Digital Whisper Technologic Papers

הסודות החבואים ב-WebSocket

מאת רזיאל בקר

הקדמה

רשת האינטרנט "הקלאסית" כיום עובדת בתבנית המבוססת על "בקשה-תגובה" של פרוטוקול HTTP. הדפדפן טוען את עמוד האינטרנט, ועד שהמשתמש לא לוחץ על קישור בדף לא קורה שום דבר מבחינת הרשת. בשנת 2005, טכנולוגיית ה-Ajax נכנסה לשוק ובעזרתה אתרים הפכו ליותר דינמיים. אך עם זאת, כל התקשורת בין הדפדפן לאתר התבצעה על ידי הפרוטוקול HTTP כך שתמיד נדרשה התערבות מצד הלקוח כדי לקבל מידע מהשרת.

אתרי האינטרנט גדלו ועם גדילתם גדל הצורך להעביר יותר ויותר נתונים, הבעיה ב-HTTP הייתה שבכל פעם שהדפדפן היה צריך לקבל נתונים מהשרת, הוא היה צריך לשלוח בקשה אליו כל פעם מחדש כדי לבדוק האם ישנו נתון חדש שעליו לקבל והאם להשתמש בו או לא, אחד הפתרונות היצירתיים הוא HTTP Long Polling[1], הדפדפן שולח בקשת HTTP לשרת ומחכה לתגובה שלו, ברגע שיש מידע חדש שהדפדפן אמור לקבל מהשרת השרת מגיב עם המידע החדש והדפדפן שולח את הבקשה עוד פעם וכך יוצא מצב שבעצם מפתח האתר גורם ל-DOS על השרת שלו, בעיה נוספת היא מגבלת החיבורים למארח[2] הדפדפן מגביל את החיבורים כדי שהשרת לא יוצף בבקשות על ידי הדפדפן.

BROWSER	MAX PARALLEL CONNECTIONS PER HOST
IE 6 and 7	2
IE 8	6
IE 9	6
IE 10	8
Firefox 2	2
Firefox 3	6
Firefox 4 to 17	6
Opera 9.63	4
Opera 10	8
Opera 11 and 12	6
Chrome 1 and 2	6
Chrome 3	4
Chrome 4 to 23	6
Safari 3 and 4	4

כמות החיבורים המקסימלית למארח לכל דפדפן

אחת הבעיות הגדולות בפתרונות קשורים למודל ה-HTTP היא שבכל פעם שהדפדפן ישלח בקשת HTTP אל השרת, בבקשה יכללו המון נתונים שברב המקרים השרת לא צריך אותם, הנתונים האלו נקראים כותרים (Headers) וזה יכול להגיע למצב שיותר מ-50% מהתוכן של הבקשה או התשובה הוא הכותרים, אם אתה מתכנת WEB ואתה כותב משחק

[1] http://stackoverflow.com/a/333884
[2] http://sgdev-blog.blogspot.sg/2014/01/maximum-concurrent-connection-to-same.html

מבוסס-דפדפן, זמן התגובה הוא קריטי בשבילך כדי שהמשחק יפעל "חלק". הכותרים אינם נחוצים למימוש רב המשחק ולמרות זאת הם יופיעו בבקשה והתגובה ויאריכו את זמן התגובה.

מה שאנחנו באמת צריכים הוא יצירת חיבור קבוע בין הדפדפן לשרת מה שיקצר את זמן התגובה וכאן פרוטוקול ה-Websocket נכנס למשחק.

משה! > קראת ליי? > כן! (Handshake)

התקשורת ב-Websocket מחולקות ל-2: לחיצת היד (Handshake) והעברת נתונים. כאשר אובייקט Websocket נוצר בדפדפן מאחורי הקלעים מתבצעת "לחיצת יד" בין הדפדפן לשרת. הדפדפן שולח בקשת HTTP GET Upgrade לשרת.

```
▼ Request Headers        view source
  Connection: Upgrade
  Host: localhost:28818
  Origin: http://localhost
  Sec-WebSocket-Extensions: x-webkit-deflate-frame
  Sec-WebSocket-Key: vZf88TCN54OQHuzfHSj/zA==
  Sec-WebSocket-Protocol: local
  Sec-WebSocket-Version: 13
  Upgrade: websocket
  (Key3): 00:00:00:00:00:00:00:00
▼ Response Headers       view source
  Connection: Upgrade
  Sec-WebSocket-Accept: DtYQ/WVT2ttfcx6paWiJjI2jcvg=
  Sec-WebSocket-Protocol: local
  Upgrade: WebSocket
  (Challenge Response): 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
```

בתמונה למעלה אנחנו יכולים לראות את הכותרת Upgrade: websocket שמצביעה על בקשת "שדרוג" הפרוטוקול. השרת "משדרג" את הפרוטוקול ל-Websocket שהוא פרוטוקול מבוסס TCP, בקשת ה"שדרוג" היא גשר בין פרוטוקול ה-HTTP ל-Websocket,בלחיצת היד הנתונים המועברים הם "משא ומתן" כדי לוודא שאין בעיות אבטחה.

