גירסה 1.00 – 30.12.2003

בעיות אבטחה נפוצות ב-ASP

ניר אדר

מסמך זה הורד מהאתר http://underwar.livedns.co.il.
אין להפיץ מסמך זה במדיה כלשהי, ללא אישור מפורש מאת המחבר.
מחבר המסמך איננו אחראי לכל נזק, ישיר או עקיף, שיגרם עקב השימוש במידע המופיע במסמך, וכן לנכונות התוכן של הנושאים המופיעים במסמך. עם זאת, המחבר עשה את מירב המאמצים כדי לספק את המידע המדויק והמלא ביותר.

כל הזכויות שמורות ל**ניר אדר**

Nir Adar
Email: underwar@hotmail.com
Home Page: http://underwar.livedns.co.il

אנא שלחו תיקונים והערות אל המחבר.

בעיות האבטחה המוצגות במסמך:

- URL Guessing
- HTML Injection (Form Based)
- HTML Injection (Querystring Based)
- Cookies Manipulation
- SQL Injection
- Raise Errors
- Upload
- Path Traversal and Path Disclosure
- Null Bytes
- URL Encoding
- HTTP Header Manipulation
- HTML Form-Fields Manipulation
- HTML Comments
- Old, backup and unreferenced files
- Default Accounts
- Broken Access Control

URL Guessing

התקפה 1:

ספקי שטח אכסון רבים אינם מאפשרים להגדיר בסיס נתונים מוגן, וניתן להוריד את בסיסי הנתונים של האתרים המאוכסנים אצלם על ידי הקלדת כתובת בדפדפן.

אנשים נוטים לתת שמות דומים לבסיסי הנתונים, והתקפה זו מנצלת את הבעיה:

נניח שהכתובת בה אנו מעוניינים היא http://www.hostname.com/user. לעתים קרובות בסיס הנתונים נמצא בספריה db או database, למשל: http://www.hostname.com/user/db/.

שם קובץ בסיס הנתונים יכול להיות db.mdb, user.mdb, database.mdb, או למשל שם האתר.

עבור שרתים חינמיים התקפה זו קלה יותר. למשל, הספק Brinkster - www.brinkster.com, מרשה למשתמשים בו לשמור את בסיסי הנתונים רק בספרייה db.

הגנה:

במידה וניתן, שימוש בספק המאפשר לשים את בסיס הנתונים בספריה מאובטחת.

אם לא ניתן למצוא ספק כזה, הצעדים הבאים יכולים להינקט:

- שם בסיס הנתונים צריך להיות קשה לניחוש.
- יש לדאוג שלא יהיה ניתן לעשות listing לספריה בה נמצא בסיס הנתונים. הוספת קובץ index.html ריק תבצע את העבודה בד"כ.
- הגנת בסיס הנתונים באמצעות סיסמא.
- הצפנת המידע (סיסמאות ומידע אחר) בתוך בסיס הנתונים, על מנת שגם אם יושג תהיה הגנה על המידע.

ASP3 אינו כולל אלגוריתמי הצפנה משל עצמו, אולם ניתן למצוא כאלו באינטרנט (www.planetsourcecode.com). לעומתו ASP.Net כולל אלגוריתמי הצפנה כחלק מהשפה.

במידה ובסיס הנתונים כולל משתמשים וסיסמאות, כדאי להצפין כל סיסמא בעזרת מפתח אחר - למשל, ה-id של אותו משתמש.

התקפה 2:

התקפה נוספת היא על דפי המנהל של האתר. במקרים רבים הדפים נמצאים תחת הכתובות:

www.hostname.com/admin.asp
www.hostname.com/login.asp
www.hostname.com/admin/default.asp
www.hostname.com/admin/admin.asp

במקרים רבים המנהלים מניחים שאף אחד מלבדם אינו מכיר את דפים אלו, ולכן אף לא מגנים עליהם בסיסמא. במקרה ודפים אלו כן מוגנים, ניתן להריץ תוכנת brute-force לגילוי הסיסמא. (למשל, התוכנה Brutus הניתנת להורדה מ- www.hoobie.net).

