HTML5 מנקודת מבט אחרת - חלק ב'

מאת: לירן בנודיס ואלעד גבאי

הקדמה

עד לפני כמה שנים האינטרנט עבד בעזרת HTML4.01 ותוספות כמו JS, CSS וכו', בשנים האחרונות האינטרנט מתקדם לקראת עידן חדש, HTML5. כבר עכשיו אנו מתחילים לראות איך אפליקציות נוצצות שנכתבו ב-HTML5 משתלטות אט אט על הרשת.

HTML5 הינה שילוב של כמה טכנולוגיות חדשות ושינוי בשפות קיימות. ב-HTML5 נוספו תגיות חדשות ל-HTML, אך זהו בכלל לא השינוי המהותי שנעשה. HTML5 מוסיפה פונקציונאליות חדשה ומורידה איסורים קודמים שהיו על דפדפנים ועל מפתחי אפליקציות הרשת עד כה.

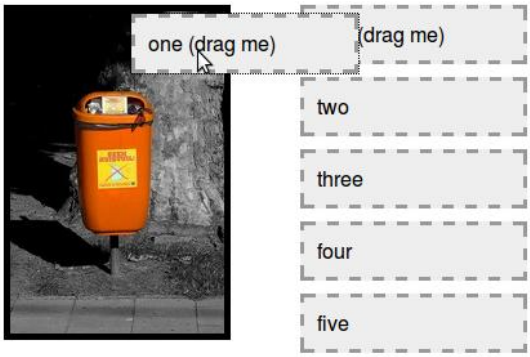
האינטרנט זהו לא המקום היחידי בו נראה HTML5, שכן היא תוכננה להוות תחליף לסביבות פיתוח רבות, כבר היום מערכות ההפעלה Android ו-IOS תומכות בכתיבת אפליקציות ב-HTML5, וגישה זו עתידה להמשיך ולצבור תאוצה, בשנים הקרובות נראה גם טלוויזיות המריצות אפליקציות HTML5 בנוסף כמובן למחשבים.

סדרת מאמרים זו סוקרת את התוספות החדשות של HTML5 ובוחנת אילו בעיות אבטחה תוספות אלו יוצרות, וכיצד ישפיעו על אבטחה בעולם הרשת.

במאמר זה, השני בסדרה, נסקור את מנגנון ה-Drag&Drop המוכר ממערכות ההפעלה, נראה כיצד הוא ממומש בדפדפן ואילו בעיות חדשות הוא יוצר למפתחי האפליקציות.

Drag&Drop (DnD)

Drag&Drop היא הפעולה (או תמיכה בפעולה) של בחירה של אובייקט גרפי כלשהו על ידי "לחיצה" עליו, וגרירה שלו למיקום אחר או לאובייקט גרפי אחר (הגדרה פורמלית חצי מובנת).



במשך שנים השתמשנו בספריות כמו DOJO ו-JQuery בכדי ליצור מנגנוני DnD, כמובן שיש צורך במנגנונים שכאלו על מנת ליצור אפליקציות מעניינות ואינטרקטיביות. אך אין שום סיבה שכל מפתח שמתכוון להשתמש במנגנון נפוץ שכזה יצטרך להשתמש בספריות חיצוניות, לכן החליטו לכלול את המנגנון בגרסת HTML5.

מנגנון זה קיים כבר שנים רבות במערכות ההפעלה הנפוצות, וכיום אנו משתמשים בו בפעולות בסיסיות כמו למקם קבצים תמונות וטקסט בתוך מסמכי Word או להעביר קבצים מתיקיה לתיקיה. לפני שנתחיל להסביר את הבעייתיות במנגנון שכזה, נסביר כיצד משתמשים בו וקצת על כיצד הוא פועל מאחורי הקלעים.