2 הידרים חשובים הם "Sec-WebSocket-Key" ו-"Sec-Websocket-Accept": הדפדפן שולח את הכותרת "Sec-WebSocket-Key" המכילה בתים אקראיים מקודדים ב-base64 והשרת מגיב עם המפתח המגובב ב-"Sec-Websocket-Accept", לחיצת היד נועדה כדי למנוע שהמטמון ישלח "שיחה" קודמת מסוג Websocket ואינה מספקת שום אימות עם השרת או פרטיות המידע. פונקציית הגיבוב מצרפת את המחרוזת הקבועה "258EAFA5-E914-47DA-95CA-C5AB0DC85B11 " (GUID) לערך "Sec-Websocket-Key" (שאינו מפוענח ב-base64), חלה על המחרוזת SHA1 ומקודדת ב-base64. השרת שלכם יצטרך

לעקוב אחרי החיבורים הפעילים של כל המשתמשים, כך כדי שלא לשמור על לחיצת היד עם הלקוחות שכבר השלימו את לחיצת היד. כתובת IP יכולה להתחבר מספר פעמים.

במידה והחיבורים רבים הדפדפן יחסום אותם כדי להמנע מהתקפת מניעת שירות (DoS).

העברת הנתונים

הדפדפן והשרת יכולים לבחור לשלוח הודעה בכל רגע זה אחד מהיתרונות של הפרוטוקול Websocket. הנתונים העוברים בין הדפדפן לשרת מוצפנים באמצעות XOR[3] (עם מפתח של 32 סיביות). כל frame המועבר בין הדפדפן לשרת נשלח במסגרת הזאת:

```
 0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-------+-+-------------+-------------------------------+
 |F|R|R|R| opcode|M| Payload len |    Extended payload length    |
 |I|S|S|S|  (4)  |A|     (7)     |             (16/64)           |
 |N|V|V|V|       |S|             |   (if payload len==126/127)   |
 | |1|2|3|       |K|             |                               |
 +-+-+-+-+-------+-+-------------+ - - - - - - - - - - - - - - - +
 |     Extended payload length continued, if payload len == 127  |
 + - - - - - - - - - - - - - - - +-------------------------------+
 |                               |Masking-key, if MASK set to 1  |
 +-------------------------------+-------------------------------+
 | Masking-key (continued)       |          Payload Data         |
 +-------------------------------- - - - - - - - - - - - - - - - +
 :                     Payload Data continued ...                :
 + - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
 |                     Payload Data continued ...                |
 +---------------------------------------------------------------+
```

שדה ה-Mask (סיבית) מייצג האם ה-Payload מקודד או לא, מידע שנשלח על ידי הדפדפן חייב לעבור את תהליך המיסוך ולכן השרת מצפה ל-1 (אם הסיבית תיהיה 0 השרת ינתק את החיבור עם הדפדפן), המידע שנשלח על ידי השרת לא מקודד ולכן שדה הMask כבוי, אני אסביר על תהליך המיסוך בהמשך.

ההודעות ממוסכות על ידי הדפדפן גם בעת שימוש בחיבור מאובטח.

[3] https://en.wikipedia.org/wiki/XOR_cipher

שדה ה-opcode (4 סיביות), מגדיר את היחס למטען (Payload) שהתקבל:

- 0x01 - טקסט
- 0x02 - נתונים בינאריים

בגרסא הנוכחית של Websocket, ל-0x3 והלאה אין משמעות.

שדה ה-Fin (סיבית) מייצג האם המטען שהתקבל הוא המטען האחרון בסדרה, אם השדה כבוי השרת יחכה לעוד חלקים של ההודעה.

פענוח אורך ה-Payload

כדי לקרוא את המטען, אנחנו צריכים לדעת מתי להפסיק לקרוא אותו. זו הסיבה מדוע אורך המטען חשוב לנו, קחו את הסיביות ב-offest 9 עד ה-offest 15 והעבירו אותם לפורמט unsigned int, זה אורך המטען, אם אורך המטען הוא 126, תקראו את הבית (16 סיביות) הבא אם אורך המטען הוא 127 תקראו את 8 הבתים (64 סיביות) הבאים והעבירו אותם לפורמט unsigned int.

