Technologic Papers

חולשות ב-SSL

נכתב ע"י ישראל חורז'בסקי / Sro (AppSec Labs)

הקדמה

מאמר זה מיועד לכאלה ששמעו על המושג SSL, שמעו על המושג מפתח ציבורי, מפתח פרטי ומונחים דומים, ורוצים להכיר קצת ממגוון החולשות של SSL. במאמר זה ניגע בחולשות אלו בקצרה. לא יהיו פה אלגוריתמים, אותם ניתן להשיג בקלות בויקיפדיה. החלק הראשון נכתב באופן קצת סאטירי, אבל אל דאגה, בסיומו תצאו עם רשימת בעיות ב-SSL שאת חלקן מרבית האנשים לא מכירים. החלק השני כתוב בסגנון רציני יותר והוא מסכם בדיקות שביצעתי בשביל למצוא, נכון להיום, עד כמה החולשות מסוכנות.

קצת מונחים לפני הכל...

מי אתה מר SSL?

כינוי: SSL

ר.ת: Secure Socket Layer

שכבה: 6 במודל OSI

מה התפקיד שלך?

אני בא לפתור בעיית אבטחה.

מה הבעיה?

כאשר מתבצעת תקשורת בין מחשבים שונים על גבי הרשת (כמו למשל, בעת גלישה לאתר אינטרנט), ישנן שיטות רבות לחבל בתעבורה, ולגרום לה לעבור דרך מחשב שלישי - מחשב הנמצא ברשותו של התוקף. מתקפה זו שבה אנחנו מעבירים את התעבורה דרך מחשב הנמצא ברשותינו נקראת מתקפת Man In The Middle (או בקיצור "MITM").

תוכל למנות מספר דוגמאות למימוש MITM?

בהחלט! אמנה מספר שמות אך לא אכנס לעומק, במידה ותרצו, תוכלו לחפש עליהן בגוגל:

- **ARP Poisoning** - מתקפה אשר ניתן לבצע ברשתות LAN, במהלך המתקפה התוקף מרעיל את טבלת ה-ARP של הקורבן ושל הנתב, על מנת לגרום לכל אחד לחשוב שהוא השני. למידע נוסף:

http://en.wikipedia.org/wiki/ARP_spoofing

- **DNS Hijacking** - מתקפה אשר במסגרתה תוקף גורם לקורבן לחשוב כי כתובת IP של שרת הנמצא ברשותו הינה ה-Resolution של כתובת DNS מסויימת שאותה הקרבן מחפש. למידע נוסף:

http://en.wikipedia.org/wiki/DNS_hijacking

- **MAC flooding** - מתקפה אשר באמצעותה תוקף יכול לגרום למתג (Switch) לנתב אליו / לכלל הרשת מידע שהיה אמור להגיע לנמען (כתובת mac) ספציפי, למיע נוסף:

http://en.wikipedia.org/wiki/MAC_flooding

אוקי, אז המידע עובר דרך צד שלישי, מה הבעיה בזה?

כאשר מידע עובר לנמען דרך צד שלישי, הצד השלישי יכול לבצע במידע כרצונו - הוא יוכל לחשוף את המידע, להשתמש בו לרעה, לפגום במידע, לשנותו ולפגוע באמינותו, למנוע את העברתו ליעד המקורי, ולבצע בו עוד פעולות רבות.

וואלה, בעיה. אז איך אתה מציע לפתור את הבעיה?

על מנת למנוע ממקרים כאלה להתרחש, אני מציע להגביר את רמת האמינות של תווך התקשורת. לדוגמא, ע"י זיהוי הגורמים בשיחה ואימות זהותם מול גורם אמין, על ידי בדיקה כי המידע אשר נשלח ממחשבו של צד אחד אכן הגיע לצד השני בשלמותו וללא שום שינוי. אני מציע להצפין את המידע הנשלח באופן כזה שרק הגורמים בשיחה יוכלו לפענח את המידע מבלי שגורם שלישי יוכל לעשות זאת.

אבל היי, בשביל להצפין צריך שגם למחשב שלי וגם לשרת שאליו אני פונה יהיה מפתח הצפנה זהה, לא?

כן ולא! וזה בדיוק הנושא הבא שלנו ☺

דיפי הלמן ושות'

שם: אחד מאיתנו Diffie השני Hellman.

מקצוע: מתמטיקאים

אז מה גילו דיפי-הלמן ששווה להזכיר אותם?

דיפי והלמן מצאו שיטה שבה כל אחד יכול לבחור מספר (להלן: מפתח פרטי) לבצע עליו חישובים מסוימים ולפרסם את התוצאה (להלן: מפתח ציבורי). נניח כי דיפי רוצה לכתוב משהו להלמן, הוא לוקח את המפתח הציבורי שהלמן פרסם, ואת המפתח הפרטי שלו עצמו, ומבצע חישוב מתמטי שכולל את שני המפתחות האלה. תוצאת החישוב תהיה מספר חדש (להלן: מפתח ההצפנה), עם המפתח הזה הוא מצפין את התעבורה, ושולח את התעבורה המוצפנת להלמן. הלמן שמקבל את התעבורה המוצפנת מדיפי, לוקח את המפתח הפרטי שלו (של הלמן) ואת המפתח הציבורי של דיפי ומבצע את אותו חישוב. המעניין בחישוב הוא, שניתן גם באמצעות חישוב של המפתח הפרטי של הלמן והמפתח הציבורי של דיפי, וגם באמצעות המפתח הפרטי של דיפי והמפתח הציבורי של הלמן, להגיע לאותו מספר.

כעת, כאשר יש להלמן את מפתח ההצפנה, בקלות הוא יכול לפענח את התעבורה המוצפנת.

נו, אז מה הבעיה?

הבעיה? השאלה היא איך דיפי יידע מה המפתח הציבורי של הלמן, ואיך הלמן יידע מה המפתח הציבורי של דיפי.

שישאלו אחד את השני...

זה אכן מה שעושים, אבל אם הפעולה הזו מתבצעת במרחב האינטרנטי, באמצעות מתקפת MITM הנ"ל, ניתן "לעבוד" על שניהם ולפענח את התעבורה.

מה זאת אומרת?

