Technologic Papers

The POODLE Attack

מאת שחר קורוט (Hutch)

הקדמה

בחודש אוקטובר האחרון, פרסמו שלושה מהנדסי אבטחת המידע של גוגל Bodo Möller, Thai Duong ו-Krzysztof Kotowicz מתקפה חדשה הנקראת POODLE (CVE-2014-3566). המתקפה למעשה מאפשרת לתוקף לגלות סוד הנמצא בבקשת לקוח העוברת בטווח תקשורת המוצפן באמצעות SSL. בכך, למשל, יכול תוקף לגנוב ממשתמש את ה-Cookie שלו גם אם העוגייה הוגדרה עם מאפייני אבטחה כמו Secure ו-HTTP Only. המתקפה תוקפת תקשורת מוצפנת מבוססת SSLv3.0, ומזכירה ברעיון שלה מתקפות מוכרות כמו ה-BEAST וה-CRIME. נזכיר שגירסא 3.0 שוחררה ב-1996 על ידי מהנדסי Netscape שפיתחו במקור את פרוטוקול ה-SSL.

מה החולשה?

השם POODLE הינו ראשי תיבות של **P**adding **O**racle **O**n **D**owngraded **L**egacy **E**ncryption. וכמו שהשם מרמז, המתקפה מבוססת על מניפולציות שהתוקף מבצע על ה-Padding בפרוטוקול. ה-**Padding** ("ריפוד") הינו ערך רנדומלי שנוסף למסר סודי שנשלח על ידי המשתמש כדי להקשות על תוקפים לפענח את התוכן. הריפוד נעשה על ידי אלגוריתם מוסכם לפני שלב ההצפנה, ולרוב אינו תהליך סודי. אפשר לשרשר מספר אקראי כלשהו למסר עצמו לפני ההצפנה כך שהוא מטשטש את המבנה המקורי של המסר. **Oracle** הוא פשוטו כמשמעו - נביא, כלומר גילוי התוכן המוצפן על ידי חולשה במנגנון ה-Padding (למי שרוצה להרחיב בנושא ממליץ בחום לקרוא את הכתבה של An7i, מבוא למתקפת Padding Oracle).

החולשה נמצא למעשה באופן שבו הפרוטוקול SSL 3.0 מבצע את תהליך ה-Padding ואינו חלק מהצפנת ה-MAC (Massage Authentication Code), לפיכך לא מתבצעת וולידציה בדבר אחר ערך ה-Padding והאם הוא זהה לערך שאותו שלח המשתמש.

על MAC מתוך ויקיפדיה:

> "**MAC**, הוא שם כולל לקבוצה של פונקציות עם מפתח סודי המשמשות לאימות (Authentication) והבטחת שלמות מסרים (Message Integrity). פונקציית MAC מקבלת מפתח סודי ומסר באורך שרירותי ומפיקה פיסת מידע קצרה הנקראת תג המשמשת לאימות וזיהוי מסר. אלגוריתם קוד אימות מסרים הינו סימטרי במובן שהשולח והמקבל חייבים לשתף ביניהם מראש מפתח סודי, באמצעותו המקבל יכול לוודא שהמסמך שקיבל אותנטי ושלא נעשה שינוי בתוכנו. ללא ידיעת המפתח לא ניתן לייצר את התג הנכון, ולכן אם נעשה שינוי כלשהו בהודעה, התוקף לא יצליח להתאים את התג בצורה שמתאימה להודעה ששונתה, והמקבל יבחין בשינוי."

משום שה-Padding אינו חלק מה-MAC, ניתן לערוך אותו וע"י כך לגלות מידע סודי הנמצא בבקשת ה-Request כמו למשל Secure & HTTP only Cookie.

מה ההתקפה?

הצפנת ה-SSL(3.0) עובדת כך שהלקוח לוקח את ההודעה הגלויה ומצפין אותה כ-MAC. לאחר מכן מוסיפים את ערך ה-Padding ומשלמים למספר הביטים הנדרש במסגרת ההצפנה (למשל AES128 משתמש במפתחות בגודל 16Byte) כאשר הביט האחרון ב-Padding משמש לספירת גודל ה-Padding. בשלב האחרון מצפין הלקוח את ההודעה באחת משיטות השונות של ההצפנה ב-SSL כך שכל חבילה נראת כך:

[((Message in Plain text)HMAC-SHA-1) + Padding]AES128

נראה איך זה נראה בפועל, נגיד והבקשה שלנו היא:

```
My secret password is So8dYAd14V. I want you to log me in.
```

לפני ההצפנה ההודעה שלנו נראת כך (בהקסדצימלי כאשר כל שורה היא 16 ביטים):

```
4d79 2073 6563 7265 7420 7061 7373 776f  My secret passwo
7264 2069 7320 536f 3864 5941 6431 3456  rd is So8dYAd14V
2e20 4920 7761 6e74 2079 6f75 2074 6f20  . I want you to
6c6f 6720 6d65 2069 6e2e 0abc 3a1f 0cf3  log me in.
02c5 80dd c869 66c0 1b2c 2a53 3c9d 5b00
```

כאשר הקטע המסומן בצהוב מ-BC עד 5b הינו ה-Tag של ה-HMAC-SHA-1 שהוסף אל ההודעה, ו-00 מסמל את גודל ה-Padding.

