Digital Whisper
Technologic Papers

תיאור מתקפת BEAST על הפרוטוקול SSL

מאת יוחאי אייזנרייך

פתח דבר

את המאמר על מתקפת BEAST (מעתה תקרא בפשטות BEAST) החלטתי לכתוב ממספר סיבות: ראשית, כשרציתי ללמוד על המתקפה בעצמי, לקח לי לא מעט זמן למצוא את כל החומר הנדרש ולהבין אותו כמו שצריך. לא מצאתי אתר שמאגד בתוכו את כל האינפורמציה בצורה מאוד נוחה וברורה, וחשבתי שיהיה נחמד לכתוב מאמר על BEAST בשפת הקודש.

שנית, לרוב כאשר אני צריך להסביר משהו לאנשים אחרים אני נתקל בפינות חשוכות שאין לי תשובות אליהן וזה מאלץ אותי לחקור את הנושא לעומק. ממש לעומק. לשם כך, החלטתי לקחת על עצמי לכתוב מאמר בנושא, ולהכנס גם לפרטים הקטנים, בכדי שהמתקפה תהיה לי ברורה לחלוטין. לבסוף, אני מוצא את הנושא מאוד מעניין, ואני מקווה שאצליח להצית התלהבות בקרב חוקרים וחובבי אבטחת מידע נוספים.

המאמר מתאר הרבה דברים בסיסיים שכנראה שכל קורא טכנולוג הדיוט יודע. השתדלתי לכתוב את המאמר כך שגם קוראים שהם לא בהכרח אוכלים HTTP לארוחת בוקר יבינו אותו ויגלו בו עניין. החלקים היישומיים מגיעים לקראת הסוף.

עוד מילה קטנה לפני שאני מתחיל - הרבה פעמים כשאני נתקל במערכות הצפנה (כגון SSL) אני מוותר לעצמי בטענה שזה "קשה מדי, מתמטי מדי, מסובך" ובפועל אחרי שאני מבין את הדברים לעומק, אני מגלה שהם לא היו מורכבים כמו שחשבתי. משום כך, השתדלתי לכתוב את המאמר בצורה הכי מובנת ופשוטה שיש, בכדי שגם מי שהוא טכנולוג אך לא התעסק בהצפנה בעבר יוכל להרגיש בבית. כן נדרשת הבנה בסיסית בפרוטוקולים ועולם האינטרנט, אך לא משהו יוצא מן הכלל.

מבוא

BEAST או בשמה המלא **Browser Exploit Against SSL/TLS** היא מתקפה על פרוטוקול SSL[1] אשר מאפשרת בהינתן תנאים מסוימים לקרוא תוכן של תעבורה מוצפנת. מה זה אומר בעברית פשוטה? שאני

[1] Secure Sockets Layer – תקן RFC5246. הפרוטוקול המעודכן הוא Transport Layer Security, (TLS). המתקפה פועלת גם על TLSv1.0, אך למען הנוחות ושמירה על קונבנציה, נכנה את הפרוטוקול SSL.

יכול להבין מה עובר בין המחשב שלכם לאינטרנט - גם אם אתם גולשים תחת SSL ויש לכם מנעול נחמד בדפדפן שמשרה בכם תחושת בטחון.

את המתקפה הציגו בכנס ekoparty שני חוקרים בשם Thai Duong ו-Juliano Rizzo בספטמבר 2011[2]. מתקפת BEAST מבוססת בעצמה על מתקפת צופן שעד אז הייתה ידועה אך נחשבה תיאורטית בלבד. המתקפה התיאורטית התגלתה כבר בשנת 2002 על ידי Phil Rogaway[3].

לאחר הפרסום, המתקפה עוררה הדים רבים בעולם אבטחת המידע, ונעשו מספר שינויי תקנים ותיקונים בדפדפנים שונים על מנת להתמודד עם המתקפה. לכאורה, מתקפה מסוג זו אינה אפשרית היום - אך הרבה מהעקרונות אשר מאפשרים את ההתקפה פועלים באותה צורה עוד היום.

אז איך כל הסיפור הזה עובד? Let's get down to business!

רקע - מספר מילים על הצפנה ואיך SSL עובד

אוקיי, אז SSL, למי שלא לגמרי מעודכן ברזי האינטרנט - הוא אולי הפרוטוקול המרכזי והמשמעותי ביותר בהעברת מידע בצורה מאובטחת מנקודה א' לנקודה ב', בלי שגורם צד ג' יוכל לפענח את המידע. קריאת אימייל, טרנזאקציות אשראי, פייסבוק, וכמעט כל אתר היום באינטרנט כבר משתמש ב-SSL. קצרה היריעה מלתאר אותו על כל רבדיו אבל אתחיל בפתיח כללי מאוד, וממנו נצלול להבנת המנגנונים הרלוונטים לטובת BEAST. מי שמכיר את הפרוטוקול מעט ואיך עובדת הצפנה באופן כללי יכול לדלג על פרק זה.

