέψει κάποιος επιτιθέμενος κάνοντας χρήση του XSS?

Οι επιτιθέμενοι μπορούν να υποκλέψουν το session ID ενός έγκυρου χρήστη χρησιμοποιώντας το XSS. Το session ID είναι πολύτιμο καθώς αποτελεί τον μυστικό συνδετικό κρίκο ανάμεσα στον client και τον server για όλο το χρονικό διάστημα από τη στιγμή που ο χρήστης κάνει login μέχρι τη στιγμή που θα βγει από το σύστημα. Αν το session ID αποθηκεύεται σε κάποιο cookie, ο επιτιθέμενος μπορεί να γράψει κάποιο script που θα τρέξει στον browser του χρήστη και θα ζητάει την τιμή του cookie και θα το στέλνει στον επιτιθέμενο. Ο επιτιθέμενος τότε μπορεί να χρησιμοποιήσει το έγκυρο session ID για να πλοηγηθεί στον δικτυακό τόπο, χωρίς όμως να κάνει login. Το script θα

μπορούσε επίσης να συλλέξει άλλες πληροφορίες από τη σελίδα, συμπεριλαμβανομένων των περιεχομένων αυτής.

Εκτός από τα links ψεύτικων σελίδων που αποστέλλονται με e-mail, υπάρχουν άλλοι τρόποι επίθεσης με XSS;

Ναι. Υπάρχουν και άλλες μέθοδοι. Ας πάρουμε το παράδειγμα μιας εφαρμογής ενός πίνακα ανακοινώσεων, όπου τα δεδομένα που εισάγει ένας χρήστης μπορούν να προσπελαστούν από άλλους χρήστες. Ο επιτιθέμενος εισάγει ένα script στη σελίδα αυτή. Όταν ένας έγκυρος χρήστης προσπαθήσει να δει τη σελίδα, το script εκτελείται στον browser του. Έτσι στέλνει τις πληροφορίες του χρήστη στον επιτιθέμενο.

Πως μπορώ να προστατευτώ από το XSS;

Η προστασία από το XSS μπορεί να προστεθεί κατά την υλοποίηση της εφαρμογής. Θα πρέπει να διαπιστώνουμε την εγκυρότητα όλων των εισόδων και των εξόδων προς και από την εφαρμογή και να αποφύγουμε όλους τους ειδικούς χαρακτήρες που μπορεί να χρησιμοποιηθούν σε κάποιο script. Αν ο κώδικας αντικαθιστά τους ειδικούς χαρακτήρες από τους ακόλουθους, πριν προβάλλει την έξοδο, το XSS μπορεί να αποκλειστεί για κάποιες περιπτώσεις.

αντί	<	βάζουμε	&lt;
αντί	>	βάζουμε	&gt;
αντί	(	βάζουμε	&#40;
αντί	)	βάζουμε	&#41;
αντί	#	βάζουμε	&#35;
αντί	&	βάζουμε	&#38;

Ο Gunter Ollmann έχει γράψει ένα εξαιρετικό δοκίμιο σχετικά με τη χρήση ειδικών χαρακτήρων στις επιθέσεις μέσω XSS. Για παράδειγμα, η παραπάνω μέθοδος προστασίας από ειδικούς χαρακτήρες δεν μπορεί να μας διαφυλάξει από ένα script που έχει τη μορφή
«javascript:self.location.href='http://www.evil.org'»
καθώς το script αυτό δεν χρησιμοποιεί κανέναν ειδικό χαρακτήρα.

Μπορούν οι XSS επιθέσεις να αποφευχθούν χωρίς να μεταβάλλουμε τον πηγαίο κώδικα;

Υπάρχει μια μέθοδος που απαιτεί μικρές αλλαγές στον κώδικα σε σύγκριση με τη δοσμένη είσοδο και έχει να κάνει με την εξακρίβωση της εξόδου, με στόχο να αποτρέψουμε την υποκλοπή των cookies μέσω XSS. Ο Internet Explorer 6 έχει ένα χαρακτηριστικό που ονομάζεται HTTP Only που αφορά τα cookies. Κάνοντας χρήση αυτού, σιγουρευόμαστε ότι το cookie δεν μπορεί να προσπελαστεί από κάποιο script. Περισσότερες λεπτομέρειες είναι διαθέσιμες στο site της MSDN στο άρθρο httpcookies στο
http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dhtml/httponly_cookies.asp
Ο Mozilla έχει επίσης τη δυνατότητα να προσαρμόσει στο μέλλον παρόμοια χαρακτηριστικά Ερευνητές έχουν βρει τρόπο να νικήσουν κάτι τέτοιο. Η μέθοδος είναι γνωστή ως Cross Site Tracing.

Τι είναι το Cross Site Tracing (XST); Πως μπορώ να προστατευτώ από αυτό;

Οι επιτιθέμενοι μπορούν να προσπεράσουν τα χαρακτηριστικά του HTTP για να υποκλέψουν πληροφορίες κάνοντας χρήση της μεθόδου Cross Site tracing (XST). Η TRACE είναι μια HTTP μέθοδος που μπορεί να σταλεί στον server. Ο server στέλνει πίσω στον browser οτιδήποτε περιέχεται στο TRACE αίτημα. Σε κάποιο site που χρησιμοποιεί cookies, οι πληροφορίες των cookies στέλνονται προς τον server σε κάθε αίτημα. Αν στείλουμε ένα TRACE αίτημα σε ένα URL ενός τέτοιου site, ο server θα στείλει πίσω όλες τις πληροφορίες των cookies στον browser. Σκεφθείτε τώρα μια περίπτωση παρόμοια με αυτήν που περιγράψαμε στο XSS αλλά το site στην περίπτωση αυτή χρησιμοποιεί HTTP Only cookies.