וכך נראת ההודעה לאחר ההצפנה:

```
33c5 3aa2 08a4 0ecf c408 c6df 0c7d ac47
c9fb a2e4 54c8 a316 78de 1ec2 cc6e 9600
0572 dcfd 25fc c941 e9fd 6ea9 9ca2 9e50
e4c3 5bc9 f4c6 d973 2412 03fb 1615 be93
7faf 2119 c740 becc 9095 c595 034a 6a61
```

במידה ונשנה את הביט האחרון בשורה הראשונה מ-47 ל-46, התוצאה לאחר Decrypt תראה כך:

```
860b 238a 9fc2 b909 6dd6 7642 05a8 85fe
7264 2069 7320 536f 3864 5941 6431 3457  rd is So8dYAd14W
2e20 4920 7761 6e74 2079 6f75 2074 6f20  . I want you to
6c6f 6720 6d65 2069 6e2e 0abc 3a1f 0cf3  log me in.
02c5 80dd c869 66c0 1b2c 2a53 3c9d 5b00
```

השורה הראשונה נהרסה משום שערכנו את המידע, אך אפשרי לראות עכשיו כי האות האחרונה בסיסמא השתנתה מ-V ל- W, הדבר נובע מכך שרוב שיטות ההצפנה של SSL3.0 עובדת בשיטת CBC. CBC עובד כך שלפני שלב ההצפנה בפועל כל בלוק מידע (16 ביטים כמו השורות שאנו עובדים איתם) עושה XOR עם הבלוק לפניו. (שימו לב כי השינוי מ-47 ל-46 הוא שינוי מתמתי כלשהו אלא עריכה שרירותית שלנו)

נחזור לדוגמא, נקח את ההודעה ונוסיף לה בלוק חדש של Padding:

```
4d79 2073 6563 7265 7420 7061 7373 776f  My secret passwo
7264 2069 7320 536f 3864 5941 6431 3456  rd is So8dYAd14V
2e20 4920 7761 6e74 2079 6f75 2074 6f20  . I want you to
6c6f 6720 6d65 2069 6e2e 0abc 3a1f 0cf3  log me in.
02c5 80dd c869 66c0 1b2c 2a53 3c9d 5b00
```

בעקבות כך שב-SSL3.0 ה-Padding הם ערכים רנדומלים ואין עליהם וולידציה, נוכל להשתמש באיזה ערכים שנרצה ל-Padding. נצפין את ההודעה:

```
33c5 3aa2 08a4 0ecf c408 c6df 0c7d ac47
c9fb a2e4 54c8 a316 78de 1ec2 cc6e 9600
3c87 de66 3db9 6961 3cee 12b2 0391 e2ba
e68c 5ff0 800c 72f7 78a6 78be 0866 826e
6889 b648 f1bd cbd7 294a 76b9 a51c 0632
08ab db46 cf99 bc60 c772 e3ce 3d15 c11b
```

ולאחר מכן נקח את המידע המוצפן ונעתיק את השורה בה מופיעה הסיסמא שלנו לסוף ההודעה במקום השורה האחרונה הנוכחית בתור ה-Padding שלנו כך:

```
33c5 3aa2 08a4 0ecf c408 c6df 0c7d ac47
c9fb a2e4 54c8 a316 78de 1ec2 cc6e 9600
0572 dcfd 25fc c941 e9fd 6ea9 9ca2 9e50
e4c3 5bc9 f4c6 d973 2412 03fb 1615 be93
c9fb a2e4 54c8 a316 78de 1ec2 cc6e 9600
```

אם נבצע Decrypt להודעה כעת נקבל את התוצאה הבאה:

```
4d79 2073 6563 7265 7420 7061 7373 776f  My secret passwo
7264 2069 7320 536f 3864 5941 6431 3456  rd is So8dYAd14V
2e20 4920 7761 6e74 2079 6f75 2074 6f20  . I want you to
6c6f 6720 6d65 2069 6e2e 0abc 3a1f 0cf3  log me in.
a562 4102 8f42 84d3 d87e 9c65 7e59 2682
```