אם כן, נתחיל מדוגמא פשוטה - ברצוני לפנות אל הבנק ולברר מה היתרה שלי. לרוע המזל, השכן המרושע שלי (מעתה נכנה אותו "מיצי") משתמש ברשת האלחוטית שבביתי ומאזין לכל המידע שאני מעביר. אם לדוגמא הבנק יבקש ממני סיסמא בכדי להיכנס אל השירות ואני אשלח אותה אל הבנק, מיצי מיד יגלה את הסיסמא שלי, כיוון שהוא מאזין למידע שאני שולח אל הבנק, ובאמצעותה הוא יוכל לגנוב את כל הכסף שהרווחתי בעמל רב (שזה תרחיש גרוע רק בקצת מ"פיגוע פייסבוק").

בכדי למנוע את תרחיש האימים הזה, אני והבנק מסכימים להצפין את כל המידע בינינו כך שאם מיצי פתאום מחליט להתלבש לי על הרשת, הוא רק יראה מידע מוצפן, ולא יוכל לחלץ את הסיסמא שלי. ניצחון!

[2] http://www.ekoparty.org/2011/thai-duong.php
[3] https://web.archive.org/web/20120630143111/http:/www.openssl.org/~bodo/tls-cbc.txt

אבל רגע. מה זה אומר שאני והבנק "מצפינים את המידע"? איך פתאום הקסם הזה קורה? אז הקסם הזה מורכב משני רכיבים משמעותיים - מפתח וצופן.

צופן הוא השיטה בה אנחנו מצפינים את המידע, כלומר - אני והבנק צריכים להסכים בינינו על שיטה מסוימת ששנינו יודעים אותה, והיא מאפשרת לקודד את המידע. לדוגמא אפשר להחליט שכל אות באל"ף-בי"ת העברי נחליף באות אחרת. לדוגמא: האות "א" תהפוך לאות "ג", האות "ב" תהפוך ל"ד" וכן הלאה (שיטה זו נקראת "צופן קיסרי" או "צופן היסט").

מפתח הוא הסוד המשותף שמשמש את השיטה להצפין, כלומר - אם נחזור לדוגמא הקודמת - כל אות באל"ף-בי"ת הופכת לאות שנמצאת במרחק +2 ממנה. כך ש-"א"->"ג" "ב"->"ד" וכן הלאה. אם כן - השיטה שלנו היא צופן קיסרי, והמפתח במקרה זה יהיה **+2**. המפתח הוא סוד משותף ביני ובין הבנק, שאסור שיוודע לאנשים אחרים! (אז בחיאת, אל תגלו...)

אם לדוגמא אני רוצה לשלוח לבנק את המילה "סיסמא", בפועל אני אשלח לו "פלפסג". הבנק יודע את שיטת הצופן ואת המפתח, ולכן כשיחליף כל אות בזו שנמצאת 2 מקומות לפניה באל"ף-בי"ת, הוא יבין שאני רוצה לשלוח לו "סיסמא". דוגמא לטבלה שממחישה את ההצפנה עם מפתח +2:

א	ב	ג	ד	ה	ו	ז	ח	ט	י	כ	ל	מ	נ	ס	ע	פ	צ	ק	ר	ש	ת
ג	ד	ה	ו	ז	ח	ט	י	כ	ל	מ	נ	ס	ע	פ	צ	ק	ר	ש	ת	א	ב

עכשיו, בואו נזכר בשכן המרושע שלי - מיצי. הוא עשוי לדעת שהשיטה שאני והבנק משתמשים בה היא צופן קיסרי, אבל זה לא יעזור לו, כל עוד הוא לא ניחש את מפתח ההצפנה הנכון. הוא עלול לראות שאני שולח "פלפסג", אבל כל עוד הוא לא הבין שהמפתח הוא +2, הוא יכול לשבור את הראש בניסיון לשים את הידיים המטונפות שלו על הכסף שלי. למי שאיבד אותי - סיפקתי עזר חזותי שאמור לסייע בהבנת הנכתב.