ב-HTML5 פעולת DnD מוגדרת כסדרה של אירועים, קודם כל המשתמש לוחץ על האובייקט וגורר אירוע mousedown, לאחר מכן המשתמש "גורר" את האובייקט ומתקיימים מספר אירועי mousemove ולבסוף, המשתמש משחרר את לחצן העכבר ואירוע mouseup קורה. בכדי ליצור אובייקט שניתן לגרירה עלינו להוסיף לאובייקט את תכונת ה-draggable, מכשעשינו זאת ניתן ליצור EventListener המאזין לאירוע "dragstart". כעת כשיש לנו אובייקט שניתן לגרירה נרצה לדעת לאן גוררים אותו, נוכל להוסיף לאובייקט נוסף כלשהו את תכונת ה-dropzone, ולהאזין לאירוע drop. המידע המועבר בתהליך ה-DnD נקרא DDS קיצור של (drag data store) והוא מורכב (בין השאר) מהבאים:

- DDSIL (drag data store item list)
- סוג הגנה.

עבור כל פעולת DnD נוצרת רשימה של (drag data store item) DDSI, המידע ברשימה זו מסודר בצורת מחסנית, כך שהמידע האחרון שנוסף נמצא בראש הרשימה. כל DDSI מכיל את:

- סוג המידע המועבר (טקסט, קובץ - מידע בינארי).
- מחרוזת MIME הקובעת את הסוג המידע.
- המידע.

ישנם שלושה סוגי הגנה על הרשימה:

- **קריאה / כתיבה** - עבור אירוע ה-datastart. ניתן להוסיף מידע חדש לרשימה.
- **קריאה בלבד** - עבור אירוע ה-drop. ניתן לקרוא את המידע מן הרשימה.
- **מוגן** - עבור כל אירוע אחר. ניתן לקרוא את סוגי המידע ואת הפורמטים שלהם (בינארי או טקסט) אך המידע עצמו אינו חשוף ולא ניתן להוסיף מידע חדש.

ניקח לדוגמה מצב שכזה:

```
<span draggable="true" ondragstart="startDragFunc(event)">text</span>
<div dropzone="copy s:text/plain" ondrop="dropFunc(event,this)">DROP ZONE</div>
```

לא ניכנס יותר מידי לפרטים הטכניים של הקוד, אך ניתן לראות כי קיים אלמנט span הניתן לגרירה ואלמנט div המוגדר כ-dropzone, עבור כל אחד מאלה קיימת פונקציה המטפלת באירועים הרלוונטים. עבור תג ה-span קיימת הפונקציית startDragFunc(event) אשר תוסיף לרשימה DDSI חדש המכיל את הטקסט שגררנו ובעת "זריקה" ב-dropzone הפעולה dropFunc תוכל להוציא את המידע מתוך הרשימה. בין תחילת הגרירה לסיומה, כל אובייקט שנעבור מעליו יוכל לדעת שאנו גוררים מידע מסוג טקסט, אך לא ידע מהו הטקסט.

תוכלו לראות דוגמה לשימוש במנגנון זה בלינק הבא:

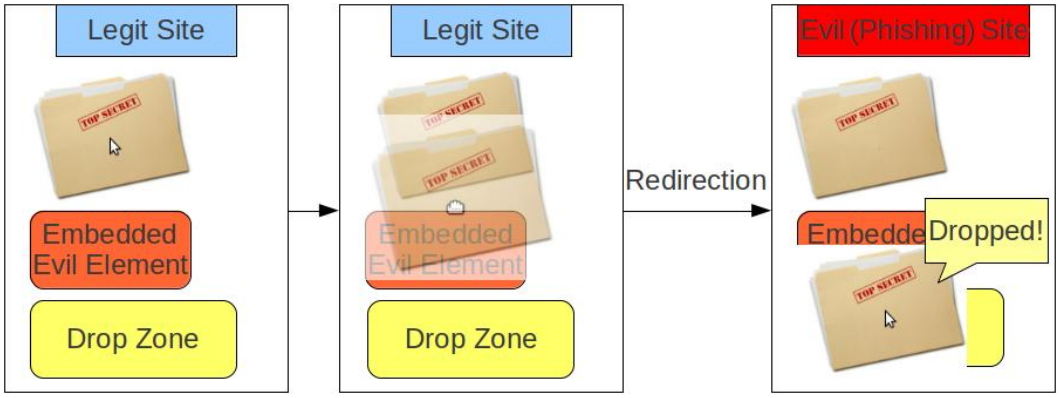
http://html5demos.com/drag

מנגנון ה-DnD יוצר כמה סוגים של בעיות, חלקן ניתן לפתור בעזרת מימוש נכון מצד המפתח, חלקן ניתן לפתור בעזרת מימוש נכון מצד הדפדפן ולחלקן אין פתרון כרגע.

דליפת מידע - 1

DnD יוצר כמה בעיות שעלולות לגרום לדליפת מידע לגורמים חיצוניים, למרות שמנגנון ההגנות על ה-DDSIL מונע מאובייקטים ש"גוררים" מעליהם לדעת מהו בדיוק המידע אותו גוררים הם עדיין יכולים לדעת את סוג המידע. כמו הרבה מקרים בעבר אני בטוח שגם את המידע הזה ילמדו לנצל לרעה, אך כרגע זאת נחשבת לבעיה הקטנה ביותר במנגנון זה.

ניקח לדוגמה מצב בו קיים עמוד אינטרנט לגיטימי המשתמש במנגנון ה-DnD בכדי להעביר מידע רגיש. תוקף מטמיע בעמוד אלמנט HTML זדוני בדרך כלשהי (למשל בתור פרסומת הנראית לגיטימית באתר המותקף), אשר בעת מעבר עליו מריץ קוד JS הגורם ל-redirection אל אתר phishing הנראה כמו העמוד הלגיטימי. משתמש תמים הנכנס לאתר, מתחיל לגרור את המידע הרגיש אל עבר ה-drop zone. בתחילת פעולת הגרירה המידע הרגיש מוסף ל-DDSIL. בעת ביצוע הגרירה המשתמש עובר מעל האלמנט הזדוני. המשתמש מקושר אל העמוד הזדוני. כעת, כשהמשתמש "יזרוק" את האובייקט ב-drop zone. התוקף יקבל את כל המידע הזדוני.



את הבעיה הזו ניתן לפתור גם מצד המפתח וגם בעזרת תכנון נכון של הדפדפן:

מצד המפתח - המפתח יכול לחסום כל redirection בטיפול באירוע ה-dragstart ולבטל את החסימה באירוע ה-drop, כמובן שבשביל לעשות זאת צריך להיות מודע לבעיה, וכל מפתח שלא מתמצה באבטחת מידע לא יטפל בה.

מצד הדפדפן - דפדפנים יכולים לפתור את הבעיה בקלות יחסית, כל שעליהם לעשות זה למחוק כל פעילות DnD קודמת לפני ביצוע redirection.

בנוסף, על דפדפנים למנוע התחלה או סיום של פעולת DnD על ידי סקריפט ולהתייחס בזהירות לכל סקריפט הרץ בזמן שפעולת DnD קוראת.

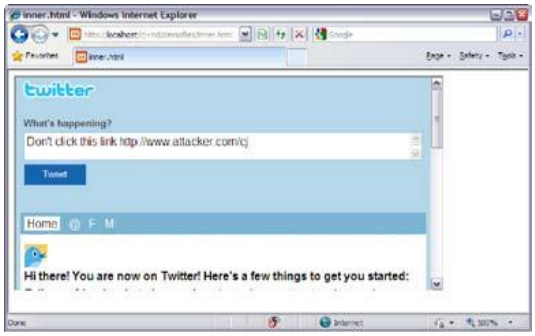
ניתן כמה רעיונות בכדי להבין את הבעיה:

- סקריפט עלול להאזין ללחיצת המשתמש על העכבר ולגרום להתחלת פעולת DnD על ידי הזזת חלון הדפדפן.
- סקריפט עלול לגרום לפעולת drop ללא כוונת המשתמש.