הגנה:

- אסור להניח שפורצים לא ינחשו את דף המנהל. יש לאבטח באמצעות מערכת זיהוי את דפי הניהול.
- יש לבצע Log של ניסיונות כניסה כושלים על מנת לאתר פריצות.
- ניתן, למשל, לנעול את דפי המנהל אחרי מספר מסוים של ניסיונות כושלים, כך שרק באמצעות FTP יהיה ניתן לשחררם. בצורה כזו ניתן להתגונן מהתקפות Brute-Force.

HTML Injection (Form Based)

ההתקפה:

בעיה הנפוצה בעיקר באתרים המכילים פורומים או אמצעי אחר בו תוקפים יכולים לשלוח מידע שיוצג בפני משתמשים אחרים. לדוגמא:

נניח שבפורום קיימים שני Text Inputs: txtUser ו-txtContent, שהוספת המידע נעשית בצורה הבאה:

```
Con.Execute("INSERT INTO datadb (txtUser, txtText) VALUES('"& _
Request.Form("txtUser")& "','"& _
Request.Form("txtContent")& "')")
```

התוקף עלול לכתוב בשדה txtUser קוד הכולל תגי HTML שיגיעו אל משתמשים אחרים וירוצו על הדפדפן שלהם.

ההגנה:

- סינון תגי HTML מקטעי הקוד המגיעים מן המשתמשים.

HTML Injection (Querystring Based)

ההתקפה:

ההתקפה מתאפשרת כאשר מתכנתים משתמשים במחרוזת המתקבלת כפרמטר כדי להציג תוכן בדפים.

נביט בתרחיש הבא:

בדף אחד קיים הקוד הבא:

```
If Request.Form("txtPass") <> "Password" Then
Response.Redirect "login.asp?err=Wrong+Password"
End If
```

כעת, אם בדף login.asp מופיע הקוד:

```
If Request.Querystring("err") <> "" Then Response.Write
Request.Querystring("err")
```

הרי שתוקף יוכל לגרום לקוד להתווסף לדף.

יותר בעייתי הוא המקרה בו קוד זה יהיה מצוי בתוך Form:

```
<form name="form1" method="post" action="check.asp">
<% If Request.Querystring("err") <> "" Then Response.Write
Request.Querystring("err") %>
(...)
```

במקרה כזה התוקף יכול לשנות את התנהגותו של הטופס, ולגרום, למשל, שמידע יישלח אל התוקף.

ההגנה:

- סינון תגי HTML מקטעי הקוד המגיעים מן המשתמשים.

Cookies Manipulation

ההתקפה:

משתמש עלול לשנות את ה-cookies שהשרת שלח אליו, ולהשיג גישה מיוחדת אל השרת.

למשל, ה-cookie המקורי עשוי להיות:

lang=en-us; admin=no; y=1;

התוקף עשוי לשנוי את ה-cookie כך:

lang=en-us; admin=yes; y=1;

ולקבל גישת מנהל.

ההגנה:

לשרת אסור להניח שהמידע המגיע מהמשתמש אמין. ניתן להשתמש במידע כדי לבצע קיצורי דרך, אולם אסור שמידע המגיע מן המשתמש יהיה האישור היחידי לזהותו של המשתמש.

לחילופין, ניתן להשתמש ב-cookies אם ננקטים מספר אמצעים:

- בעת שליחת ה-cookie למשתמש מייצרים hash value כחלק מה-cookie. כאשר ה-cookie נשלחת לשרת, משווים את ה-hash מול ה-hash המצופה.
- הצפנת כל המידע ב-cookie, על מנת שיהיה קשה לזייף אותו.
- בנוסף למידע השמור ב-cookie, נוסיף מידע בשרת המזהה את ה-cookie המסוימת, ומאשר אם היא שונתה או לא.