אז ככה, יש לנו את המחשב של **דיפי** שפונה לשרת של **הלמן** ויש את ה**תוקף** שמאזין באמצע. כאשר **דיפי** פונה ל**הלמן** על מנת לקבל ממנו את המפתח הציבורי שלו, ה**תוקף** עוצר את הבקשה בדרך, מייצר מפתח פרטי ומפתח ציבורי משל עצמו ואומר ל**דיפי** (בשם **הלמן**) כי זה המפתח הציבורי שלו - (של **הלמן**). **דיפי** מצפין את התעבורה עם מפתח ההצפנה שהוא יוצר באמצעות המפתח שלו יחד עם המפתח הציבורי של ה**תוקף** כשהוא חושב שזה המפתח הציבורי של **הלמן**.

בינתיים התוקף לא נח, פונה ל**הלמן** ומזדהה כ**דיפי**, ואומר לו: "זה המפתח הציבורי שלי, תעביר לי בבקשה את המפתח הציבורי שלך כדי שנוכל לתקשר בצורה מאובטחת". **הלמן**, בטוח שהוא מדבר עם **דיפי** ומביא לו את המפתח הציבורי שלו - של **הלמן**, ולוקח מה**תוקף** את המפתח הציבורי של ה**תוקף**

בתור המפתח הציבורי של **דיפי**. **הלמן** לוקח את המפתח הפרטי שלו - של **הלמן** יחד עם הציבורי של ה**תוקף** ויוצר מפתח הצפנה.

כעת **דיפי** פונה ל**הלמן**, שולח תעבורה שמוצפנת למעשה עם מפתח ההצפנה **שלו** ושל ה**תוקף** (כל שילוב של אנשים יוצר מפתח שונה). **התוקף עוצר את התעבורה באמצע, מפענח אותה, קורא/משנה אותה, מצפין עם מפתח ההצפנה שהוא יצר עם הלמן ומעביר להלמן.**

ל**הלמן** אין אפשרות לדעת שיד התוקף בדבר, ו**הוא** יחזיר חזרה תעבורה ל**דיפי** דרך התוקף. **הוא** יצפין עם המפתח שהוא חושב שהוא של **דיפי**, התוקף יפענח ויצפין עם המפתח **שלו** עם **דיפי**.

כולם בטוחים שהכל בטוח, וזה בדיוק המתכון להתרסקות, Gave Over.

סיפור יפה, אבל... מישהו במרחב האינטרנטי מסכים להשתמש בשיטה הזו?

תתפלא, שרת WAMP תומך בהצפנה שכזו... היא נקראת ADH - Anonymous Diffie Hellman.

באמת?!

בעיקרון אל דאגה, אם הדפדפן שלך עדכני, הוא לא יסכים להתחבר בהצפנה הזו, ויגיד שיש בעיה עם ההצפנה.

יש אלגוריתמים נוספים?

אלגוריתם אחר שייך לשלישיית RSA, שלושה חוקרים שיצרו מנגנון שמזכיר את אלגוריתם החלפת המפתחות של דיפי-הלמן, אך יש בו שינוי מהותי. ההצפנה ב-RSA היא באמצעות המפתח הציבורי של הנמען נטו, בלי שום מפתח של השולח. כך שאם דיפי רוצה לשלוח משהו להלמן, הוא מצפין עם המפתח הציבורי של הלמן בלבד. דיפי יכול רק להצפין עם המפתח הציבורי של הלמן, הוא לא יכול לפענח חזרה את מה שהוא הצפין. רק הלמן שיש לו את המפתח הפרטי (שהמפתח הציבורי הוא תוצאה של חישוב שבוצע עליו) יכול לפענח את ההצפנה.

אז מה, מצפינים את כל התעבורה עם המפתח הציבורי? זה לא לוקח הרבה זמן?

לא, מצפינים רק את המפתח, את היתר מצפינים באמצעות מספר שיטות סימטריות אחרות - פעולה יותר פשוטה מבחינת משאבים.

סאטירי מה?

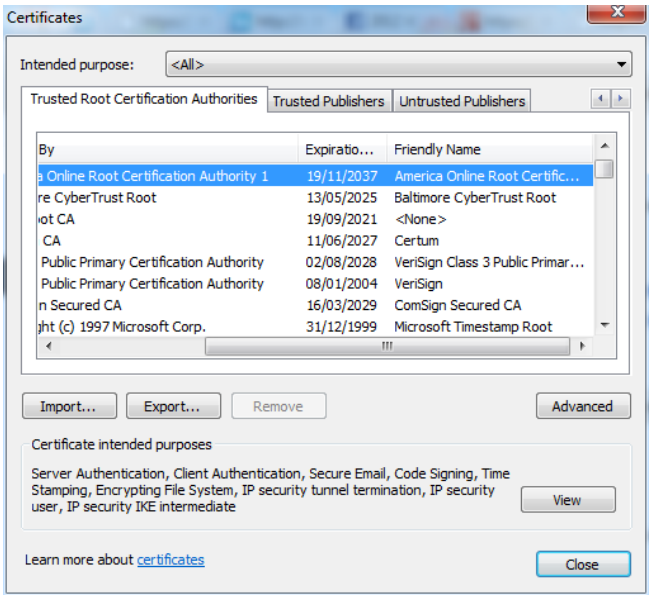
סימטרי. כל הצפנה נעשית באמצעות מפתח הצפנה. אם ניתן באמצעות אותו מפתח לפענח את ההצפנה (צפנים שכאלה הן המוכרות כמו א"ת-ב"ש, צופן קיסר וכן הלאה) הוא נקרא צופן סימטרי מכיוון שמפתח ההצפנה זהה למפתח הפענוח. אם באמצעות מפתח ההצפנה לא ניתן לפענח את ההצפנה (וזה מה שייחודי בצופן RSA) הוא נקרא א-סימטרי (א = לא), בצופן א-סימטרי חוץ ממפתח ההצפנה קיים מפתח שמשמש לפענוח המידע המוצפן.

אפשר סיכום ביניים קצר?