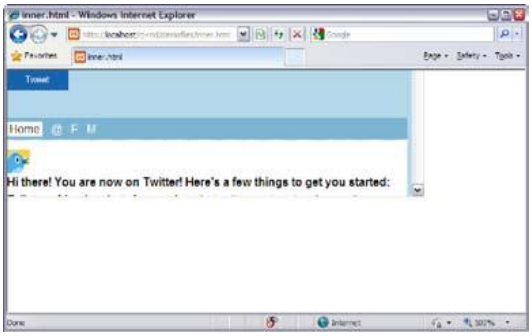
כמובן שהדפדפן גם צריך לממש את מנגנון ההגנה על ה-DDSIL בצורה נכונה, אחרת כל סקריפט הרץ בעמוד יוכל לקבל בקלות גישה לכל מידע רגיש העובר בפעולת ה-DnD.

Click Jacking

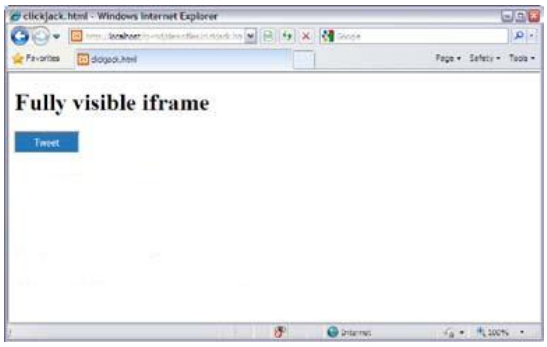
Click Jacking הינה שיטה המשמשת לביצוע התקפות Cross-domain על ידי "גניבת" קליקים שבוצעו בכוונה על ידי המשתמש בכדי לבצע פעולות שהוא לא התכוון אליהן. רובנו מכירים את סוג המתקפה הזו משצף הקבוצות בפייסבוק שקוראות לכם ללחוץ על הרבה כפתורים בכדי "לוודא שאתם לא רובוטים" ולחשוף בפניכם סרטון אמיתי לגמרי המציג המבורגר אוכל ילד בן 9. קודם כל נוצר עמוד בשם inner.html המכיל תג iframe ובתוכו את העמוד המותקף:



צריך לסדר את ה-iframe כך שהאלמנט שנרצה שהמשתמש ילחץ עליו יהיה בפינה השמאלית למעלה:



ולבסוף, נטען את העמוד inner.html אל תוך iframe קטן יותר המספיק להכיל בדיוק את הכפתור "Tweet":



תוכלו לראות דוגמה ל-Click Jacking בלינק הבא:

http://www.planb-security.net/notclickjacking/iframetrick.html

יש כמובן עוד שיטות רבות ומגוונות לבצע Click Jacking בסגנון זה, אך לא נרחיב בעניין זה.

Click Jacking בצורתו המוכרת הינו כלי חזק ויכול להיות מאוד שימושי בשילוב עם שיטות אחרות כגון CSRF ו-XSS, אך עדיין הוא יחסית מוגבל, הוא מאפשר לחיצה על כפתורים ו-check-boxes אך לא הוא לא מאפשר שינוי של שדות טקסט.

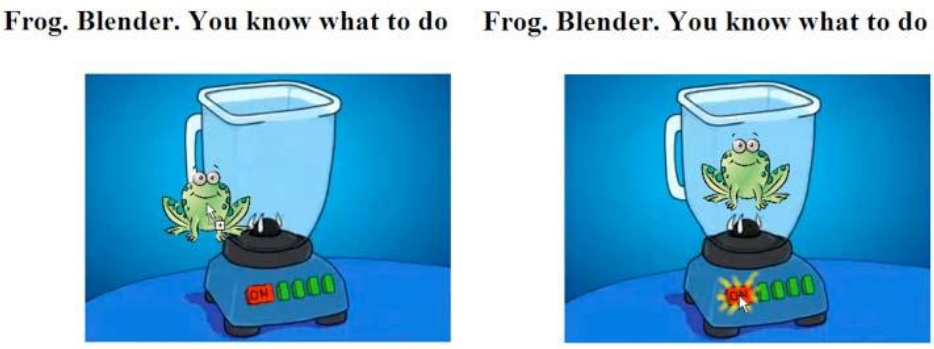
HTML5 ומנגנון ה-DnD שכבר היום ממומש בהרבה דפדפנים (גם אם לא בצורה מלאה), מאפשר בתוקף לבצע מניפולציה גם על שדות טקסט. תרגיל קטן שתוכלו לעשות כדי לקבל את הרעיון הוא לפתוח את הדפדפן החביב עליכם לסמן טקסט כלשהו עם העכבר ולגרור אותו אל תוך תיבת טקסט, עבד? עכשיו תחשבו מה אפשר לעשות עם זה...