SQL Injection

ההתקפה:

בדומה להתקפות הקודמות, אם מידע עובר מפרמטרים של הדף ישירות אל Statement של SQL מתעוררת בעיית אבטחה. התוקף עלול להיות מסוגל לשנות את פעולת השאילתה, ולבצע פעולות על בסיס הנתונים.

לדוגמא:

```
Set rsQuery = Con.Execute("SELECT * From X WHERE User_Name='" & _
Request.Form("User") & "' AND User_Pass = '" & _
Request.Form("Pass")&amp; "'")

If rsQuery.EOF = True Then
      Response.Write "Wrong password!"
Else
      Session("User") = rsQuery("username")
      Session("Level") = rsQuery("level")
      Response.Redirect "admin.asp"
End If
```

אם המשתמש מקיש שם וסיסמא, למשל, Underwar/12345, שאילתת ה-SQL עשויה להראות כך:

```
SELECT * FROM X WHERE User_Name='Underwar' AND User_Pass='12345'
```

אם המשתמש יקיש ' or ' כשם משתמש ו- ' or ' כסמא, השאילתה תראה כך:

```
SELECT * FROM X WHERE User_Name='' OR '' AND User_Pass='' OR ''
```

אם הקוד אינו מאובטח כראוי, המשתמש יורשה להגיע אל דף המנהל.

הגנה:

יש צורך לסנן ולנתח את המידע לפני שמשלבים אותו בשאילתות SQL.

סינון וניתוח המידע משתנה בהתאם למידע שאמור להגיע.

הגנה למשל לדוגמא שהוצגה, יכולה להיות:

```
Set rsQuery = Con.Execute("SELECT * From X WHERE User_Name='" & _
Replace(Request.Form("User"), "'", "''") & "' AND User_Pass = '"& _
Replace(Request.Form("Pass"), "'", "''") & "'")

If rsQuery.EOF = True Then
      Response.Write "Wrong password!."
Else
      Session("User") = rsQuery("Benutzername")
      Session("Level") = rsQuery("Zugangslevel")
      Response.Redirect "admin.asp"
End If
```

- סינון סימנים מיוחדים מפחית את הסיכוי להתקפת SQL Injection אולם לא מונע אותה לגמרי.

Raise Errors

ההתקפה:

במידה ופרמטרים שהאתר מקבל מהמשתמש לא נבדקים, ייתכן מצב שהמשתמש ייצר הודעות שגיאה אצל השרת.

למשל, אם פרמטר מסוים הוא מסוג int, ומשתמש מקיש רצף אותיות, או מספר גדול מאוד, תיווצר שגיאה במקרה והפרמטר אינו נבדק. לדוגמא: אם הדף הוא show.asp?id=34, ומשתמש מחליף את הכתובת ל-show.asp?id=trghrt עלולה להיווצר שגיאה.

ההגנה:

ניתוח הפרמטרים המגיעים לפני השימוש בהם.

Upload

ההתקפה:

אתרים מסוימים מאפשרים להעלות אליהם קבצים.

בעיית אבטחה הנוצרת לעיתים היא תוקף המעלה קבצי asp, למשל, ואז מפעיל אותם על השרת.

במידה והוא מצליח, יש לו את אותן זכויות שיש לשרת באותו מחשב. התוקף עלול להעלות דפי ASP שיאפשרו לו לשוטט בין קבצי המערכת, להעלות, להוריד ולמחוק קבצים, ועוד.

ההגנה:

אם ניתנת האפשרות להעלות קבצים, יש להגביל את סוגי הקבצים המותרים להעלאה.

במידה ולא ניתן להגביל את סוג הקבצים, יש לנתח את הקבצים המועלים כדי למצוא קוד פוטנציאלי העלול לגרום לבעיות אבטחה.

Path Traversal and Path Disclosure

ההתקפה:

לעתים דפי ASP מקבלים כפרמטר כתובת של דף אחר. לדוגמא:

http://www.website.com/main.asp?page=view.asp

אם הפרמטר שנבדק לא מתקבל, התוקף יכול להגיע גם לדפים אחרים שמתכנן האתר לא התכוון שיגיע אליהם.