כן. עד כה למדנו על 2 אפשרויות להחליף מפתח הצפנה בין דיפי להלמן בשביל להצפין עם מפתח ההצפנה את התעבורה. בשיטת החלפת המפתחות של דיפי-הלמן, המפתח נוצר באמצעות חישוב המפתחות של שני הצדדים, ואיתו מצפינים את התעבורה באמצעות צופן סימטרי. בשימוש באלגוריתם RSA, הגולש מגריל מספר שישמש כמפתח הצפנה (במילים אחרות: בוחר מספר אקראי) מצפין אותו עם המפתח הציבורי של השרת, ושולח אותו לשרת. מכאן ואילך, התקשורת נעשית באמצעות מפתח ההצפנה שהגולש שלח והכל מוצפן בהצפנה סימטרית.

אז יש לנו שני אלגוריתמים, שניהם דורשים שהגולש יידע בוודאות את המפתח הציבורי של השרת כדי למנוע מתקפת MITM, איך אכן מוודאים שהמפתח לא זויף?

אה, נגעת בנקודה רגישה, אז ככה: מערכת ההפעלה באה עם רשימת מפתחות מובנית ששייכים לכל מיני חברות שנמצאות במדינות שונות ונתונות לחוקים של שלטונות שונים. כעת, כאשר דיפי רוצה לפנות לשרת של הלמן מה שהוא צריך לעשות זה לשאול את הלמן מה המפתח הציבורי שלו, לאחר מכן לשאול בצורה מוצפנת את אחד השרתים של החברות הנ"ל שנתונות תחת ממשלות שונות, האם המפתח שהוא קיבל מהלמן אכן של הלמן ואף אחד לא דחף את אפו לאמצע, אם הממשלה מאשרת, סימן שזה נכון. אני חוזר - אם הממשלה מאשרת, סימן שזה נכון. את ההמשך אנחנו יודעים, כעת כשיש לו את המפתח הציבורי של הלמן, הוא מצפין איתו מפתח הצפנה, שולח להלמן ואז עובר להצפנה סימטרית באמצעות מפתח ההצפנה ואיתה מצפין את יתר התעבורה. הלמן שזה עתה קיבל את מפתח ההצפנה שדיפי רוצה לתקשר באמצעותו, מצפין ומפענח בהתאם את התעבורה בינו לבין דיפי.



אז כל אחת מהחברות המקושרות יכולה לפענח את התעבורה כי אני סומך עליה?

זו חולשה נוספת במנגנון ה-SSL, כפי שהוא מיושם כיום, שלא רק שכל אחת מהחברות שהמפתח שלהן מותקן בברירת מחדל על המחשב בתור Root CA יכולה לפענח את כל התעבורה, (כמובן שבמדינה מתוקנת, זו עבירה על החוק, כיוון שהיא דורשת התערבות בתעבורה ומסירת מפתח פומבי מזויף) אלא שגם כל מי שגונב ממנה את המפתח הפרטי שלה יכול...

וזה קורה בימינו?

תשאל את חבר שלי (Google) לגבי Veri-sign.

נחזור לנושא ההצפנה, למה פתאום הכנסת הצפנה סימטרית גם בצופן RSA, למה שהגולש לא יעביר באמצעות הצפנה א-סימטרית את המפתח הציבורי שלו לשרת וכך כל אחד יצפין את התעבורה בצופן א-סימטרי עם המפתח הציבורי של השני?

Performance. להצפין ולפענח בשיטת הצפנה א-סימטרית דורשת הרבה יותר משאבים מאשר הצפנה בשיטה סימטרית.

יש עוד החלטות שקשורות לביצועים?

יאפ. אורך מפתח ההצפנה.

פרט, בבקשה.

יותר קל להצפין ולפענח באמצעות מפתח קצר, ולכן בתחילת הדרך אורך המפתחות היה 40 ביט, עד שראו שזה לא מספיק חזק בהתחשב באמצעי המחשוב המתקדמים, כיוון שהתוקף יכול לנסות את כל האפשרויות עד שהוא יצליח לפענח את התעבורה. לכן פיתחו תמיכה במפתחות ארוכים יותר.

והפסיקו את התמיכה במפתחות חלשים?

מה פתאום, יש 2 עקרונות "חשובים":

1. עובד? - אל תיגע!
2. תמיכה לאחור.

שני עקרונות חשובים אלה מוודאים שתמיד ניתן יהיה להתחבר עם מפתח חלש.

מריח כמו בעיית אבטחה...

כמובן. בתהליך ההצפנה הגולש שולח לשרת את הפרוטוקולים שהוא יודע לתקשר באמצעותם ומתוכם השרת בוחר באיזה להשתמש. אם הן השרת והן הגולש תומכים גם בפרוטוקול מאובטח וגם באחד חלש יותר, התוקף יכול לשנות את הרשימה שהגולש שולח ולהשאיר שם רק את הפרוטוקול החלש. השרת שיחשוב שזה הפרוטוקול היחיד שהגולש תומך בו, ישתמש בו להצפנת התעבורה. כעת מה שנותר לתוקף

זה לנסות את כל האפשרויות שיכולות להיות. אם נניח אורך המפתח הוא 40 ביט, יש 2^40 אפשרויות לערכו של המפתח. רק לשם הבנה, מדובר על:

```
1,099,511,627,776
```

אפשרויות. למפתח באורך 128 ביט יש (2^128):

```
340,282,366,920,938,463,463,374,607,431,768,211,456
```

אפשרויות. מינימום אורך המפתח שמקובל כיום כמאובטח הוא 128 ביט.

חזקות נוספות של המספר 2, ניתן למצוא כאן:

http://en.wikipedia.org/wiki/Power_of_two

במקרה והתוקף לא מתערב בשלב בחירת סוג ההצפנה, האם השרת בוחר את מפתח ההצפנה הכי ארוך / הכי קצר?

סוג ההצפנה ואורך המפתח הדיפולטיבי נקבע בשרת בהתאם לקנפוג, כמובן שניתן למצוא אתרים שמתעדפים מפתחות קצרים, כך שגם אם התוקף מגיע אחרי ה-Hand shake הוא יוכל לפצח/למצוא את מפתח ההצפנה עקב כך שהשרת בחר מפתח קצר ו/או פרוטוקול פגיע.