קריאה ומיסוך המטען

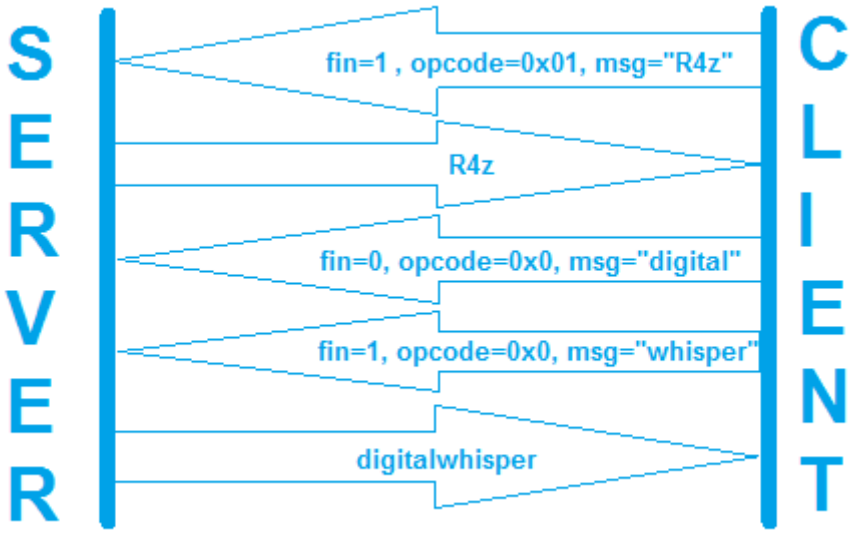
אם שדה ה-Mask דולק (והוא יידלק במידה וההודעה נשלחת על ידי הדפדפן אל השרת), הבתים (32 סיביות) הבאים הם מפתח המיסוך. כעת מפתח המיסוך ואורך המטען בידינו, שאר המידע הוא המטען שלנו. פענוח המטען יתבצע כך:

```
var DECODED = "";
for (var i = 0; i < ENCODED.length; i++) {
    DECODED[i] = ENCODED[i] ^ MASK[i % 4];
}
```

נרוץ בלולאה על המטען שקיבלנו ונבצע XOR עם הבית הנוכחי של המטען והבית ה-i % 4 של המפתח. וקיבלנו את המטען המקורי שהדפדפן שלח אל השרת ☺.

חלקת חבילות

כדי להעביר מידע, שדות ה-FIN וה-opcode משתלבות בהעברת הנתונים בין הדפדפן לשרת, תהליך זה נקרא חלקת חבילות (Message Fragmentation) וזמין רק כאשר הערך ב-opcode הוא בין 0x0-0x2. שדה ה-opcode מייצג את סוג המטען, 0x1 מייצג את המטען כטקסטואלי, 0x2 מייצג את המטען כנתונים בינאריים לעומת זאת - אם הערך בשדה הopcode הוא 0x0 המסגרת היא מסגרת המשך, זאת אומרת שהשרת צריך לשרשר את ההודעה עם המסגרת האחרונה שקיבל מהדפדפן.



במסגרת הראשונה דגל ה-FIN דולק (fin=1 ו-opcode!=0x0), כך הדפדפן מודיע לשרת שזה סוף ההודעה. במסגרת השנייה הדפדפן שלח הודעת טקסט(opcode=0x1) אבל השרת עדיין מחכה לסוף ההודעה(דגל ה-FIN כבוי) ואת המסגרת האחרונה הדפדפן ישלח כאשר דגל ה-FIN דולק.

פינג-פונג ב-Websocket

אחרי לחיצת היד בין הדפדפן לשרת, הדפדפן או השרת שולחים חבילת פינג. כשהפינג מתקבל - הנמען חייב לשלוח חבילת פונג, הפרוטוקול משתמש בזה כדי לוודא שהלקוח עדיין מחובר. פינג פונג או הוא רק מסגרת קבועה, אבל זה מסגרת שליטה. יש לי Pings opcode של 0x9, ויש לי pongs opcode של 0xA. כאשר אתה מקבל פינג, לשלוח בחזרה פונג עם אותם נתונים המדויקים כמטען פינג (ל-pings ו-pongs, אורך המטען המרבי הוא 125). ייתכן גם לקבל פונג מבלי לשלוח פינג; להתעלם מכך אם זה קורה.

חבילות הפינג או הפונג הן בעצם מסגרת קבועה, הערך ב-opcode 0x9 מייצג פינג והערך 0xA מייצג פונג, כאשר הדפדפן מקבל פינג הוא חייב לשלוח בחזרה פונג עם אותה הודעה מדויקת, לחבילות הפינג-פונג אורך המטען הוא 125 - השרת יכול גם לקבל פונג מבלי שהוא ישלח פינג.

אם אתה מקבל פינג אחד או יותר לפני שאתה שולח חבילת פונג, אתה רק צריך לשלוח חבילת פונג אחת.

סגירת החיבור

כדי לסגור חיבור בין הלקוח או שרת יכול לשלוח מסגרת שליטה בנתונים המכילים רצף שליטה צוין להתחיל לחיצת יד הסגירה (מפורט בסעיף 5.5.1). עם קבלת מסגרת כזו, העמיתים האחרים שולח מסגרת לסגור בתגובה. העמיתים הראשונים אז סוגרים את החיבור. כל הנתונים נוספים שהתקבלו לאחר סגירת החיבור אז הושלכה.

סגירת החיבור בין הדפדפן לשרת מתבצעת על ידי שליחת חבילת סגירה (opcode=0x8), הנמען שולח מסגרת סגירה בתגובה ולאחר מכן סגירת החיבור מתבצעת על ידי שני הצדדים. המטען מכיל את הסיבה לסגירת החיבור (אם החבילה שהתקבלה גדולה משמעותית או לא עמדה בתקן). גם חבילות סגירה הנשלחות מהדפדפן אל הלקוח עוברות תהליך מיסוך.

אז איפה הכיף?