יתר על כן, על ידי שימוש בסימון /.. התוקף יכול להגיע גם לחלקים בשרת שאינם אמורים להיות גלויים לציבור. למשל, יהיה ניתן לשנות את ה-URL לכתובת הבאה:

http://www.website.com/main.asp?page=..\..\Windows\users.dat

ולהגיע לקבצים אחרים במחשב המשתמש.

ההגנה:

סינון הפרמטרים המתקבלים. במידה והפרמטר מציין נתיב, יש לסנן את הצירוף "..".

יש לשים לב שיש יותר מדרך אחת לייצג סימנים אלו. יש לנתח היטב את הפרמטרים כדי לא לפספס מקרים.

Null Bytes

ההתקפה:

שרתים רבים, ללא קשר לשפה בה כתובים הסקריפטים בהם, משתמשים בפונקציות של C/C++ כדי לנתח נתונים. שפת C/C++ מחשיבה את תו ה-NULL המסומן 0\ כתו סיום המחרוזת.

אם תוכנית Web מקבלת כפרמטר את המחרוזת "AAA\0BBB", הפונקציות בשפת C עלולות לפרש אותה כ-"AAA".

ניתן לרמות תוכנות גם על ידי הכנסת תו ה-NULL כערך של פרמטר חיוני. ההתקפה בדרך כלל נעשית על ידי קידוד URL של תו ה-NULL: 00%, ולעיתים גם על ידי Unicode Chars.

ההתקפה הזו יכולה להצטרף למגוון מההתקפות שתוארו קודם.

התוקף יכול ליצור הודעות שגיאה, לחשוף את הנתיבים של בסיסי הנתונים.
התוקף יכול להשתמש בתו זה כדי לסיים שאילתת SQL לפני המקום בו היא הייתה אמורה להסתיים, ולגרום לאינטרפרטר להתעלם מהמשכה.
התוקף יכול להשתמש בתו כזה כדי לשטות בבדיקת תקינות קלט עבור הפרמטר.

אתרים המשתמשים ב-Java פגיעים ביותר להתקפה זו, בייחוד כאלו הכוללים שימוש במחלקות כגון File, RandomAccessFile וכדומה.

ההגנה:

ניתוח הפרמטרים, וסינון של Null Byte או סימנים אחרים, העלולים להתפרש לתו זה.

URL Encoding

ההתקפה:

ניתן לייצג תווים על ידי סימן אחוזים ואז שתי ספרות הקסדצימליות המתארות את ערך התו. (סה"כ 8 ביטים). לדוגמא, התו רווח יכול להיות מצוין כ-20%.
על ידי שימוש ב-URL Encoding התוקף יכול להסתיר במקרים רבים קוד זדוני.
ניתן לשנות את הפרמטרים המועברים אל הדף, ולהתחמק מבדיקות תקפות רבות.

ההגנה:

- בחירה של סטנדרט בו יקודדו כל הנתונים המגיעים מהמשתמש, והמרת נתוני המשתמש לאותו סטנדרט לפני שנעשית כל החלטה האם הנתונים מאושרים או לא.
- ביצוע בדיקות תקינות קלט רק לאחר שפענוח הקלט הושלם לגמרי.

HTTP Header Manipulation

דפדפנים רגילים אינם נותנים למשתמש את האפשרות לשנות את ה-HTTP Headers הנשלחים על ידם, ולפיכך HTTP Headers נחשבים על ידי בוני אתרים כאמינים.
עם זאת, גישה זו איננה נכונה - המשתמש יכול לכתוב תוכנית קצרה בת כ-15 שורות ב-Perl או ב-Visual Basic, למשל, שתהיה מסוגלת לשלוח בקשות HTTP אל שרתים.
ישנן מספר התקפות שניתן לבצע בעזרת HTTP Headers:

התקפה 1 - Referrer Header:

Referrer Header מכיל את הכתובת ממנה המשתמש הגיע אל הדף הנוכחי. אתרים שונים מבססים את ההגנה שלהם על כותרת זו, ובודקים שהמשתמש הגיע אל הדף הנוכחי דרך דף אחר שיוצר על ידי אותו שרת. תוקף יכול לשנות את Referrer Header כדי שהשרת יחשוב שהוא הגיע מדף לגיטימי.