נקודה שחשוב לציין היא ש-DnD אינו מוגבל על ידי "Same Origin Policy", אשר מונעת מאתר לגשת למידע השייך לדומיין אחר. דפדפנים מאפשרים זאת מכוון ובמימוש נכון, פעולת DnD מתחילה בפעולה של המשתמש ולא על ידי סקריפט כלשהו.

מהלך התקיפה הוא כזה:

1. עמוד זדוני משכנע משתמש לגרור אובייקט כלשהו על פני העמוד
2. כאשר גרירה מתחילה מבוצע שימוש ב-DnD API בכדי להוסיף את המידע הרצוי ל-DDSIL
3. כאשר המשתמש "זורק" את האובייקט המידע שנוסף ל-DDSIL יעבור אל iframe מוסתר שבו מוקם שדה טקסט, כך שהמידע יכנס לתוך שדה הטקסט. יש לזכור שבעל העמוד הזדוני שולט במידע זה.

לאחר שהתוקף גרם למשתמש להכניס את כל ערכי הטקסט הנחוצים הוא יוכל להשתמש ב-Click Jacking פשוט בכדי לגרום למשתמש ללחוץ על כפתור ה-Submit.



טכניקה זו יכולה להיות יעילה מאוד במקרים בהם התקפות CSRF ו-Click Jacking רגיל לא אפשריות. בנוסף, ניתן להשתמש בטכניקה בשילוב עם טכניקות אחרות בכדי ליצור מתקפות חדשות.

דליפת מידע - 2

בעקבות בעיות אבטחה שנתגלו בשימוש ב-iframes נוספו לכל דפדפן מנגנוני הגנה, אחת משיטות ההגנה היא לחסום עמוד מלקרוא תוכן של עמוד המוטמע בתוכו בעזרת תג iframe אם שני אלו אינם נמצאים באותו דומיין, הגנה זאת נקראת same-origin-policy (שהוזכרה כבר קודם). אך שיטה זאת, כמו שציינו מקודם, אינה מיושמת על מנגנון ה-DnD, זאת אומרת, שאפשר להשתמש במנגנון זה על מנת לקרוא מידע של עמוד המוטמע בעזרת תג iframe גם אם אותו עמוד לא נמצא באותו דומיין של העמוד המטמיע.

בכדי להתקיף אתר בצורה שכזאת, על התוקף לזהות איזה סוג של מידע הנמצא בעמוד המותקף יכול המשתמש לגרור, רוב הדפדפנים מאפשרים לגרור קישורים ותמונות בברירת מחדל. כאשר גוררים תמונה או לינק, ה-DDS (Drag Data Store) יכיל את ה-URL של התמונה או הלינק. בדרך כלל מידע שכזה הינו סטטי, אך לעתים כתובת ה-URL מכילה מידע רגיש כמו ID של מסמך, Token אבטחה, או מידע מזהה על המשתמש (לדוגמה השפה בה הוא רואה את האתר).

למרות שכתובות URL יכולות להיות מעניינות מאוד, בדרך כלל המידע הטקסטואלי הוא זה שיספק לנו את כמות המידע הרבה ביותר. ניתן לגרור כל אזור בעמוד על ידי סימון שלו, סימון נעשה באותו אופן שנעשית פעולת DnD, לחיצה על מקש העכבר השמאלי, גרירה מעל האזור אותו נרצה לסמן (בדרך כלל טקסט) ושחרור מקש העכבר.