מדוע 82?, בתהליך ההצפנה אנו עושים XOR עם הערך 93 שהוא בבלוק שמעל לבלוק הכתום. בהצפנה המקורית ה-XOR נעשה עם הערך 47 (ניתן להסתכל באיור הראשון המציג את המצב לאחר ההצפנה, בערך שבסוף השורה הראשונה). עכשיו באמצעות חישוב מתמטי פשוט נוכל לגלות מה הערך: במידה ו-X הוא האות האחרונה בסיסמא אותה אנחנו רוצים לגלות, נחשב:

$$X \oplus 47 \oplus 93 = 82 \underset{47 \oplus 93 = D4}{\Leftrightarrow} X \oplus D4 = 82 \underset{X \oplus A = Y \Leftrightarrow X = Y \oplus A}{\Leftrightarrow}$$

$$X = D4 \oplus 82 \Leftrightarrow X = 56$$

56 בהקסדצימלי משמעותו V, כפי שאנו יודעים זאת אכן האות האחרונה מהסיסמא המקורית שלנו.

[מקור התמונות: https://www.dfranke.us/posts/2014-10-14-how-poodle-happened.html]

כיצד מתבצעת ההתקפה?

כלפי תעבורת הלקוח. לרוב, יידרש התוקף MITM על מנת לבצע מתקפה זו תוקף צריך להיות במצב של , פאקטה זו מבקשת TLS_FALLBACK_SCSVלהוריד את פרוטוקול החיבור של המשתמש באמצעות על JavaScript, בשלב לאחר מכן התוקף יצטרך להריץ קובץ SSL 3.0מהשרת לעשות שימוש בפרוטוקול ה-הלקוח שיגרום לו לשלוח את אותה הודעה עם הסוד באופן קבוע

כפי שאנחנו יודעים בחיים האמיתיים אין לנו את המידע המפוענח שהשתמשננו בו קודם כדי לחשב את ערך הסוד, אז למעשה נצטרך להשתמש בטריק קצת שונה אך עם מתמתיקה דומה.

מכיוון של SSL 3.0 בודק את תקינות ה-Padding רק לפי גודל ה-Padding (Padding byte) נוכל לנחש את הערך שנמצא בבלוק שמעל הסיסמא. כלומר הערך ההקסדצימלי 93 יוחלף בערכים הקסדצימלים (מ-00 עד ff) עד שנקבל את גודל ה-Padding Byte המקורי, או במקרה שלנו 00. למשל, אם נחליף את הערך 93 ב-11, שזה 82 (מהמסר שפענחנו קודם) ⊕ 93 כך:

```
e4c3 5bc9 f4c6 d973 2412 03fb 1615 be11
c9fb a2e4 54c8 a316 78de 1ec2 cc6e 9600
```
→ Y

93⊕00⊕Y(00-ff)=Z(82)

93⊕ Y(00-ff)=Z(82)

כאשר Y הוא מי שהחלפנו ב-11, ו-Z הוא ערך שאינו ידוע. אך ברגע ש-Y ישתווה במשוואה ל-Z השרת אמור לאשר את ההודעה, ולא להוציא הודעת שגיאה, משום שה-Padding Byte לאחר Decrypt יהיה 00 כפי שהוא ציפה. ובכך אפשר לחשב את 82 אחורה ולפיכך לחשב את ערך הסיסמא כפי שראינו בדוגמא קודם:

$$X \oplus 47 \oplus 93 = Z \ ,Z=82$$

$$X \oplus 47 \oplus 93 = 82 \underset{47 \oplus 93 = D4}{\Leftrightarrow} X \oplus D4 = 82 \underset{X \oplus A = Y \Leftrightarrow X = Y \oplus A}{\Leftrightarrow}$$

$$X = D4 \oplus 82 \Leftrightarrow X = 56$$

למעשה ההתקפה על ה-Padding עובדת בעצם פעם אחת מתוך 256 פעמים כאשר 255 פעמים השרת יחזיר שגיאה, אך פעם אחת מתוך 256 הפעמים המתקפה תעבוד ונוכל לבצע את החישוב ה-XOR. כאשר שנרצה להתקדם כדי לגלות עוד חלק מהסוד בעזרת ה-Padding Byte, נצטרך לקדם את כל הסוד בביט אחד, נניח והסוד שלנו הוא ה-Cookie אנחנו צריכים להוסיף עוד תו אחד למשל אל ה-URL כדי שנוכל לחשוף את האות הבאה בעוגיה, למשל אם ביקשנו מהמשתמש לבקר בדף הבא למשל:

http://www.digitalwhisper.co.il/0x76/

לאחר שגילינו את האות הראשונה בסוד נשלח את המשתמש אל:

http://www.digitalwhisper.co.il/0x76/?