אוקיי, מה בדבר בעיות אבטחה?

RSA, דיפי והלמן, הכל טוב ויפה, אבל צריך לממש אותם, וכידוע - נדיר למצוא משהו בלי באגים. SSLv1 לא פורסם פומבית מעולם, אז אין מה להזכיר אותו. SSLv2 עדיין נתמך לצערנו בשרתים רבים למרות שנדיר למצוא דפדפנים / אפליקציות שלא תומכות ב-SSLv3.

אתה בעצם אומר ש-SSLv2 מכיל מספר בעיות אבטחה ולכן הוא פסול לגמרי?

אכן.

אז תן סיכום ביניים לגבי החולשות שראינו עד כה ב-SSL.

ובכן, כעת אתה אמור להבין את הרשימה הבאה:

- תמיכה בהחלפת מפתחות בשיטת ADH (Anonymous Diffie Hellman)
- תמיכה ב-SSLv2
- תמיכה במפתחות קצרים מ-128 ביט
- תעדוף של הצפנות חלשות

- וכמובן שאם מפתח פרטי של אחת מהחברות שמוגדרות במחשב האישי כאמינות בשביל לאמת מפתחות ציבוריים (מוכר גם כ-"תעודה" או "Certificate") דולף, ניתן לאמת באמצעותו כל תעודה בתור מקורית, כולל את של התוקף... הפתרון בכזה מקרה הוא להסיר במחשב את ה"אמון" בחברה הזו.

רשימה נאה, תביא עוד בבקשה...

אוקי. למעשה ב-SSL יש 2 הגנות, אחת מונעת קריאה של החומר (שמירה על Confidentially) ע"י גורם זר והיא ההצפנה המדוברת, השניה מונעת שינוי (שמירה על Integrity) והיא מתבצעת על ידי האשינג של חלקים / בלוקים של התעבורה. לא נאריך בנושא Hash, אני מניח ששמעת עליו. אלגוריתמים מוכרים שלו הם MD5, SHA1. במקרה הזה אין בעיה שמבוצע שימוש ב-MD5 כיוון שגם זיוף של הבלוק אולי יהיה לא קריא, אבל יהיה צורך בידיעת מפתח ההצפנה בשביל לעשות איתו משהו מעשי. בכל זאת מומלץ להשתמש ב-SHA1. גם RC4 (אלגוריתם הצפנה סימטרי) מכיל חולשות, אולם נכון להיום הוא עדיין מוגדר כחזק מספיק. למרות זאת, אם אתם נדרשים להישמע ל-FIPS, שני אלגוריתמים אלה (MD5 ו-RC4), כמו גם האלגוריתמים IDEA ו-Blowfish, נחשבים ל"ישנים" ולכן אין להשתמש בהם. בחלק של הצפנות סימטריות, רק האלגוריתמים 3DES-EDE ו-AES מאושרים ע"י FIPS. (למידע נוסף: http://csrc.nist.gov/publications/nistpubs/800-52/SP800-52.pdf).

בנוסף, בשביל מקרי דיבאגינג וסיבות שונות, שרתים תומכים ב-Clear text encryption, או אם תרצה סוג הצפנה: Null. זה אומר שהבלוקים לא עוברים הצפנה, אלא רק האשינג.

אתה לגמרי שופע הפתעות היום!

יאפ, שופע הפתעות תמיד. תעקוב מספיק תגלה 0-days מפעם לפעם...

לדוגמא?

BEAST attack, מתקפה שיצאה ב-2011 על פרוטוקולי הצפנה CBC (Cipher block chaining, בעברית מדוברת הכוונה להצפנה לפי בלוקים בגודל קבוע, להבדיל מהצפנה של Streaming שבה ההצפנה לא נעשית לפי אורכי בלוקים קבועים). במתקפה זו החוקרים ניצלו באג (שמאפשר לעקוף SOP) בפלאש שבדפדפן, שבאמצעותה יכלו ליצור ושוב ושוב בקשות HTTP מותאמות אישית לדומיין המותקף. כעת ניצלו באג בצורה שבה מומש CBC ב-SSLv3 ו-TLSv1 והצליחו באמצעות Brute Force (בעברית: כח גס, בעברית מדוברת: ניסוי כל האפשרויות) למצוא את כל פרטי הבקשה, כשמבחינתנו החלק המעניין בבקשה הוא תוכן הכותר Cookie (Header). הבאג בפלאש תוקן, הבאג בפרוטוקול עדיין לא, ב-TLS1.1 ומעלה הבעיה לא קיימת. TLS זה השם החדש של SSL, TLS1 מגיע אחרי SSL3.

אז תוסיף לרשימת הבעיות: שימוש בהצפנה עם CBC (יש להבדיל אותו מ-CBC3 שאינו פגיע).

יש עוד?

בהחלט! יש עוד הרבה בנושא SSL... אבל באמת הגיע הזמן לסיים. אז הנה אחת לסיום:

אמרנו שהסרבר מציג את המפתח הציבורי שלו, והגולש משתמש בתעודות שמוכרות אצלו במחשב (שייכות לחברות שונות) והוא פונה איתן לשרתים מסוימים בשביל לאמת את המפתח הציבורי / תעודת ה-SSL. בפועל, האימות לא נעשה לכל התעודה / מפתח ציבורי, אלא ל-Hash (גיבוב / ערבול)של מספר מאפיינים של האתר כמו שם דומיין (ובשביל אבשלום קור: שם מתחם), תאריך תפוגה, המפתח הציבורי ועוד.

אמרת Hash, תן לי לנחש, החולשה קשורה ל-MD5...

אכן, גם MD5, וגם אורך התעודה. אם האורך קצר (מתחת ל-1024 ביט, ומאז 2010 [לפי NIST] גם 1024 נחשב לקצר ועל תעודה להיות ארוכה יותר) ו/או שיטת החתימה (Hash) הינה MD5, התוקף יוכל (בעזרת כח מחשוב לא קטן) למצוא תעודה משלו, כך שכאשר תתבצע עליה פעולת ה-Hashing הפלט שיוחזר יהיה זהה ל-Hash של התעודה המקורית (תופעה המכונה "התנגשות", או באנגלית: Collision) וכך התוקף יוכל "לחקות" את התעודה המקורית מבלי שתהיה לו אותה במציאות. התנגשויות הן תופעות מאוד נדירות, בייחוד באלגוריתמים אלו. על מנת לאתרן יש צורך בכח חישוב עצום! עם זאת, פורסמו בעבר מספר מקרים כאלו, כדוגמת:

http://www.mathstat.dal.ca/~selinger/md5collision/

חולשות, חולשות, האם הן עדיין קיימות?