Οι επιτιθέμενοι κατευθύνουν έναν έγκυρο χρήστη να κάνει κλικ σε ένα link που περιέχει κάποιο script που καλεί τη μέθοδο TRACE. Όταν ο χρήστης κάνει κλικ στο link αυτό, το αίτημα TRACE, όπως επίσης και όλες οι πληροφορίες των cookies αποστέλλονται στον server. Τότε ο server επιστρέφει τις πληροφορίες των cookies πίσω στο script, στον browser. Υποθέστε ότι το κακόβουλο script περιέχει ακόμη κώδικα για να στείλει τις πληροφορίες αυτές σε επιτιθέμενους. Οι επιτιθέμενοι πέτυχαν και πάλι να υποκλέψουν τα cookies, παρότι χρησιμοποιήθηκαν HTTP Only Cookies. Για να συνοψίσουμε, τα HTTP Only cookies εμποδίζουν τα JavaScript από το να έχουν πρόσβαση απευθείας στα cookies, αλλά οι επιτιθέμενοι έχουν ακόμη τη δυνατότητα να πάρουν τις πληροφορίες αυτές μέσα από μια έμμεση μέθοδο. Το XST μπορεί να αποφευχθεί απενεργοποιώντας τη μέθοδο TRACE στον web server. Το δοκίμιο αυτό από τον Jeremiah Grossman πραγματεύεται το θέμα του XST με περισσότερες λεπτομέρειες στη διεύθυνση: http://www.cgisecurity.com/whitehat-mirror/WhitePaper_screen.pdf

"Δακτυλικά αποτυπώματα" των Web Servers
Πως μπορούν οι επιτιθέμενοι να αναγνωρίσουν ποιον web server χρησιμοποιώ;

Η αναγνώριση της εφαρμογής που τρέχει σε έναν απομακρυσμένο web server είναι γνωστή ως server fingerprinting. Ο πιο απλός τρόπος για να γίνει αυτό, είναι να σταλεί ένα αίτημα στον server και να δούμε το banner που αποστέλλεται στην απάντηση. Τα banners γενικά περιέχουν το όνομα του server και την έκδοσή του. Μπορούμε να χειριστούμε το πρόβλημα αυτό είτε ρυθμίζοντας τον server να μην εμφανίζει καθόλου banners ή αλλάζοντας τα στοιχεία αυτά, κάνοντας τον server να φαίνεται ότι είναι κάποιο άλλο μηχάνημα.

Πως μπορώ να παραχαράξω τα banners ή να ξαναγράψω τις κεφαλίδες του web server μου;

Υπάρχουν κάποια εργαλεία ου βοηθούν να φτιάχνουμε ψεύτικα banners.

- Το URLScan είναι ένα από αυτά, και μπορεί να αλλάξει το banner ενός IIS web server. Μπορούμε να το βρούμε στη διεύθυνση

http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/tools/URLScan.asp

- Το mod_security περιέχει ένα χαρακτηριστικό για να λλάξει την ταυτότητα ενός Apache web server. Μπορούμε να το βρούμε στο

http://www.modsecurity.org/

- Το Servermask είναι ένα εργαλείο για να φτιάχνουμε ψεύτικα banners για IIS servers, και μπορούμε να το βρούμε στο

http://www.servermask.com/

Από τη στιγμή που δημιουργώ ψεύτικα banners, μπορεί ακόμη ο server μου να υποστεί fingerprinting;

Ναι. Δυστυχώς υπάρχουν κάποια εργαλεία που κάνουν fingerprinting σε web server χωρίς να στηρίζονται στα banners. Διαφορετικοί web servers μπορεί να περιλαμβάνουν χαρακτηριστικά που δεν αναφέρονται στα HTTP RFCs ξεχωριστά. Υποθέστε ότι φτιάχνουμε μια βάση δεδομένων αυτών των ειδικών αιτημάτων και αντιστοίχως απαντήσεων κάθε web server. Μπορούμε τώρα να στείλουμε τα αιτήματα αυτά στυον server που θέλουμε να κάνουμε fingerprint και συγκρίνουμε τις απαντήσεις με αυτές που έχουμε στη βάση. Αυτή είναι η τεχνική που χρησιμοποιείται από εργαλεία όπως το Fire & Water. Το εργαλείο αυτό το βρίσκουμε στη διεύθυνση:
http://www.ntobjectives.com/products/firewater/
Υπάρχει ένα δοκίμιο του Saumil Shah που πραγματεύεται το εργαλείο httprint στο σημείο
http://net-square.com/httprint/httprint_paper.html
Το httprint μπορεί να βρεθεί στο σημείο:
http://net-square.com/httprint/

Ένας φίλος μου είπε ότι είναι πιο ασφαλές να τρέχω τον web server σε μια μη – προκαθορισμένη θύρα Είναι αυτό σωστό;

Ένας web server γενικά χρειάζεται να προσπελαύνεται από πολύ κόσμο στο internet. Καθώς, κανονικά τρέχει στην 80 θύρα, και όλοι οι browser έχουν καλιμπραριστεί να παίρνουν πρόσβαση στην 80, πόρτα του web server, έτσι οι χρήστες έχουν τη δυνατότητα να πλοηγηθούν στο εκάστοτε site. Αν αλλάξουμε τη θύρα, οι χρήστες θα πρέπει να ορίσουν, εκτός του domain name, και τον αριθμό της θύρας. Αυτή είναι μια καλή ιδέα για μια εφαρμογή σε κάποιο intranet όπου όλοι οι χρήστες ξέρουν που να συνδεθούν. Αυτό είναι πιο ασφαλές καθώς ο web server δεν γίνεται στόχος από αυτόματα εργαλεία όπως worms που ανιχνεύουν την 80 θύρα και άλλες default θύρες.