בעזרת שימוש בשיטת ה-Click Jacking שהראנו קודם, תוקף יכול "לעבוד" על המשתמש כך שיבצע סידרה של לחיצות על המקש השמאלי, גרירות של העכבר ושחרור כך שהמשתמש, ללא ידיעתו, יסמן אזור באתר imbeingframed.com המוכל ב-iframe נסתר המוטמע בעמוד, ולאחר מכן יגרור את האזור המסומן לתיבת טקסט בעמוד התוקף.

הנה דוגמה פשוטה הממחישה אפשרות תקיפה שכזו:

```
<style>
  div,iframe,textarea{
      text-align:center;
      width:500px;
      position:absolute;
      left:0;
  }
  iframe,textarea{
      opacity:0.0;
      filter:alpha(opacity=0);
  }
  iframe:hover,textarea:hover{
      opacity:0.5;
      filter:alpha(opacity=50);
  }
</style>
<div style="top:0; z-index:0; height:500px; background-color:lightgreen">
  Try to select text from within the iframe and drag it to the textbox below
  <span style="position:absolute; left:150px; top:190px;">Move you mouse
here</span>
  <span style="position:absolute; left:150px; top:440px;">Move you mouse
here</span>
</div>
<iframe src="http://www.digitalwhisper.co.il/AboutUs" width="500px"
height="400px" style="top:0; z-index:1;"></iframe>
<textarea style="top:400; height:100px; z-index:1;"></textarea>
```

כמובן שבמקרה אמיתי יש צורך לשכנע את המשתמש לבצע את הפעולות שהוזכרו, לדוגמה על ידי משחק כמו שראינו קודם.

זוהי רק דרך אחת לבצע את ההתקפה, תוקף יותר מתוחכם יבין שכל מה שצריך זה שתי פעולות גרירה של המשתמש, לא משנה מאיפה ולאן, מכוון והתוקף יכול, בעזרת סקריפט כלשהו הרץ ברקע, לעקוב אחרי העכבר של המשתמש ולמקם את האובייקטים בהתאם לאירועים. סדר אירועים של תקיפה שכזו יכול להיות כזה:

1. תג iframe יעקוב אחרי סמן העכבר של המשתמש.
2. לאחר לחיצה שבוצעה על ידי המשתמש התוקף יגרום (שוב בעזרת סקריפט) לגלילה של ה-iframe.
3. ברגע שהמשתמש יזיז את העכבר, הסימון יתבצע.
4. לאחר מכן המשתמש ישחרר את מקש העכבר ויסיים את פעולת הסימון.

5. כעת פעולת לחיצה והזזת העכבר תגרום להתחלת פעולת גרירה של הטקסט שסומן בשלב הקודם.
6. התוקף יפסיק לעקוב אחרי סמן העכבר עם ה-iframe ובמקום, יעקוב אחרי הסמן עם תיבת טקסט.
7. ברגע שהמשתמש ישחרר את מקש העכבר המידע יכנס לתיבת הטקסט.
8. מכאן, התוקף יכול לשלוח לעצמו את המידע בעזרת בקשת AJAX או HTTP.