<u>הערת צד: ישנן בעיות חוקיות בכל הנוגע להנדסה לאחור. מדריך זה נועד ללימוד עצמי בלבד!</u>

אחרי שאנחנו מכירים את הפרוטוקול לעומק, מבינים איך הוא עובד - מה קורה ברקע ומה עומד מאחורי כל תכונה, ואף אם ניקח את זה רחוק יותר, יישום לקוח או שרת מבוססים Websocket. עכשיו אפשר לעבור לצד המעשי או במילים אחרות - הגענו לחלק הכיפי ☺.

הפרוטוקול נכנס לשוק לא מזמן ולכן שירותים רבים משתמשים ב-Websocket (משחקי דפדפן, Skype, Whatsapp ועוד) וזה נובע מכך שהפרוטוקול עדיין טרי - מפתחים רבים לא מבינים איך לעבוד איתו ואיך לשלב אותו נכון במערכות שונות. בהמשך, ננתח משחק דפדפן (MMORPG) מבוסס Websocket ובהמשך:

- נבין את המחשבה שעומדת מאחורי כל החלטה שהמערכת מקבלת ואיך אפשר לנצל אותה.
- נלמד "לרמות" במשחק ואני מתכוון לעריכת שם משתמש, חפצים וכל נתון אחר שתבחרו.
- כתיבת סקריפט שמתחבר אוטומטית לכל שחקן.

תצטרכו:

- גוגל כרום כדי להנדס לאחור (Reverse Engineering)
- ידע בסיסי ב-Javascript
- Wireshark למי שלא אוהב להשתמש בכרום כדי לנתח חבילות
- להשתמש בראש שלכם ☺

BrowserQuest

BrowserQuest הוא משחק MMORPG שנכתב על ידי Little Workshop ו-Mozilla Foundation, המשחק נכתב ב-HTML5 ו-WebSocket ותומך בדפדפנים העדכניים. BrowserQuest שוחרר כ-Open Source ב-GitHub. הקוד תחת הרישיון MPL 2.0 התוכן תחת CC BY-SA 3.0.

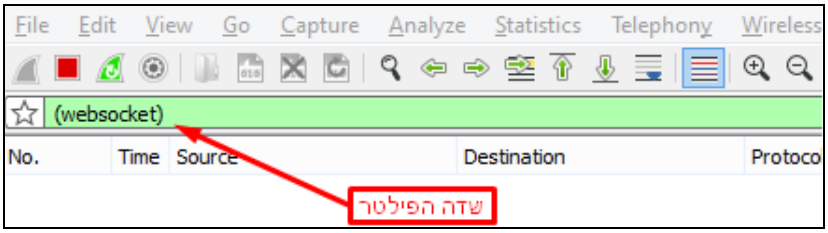
מידע נוסף:

- HTTP://www.littleworkshop.fr/browserquest.HTML
- Https://hacks.mozilla.org/2012/03/browserquest/
- HTTP://www.engadget.com/2012/03/28/mozilla-browserquest-HTML5-game/

הסנפת חבילות Websocket עם Wireshark וGoogle Chrome

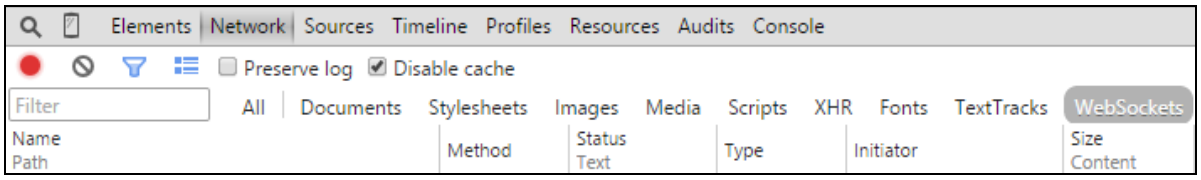
הסנפת חבילות Websocket ב-Wireshark:

1. Capture Options
2. בחר בכרטיס הרשת
3. מלא את שדה הפילטר ב-(websocket) או tcp port 8001:



באמצעות Chrome Developer tools אפשר לצפות במטענים שעוברים בין הדפדפן לשרת ולצפות בו אם המידע אינו נתונים בינאריים (opcode=0x1). הסנפת חבילות Websocket ב-Google Chrome:

1. אתחלו את Google Developer tools (ctrl+shift+c או F12)
2. רעננו את הדף
3. לחצו על "Network" בטאבים.
4. בחרו בחיבור ה-Websocket מהרשימה (אם הסטטוס הוא 101 Switching Protocol) או בטאב WebSockets:

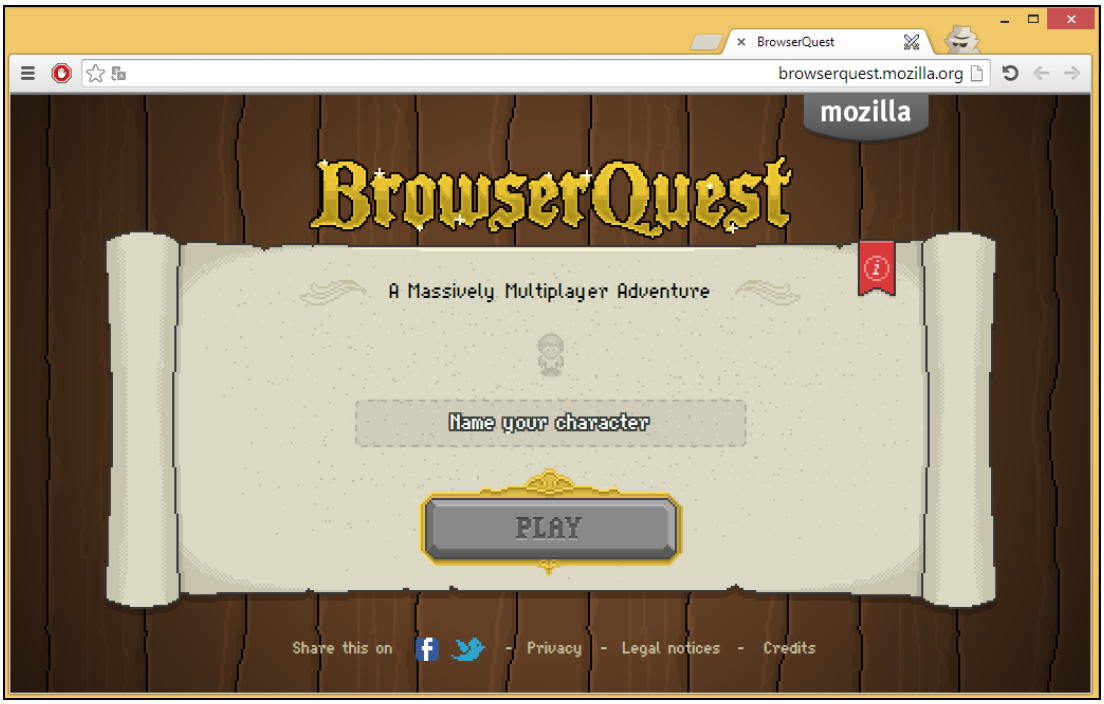


אופציה נוספת היא להשתמש ב-XSS-Track - לקריאה נוספת:

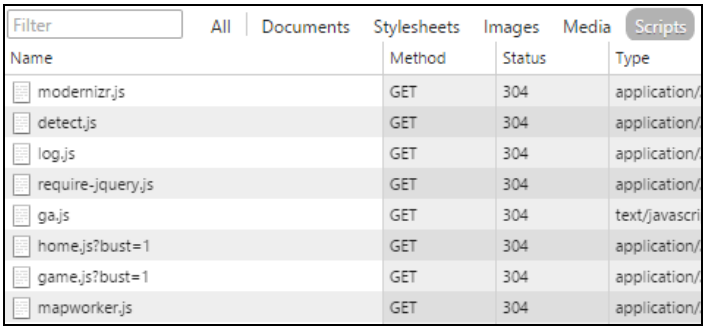
HTTP://blog.kotowicz.net/2011/01/xss-track-as-HTML5-websockets-traffic.HTML

וַיְדַבֵּר

יאללה בואו נתחיל, פתחתי את המשחק BrowserQuest ב-Google Chrome גרסא 45:



המשחק הוא משחק דפדפן, כדי שהדפדפן יבצע פעולה מסוימת, צריך קוד javascript שיוצר אותה. מה שאומר שהקוד javascript אחראי על מה שהדפדפן מראה לנו והדפדפן מראה לנו את מה שהמפתח רצה שנראה - חושבים ששווה לעבור על הקוד?



Filter	All	Documents	Stylesheets	Images	Media	Scripts

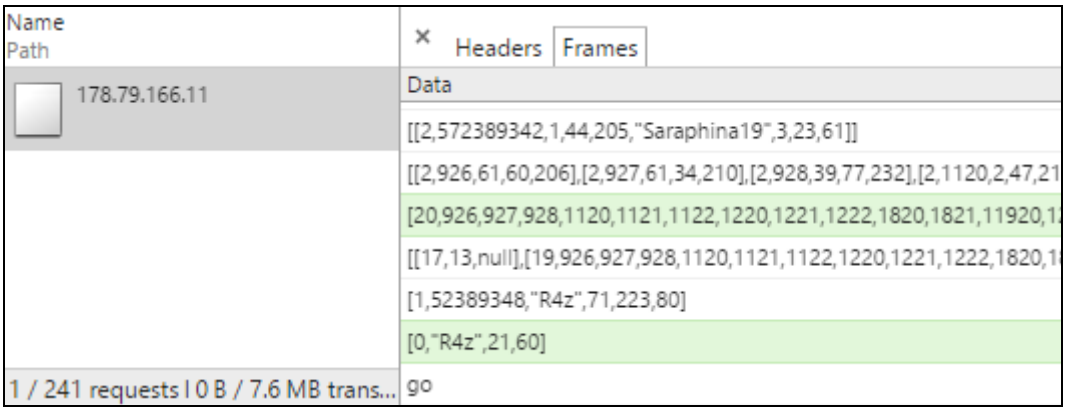
Name	Method	Status	Type
modernizr.js	GET	304	application/
detect.js	GET	304	application/
log.js	GET	304	application/
require-jquery.js	GET	304	application/
ga.js	GET	304	text/javascri
home.js?bust=1	GET	304	application/
game.js?bust=1	GET	304	application/
mapworker.js	GET	304	application/

משמות הקבצים ניתן להבין שהכי מעניינים הם home.js ו-game.js (בדקתי ואכן הם קבצים ששווה לנו לעבור עליהם) אם תנסו לקרוא את הקוד, תצטערו לגלות שהקוד מסורבל וקשה לקריאה בדרך כלל מפתחים משתמשים בזה כדי שהקובץ יתפוס פחות מקום או Anti-debugging, במקרים כאלה אפשר להשתמש בכלי codebeautify.org. באמצעותו נהפוך את הקוד לקריא, לאחר מכן נעתיק אותו לעורך שלנו (אני משתמש ב-Notepad++).