התקפה 2 - Accept-Language Header:

Accept-Language זוהי כותרת המציינת את השפה המועדפת על המשתמש.

אתרים התומכים במספר שפות עשויים לקחת כותרת זו ולבצע בעזרתה חיפוש בבסיס הנתונים, על מנת להציג את הטקסט בשפתו של המשתמש.

אם השרת משתמש בכותרת כדי לחפש מידע בבסיס הנתונים, התוקף יכול לבצע SQL Injection על ידי שינוי הכותרת.

אם השרת משתמש בכותרת כדי לבנות שם קובץ אליו הוא ייגש, ניתן לבצע Path Traversal.

ההגנה:

- אין לסמוך על כותרת HTTP מבלי לבדוק אותן.
- אין להחליט החלטות לגבי המשתמש רק בעזרת המידע המגיע מן הכותרות (Referrer Address למשל).

HTML Form-Fields Manipulation

ההתקפה:

תוקף עשוי לשנות טופס באתר, כך שישלח כל מידע לפי רצונו.

שינויים יכולים לעשות בפשטות על ידי שמירת קוד הטופס מהאתר, שינויו, והעלאת הקוד החדש בדפדפן של התוקף.

התוקף עשוי, למשל, לשנות את השדות maxlen, disabled, readonly ואחרים, הנמצאים בטופס.

התוקף עשוי לשנות ערך של שדות בלתי נראים. למשל, נניח קיים בדף השדה:

```
<input name="masteraccess" type="hidden" value="N">
```

התוקף עשוי לשנותו ל-

```
<input name="masteraccess" type="hidden" value="Y">
```

ולקבל גישת מנהל.

ההגנה:

- אסור להניח דבר על השדות המגיעים מטפסים - יש לבדוק את כולם.
- זיהוי המשתמש צריך להיעשות בצד השרת על ידי משתני session, ולא להתבצע כשדות נסתרים בצד המשתמש.

HTML Comments

ההתקפה:

דפי HTML הנשלחים אל המשתמש עשויים להכיל הערות HTML, המכילות מידע על המבנה הפנימי של האתר, הרלוונטי רק עבור המפתחים ומנהלים האתר.
ההערות יכולות להיות מיוצרות על ידי רכיבים אוטומטיים המשולבים באתר, או על ידי מתכנתי האתר.
לדוגמא: "שדה נסתר זה חייב להיות 1 או שהדף לא יעבור כראוי", "אסור לשנות את סדר התאים בטבלה" וכו'. תוקף יכול להשתמש בהערות אלו כדי לקבל מידע על נקודות החולשה של האתר.

ההגנה:

מחיקת כל הערות ה-HTML מהדפים הנשלחים אל המשתמשים. עבור הערות הנוצרות באופן אוטומטי - יש להתקין תוכנה שתסנן את הדף רגע לפני שנשלח אל המשתמש.

Old, backup and unreferenced files

ההתקפה:

אתרים רבים מכילים דפים ישנים שנשמרו לגיבוי, או סתם לא נמחקו.
כמו כן, שרתים עשויים להכיל קבצי דוגמא שנכללו בהתקנת השרת.
קבצי הדוגמא מכילים פרצות ידועות, וקבצי הגיבוי עשויים להכיל באגים שטופלו בגרסאות האחרונות של האתר.
תוקפים לרוב מריצים סדרה של בדיקות על מנת למצוא בעיות אבטחה ידועות (showcode.asp למשל).
הודעות שגיאה שונות - למשל 404, יכולות לגלות לתוקף איזה תוכנה רצה על השרת, ואילו חבילות מותקנות עליו.