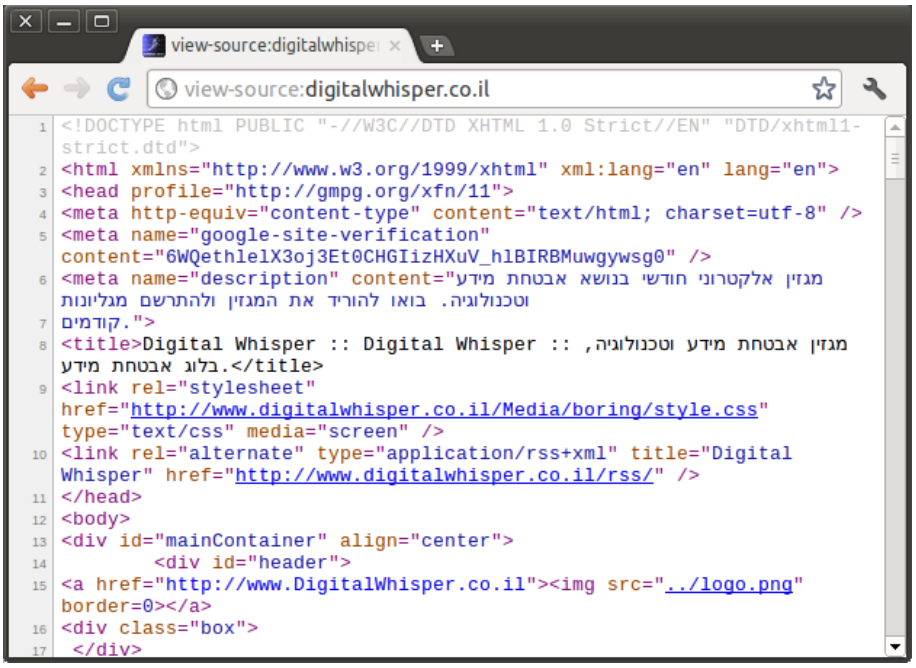
עד כה הסברנו מהי הבעיה, אבל עדיין לא הסברנו למה זו בעיה. הרי אם עמוד כלשהו נמצא באינטרנט הרי שגם התוקף יכול לראות אותו, בשביל מה הוא צריך אותנו?

נניח שאנחנו נכנסים לחשבון הבנק שלנו, אנו מחכים לסכום נכבד שיופקד לנו בחשבון כתשלום על עמלנו במשך החודש. בשלב מסוים נמאס לנו לבהות בעמוד שלא מתעדכן ואנו משאירים אותו פתוח בצד ופותחים משחק Angry Birds לדפדפן ב-Tab אחר. אך בעוד אנחנו מפציצים חזירים בציפורים זועמות אנו לא שמים לב כי נפתח iframe מוסתר בעמוד אשר מטמיע את עמוד הבנק שלנו בו, מכוון שכבר יש לנו session פתוח עם הבנק לא נצטרך לבצע כניסה עם שם משתמש וסיסמה ובעמוד (המוסתר) יופיע המידע שלנו. כמובן שאנו לא נראה את ה-iframe הזה, אך כאשר נגרור ציפורים על הרוגטקה ונירה בחזירים מסכנים שכל פשעם היה להעמיד כמה עצים בצורה מאוזנת אנו נסמן (ללא ידיעתינו) את הטקסט בעמוד הבנק שלנו ונגרור את התוכן אל תיבת טקסט מוסתרת, ובכך ניתן לתוקף גישה לכל הפיקדונות שנעשו בחשבונינו.

קוד מקור

עד כה השתמשנו בשיטה בכדי לקבל את התוכן של האתר: טקסט, תמונות, קישורים וכדו'... אך ניתן אפילו לקבל את קוד המקור! כלל הסקריפטים, הערות, ואפילו, בחלק מן הדפדפנים, מידע אשר מוסתר מן הגולש עצמו (למשל תיבת קלט מסוג hidden או אובייקט עם display:none).

איך עושים זאת? הדרך הקלה היא להשתמש באחת ממני הכלים שהדפדפן שלנו נותן לנו. נסו זאת בעצמכם, כנסו לכתובת view-source:http://digitalwhisper.co.il בדפדפנים Firefox או Chrome. מכאן והלאה, משתמשים באותה שיטה שהראנו. נכון שדפדפנים זה דבר נפלא?



(זה אפילו צבעוני!)

ישנה שיטה נוספת, אך עליה לא נרחיב, אתם מוזמנים לקרוא עליה במאמר למטה:

http://www.contextis.com/research/white-papers/clickjacking/ContextClickjacking_white_paper.pdf

אז איך מתגוננים?