Θα έπρεπε πραγματικά να ανησυχήσω ότι ο web server μπορεί να δεχθεί επίθεση με fingerprinting;

Υπάρχουν δυο τάσεις για το θέμα αυτό. Σύμφωνα με την πρώτη σκέψη, θα πρέπει να πάρετε μέτρα ασφαλείας ενάντια στην περίπτωση επίθεση μέσω fingerprinting καθώς η αναγνώριση του web server μπορεί να είναι το πρώτο βήμα για μια πιο επικίνδυνη επόμενη επίθεση. Από τη στιγμή που οι επιτιθέμενοι έχουν ανακαλύψει ότι ο web server είναι ένας IIS 5 για παράδειγμα , θα αναζητήσουν γνωστές αδυναμίες και αχίλλειες πτέρνες για τον IIS 5. Αν ο web server δεν έχει προσαρμοστεί με τα νέα updates για όλους τους κινδύνους και τις ελλείψει ασφαλείας, ή οι επιτιθέμενοι βρουν κάποιο από αυτά, για τα οποία δεν έχει βγει κάποιο patch που να το διορθώνει, δεν υπάρχει κάτι που να μπορεί να τους σταματήσει πριν επιτεθούν. Ακόμη, αυτόματα εργαλεία και worms μπορεί να ξεγελαστούν, αλλάζοντας τις πληροφορίες έκδοσης. Κάποιοι αποφασισμένοι και συγκεντρωμένοι στο σκοπό του επιτιθέμενοι, μπορεί να ελέγξουν άλλα στοιχεία για να συμπεράνουν τα στοιχεία του server αλλά τα κολλήματα που οι επιτιθέμενοι πρέπει αν ξεπεράσουν είναι αυξημένα όταν είναι δυσκολότερο να βρουν τα στοιχεία του web server όπως το όνομα και η έκδοση.

Ο Jeremiah Grossman υποστηρίζει μια δεύτερη τάση σχετικά με το θέμα αυτό. Προληπτικά μέτρα σαν αυτά, θεωρούνται άχρηστα καθώς οποιοδήποτε αυτόματο εργαλείο που σκανάρει δικτυακούς τόπους εφόσον βάλει στόχο έναν web server, δεν θα το ενδιαφέρει το όνομα και η έκδοσή του. Το αυτόματο εργαλείο θα τρέξει ΟΛΑ τα test ανεξαρτήτως αν έχουν επίδραση στο σύστημα ή όχι. Αυτή είναι μια τυπική προσέγγιση στο θέμα. Κάποιος κακός που βάζει στόχο το site, μπορεί να εμποδιστεί να δεν ξέρει την ακριβή έκδοση αλλά αν είναι αποφασισμένος, θα εξακολουθήσει να δοκιμάζει όλους τους πιθανούς τρόπους για να μπορέσει να σπάσει την ασφάλεια του συστήματος.

Κάνοντας δοκιμές
Θέλω να συνδέσω ένα proxy εργαλείο με τον proxy server μου. Υπάρχουν αυτόματα Utilities που μου επιτρέπουν να κάνω κάτι τέτοιο;

Ναι, υπάρχουν διάφορα εργαλεία που επιτρέπουν τη σύνδεση των proxy. Κάποια από αυτά είναι:

- WebScarab στο

http://www.owasp.org/development/webscarab

- Exodus στο

http://home.intekom.com/rdawes/exodus.html

- Odysseus στο

http://www.wastelands.gen.nz/odysseus/index.php

Δεν μπορεί η δοκιμή δικτυακών εφαρμογών να γίνει αυτόματα; Υπάρχουν ανάλογα εργαλεία για τη δουλειά αυτή;

Υπάρχουν εργαλεία που ελέγχουν εφαρμογές για διαρροές ασφαλείας. Αλλά τα εργαλεία αυτά μπορούν να αναζητήσουν μόνο περιορισμένο αριθμό αδυναμιών και δεν εντοπίζουν όλα τα πιθανά προβλήματα της εφαρμογής. Ακόμη, πολλές επιθέσεις απαιτούν να αντιλαμβάνεται ο επιτιθέμενος τη δομή της εφαρμογής για να αποφασίσει σχετικά με με μεταβλητές που μπορεί να χρησιμοποιήσει σε συγκεκριμένο ερώτημα. Κάτι τέτοιο δεν μπορεί να γίνει από ένα εργαλείο. Μια παρουσίαση του Jeremiah Grossman από το White Hat Security που αναφέρεται στους περιορισμούς του αυτόματου ελέγχου μπορούμε να βρούμε στο
http://www.blackhat.com/presentations/bh-federal-03/bh-fed-03-grossman-up.pdf
Στους ελέγχους μας, κατά τη χρήση ενός WebGoat, το καλύτερο Black-box εργαλείο βρήκε λιγότερα από το 20% των προβλημάτων Μερικά εργαλεία για αυτόματους ελέγχους είναι:
SpikeProxy, με ελεύθερο κώδικα και δωρεάν διάθεση στο
http://www.immunitysec.com/spikeproxy.html
WebInspect, που μπορούμε να βρούμε στο
http://www.spidynamics.com/productline/WE_over.html

Πως μπορώ να ελέγξω τις ικανότητές μου στη δοκιμή των εφαρμογών; Υπάρχει μια sample εφαρμογή που μπορώ να πειραματιστώ;

Το OWASP παρέχει μια sample εφαρμογή που μπορεί να χρησιμοποιηθεί για τον σκοπό αυτό. Οι στόχοι του WebGoat project είναι να διδάξει σχετικά με τη δικτυακή ασφάλεια σε ένα interactive περιβάλλον εκπαίδευσης. Υπάρχουν μαθήματα για τα περισσότερα