אבל קודם כל, אני אראה לכם איך אפשר להתחבר לשרת websocket ב-javascript.

כתיבת ה-client בjavascript

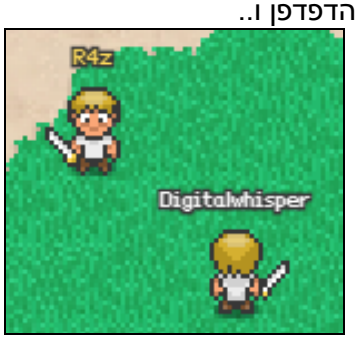
נאתחל את Chrome Developer tools כדי להסניף את המטענים שעוברים ב-Websocket:



בלשונית Frames אפשר לראות את המטענים שעוברים בין הדפדפן לשרת. המטענים שהשרת שולח אל הדפדפן מודגשים בלבן והמטענים שהדפדפן שולח אל השרת מודגשים בירוק. נכתוב את ה-client ב-javascript (כמובן שאפשר לכתוב אותו בכל שפה, בחרתי לכתוב אותו ב-javascript מכיוון שנוח להשתמש ב-API של javascript - לעוד מידע: HTTP://bit.ly/1FziUWg):

```
var host = "ws://178.79.166.11/"; // הכתובת של השרת שאנחנו רוצים להתחבר אליו
var socket = new WebSocket(host); // פתיחת החיבור בין הדפדפן לשרת(תהליך לחיצת היד מתבצע כאן)
console.log(socket.readyState);
socket.onopen = function(){
         console.log('Socket Status: '+socket.readyState+' (open)');
}
socket.onmessage = function(msg){
   console.log('Received: '+msg.data);
   if(msg.data=="go") {  // כשאנחנו מקבלים מהשרת הודעה כזו אנחנו מבינים שאנחנו צריכים לשלוח לו מידע על השחקן שלנו
      socket.send('[0,"Digitalwhisper",23,65]');  // שם משתמש + מיקום
      socket.send("[20,90,924,925,926,927,1021,1120,1121,1122,1220,1221,1222,1320,1321,1322,1820,
      1821,84195,87195,815222,818209,837200,841195,847195,867220,870197,873197,8154138,540386503,
      551386501,564386450]");
   }
}
socket.onclose = function(){
         console.log('Socket Status: '+socket.readyState+' (Closed)');
}
```

נריץ את הקוד javascript באמצעות הדפדפן ו..



home.js

נתחיל ב-home.js, נסתכל עליו קצת ונראה משהו כזה:

```
define("storage", [], function() {
        …
   …
});
```

השימוש ב-define משמש להגדרת מודלים חדשים ב-requireJS, requireJS טוען מודלים, הם עוברים תהליך אופטימיזציה בטעינת המודל על ידי הדפדפן. requireJS יכול לשמש גם סביבות נוספות כמו Rhino או Node.

RequireJS מאפשר לך מהירות ואיכות גבוהה יותר. הקוד javascript יוצר מודל חדש בשם storage, כנראה שזה המודל שאחראי על אחסון הנתונים - לא חושבים ששווה להעביר מבט?

```
var a = Class.extend({
   init: function() {
             this.hasLocalStorage() && localStorage.data ? this.data =
JSON.parse(localStorage.data) : this.resetData()
},
```

מפתח המשחק השתמש ב-Local Storage כדי לאחסן נתונים מסויימים בדפדפן שלנו, אחסון נתונים לוקאלית בדפדפן הוא כלי חזק עבור מפתחי Web. שימוש ב-Local Storage[4] בדפדפנים עדכניים:

```
localStorage.setItem('username','R4z'); // אחסון נתונים
var username = localStorage.getItem('username'); // קריאת נתונים
localStorage.removeItem('username'); // מחיקת נתונים
                                        כדי לאחסן נתונים באופן זמני ניתן להשתמש בSession Storage (עד שהדפדפן יסגר).
resetData: function() {
        this.data = {
                hasAlreadyPlayed: !1,
                player: {
                        name: "",
                        weapon: "",
                        armor: "",
                        image: "",
…
},
```

התכונה resetData (מודגש בצהוב), מהשם resetData אפשר להבין שהוא אחראי על איפוס המידע, המתודה מאפסת נתונים על השחקן (מודגש בירוק), אפשר להבין שנתונים כמו שם השחקן, הנשק, שריון והתמונה מאופסים מה שאומר שהמשחק משתמש בנתונים האלה בשביל ביצוע לפחות פעולה אחת במשחק.