ההגנה:

- מחיקת כל קובץ שאינו בשימוש ושאינו חיוני לפעולת האתר התקינה - כולל קבצי הדוגמאות שבאים ביחד עם השרת.
- החלפת הודעות השגיאה הסטנדרטיות בהודעות שלא ירמזו על התוכנה המותקנת באותו מחשב.
- בדיקה תקופתית של קבצי השרת - האם קיימים ביניהם כאלו שלא בשימוש.

Default Accounts

ההתקפה:

שרתים רבים כוללים רכיבים כגון פורומים או מערכות משתמשים אחרות, הכוללים משתמשים הנוצרים כברירת מחדל על ידי ההתקנה, כגון Administrator, Guest, Test וכדומה, להם סיסמאות ידועות. תוקף עשוי להשתמש בחשבונות אלו, במידה ולא שונתה הסיסמא ההתחלתית שלהם.

ההגנה:

- תמיד צריך לשנות את הקונפיגורציה ההתחלתית בה מגיעה חבילת התוכנה בה משתמשים.
- מחיקת כל חשבונות המשתמשים שאינם בשימוש.
- ביטול, במידת האפשר, של האפשרות לתת למשתמשים מרוחקים להתחבר לחשבון המנהל.
- שינוי כל סיסמאות ברירת המחדל של המערכת.

Broken Access Control

ההתקפה:

בביטוי Access Control אנו מתכוונים לחלוקת הזכויות בין המשתמשים המורשים בשרת מסוים - הכוונה - לאחר שמשתמש כלשהו אושר להיכנס לאתר - אילו פעולות מותרות עבורו, ואילו אסורות. משתמש עשוי להיות שייך גם למספר קבוצות משתמשים, שלכל אחת זכויות משלה. השרת צריך לנהל את זכויות המשתמש, ולתת לו לגשת רק לאזורים המורשים עבורו.

באתרים רבים מערכת ניהול המשתמשים צומחת ביחד עם האתר, ללא תכנון מסודר. עובדה זו יוצרת בעיות רבות. קוד ה-Access Control מוסף תוך כדי התפתחות האתר, ובמקרים רבים נשארים קטעים באתר שאינם מוגנים כראוי.

רבות מבעיות אלו אינן קשות לאיתור וניצול - כל שעל התוקף לעשות הוא לנסות לבצע פעולות שאמורות להיות חסומות עבורו, ולמצוא חור באבטחה.
אם נמצאה פרצת אבטחה, תוקף עלול להיות מסוגל לסייר בחלקים פרטיים של האתר, להוסיף או למחוק תוכן, ואף לקבל זכויות מנהל.

כמעט כל אתר צריך Access Control. לפיכך - מפתחי האתר צריכים להגדיר חוקים ברורים לגבי איך צריך לממש את מדיניות בקרת הגישה. אם לא קיים מסמך המתאר מדיניות זו, סביר להניח שבקרת הגישה לא נעשית בצורה מאורגנת, ושניתן לפרוץ לאתר.
בעיות נפוצות שניתן לנצלן להתקפה זו:

- IDs לא מאובטחים - רוב האתרים הגדולים משתמשים במפתח (id) כדי לזהות את המשתמשים. במערכות רבות, אם התוקף מנחש את ה-id של משתמש מסוים, הוא יכול לשוטט באתר עם הזכויות של אותו משתמש.
- לעיתים מערכת ההגנה של האתר מונעת מהמשתמשים להיכנס לדפים שנמצאים עמוק באתר. עם זאת, במקרים רבים הדפים הנמצאים עמוק יותר אינם מוגנים בעצמם, ומסתמכים על כך שהגולש לא ינחש את כתובתם.
- הרשאות קבצים - קבצים רבים הנמצאים באתר ניתנים לגישה על ידי הדפדפן, למרות שאינם אמורים להיות נגישים לכל משתמש. תוקף המנחש את כתובתם יכול לגשת אליהם ולקרוא את תוכנם. הבעיה נובעת בד"כ עקב העובדה שמערכת ההפעלה עליה מותקן השרת מאפשרת גישה לקבצים בדרך זו. הפתרון לבעיה הוא גם על ידי מערכת ההפעלה - ארגון נכון של זכויות הגישה לקבצים עבור השרת, זכויות ההפעלה, זכויות הקריאה וזכויות הכתיבה, על מנת למנוע ממשתמשים לא מאושרים לגשת אל הקבצים.