γνωστά προβλήματα ασφαλείας. Ακόμη ένα ενδιαφέρον site είναι το Hackingzone που έχει ένα παιχνίδι στην SQL Injection στο
http://www.hackingzone.org/sql/index.php

Υπάρχουν εργαλεία ελέγχου του πηγαίο κώδικα για .NET γλώσσες, Java, PHP και άλλες που να προβλέπουν προβλήματα ασφαλείας σε αυτόν;

- Το Rough Auditing Tool for Security (RATS) είναι ένα εργαλείο που ελέγχει τον πηγαίο κώδικα για προβλήματα ασφαλείας για C, C++, Python, Perl και PHP προγράμματα. Μπορούμε να το βρούμε στο
  http://www.securesoftware.com/download_rats.htm
- Το FX Cop δημιουργήθηκε από τη Microsoft στην κοινότητα GotDotNet για να ελέγχει .NET Framework guidelines που περιλαμβάνουν και την ασφάλεια.
- Το Prexis είναι ένα εμπορικό πρόγραμμα που αναλύει τον πηγαίο κώδικα και το τρέξιμο μιας εφαρμογής.
- Το Flawfinder είναι ένας στατικός αναλυτής πηγαίου κώδικα.
- Το Compaq ESC είναι ένα πρόγραμμα που ελέγχει το τρέξιμο Java προγραμμάτων.
- Το Parasoft AEP είναι ένα εμπορικό πρόγραμμα που αναλύει πηγαίο κώδικα για Java.

Μπορούν ακόμη και τα μη HTTP πρωτόκολλα να διαχειρισθούν με τον τρόπο αυτό;

Ναι. Το Interactive TCP Replay είναι ένα εργαλείο που δρα σαν proxy for για μη HTTP εφαρμογές και επιτρέπει επιπλέον τη μεταβολή της κίνησης. Επιτρέπει την επεξεργασία των μηνυμάτων σε έναν hex editor. Το ITR επίσης κρατάει αρχείο καταγραφής όλων των μηνυμάτων που περνούν ανάμεσα στον client και τον server. Μπορεί να χρησιμοποιήσει διαφορετικούς τύπους κωδικοποίησης χαρακτήρων όπως ASCII ή EBCDIC για επεξεργασία και καταγραφή. Περισσότερες πληροφορίες σχετικά με αυτό μπορούμε να βρούμε στο
http://www.webcohort.com/web_application_security/research/tools.html

Κρυπτογραφία/SSL
Τι είναι το SSL;

Το Secure Socket Layer (SSL) μας ασφαλίζει για δύο πράγματα: Πρώτον, όταν ένας client συνδέεται σε έναν web server, ο client μπορεί να είναι σίγουρος ότι επικοινωνεί με τον σωστό server ελέγχοντας το πιστοποιητικό που ο server του στέλνει. Δεύτερον, το SSL μας διαβεβαιώνει για την εμπιστευτικότητα των δεδομένων καθώς ο client και ο server ανταλλάσσουν κωδικοποιημένα μηνύματα που δεν μπορούν να γίνουν αντιληπτά από κανέναν άλλον. Το SSL λειτουργεί ως εξής: Όταν ο client ζητάει μια SSL σελίδα, ο server στέλνει ένα πιστοποιητικό που έχει προμηθευτεί από μια αρχή έκδοσης πιστοποιητικών που εμπιστευόμαστε. Το πιστοποιητικό αυτό περιέχει το δημόσιο κλειδί του server. Αφού πειστεί ότι το πιστοποιητικό είναι σωστό και ο server είναι ο αυθεντικός, ο client δημιουργεί έναν τυχαίο αριθμό, το session key. Το κλειδί αυτό κωδικοποιείται από το δημόσιο κλειδί του server και αποστέλλεται πίσω. Ο server αποκωδικοποιεί το μήνυμα με το ιδιωτικό του κλειδί Τώρα, και οι δύο πλευρές έχουν ένα session key που είναι γνωστό μόνο σε αυτές. Όλη η επικοινωνία από και προς κωδικοποιείται και αποκωδικοποιείται με το session key. Ένα ενδιαφέρον link σχετικά με το SSL είναι το

http://www.rsasecurity.com/standards/ssl/basics.html

Πρέπει να χρησιμοποιήσω 40-bit ή 128-bit SSL;

Υπάρχουν δύο αρχιτεκτονικές στο SSL – το 40-bit και το 128-bit. Αυτές αναφέρονται στο μήκος του μυστικού κλειδιού που χρησιμοποιείται για την κωδικοποίηση της σύνδεσης. Το κλειδί αυτό δημιουργείται για κάθε ασφαλή SSL σύνδεση και χρησιμοποιείται για το σύνολο αυτής. Όσο μεγαλύτερο είναι αυτό, τόσο πιο δύσκολο είναι να σπάσουν τα κωδικοποιημένα δεδομένα. Συμπεραίνουμε λοιπόν ότι η 128-bit κωδικοποίηση είναι περισσότερο ασφαλής από αυτή των 40-bit. Οι περισσότεροι browsers σήμερα υποστηρίζουν 128-bit κωδικοποίηση. Υπάρχουν κάποιες χώρες που έχουν browsers που χρησιμοποιούν μόνο 40-bit κωδικοποίηση. Στην περίπτωση που χρησιμοποιείτε ένα 40-bit SSL, ίσως χρειαστεί να πάρετε παραπάνω μέτρα για να προστατεύσετε τα ευαίσθητα δεδομένα σας. Το Salted hash για τη μετάδοση συνθηματικών είναι μια καλή μέθοδος. Αυτό μας διαβεβαιώνει ότι το συνθηματικό δεν μπορεί να υποκλαπεί ακόμη και αν το SSL κλειδί σπάσει.