[4] http://www.smashingmagazine.com/2010/10/local-storage-and-how-to-use-it/

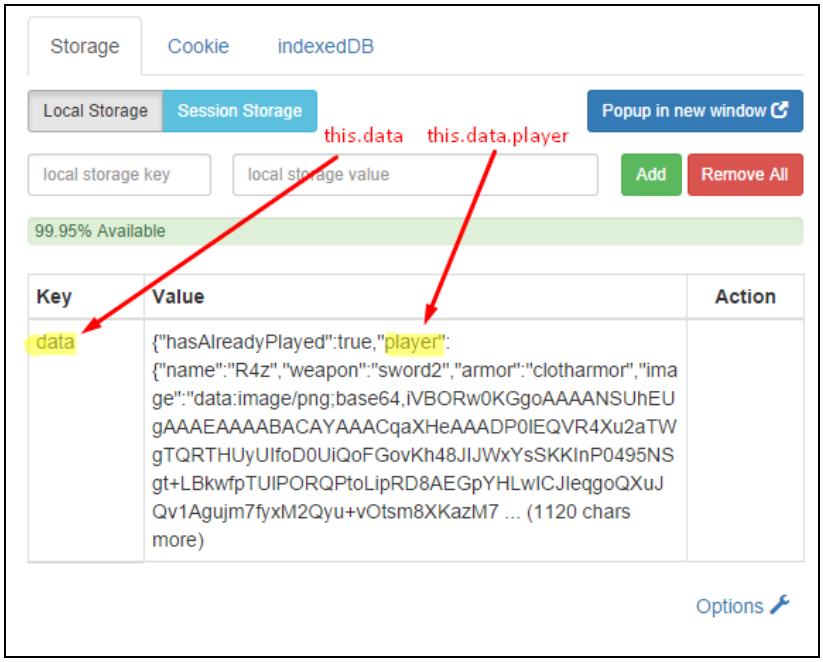
הנתונים שהאתר מאחסן בדפדפן (Local / Session Storage) יושבים בתיקיית ה-AppData בנתיב:

- %LocalAppData%\Google\Chrome\User Data\Default\Local Storage (Windows)
- ~/Library/Application Support/Google/Chrome/Default/Local Storage/ (OSX)

כדי לגשת אל הנתונים האלו בצורה נוחה יותר, אפשר להשתמש בתוספת Storage Manager של misc.im. הורדה:

https://chrome.google.com/webstore/detail/HTML5-storage-manager-all/giompennnhheakjcnobejbnjgbbkmdnd?hl=en

אחרי שהורדנו את התוסף, אפשר להתחיל לשחק ☺. אחרי שהרגתם עכבר או שניים, רעננו את העמוד ואתחלו את ה-Storage Manager:



המידע של השחקן שלנו נשמר ב-Local Storage בדפדפן, אם מעניין אותכם מה ה-image מייצג נשתמש שוב ב-codebeautify שוב: HTTP://codebeautify.org/base64-to-image-converter הכניסו את הערך של המפתח image אל שדה ה-base64 ושלחו את הטופס:

ה-image מכיל את תמונת השחקן בדף ההתחברות.

כדי לצפות ב-json בצורה יותר קריאה, אפשר להשתמש באתר HTTP://json.parser.online.fr:


```
{
   "hasAlreadyPlayed":true,
   "player": {
      "name":"R4z",
      "weapon":"sword2",
      "armor":"clotharmor",
      "image":"..."
   },
   "achievements": {
      "unlocked": [
         18,
         11,
         1
      ],
      "ratCount":4,
      "skeletonCount":0,
      "totalKills":4,
      "totalDmg":34,
      "totalRevives":0
   }
}
```


מגניב, הבנו שהמאפיינים של השחקן נשמרים אצלנו בדפדפן או ליתר דיוק ב-Local Storage ואנחנו יודעים שאפשר לערוך את מאפיינים אלו מהסיבה שהם נשמרים על המחשב שלנו, איך זה עוזר לנו? איך נערוך את ה-Local Storage כדי לרמות במשחק?

המפתח player אחראי על השחקן, armor ו-weapon מייצגים את הנשק והשריון של השחקן, הנשק והשריון משפיעים על נק' ההתקפה וההגנה של השחקן הרי ככל שהנשק והשריון ברמה גבוהה יותר, כך אנחנו יותר חזקים אם הנתונים האלו נשמרים אצלנו במחשב, ב-Local Storage אז אפשר לערוך את ה-Local Storage כדי שיהיה לשחקן שלנו את הנשקים הכי טובים. מה הנשקים או השריונים הטובים ביותר, מה האפשרויות שהמשחק מציע לנו? נקבל את התשובות האלו על ידי ניתוח קוד ה-javascript האחראי על ממשק המשחק.