אף על פי שבחלק זה הוצגו מספר טכניקות חדשות בחלק זה, מפני רובן ניתן להתגונן באותן דרכים בהם היינו מתגוננים מפני Click Jacking מסורתי.

Frame-busting הינה הטכניקה הראשונה שאיתה התמודדו עם התקפות Click Jacking. טכניקה זו מופעלת בצורה הבאה, עמוד מזהה שהוא נמצא בתוך iframe וינסה לטעון את עצמו מחוץ למסגרת ה-iframe במילים אחרות, ינסה לטעון את עצמו כעמוד הראשי ובעצם "לפרוץ מן המסגרת". כמובן שיש לקחת בחשבון שהעמוד הזדוני לא הולך לוותר בקלות ויכול לזהות שמישהו מנסה להשתלט עליו, לדוגמה הוא יכול להשתמש באירוע ה-onunload בכדי לקחת שליטה בחזרה על העמוד.

טכניקה אחרת להגנה מ-Click Jacking היא לנסות להסתיר את תוכן העמוד כאשר מזהים שהוא נמצא בתוך iframe.

שיטות אלו אף פעם אינן יעילות ב-100% וכרגע לא מטפלות במקרה של דליפת מידע (אתר זדוני עדיין יכול לקבל את קוד המקור של האתר למשל), לכן דפדפנים מימשו אפשרויות הגנה נוספות לדוגמה X-Frame-Options, אשר הוצגה לראשונה דווקא ב-Internet Explorer 8. דפדפני אינטרנט אשר תומכים ב-X-Frame-Options ימנעו מעמוד להיטען אל תוך iframe אם קיימת אצלו כותרת ה-X-Frame-Options. בכדי להגן על דפדפנים ישנים יותר אשר לא תומכים בשיטה זו ממולץ להשתמש גם בהגנות ה-JS הידועות בנוסף ל-Header.

המידע בחלק זה נלקח מן המאמר "Next Generation ClickJacking", לינק למאמר (מומלץ מאוד!):
http://www.contextis.com/research/white-papers/clickjacking/ContextClickjacking_white_paper.pdf

סיכום

במאמר זה סקרנו מנגנון חדש שהתווסף ל-HTML5 ומאפשר לבצע התקפות Click Jacking מסוג חדש וחזק הרבה יותר, ראינו ווריאציות שונות של המתקפה אך כמובן שניתן לשלב מתקפה זו ולשנות אותה בהרבה דרכים. נגענו במנגנוני הגנה מפני המתקפה, וראינו כי כיום מנגנון ההגנה החזק ביותר בעצם עוסק בביטול האפשרות להכניס את העמוד אל תוך iframe ובעצם פותר את הבעיה על ידי חסימה מוחלטת של אחד מן הגורמים היוצרים את הבעיה.

במאמרים הבאים נמשיך ונציג מגנונים חדשים שנוספו ל-HTML5 וכיצד ניתן לנצלם לרעה. נציג את מנגנוני ה-Storage השונים שנוספו לדפדפנים, נראה מה ההבדלים בין כל מנגנונים אלו, נדבר על תוספות כמו Websockets אשר מאפשר לפתוח socket מן הדפדפן תוך שימוש ב-JS וניתן כמה דוגמאות על איך ניתן לנצל מנגנון זה לרעה. נדבר על מנגנון ה-Geo-location המושמץ, ננסה להבין מה כל כך בעייתי בו, כיצד ניתן לנצלו לרעה, והאם ניתן להשיג את מיקומכם גם בלעדיו.

הנושאים הללו ועוד נושאים רבים שהשארנו לסוף יוצגו במאמרים הבאים.

לקריאה נוספת

- http://dev.w3.org/html5/spec/dnd.html
- http://www.html5rocks.com/en/tutorials/dnd/basics/
- http://blogs.msdn.com/b/ie/archive/2011/07/27/html5-drag-and-drop-in-ie10-ppb2.aspx
- https://wiki.mozilla.org/Firefox3.1/HTML5_drag_drop_Security_Review