Είναι η 40-bit SSL κωδικοποίηση, μη ασφαλής;

Το 40-bit SSL δεν είναι ακριβώς μη ασφαλές. Απλά είναι υπολογίσιμα εφικτό να σπάσει το κλειδί των 40-bit όχι όμως το κλειδί των 128-bit. Ακόμη και αν το 40-bit κλειδί σπάσει, χρειάζεται ένα υπολογίσιμα μεγάλο αριθμό υπολογιστών για να γίνει κάτι τέτοιο. Κανεί δεν θα τολμούσε να κάνει κάτι τέτοιο για μια πιστωτική κάρτα ή κάτι παρόμοιο. Αλλά υπάρχουν τρόπο να σπάσει το 40-bit RC4 κλειδί σε μερικές ώρες. Ανάλογα λοιπόν με τα δεδομένα που η εφαρμογή μας διαχειρίζεται, μπορούμε να αποφασίσουμε για τις δυνατότητες του SSL που θα χρησιμοποιήσουμε. Κάνοντας χρήση των 128-bit παρέχουμε σίγουρα μεγαλύτερη ασφάλεια.

Τι είναι αυτό που κωδικοποιείται όταν χρησιμοποιώ SSL; Κωδικοποιείται και η αίτηση σελίδας;

Μετά την αρχική SSL επικοινωνία και αφού η σύνδεση επιτευχθεί σε HTTPS, όλα κωδικοποιούνται, συμπεριλαμβανομένων και των αιτήσεων για σελίδες. Συνεπώς, ότι δεδομένα στέλνονται σε μια συμβολοσειρά ερώτησης, θα κωδικοποιηθούν επίσης.

Ποιος αλγορίθμους κρυπτογράφησης χρησιμοποιεί το SSL;

Το SSL υποστηρίζει έναν αριθμό αλγορίθμων κρυπτογράφησης. Κατά τη διάρκεια της αρχικής "handshaking" φάσης, χρησιμοποιεί τον RSA αλγόριθμο δημοσίου κλειδιού. Για να κωδικοποιήσει τα δεδομένα με το session key χρησιμοποιούνται οι ακόλουθοι αλγόριθμοι: RC2, RC4, IDEA, DES, τριπλός DES και MD5.

Θέλω να χρησιμοποιήσω SSL. Πως ξεκινάω;

Υπάρχουν διάφορες αρχές έκδοσης πιστοποιητικών από όπου μπορούμε να αγοράσουμε ένα πιστοποιητικό SSL. Όποια από αυτές και αν επιλέξουμε, η βασική διαδικασία θα έχει ως εξής:

- Δημιουργούμε ένα ζεύγος κλειδιών για τον server

- Δημιουργούμε το Certificate Signing Request. Αυτό θα μας ζητάει να του παρέχουμε συγκεκριμένες λεπτομέρειες όπως τοποθεσία, και πλήρες domain name για τον server.
- Υποβάλλουμε το CSR *(Certificate Signing Request)* στον CA *(αρχή έκδοσης πιστοποιητικών)* μαζί με την έγγραφη απόδειξη της ταυτότητας.
- Εγκαθιστούμε το πιστοποιητικό που μας έστειλε η CA

Τα δύο πρώτα βήματα γίνονται από το web server. Όλοι οι servers έχουν τα χαρακτηριστικά αυτά. Κατά την εγκατάσταση του πιστοποιητικού που εκδίδεται από τη CA, θα πρέπει να ορίσουμε ποιες σελίδες πρέπει να κωδικοποιηθούν με SSL.

Cookies και Διαχείριση Ενοτήτων
Υπάρχουν κίνδυνοι κατά τη χρήση persistent αντί non-persistent cookies;

Τα Persistent cookies είναι δεδομένα που τοποθετεί ένας δικτυακός τόπος στον σκληρό δίσκο του χρήστη για να διατηρεί πληροφορίες για παραπάνω από μια συνδέσεις του browser. Τα δεδομένα αυτά θα παραμείνουν στο σύστημα του χρήστη και μπορεί να διαβαστούν από τον δικτυακό τόπο την επόμενη φορά που ο χρήστης πλοηγηθεί σε αυτό.

Τα Non-persistent cookies από την άλλη μεριά, είναι αυτά που χρησιμοποιούνται μόνο για την σύνδεση του browser που τα δημιουργεί. Παραμένουν μόνο στη μνήμη του μηχανήματος και δεν αποθηκεύονται στον σκληρό δίσκο. Ο κίνδυνος ασφαλείας με τα persistent cookies είναι ότι αποθηκεύονται γενικά σε ένα text αρχείο στον client και ο επιτιθέμενος με πρόσβαση στο μηχάνημα το θύματος μπορεί να υποκλέψει την πληροφορία αυτή.

Μπορεί ένα άλλο web site να κλέψει τα cookies που το δικό μου site τοποθετεί στο μηχάνημα ενός χρήστη;

Όχι. Δεν είναι δυνατόν κάποιο website να προσπελάσει τα cookies ενός άλλου. Τα Cookies έχουν ένα χαρακτηριστικό domain που συνδέεται με αυτά. Μόνο μια αίτηση που προέρχεται από το συγκεκριμένο domain μπορεί να πάρει πρόσβαση στο cookie. Το χαρακτηριστικό αυτό μπορεί να έχει μόνο μια τιμή.