כפי שראינו קיימות מספר חולשות ב-SSL, בעיקר מדובר בשימוש בפרוטוקולים / הצפנות שנמצאו פגיעים ו/או אורך מפתחות קצר מדי. השלב העיקרי של המחקרון שלי היה לבדוק האם הפרוטוקולים האלה עדיין נתמכים, והאם הן אכן מסוכנים.

מבחינת מה שבדקתי, לקחתי כתשתית WAMP 2.2 (הגרסה הכי עדכנית, נכון להיום) שמכיל Apache 2.4.2. במקביל לקחתי גם Windows server 2008 שמגיע עם IIS7 (יצויין שכבר קיים IIS 8, אבל הוא חדש ונפוץ בערך כמו Windows 8).

התקנות:

על מנת להתקין SSL על Wamp, ניתן להשתמש במדריך המצוין:

http://forum.wampserver.com/read.php?2,32986

כדי להתקין SSL על IIS7, ניתן להשתמש במדריך המצוין גם הוא (אני משתדל להשתמש רק במדריכים מצויינים...):

https://learn.iis.net/page.aspx/144/how-to-set-up-ssl-on-iis/

קנפוג:

ב-WAMP, תקנפגו את ה-SSL של Apache בקובץ:

```
wamp\bin\apache\apache2.4.2\conf\extra\httpd-ssl.conf
```

בשורה SSLCipherSuite. סוגי ההצפנות המוזכרות שם מופרדות באמצעות נקודתיים. אם בתחילת השם מופיע סימן קריאה זה אומר לא לאפשר את ההצפנה המוזכרת. לדוג' השורה הבאה, מורה לשרת לאפשר את ההצפנות המוגדרות בקטגורייה Low ולא לאפשר את שיטות ההצפנה המשתמשות ב-MD5:

```
SSLCipherSuite LOW:!MD5
```

בווינדוס מגדירים הכל ב-Registry, אפשר לחפור לבד (לינק לא שימושי, אני מביא אותו רק בשביל הפרוטוקול - http://support.microsoft.com/kb/245030) או להשתמש בכלי גרפי:

https://www.nartac.com/Products/IISCrypto/Default.aspx

De facto

על מנת לבדוק מהן הפרוטוקולים שהשרת מאפשר בפועל, נשתמש בכלי sslscan:

Linux version - http://sourceforge.net/projects/sslscan/

Windows version - https://code.google.com/p/sslscan-win/

דוגמא לסריקה של localhost, והצגת רק ההצפנות המאופשרות:

```
sslscan.exe --no-failed localhost
```

דוגמא לפלט:

ההצפנות שברשימה המוצגת כולן משתמשות בפרוטוקול SSLv3 (מאובטח יחסית), השרת מאפשר הצפנות עם מפתחות קצרים מ-128 ביט (לא מאובטח), הצפנות שמשתמשות ב-CBC (לא מאובטח), והצפנות שמשתמשות ב-ADH (לא מאובטח).

בצד ימין זו רשימת האלגוריתמים שההצפנה משתמשת בהם. לדוג', השורה הראשונה - ADH, החלפת מפתחות אנונימית (אין אימות לתעודה) בשיטת דיפי-הלמן, לאחר מכן עם המפתח מצפינים את יתר התעבורה בהצפנה סימטרית עם האלגוריתם המכונה AES256, והחתימה שמיועדת לאיתור ומניעת שיבושים בתעבורה היא SHA.

אז דבר ראשון לא נגעתי בשרתים, ובדקתי מה הם מאפשרים בברירת מחדל.

ב-Apache זו ההגדרה:

```
SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5
```

ב-IIS7, פשוט אין מפתחות ברג'יסטרי עם שמות ההצפנות.

והתוצאה (באדום מסומנים הבעייתיים):

IIS7	APACHE 2.4.2
Supported Server Cipher(s): Accepted SSLv2 168 bits DES-CBC3-MD5 Accepted SSLv2 128 bits RC4-MD5 Accepted SSLv3 168 bits DES-CBC3-SHA Accepted SSLv3 128 bits RC4-SHA Accepted SSLv3 128 bits RC4-MD5 Accepted TLSv1 256 bits AES256-SHA Accepted TLSv1 128 bits AES128-SHA Accepted TLSv1 168 bits DES-CBC3-SHA Accepted TLSv1 128 bits RC4-SHA Accepted TLSv1 128 bits RC4-MD5 **Prefered Server Cipher(s):** SSLv2 168 bits DES-CBC3-MD5 SSLv3 128 bits RC4-SHA TLSv1 128 bits AES128-SHA	**Supported Server Cipher(s):** Accepted SSLv3 256 bits DHE-RSA-AES256-SHA Accepted SSLv3 256 bits AES256-SHA Accepted SSLv3 128 bits DHE-RSA-AES128-SHA Accepted SSLv3 128 bits AES128-SHA Accepted SSLv3 168 bits EDH-RSA-DES-CBC3-SHA Accepted SSLv3 168 bits DES-CBC3-SHA Accepted SSLv3 128 bits IDEA-CBC-SHA Accepted SSLv3 128 bits RC4-SHA Accepted TLSv1 256 bits DHE-RSA-AES256-SHA Accepted TLSv1 256 bits AES256-SHA Accepted TLSv1 128 bits DHE-RSA-AES128-SHA Accepted TLSv1 128 bits AES128-SHA Accepted TLSv1 168 bits EDH-RSA-DES-CBC3-SHA Accepted TLSv1 168 bits DES-CBC3-SHA Accepted TLSv1 128 bits IDEA-CBC-SHA Accepted TLSv1 128 bits RC4-SHA **Prefered Server Cipher(s):** SSLv3 256 bits DHE-RSA-AES256-SHA TLSv1 256 bits DHE-RSA-AES256-SHA
תומך ב-SSLv2. אבל אפשר לדון אותו לקו זכות, שזה שרת שיצא לפני 4 שנים, בכל זאת... סה"כ, ממש בטוח. והי, הוא לא תומך ב-CBC.	יחסית למצב גרוע יותר, זה סביר...