game.js

הקובץ game.js אחראי על ממשק המשחק (הגדרת אובייקטים וכ'ו), כדי למצוא את החלק הזה של הקוד אפשר לעבור על כולו, אבל אפשר לנסות לחפש את האובייקט שמגדיר את הנשק sword2 לצורך העניין, נסו לחפש את הביטוי: "sword2:". (זה בדר"כ עובד במערכות, פשוט תכנסו לראש של המפתח).

```
define("items", ["item"], function(a) {
       var b = {
              Sword2: a.extend({
                     init: function(a) {
                            this._super(a, Types.Entities.SWORD2, "weapon"),
this.lootMessage =
               "You pick up a steel sword"
                     }
              }),
              Axe: a.extend({
                     init: function(a) {
                            this._super(a, Types.Entities.AXE, "weapon"),
this.lootMessage =
               "You pick up an axe"
                     }
              }),
       …
});
```

אחרי שקיבלנו קצת עזרה מה-js viewer בשורה 1689 אפשר למצוא את המודול שאחראי על החפצים (items - מסומן בירוק), לפי הסדר שבו המפתח מגדיר את החפצים בקוד אפשר להבין איזה חפץ הכי משתלם לנו לקחת ועל פי השם, כמובן שיותר נכון אפשר לרדת עמוק יותר ולהבין לפי התכונות של החפצים את החפץ החזק ביותר אבל לא נעשה את זה עכשיו.

אם תעברו על כל החפצים תוכלו לראות את ה-GoldenSword ואת ה-GoldenArmor:

```
              GoldenSword: a.extend({
                     init: function(a) {
                            this._super(a, Types.Entities.GOLDENSWORD, "weapon"),
this.lootMessage = "You pick up the ultimate sword"
                     }
              }),
              …
              …
              GoldenArmor: a.extend({
                     init: function(a) {
                            this._super(a, Types.Entities.GOLDENARMOR, "armor"),
this.lootMessage = "You equip a golden armor"
```

```
                }
            },
```

תחשבו איתי - ממשק המשחק מגדיר את הנשקים שלנו לפי השם שלהם ב-Local Storage, אם נשנה את שם הנשק או השריון ב-Local Storage - נוכל לשנות את החפצים של השחקן שלנו.

נאתחל את ה-Storage Manager שלנו ונערוך את ה-Local Storage בדפדפן כך שהנשק שלנו הוא goldensword והשריון הוא goldenarmor. (שמות החפצים נשמרים באותיות קטנות - אפשר להסיק את זה מכך שכך הם נשמרים כבר אצלנו.

Key	Value	Action
data	{"hasAlreadyPlayed":true,"player":{"name":"R4z","weapon":"sword2","armor":"clotharmor","image":" ... (1126 chars more)	Dump JSON Edit Remove Show All

עריכת מידע במפתח הנתונים

שנו את הערכים weapon ו-sword, לנשקים שתרצו(בחרתי בטובים במשחק). שמרו את ה-Local Storage ותרעננו את הדף, לאחר מכן תתחברו לשחקן שלכם.

כעת, השחקן שלנו מנצח כל עכבר במכה אחת ☺

כמובן אפשר לקחת את האפשרות לשינוי ה-Local Storage קדימה (שינוי שם השחקן ואני בטוח שעוד, פשוט תתעמקו בקוד ותחפשו פירצות נפוצות).

אז... איך לא ישחקו עם המידע שלי?

אני מאמין שבמשחק BrowserQuest הכוונה הייתה להשתמש ב-Local Storage כדי להציג את היכולות של javascript ולערב כמה שפחות צד שרת, **בדרך כלל עדיף לשמור נתונים משמעותיים בשרת ולא בלקוח**.

אם עדיין אתם **חייבים** לשמור מידע משמעותי אצל הלקוח **תצפינו אותו** (אפשר להשתמש ב-CryptoJS). בכדי ליצור מפתח FingerprintJS או שתוכלו להצפין את המידע בצד הלקוח ולפצח אותו כשהוא נשלח לשרת. במילים אחרות תנסו שהמימוש והבנת התהליך של שמירת / קריאת המידע בדפדפן יהיה כמה שיותר מסובך ואם אתם יכולים להשתמש בשרת כדי לאחסן את המידע הזה - תשתמשו בו. תמיד תשתמשו ב-Obfuscation Javascript אך על תבנו על זה. כמובן שתמיד אפשר לראות איך ההצפנה מתבצעת בצד לקוח ו"לרמות" במשחק אתם צריכים להקשות על תהליך ההבנה ולשבור את רוחו של הרוורסר.

וכמובן - לעולם, אבל לעולם אל תסמכו על קלט שהגיע מהמשתמש.

על המחבר

R4z בן 17 עוסק בפיתוח Web בחברת Articoloo, ובזמנו הפנוי מתעסק באבטחת מידע לכל שאלה או יעוץ ניתן לפנות אליו בשרת ה-IRC של NIX בערוץ #Security או באימייל, בכתובת:

raziel.b7@gmail.com

בנוסף, אני מעוניין להודות לאפיק קסטיאל על עזרתו המועילה למאמר זה.

למידע נוסף

- https://developer.mozilla.org/en-US/docs/Web/API/WebSocket
- http://tools.ietf.org/HTML/rfc6455
- http://www.w3.org/TR/2011/WD-websockets-20110419/
- http://www.codeproject.com/Articles/531698/Introduction-to-HTML-WebSocket
- http://rawkes.com/articles/creating-a-real-time-multiplayer-game-with-websockets-and-node.HTML