Ποιος είναι ο καλύτερος τρόπος να μεταβιβάσουμε session ids σε cookies, σε URL ή σε κρυμμένες μεταβλητές;

Η μετάδοση των session IDs μέσα στο URL μπορεί να οδηγήσει σε διάφορους κινδύνους. Περαστικοί από τον χώρο εργασίας του χρήστη, μπορούν να δουν το session ID. Αν το URL αποθηκεύεται στην cache στο σύστημα του client, τότε το session ID θα αποθηκευτεί και αυτό. Το session ID θα αποθηκευτεί στα logs άλλων sites. Οι κρυμμένες μεταβλητές δεν είναι πάντα πρακτικές καθώς κάθε αίτηση μπορεί να μην είναι POST. Τα Cookies είναι ο ασφαλέστερος τρόπος καθώς δεν cachάρονται, δεν είναι ορατά στο W3C ή στα logs, και οι περισσότεροι χρήστες ούτως ή άλλως αποδέχονται την εγκατάστασή τους.

Τι είναι αυτά τα "ασφαλή cookies";

Ένα cookie μπορεί να χαρακτηρισθεί ως "ασφαλές" πράγμα που εξασφαλίζει ότι το cookie αυτό χρησιμοποιείται μόνο σε SSL συνδέσεις. Αν το γεγονός ότι ένα cookie είναι

ασφαλές δεν διευκρινίζεται, τότε αποστέλλεται ακωδικοποίητο σε μη ασφαλές κανάλι. Ευαίσθητα cookies σαν αυτά των συνδέσεων θα πρέπει να αναφέρονται ως ασφαλή αν όλες οι σελίδες στο web site που απαιτούν ασφαλή σύνδεση, χρησιμοποιούν SSL. Κάτι που πρέπει να θυμόμαστε είναι ότι οι εικόνες, γενικά δεν κατεβαίνουν με SSL πρωτόκολλο και συνήθως δεν απαιτούν ένα session token για να εμφανιστούν. Θέτοντας το session cookie ως ασφαλές, διασφαλίζουμε ότι ο browser δεν στέλνει το cookie καθώς κατεβάζει το γραφικό σε μη-SSL σύνδεση.

Αν χρησιμοποιώ ένα session ID που είναι συνάρτηση του IP του client, εμποδίζω έτσι την υποκλοπή της σύνδεσης;

Ο επιτιθέμενος μπορεί να υποκλέψει τη σύνδεση ενός άλλου χρήστη κλέβοντας το session token. Διάφοροι μέθοδοι έχουν προταθεί για να εμποδίσουν την υποκλοπή της σύνδεσης ακόμη και αν το session token κλαπεί. Για παράδειγμα, κάνοντας χρήση ενός session token που αποτελεί συνάρτηση της IP του χρήστη. Στην περίπτωση αυτή, ακόμη και αν ο επιτιθέμενος έκλεβε το token, θα χρειαζόταν την ίδια IP με αυτή του χρήστη για να υποκλέψει επιτυχώς τη σύνδεση. Παρόλα αυτά, η υποκλοπή της σύνδεσης μπορεί και πάλι να επιτευχθεί. Υποθέστε ότι ο επιτιθέμενος βρίσκεται στο ίδιο LAN με τον χρήστη και χρησιμοποιεί τον ίδιο Proxy, με την ίδια προφανώς IP, για να βγει προς τα έξω και να πάρει πρόσβαση στο web site. Ο επιτιθέμενος μπορεί ακόμα να υποκλέψει τη σύνδεση αν μπορεί να ανακαλύψει το session token. Θα μπορούσε να μην είναι εφικτό να εμποδιστεί κάτι τέτοιο αν η IP του client αλλάξει κατά τη διάρκεια της σύνδεσης, καθιστώντας τη σύνδεση λαθεμένη αν το token είναι συνδεδεμένο με την αρχική IP. Αυτό μπορεί να συμβεί αν ο client βγαίνει μέσα από μια ομάδα από proxy servers.

Τι θα λέγατε αν κωδικοποιούσαμε τα session id cookies αντί να χρησιμοποιήσουμε SSL;

Κωδικοποιώντας απλά το session ID σε μια μη-SSL σύνδεση δεν εξυπηρετούμε κάποιο σκοπό. Καθώς το session ID θα κωδικοποιηθεί μια φορά και η ίδια τιμή θα σταλεί πίσω και πάλι μπροστά κάθε φορά, ο επιτιθέμενος μπορεί να χρησιμοποιήσει την κωδικοπιημένη τιμή για να υποκλέψει τη σύνδεση

Ποιο είναι το σενάριο της χρήσης μιας page id, επιπλέον του session id;

Το Session ID ή token έχει ένα χρόνο ζωής μιας σύνδεσης και συνδέεται άμεσα με τον χρήστη που έχει κάνει log in. Μια page ID ή token έχει χρόνο ζωής μιας σελίδας και συνδέεται άμεσα με τη σελίδα που εξυπηρετεί. Είναι ένα μοναδικό token που δίνεται όταν μια σελίδα κατεβαίνει και παρουσιάζεται από τον χρήστη κατά την επίσκεψη στην επόμενη σελίδα. Ο server περιμένει μια συγκεκριμένη τιμή από τον χρήστη για να πάρει πρόσβαση στην επόμενη σελίδα. Μόνο αν το token που υποβάλλεται συμπίπτει με αυτό που ο server περιμένει, τότε δίνεται πρόσβαση στην επόμενη σελίδα. Μια εφαρμογή μπορεί να χρησιμοποιήσει τη δυνατότητα αυτή για να βεβαιώσει ότι ένας χρήστης έχει πρόσβαση σε σελίδες μόνο με τη σειρά που η εφαρμογή ορίζει. Ο χρήστης δεν μπορεί να επικολλήσει ένα άλλο URL στον browser και να υπερπηδήσει σελίδες έτσι απλά επειδή έχει μια ενεργή σύνδεση, καθώς το token της σελίδας δεν θα αναγνωριστεί για να πάρει πρόσβαση σε απώτερο URL απευθείας.