ננתח את הנתונים הנ"ל באמצעות רשימת הבעיות המוכרות לנו, ונראה איזה מהם מצאנו קיימים בברירת מחדל:

חולשה	בברירת מחדל ב-Apache 2.4.2	בברירת מחדל ב-IIS7
תמיכה ב-SSLv2	לא פגיע	**פגיע**
תמיכה ב-ADH	לא פגיע	לא פגיע
תמיכה במפתחות קצרים מ-128 ביט	לא פגיע	לא פגיע
תעדוף (preferred) של הצפנות חלשות	לא פגיע	לא פגיע
Beast attack	**פגיע**	לא פגיע
Null/Clear-text encryption	לא פגיע	לא פגיע

עכשיו נראה מה קורה אם מישהו מחליט להפעיל בשרתים תמיכה בכל ההצפנות, עד כמה אחראי הרשת יכול לירות לעצמו ברגל:

APACHE 2.4.2	IIS7
Supported Server Cipher(s):	**Supported Server Cipher(s):**
Accepted SSLv3 256 bits ADH-AES256-SHA	Accepted SSLv2 168 bits DES-CBC3-MD5
Accepted SSLv3 256 bits DHE-RSA-AES256-SHA	Accepted SSLv2 128 bits RC4-MD5
Accepted SSLv3 256 bits AES256-SHA	Accepted SSLv3 168 bits DES-CBC3-SHA
Accepted SSLv3 128 bits ADH-AES128-SHA	Accepted SSLv3 128 bits RC4-SHA
Accepted SSLv3 128 bits DHE-RSA-AES128-SHA	Accepted SSLv3 128 bits RC4-MD5
Accepted SSLv3 128 bits AES128-SHA	Accepted TLSv1 256 bits AES256-SHA
Accepted SSLv3 168 bits ADH-DES-CBC3-SHA	Accepted TLSv1 128 bits AES128-SHA
Accepted SSLv3 56 bits ADH-DES-CBC-SHA	Accepted TLSv1 168 bits DES-CBC3-SHA
Accepted SSLv3 40 bits EXP-ADH-DES-CBC-SHA	Accepted TLSv1 128 bits RC4-SHA
Accepted SSLv3 128 bits ADH-RC4-MD5	Accepted TLSv1 128 bits RC4-MD5
Accepted SSLv3 40 bits EXP-ADH-RC4-MD5	
Accepted SSLv3 168 bits EDH-RSA-DES-CBC3-SHA	**Prefered Server Cipher(s):**
Accepted SSLv3 56 bits EDH-RSA-DES-CBC-SHA	SSLv2 168 bits DES-CBC3-MD5
Accepted SSLv3 40 bits EXP-EDH-RSA-DES-CBC-SHA	SSLv3 128 bits RC4-SHA
Accepted SSLv3 168 bits DES-CBC3-SHA	TLSv1 128 bits AES128-SHA
Accepted SSLv3 56 bits DES-CBC-SHA	

	Accepted SSLv3 40 bits EXP-DES-CBC-SHA Accepted SSLv3 128 bits IDEA-CBC-SHA Accepted SSLv3 40 bits EXP-RC2-CBC-MD5 Accepted SSLv3 128 bits RC4-SHA Accepted SSLv3 128 bits RC4-MD5 Accepted SSLv3 40 bits EXP-RC4-MD5 Accepted TLSv1 256 bits ADH-AES256-SHA Accepted TLSv1 256 bits DHE-RSA-AES256-SHA Accepted TLSv1 256 bits AES256-SHA Accepted TLSv1 128 bits ADH-AES128-SHA Accepted TLSv1 128 bits DHE-RSA-AES128-SHA Accepted TLSv1 128 bits AES128-SHA Accepted TLSv1 168 bits ADH-DES-CBC3-SHA Accepted TLSv1 56 bits ADH-DES-CBC-SHA Accepted TLSv1 40 bits EXP-ADH-DES-CBC-SHA Accepted TLSv1 128 bits ADH-RC4-MD5 Accepted TLSv1 40 bits EXP-ADH-RC4-MD5 Accepted TLSv1 168 bits EDH-RSA-DES-CBC3-SHA Accepted TLSv1 56 bits EDH-RSA-DES-CBC-SHA Accepted TLSv1 40 bits EXP-EDH-RSA-DES-CBC-SHA Accepted TLSv1 168 bits DES-CBC3-SHA Accepted TLSv1 56 bits DES-CBC-SHA Accepted TLSv1 40 bits EXP-DES-CBC-SHA Accepted TLSv1 128 bits IDEA-CBC-SHA Accepted TLSv1 40 bits EXP-RC2-CBC-MD5 Accepted TLSv1 128 bits RC4-SHA Accepted TLSv1 128 bits RC4-MD5 Accepted TLSv1 40 bits EXP-RC4-MD5 **Prefered Server Cipher(s):** SSLv3 256 bits ADH-AES256-SHA TLSv1 256 bits ADH-AES256-SHA

שימו לב לפרוטוקולים המומלצים. למרות זאת, זה לא משנה הרבה כי ב-ADH, אם התוקף מתערב בתעבורה אחרי שהתחילה התקשורת זה מאוחר מדי - הצדדים כבר החליפו מפתחות. ואם הוא מתערב בתעבורה לפני שהתחילה התקשורת, הוא יכול לעקוף את הגדרות ההעדפה שבשרת כמו שהוזכר לעיל.	לא טעיתם, הרשימה הזו זהה לחלוטין לרשימה הקודמת. בברירת מחדל ב-IIS7 כל האפשרויות הנתמכות מופעלות...