ההגנה:

- הגדרת בקרת הגישה הרצויה לאתר, והגדרה כיצד צריך לבדוק האם המשתמש מאושר או לא.
- יצירת מנגנון שינהל את בקרת הגישה. עדיף שמנגנון זה יהיה מנגנון יחיד שישרת את כל חלקי האתר, על מנת שיהיה קל לבדוק את נכונותו ולעדכן את האתר כולו אם תתגלה בעתיד בעיה במנגנון.
- אי הסתמכות על פרמטרים המועברים על ידי המשתמש כאמצעי היחידי לזהות האם זהו משתמש חוקי.
- בדיקה "שקופה" האם המשתמש הינו משתמש חוקי בכל אחד מדפי האתר, ולא רק בדף הכניסה לאתר.
- ארגון נכון של זכויות הגישה לקבצים עבור השרת, זכויות ההפעלה, זכויות הקריאה וזכויות הכתיבה, על מנת למנוע ממשתמשים לא מאושרים לגשת אל הקבצים.

סיכום בעיות אבטחה ודרכי התגוננות

- ניתוח קלט המגיע מהמשתמש ופלט היוצא אל המשתמש, כדי לבדוק תקינותו. אין להניח שהמשתמש בדק את המידע, גם אם קיים קוד שבהכרח אמור לעשות זאת בצד המשתמש.
- כאשר מנתחים את קלט המשתמש, יש לזכור שניתן לייצג כל תו במספר אופנים. יש לבדוק את כל האפשרויות לייצוג תווים/קוד עוין.
- פישוט בדיקות האבטחה ככל שניתן, כדי שיהיה אפשרי לוודא שהן יעילות.
- "המערכת מוגנת בהתאם לרמת ההגנה של החוליה החלשה ביותר".
- החבאת מידע איננה הגנת המידע - היא איננה עובדת לטווח הרחוק, ולא תמיד גם לטווח הקצר.
- גם המודולים הפנימיים של האתר צריכים בדיקת פרמטרים, על מנת לספק הגנה במקרה שפורץ פרץ את החיצוניים.
- מימוש אסטרטגית "Default deny".
- שרת ה-Web צריך לקבל מינימום זכויות - רק את אלו המאפשרות לו לפעול, על מנת למנוע ניצול של השרת במקרה של פריצה.
- במידה והשרת צריך לקבל מידע רגיש - יש לשקול שימוש בפרוטוקול SSL להעברת המידע.
- רצוי להצפין את הנתונים על השרת - כך שגם במקרה פריצה הפורץ לא יוכל לגשת ישירות אל הנתונים.
- שימוש ב-cookies כדי לזהות את הגולש, וכן ב-Referrer Header. שני אמצעים אלו יעילים לעיתים, אך יש לקחתם בעירבון מוגבל - תוקף עלול לזייף אותם.

  ניתן להצפין את ה-cookies כדי לשפר את האבטחה, וכן להעביר אותם דרך SSL.
- בחירה נכונה של שמות משתמשים וסיסמאות:

  - הגבלת אורך מינימאלי לסיסמא.

  - נעילת חשבונות אחרי ניסיונות כושלים לכניסה.

  - החלפה תקופתית של הסיסמאות.

  - הצפנת הסיסמאות בצד השרת.
- שמירת Logs של פעולות שונות המתבצעות. בדיקה תקופתית של קבצי ה-Log.
- מניעת Listing של קבצי האתר, על מנת להסתיר את מבנהו הפנימי.