Ίχνη από Logging και Audit

Τι είναι αυτά τα W3C logs;

Το W3C είναι μια μορφοποίηση αρχείου καταγραφής που χρησιμοποιείται για Web server. Τα W3C logs καταγράφουν λεπτομέρειες προσβάσεων κάθε αιτήματος: την ώρα που αυτό έγινε, την IP της πηγής, τη σελίδα που αιτήθηκε, τη μέθοδο που χρησιμοποιήθηκε, την έκδοση του http πρωτοκόλλου, τον τύπο του browser, τη σχετική σελίδα, τον κωδικό απάντησης κλπ. Σημειώστε ότι αυτά αποτελούν logs προσβάσεων, και μια ξεχωριστή εγγραφή κρατείται για κάθε αίτημα. Όταν μια σελίδα με πολλαπλά gif αρχεία κατεβαίνει, θα καταγραφεί ως πολλαπλές εγγραφές στο W3C log. Έτσι τα W3C logs τείνουν να είναι voluminous.

Χρειάζομαι να έχω αρχεία καταγραφής στην εφαρμογή μου εάν χρησιμοποιώ W3C logs;

Ναι, είναι σημαντικό η εφαρμογή να κρατάει αρχεία καταγραφής σε "επίπεδο εφαρμογής" ακόμη και όταν χρησιμοποιείται W3C logging. Καθώς τα W3C logs περιέχουν εγγραφές για κάθε http αίτημα, είναι δύσκολο, και σε κάποιες περιπτώσει αδύνατο, να εξάγουμε συμπεράσματα από αυτά τα αρχεία καταγραφής. Για παράδειγμα, τα W3C logs είναι ιδιαιτέρως βαριά και πολύπλοκα για να αναγνωρίσουν μια συγκεκριμένη σύνδεση ενός χρήστη και τις ενέργειες που αυτός έκανε. Είναι καλύτερο η εφαρμογή να κρατάει αρχείο των σημαντικών ενεργειών από το να το αποκωδικοποιεί από τα W3C logs.

Τι θα πρέπει αν καταγράφω από την εφαρμογή μου;

Κρατήστε ένα αρχείο ενεργειών που θα μπορούσατε να θελήσετε κάποια στιγμή να αξιολογήσετε κατά την επίλυση προβλημάτων ή την εξαγωγή συμπερασμάτων. Σημειώστε ότι δεν προτείνεται να κρατάτε ευαίσθητες πληροφορίες σε αυτά τα αρχεία καταγραφής, καθώς οι administrators έχουν πρόσβαση σε αυτά για να λύνουν πιθανά προβλήματα. Ενέργειες που συχνά κρατώνται σε αρχείο είναι:

- Login και logout των χρηστών
- Κρίσιμες συναλλαγές (π.χ. μεταφορά ποσών μεταξύ λογαριασμών)
- Αποτυχημένες προσπάθειες πρόσβασης
- Κλειδώματα λογαριασμών
- Παραβίαση της πολιτικής

Τα δεδομένα που κρατώνται για κάθε από αυτές τις ενέργειες, συνήθως περιλαμβάνουν:

- User ID
- ώρα
- IP
- Κωδικούς σφαλμάτων
- Προτεραιότητα

Πρέπει να κωδικοποιώ τα αρχεία καταγραφής;

Η κωδικοποίηση χρειάζεται όταν η πληροφορία που αυτά περιέχουν πρέπει να προστατευτεί για να μη διαβαστεί από μη εξουσιοδοτημένους χρήστες. Ναι, η κωδικοποίηση υποβαθμίζει την απόδοση του συστήματος. Οπότε αν τα logs δεν περιέχουν ευαίσθητες πληροφορίες καλό είναι να αποφύγουμε τη διαδικασία αυτή. Παρόλα αυτά, προτείνουμε να προστατεύετε τα logs από την κοινή θέα, κάνοντας χρήση ψηφιακών υπογραφών. Οι ψηφιακές υπογραφές κάνουν λιγότερη χρήση των πόρων του

επεξεργαστή σε σχέση με την κωδικοποίηση και μας εξασφαλίζουν από τη μη εξουσιοδοτημένη πρόσβαση σε αυτά.

Μπορώ να εμπιστευτώ την IP ενός χρήστη που βλέπω στα logs; Μπορεί ένας χρήστης να χρησιμοποιεί ψεύτικη IP;

Ένας κακόβουλος χρήστης που θέλει να κρύψει την αληθινή του IP μπορεί να χρησιμοποιήσει κάποια υπηρεσία όπως το anonymizer, ή να κάνει χρήση open HTTP relays. [Τα HTTP open relays είναι λαθεμένα ρυθμισμένοι web servers στο δίκτυο που χρησιμοποιούνται σαν HTTP proxy για να συνδεθούν σε άλλα sites.] Σε τέτοιες περιπτώσεις, η IP που βλέπετε στα log files θα είναι αυτή που οι υπηρεσίες αυτές χρησιμοποιούν. Συνεπώς η IP που βλέπετε στα log files δεν είναι σίγουρα η σωστή και δεν μπορούμε να την εμπιστευτούμε.