ושוב ננתח את הנתונים הנ"ל באמצעות רשימת הבעיות המוכרות לנו, ונראה איזה מהם מצאנו שאפשריים בשרתים הללו:

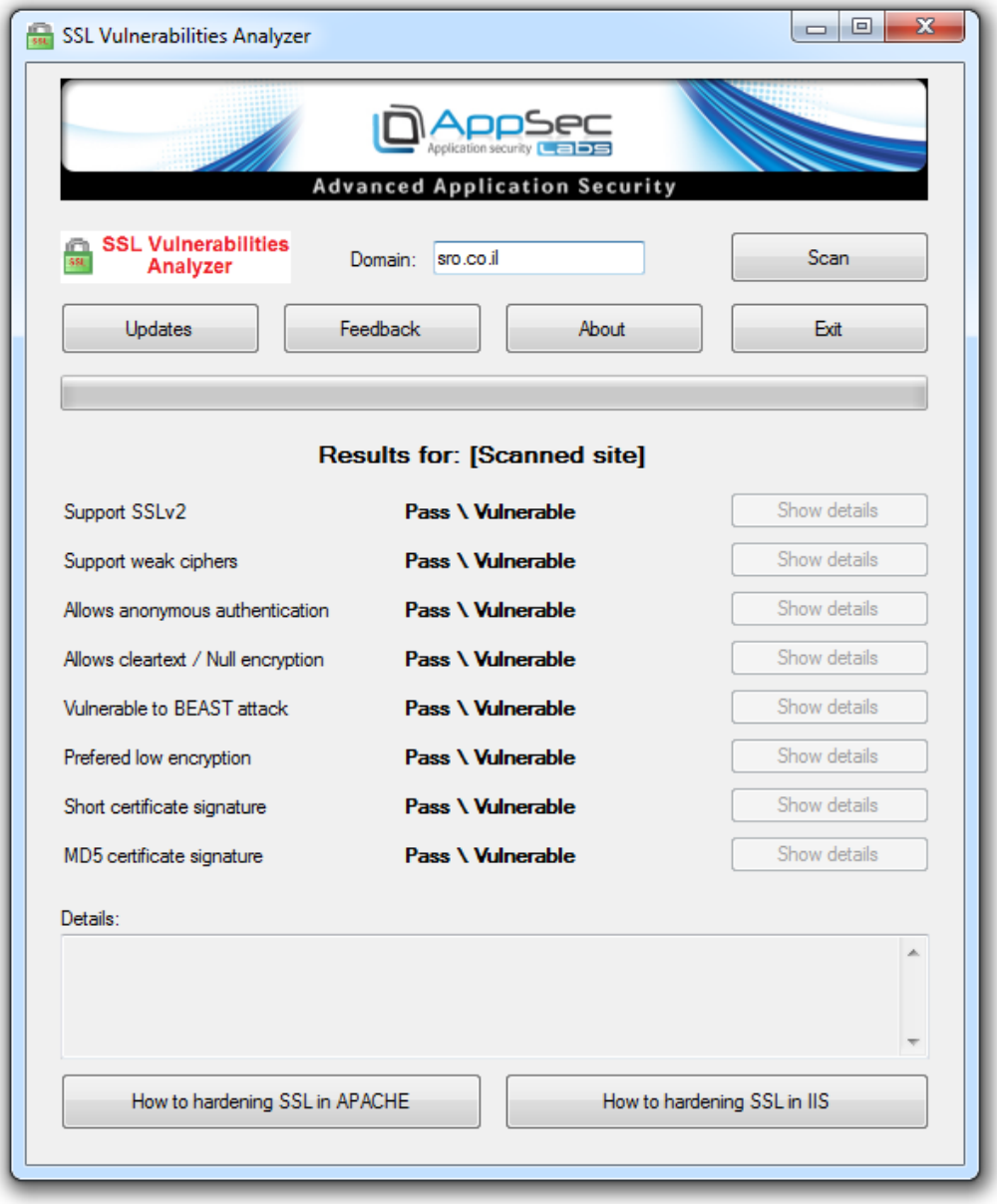
חולשה	אפשרי ב-Apache 2.4.2	אפשרי ב-IIS7
תמיכה ב-SSLv2	לא	**כן**
תמיכה ב-ADH	**כן**	לא
תמיכה במפתחות קצרים מ-128 ביט	**כן**	לא
תעדוף (preferred) של הצפנות חלשות	**כן**	לא
Beast attack	**כן**	לא
Null/Clear-text encryption	לא	לא

ברשימת ההצפנות שנתמכות בשרתים אחרים ו/או גרסאות ישנות יותר, ניתן לצפות כאן:

http://www9.atwiki.jp/kurushima/pub/pkimisc/SSLTLS_CipherSuite_Support_Table_.html

SSL vulnerabilities analyzer tool

השלב הבא הוא כתיבת כלי שמבצע לכל הניתוח הנ"ל אוטומציה ומציג תוצאה ברורה, אז עשיתי את זה...
כתבתי כלי גרפי נאה, שמקבל דומיין / IP ומנתח בצורה ברורה את הצפנות ה-SSL שהוא תומך בהם:



הכלי ניתן להורדה בכתובת:

https://appsec-labs.com/SSL_Analyzer

עד כמה אותם החולשות מסוכנות?

לאחר שסיימתי את השלב הראשון - לימוד החולשות, השלב השני - כתיבת כלי (חוקי, חוקי, אני פנטסטר), פניתי לבדוק עד כמה החולשות אכן מסוכנות.