וכך נוכל להתקדם אל האות הבאה בסוד וכאשר נגלה גם את האות הזו נוסיף עוד ביט:

http://www.digitalwhisper.co.il/0x76/?D

וכן הלאה עד שנחשוף את הסוד כולו (כאשר כל בקשה כזו מוסיפה ביט אחד).

חדי ההבחנה יבחינו ש-SSL 3.0 שוחרר בשנת 1996 ובוודאי הרהרו לעצמם למה שמישהו ישתמש ב-SSL היום ולא ב-TLS גם מבחינת השרת וגם מבחינת הדפדפנים שעדיין תומכים. העיקרון שעומד מאחורי ההחלטה הוא תמיכה לאחור, כלומר אם המחשב שלי רוצה להגיע לאתר אינטרנט אך השרת ישן ואינו יודע לדבר כלל ב-TLS. אנו נדבר איתו ב-SSL בשביל למנוע מניעת שירות לאתר.

חשוב לציין שמשתמשים במערכת Windows ושרתי Windows עד גרסאות של Windows 8 בגרסאות ה-Desktop ו-2012 בגרסאות ה-Server עדיין תומכים ב-SLL 3.0 באופן ברירת מחדל. בנוסף לכך, TLS_FALLBACK, הינה למעשה חבילה בתוך TLS המבקשת לבצע downgrade לגרסא מוקדמת יותר של TLS או ל-SSL וכבר שומשה בעבר כבסיס למתקפת ה-Renegotiation של Marsh Ray.

החבילה למעשה נכתבה במקור כדי שלא ידרשו לכתוב Extensions ל-TLS על מנת לאפשר התקשורת אל מול שרתים ישנים. כאשר לקוח שולח אל שרת בקשת TLS_FALLBACK והוא מסתיר את האפשרות שלו להתחבר בעזרת TLS ומצהיר שהוא תומך רק ב-SSL.

תוכלו להשתמש באתר הבא כדי לראות האם השרת שלכם תומך ב-TLS_FALLBACK, הידוע לשמצה:
https://www.tinfoilsecurity.com/poodle

שימו לב כי יכול להיות שהאתר שלכם תומך ב-TLS_FALLBACK אך אינו תומך ב-SSL 3.0 מה שמבטל את האופציה למתקפה.

כיצד ניתן להתגונן?

כרגע ההנחייה הינה לבטל לגמרי את התמיכה ב-SSL 3.0 משם שהפרוטוקול אינו בר תיקון בתצורתו הנוכחית, ואין הצדקה משאבית לתקן אותו. בפיירפוקס הודיעו שלא יספקו תמיכה ב-SSL 3.0 מגרסא 34 של פיירפוקס. משתמשי Chrome, Safari יוכלו למצוא הסברים כיצד לבטל את SSL3.0 כאן (משתמשי Explorer תמצאו כאן תיקון אוטומטי, יפתור לכם עוד הרבה בעיות ☺)

תודות

תודה **לליאור ברש** שהכניס אותי לעולם אבטחת המידע ועל שנים של הדרכה והכוונה. תודה **לשייע פידמן** ותודה **לכל משפחת באגסק** שמלווה תומכת ומלמדת. תודה אחרונה ומיוחדת **לעידן כהן** שעזר לי לערוך את המאמר.

מקורות מידע וקישורים להמשך קריאה

- http://googleonlinesecurity.blogspot.in/2014/10/this-poodle-bites-exploiting-ssl-30.html
- https://en.wikipedia.org/wiki/Transport_Layer_Security#SSL_3.0
- https://www.openssl.org/~bodo/ssl-poodle.pdf
- http://thehackernews.com/2014/10/poodle-ssl-30-attack-exploits-widely_14.html
- https://www.dfranke.us/posts/2014-10-14-how-poodle-happened.html
- http://en.wikipedia.org/wiki/Padding_(cryptography)
- http://en.wikipedia.org/wiki/Message_authentication_code
- http://he.wikipedia.org/wiki/%D7%A7%D7%95%D7%93_%D7%90%D7%99%D7%9E%D7%95%D7%AA_%D7%9E%D7%A1%D7%A8%D7%99%D7%9D
- http://www.digitalwhisper.co.il/files/Zines/0x10/DW16-3-PaddingOracle.pdf
- http://www.jbisa.nl/download/?id=17683062
- http://www.uniroma2.it/didattica/netsec/deposito/4_tls3.pdf