Miscellaneous
Τι είναι οι εφαρμογές – firewalls; Πόσο καλές είναι;

Τα firewalls – εφαρμογές αναλύουν τις αιτήσεις σε επίπεδο εφαρμογής. Αυτά τα firewall χρησιμοποιούνται για συγκεκριμένες εφαρμογές όπως είναι ο web server ή ένας database server. Τα firewalls για δικτυακές εφαρμογές προστατεύουν τον web server από HTTP επιθέσεις. Ελέγχουν τα αιτήματα που μπορεί να περιλαμβάνουν SQL Injection, XSS, URL encoding κλπ. Αλλά δεν μπορούν να προστατεύσουν από επιθέσεις που απαιτούν την αντίληψη του περιεχομένου. Η περίπτωση αυτή περιλαμβάνει τις περισσότερες επιθέσεις που βας΄΄ίζονται σε υπολογισμό μεταβλητών. Παρόλα αυτά, τα firewalls σε πείπεδο εφαρμογής δεν μπορούν να προστατεύσουν από επιθέσεις που απαιτούν να γίνει αντιληπτή η φιλοσοφία της εφαρμογής για να προστατευτούμε από αυτές. Κάποιες εφαρμογές λογισμικού firewall είναι:
Netcontinuum's NC-1000 Kavado Inc.'s InterDo Teros Inc.'s Teros-100 APS

Τι είναι όλα αυτά σχετικά με τα "referrer logs", και τα sensitive URLs;

Ο HTTP header περιέχει ένα πεδίο γνωστό ως Referrer. Για να επισκεφτούμε μια σελίδα μπορούμε είτε:

- Να πληκτρολογήσουμε το URL κατευθείαν στη γραμμή της διεύθυνσης του browser
- Να κάνουμε κλικ σε κάποιο link σε μια άλλη σελίδα που θα μας στείλει εκεί
- Να μεταφερθούμε αυτόματα από μια άλλη σελίδα.

Στην πρώτη περίπτωση, το referrer field θα είναι άδειο αλλά στις άλλες δύο περιπτώσεις θα περιέχει το URL της προηγούμενης σελίδας. Το URL της πρώτης σελίδας θα αποθηκευτεί στο αρχείο καταγραφής επισκέψεων του web server όταν ο χρήστης φτάσει στη $2^{η}$ από την $1^{η}$ σελίδα. Υποθέστε τώρα ότι οι δύο σελίδες ανήκουν σε διαφορετικά sites και το πρώτο URL περιέχει ευαίσθητες πληροφορίες όπως το session ID ενός χρήστη. Αν η $2^{η}$ σελίδα ανήκει σε επιτιθέμενους, ,πορούν να πάρουν την πληροφορία αυτή απλά ψάχνοντας μέσα στα logs. Πληροφορίες στα URLs θα αποθηκεύονται στα referrer logs όπως και το ιστορικό του browser. Ωστόσο, θα πρέπει να είμαστε προσεκτικοί να μην συμπεριλαμβάνουμε ευαίσθητες πληροφορίες στα URL.

Θέλω να χρησιμοποιήσω την πιο ασφαλή γλώσσα. Ποια γλώσσα προτείνετε;

Οποιαδήποτε γλώσσα μπορεί να χρησιμοποιηθεί καθώς οι ασφαλείς πρακτικές προγραμματισμού είναι αυτές που κάνουν τις εφαρμογές πιο ασφαλείς. Οι περισσότερες τεχνικές ασφαλείας μπορούν να εφαρμοστούν σε οποιαδήποτε γλώσσα. Η συμβουλή μας θα είναι να χρησιμοποιήσετε οποιαδήποτε γλώσσα με την οποία νιώθετε άνετοι. Αλλά κάποιες γλώσσες όπως η Java έχουν επιπλέον χαρακτηριστικά σαν συνδετικές μεταβλητές που αυξάνουν την ασφάλεια. Μπορείτε να χρησιμοποιήσετε αυτά τα επιπλέον χαρακτηριστικά αν αποφασίσετε να προγραμματίσετε στη γλώσσα αυτή.

Ποια είναι καλά βιβλία για να μάθουμε πρακτικές ασφαλούς προγραμματισμού;

- Το OWASP Guide to Building Secure Web Application and Web Services είναι ένας καλός οδηγός για να αναπτύξετε δικτυακές εφαρμογές. Μπορείτε να το κατεβάσετε από το

http://www.owasp.org/documentation/guide

- Το Writing Secure Code του Michael Howard και David LeBlanc έχει ένα κεφάλαιο πάνω στην ασφάλεια Web-Based υπηρεσιών. Περισσότερες πληροφορίες σχετικά με το βιβλίο αυτό μπορείτε να βρείτε στο:

http://www.microsoft.com/mspress/books/toc/5612.asp

- Το Secure Programming for Linux and Unix HOWTO του David Wheeler αναφέρεται στο πως μπορείτε να γράψετε ασφαλείς εφαρμογές συμπεριλαμβανομένων δικτυακών εφαρμογών. Περιέχει επίσης οδηγίες για μια σειρά από γλώσσες. Το βιβλίο μπορείτε να το βρείτε στο:

http://www.dwheeler.com/secure-programs

Υπάρχουν προγράμματα εκπαίδευσης πάνω στον ασφαλή προγραμματισμό που μπορώ να παρακολουθήσω;

Η Microsoft προσφέρει προγράμματα εκπαίδευσης πάνω στην ανάπτυξη ασφαλών δικτυακών εφαρμογών και Developing and Deploying Secure Microsoft .NET Framework Application. Περισσότερες πληροφορίες μπορείτε να βρείτε στο
http://www.microsoft.com/traincert/syllabi/2300AFinal.asp
και
http://www.microsoft.com/traincert/syllabi/2350BFinal.asp
Το Foundstone προσφέρει εκπαίδευση πάνω στο γράψιμο ασφαλούς κώδικα μέσα από το Global Knowledge
Το Aspect Security προσφέρει παρόμοια μαθήματα.