הגדרתי את השרתים שיתמכו רק בהצפנות פגיעות, ב-IIS זה היה ע"י חסימה של SSL3 ו-TLS, והפעלת תמיכה ב-SSL2, הכל ברג'יסטרי. ב-APACHE הגדרתי בקובץ httpd-ssl.conf:

```
SSLCipherSuite ADH-AES256-SHA:ADH-AES128-SHA:ADH-DES-CBC3-SHA:ADH-DES-
CBC-SHA:EXP-ADH-DES-CBC-SHA:ADH-RC4-MD5:EXP-ADH-RC4-MD5:EDH-RSA-DES-CBC-
SHA:EXP-EDH-RSA-DES-CBC-SHA:DES-CBC-SHA:EXP-DES-CBC-SHA:IDEA-CBC-
SHA:EXP-RC2-CBC-MD5:EXP-RC4-MD5:ADH-AES256-SHA:ADH-AES128-SHA:ADH-DES-
CBC3-SHA:ADH-DES-CBC-SHA:EXP-ADH-DES-CBC-SHA:ADH-RC4-MD5:EXP-ADH-RC4-
MD5:EDH-RSA-DES-CBC-SHA:EXP-EDH-RSA-DES-CBC-SHA:DES-CBC-SHA:EXP-DES-CBC-
SHA:IDEA-CBC-SHA:EXP-RC2-CBC-MD5:EXP-RC4-MD5:DES-CBC3-MD5:RC4-MD5
```

פתחתי Chrome, FF, IE, Opera וגלשתי לשרתים ב-https. כפי שניתן לראות בטבלה למעלה, כל הדפדפנים שנבדקו הציגו חזית אחידה וחסמו את כל הפרוטוקולים הפגיעים. כרום ופיירפוקס הציגו שגיאה מובנת, IE ו-Opera הציגו דף דיפולטיבי והשגיאה הופיעה רק ב-title.

הצפנה	נתמך ב-IIS7?	נתמך ב-Apache2.4.2?	דפדפנים פגיעים?
Accepted SSLv3 256 bits ADH-AES256-SHA	X	V	X
Accepted SSLv3 128 bits ADH-AES128-SHA	X	V	X
Accepted SSLv3 168 bits ADH-DES-CBC3-SHA	X	V	X
Accepted SSLv3 56 bits ADH-DES-CBC-SHA	X	V	X
Accepted SSLv3 40 bits EXP-ADH-DES-CBC-SHA	X	V	X
Accepted SSLv3 128 bits ADH-RC4-MD5	X	V	X
Accepted SSLv3 40 bits EXP-ADH-RC4-MD5	X	V	X
Accepted SSLv3 56 bits EDH-RSA-DES-CBC-SHA	X	V	X
Accepted SSLv3 40 bits EXP-EDH-RSA-DES-CBC-SHA	X	V	X
Accepted SSLv3 56 bits DES-CBC-SHA	X	V	X
Accepted SSLv3 40 bits EXP-DES-CBC-SHA	X	V	X
Accepted SSLv3 128 bits IDEA-CBC-SHA	X	V	X
Accepted SSLv3 40 bits EXP-RC2-CBC-MD5	X	V	X
Accepted SSLv3 40 bits EXP-RC4-MD5	X	V	X

X	V	X	Accepted TLSv1 256 bits ADH-AES256-SHA
X	V	X	Accepted TLSv1 128 bits ADH-AES128-SHA
X	V	X	Accepted TLSv1 168 bits ADH-DES-CBC3-SHA
X	V	X	Accepted TLSv1 56 bits ADH-DES-CBC-SHA
X	V	X	Accepted TLSv1 40 bits EXP-ADH-DES-CBC-SHA
X	V	X	Accepted TLSv1 128 bits ADH-RC4-MD5
X	V	X	Accepted TLSv1 40 bits EXP-ADH-RC4-MD5
X	V	X	Accepted TLSv1 56 bits EDH-RSA-DES-CBC-SHA
X	V	X	Accepted TLSv1 40 bits EXP-EDH-RSA-DES-CBC-SHA
X	V	X	Accepted TLSv1 56 bits DES-CBC-SHA
X	V	X	Accepted TLSv1 40 bits EXP-DES-CBC-SHA
X	V	X	Accepted TLSv1 128 bits IDEA-CBC-SHA
X	V	X	Accepted TLSv1 40 bits EXP-RC2-CBC-MD5
X	V	X	Accepted TLSv1 40 bits EXP-RC4-MD5
X	X	V	Accepted SSLv2 168 bits DES-CBC3-MD5
X	X	V	Accepted SSLv2 128 bits RC4-MD5

הדפדפנים היו כמעט כולם גרסאות עדכניות. כך שאם הדפדפן שלכם מעודכן, אתם מוגנים ממנהלי רשת לא אחראיים.

נ.ב. למי שממש רוצה שהדפדפן שלו יתמוך בפרוטוקולים חלשים, ניתן לבצע זאת בדרך כלל. לדוגמא:

http://www.techbug.com/en/knowledge-base/firefox-cant-connect-securely-to-url-because-the-site-uses-a-security-protocol-which-isnt-enabled/

לינקים לקריאה נוספת

SSL Protocol: http://en.wikipedia.org/wiki/Transport_Layer_Security

SSL Protocol: http://technet.microsoft.com/en-us/library/cc767139.aspx

OWASP Testing Guide: https://www.owasp.org/index.php/Testing_for_SSL-TLS_(OWASP-CM-001)

Beast attack: https://isc.sans.edu/diary.html?storyid=11722

לסיכום

במאמר זה סקרתי בקצרה מגוון תחומי אבטחה בהקשר של SSL, החל מהסברים על שיטות ההצפנה והחלפת המפתחות, המשך בסקירת חולשות שונות הקשורות ל-SSL, סקירת הכלי sslscan וניתוח תוצאותיו, הצגת הכלי שפיתחתי SSL Vulnerabilities Analyzer שמנתח באופן ברור ונאה את החולשות שקשורות ב-SSL בשרתים. התקדמתי לקראת סיום בבדיקת האלגוריתמים שנתמכים בשני סוגי שרתים פופולריים ולסיום הראיתי שנכון להיום, מי שגולש בדפדפן מודרני מעודכן (מאלה שנבדקו) מוגן מפני אלגוריתמים פגיעים אלה.

במאמר הבא אסקור את שפות התוכנה, איך והאם הן מתמודדות עם אלגוריתמים חלשים ב-SSL ו/או חותמים לא מוכרים. האם אכן VBS מאובטח יותר מפייתון?.. על כך ועוד, המתינו לחלק הבא.

כל הנ"ל ניתן להרחבה וחלקים שונים נמצאים במחקר מתקדם. תרגיש/י חופשי להשאיר פידבק (israel@appsec-labs.com), תודה מראש. אני שמח לתרום את המאמר לקהילה, ומקווה שהתרומה הישראלית לקהילה הישראלית תתרחב.

ישראל חורז'בסקי [Sro.co.il]

ראש צוות Penetration Testing ב-AppSec Labs