אז אני והבנק מבסוטים אחד על השני, אנחנו רק צריכים להסכים על צופן ועל מפתח - לכאורה, לא בעיה מאוד קשה. כאן נכנס לתמונה פרוטוקול SSL. במקום שאני אצטרך ללכת פיזית לבנק (או לכל הפחות לדבר עם מוקדנית), פרוטוקול SSL מבצע את כל הטרחה הזו בשבילי, על גבי תשתית האינטרנט הקיימת. הוא כזה גבר, שהוא גם עושה את זה מול פייסבוק, ג'ימייל, ורחמנא ליצלן - קופת חולים (לא רוצה לדמיין אפילו את המוקד שם). אז איך הוא עובד כל כך הרבה בלי להתעייף?

ב-3 שלבים מרכזיים

א. הסכמה על שיטת הצפנה - חשוב מאוד שאני והבנק נעבוד באותה השיטה. SSL עושה בינינו בוררות עד שנבחר שיטת הצפנה שטובה לשנינו. זכרו - אין שום בעיה שמיצי ידע באיזו שיטה אני והבנק משתמשים, כל עוד אין בידיו את המפתח.

ב. העברת מפתח - פה כבר מגיע חלק מאוד מאוד טריקי ומסובך. אני והבנק צריכים להעביר בינינו איזשהו סוד, בלי שמיצי החטטן יגלה אותו. אני לא אכנס לעומק הדברים ואסביר איך זה עובד, אז פשוט אציב כעובדה ש-SSL מסוגל להעביר סודות מסוימים בלי שמיצי יראה. למתעניינים - חפשו עוד על החלפת מפתחות, RSA, והצפנה א-סימטרית. השורה התחתונה היא שאני והבנק הסכמנו על איזשהו סוד בינינו שיעזור לנו להצפין את התעבורה בינינו. בכל אופן - זה לא חלק קריטי בכדי להבין את BEAST.

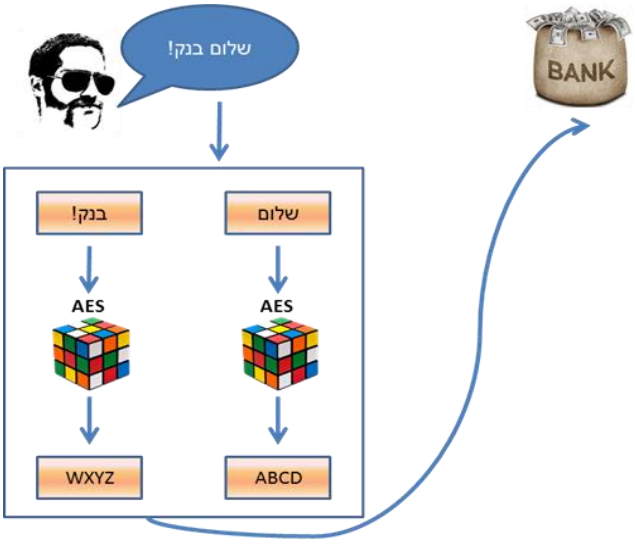
ג. העברת המידע עצמו - אחרי שאני והבנק הסכמנו על שיטה ומפתח - כל מה שנותר לנו הוא להעביר בינינו את המידע. אני מדגיש שההצפנה שאנו משתמשים בה היא סימטרית! בעברית: טקסט מוצפן ("פלפסג") יכול להפוך באופן ישיר לטקסט קריא ("סיסמא") באמצעות המפתח הנכון ("+2"). הכוונה היא שזו פעולה הפיכה.

צוללים פנימה - Block Cipher

אחרי שעברנו את הבסיס התיאורטי הזה, אפשר להתחיל להגיע לפרטים העסיסיים והטכניים שמרכיבים את BEAST. פרוטוקול SSL מאפשר להשתמש במספר מנגנוני הצפנה, אשר אחד מהם, וניתן אף בזהירות להגיד - הפופולרי מביניהם (או לפחות עד סוף 2011), נקרא בשם Advanced Encryption Standard (AES)[4]. הצופן מהווה סטנדרט מוכר בעולם ונפרט מעט על דרכי פעולתו בכדי להבין למה הוא יכול להיות בעייתי.

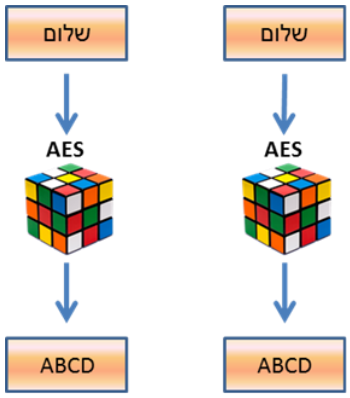
[4] הסטנדרט של שיטת ההצפנה - http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

אז AES עובד בשיטה שנקראת "Block Cipher" או "צופן בלוקים". זה אומר שהוא מחלק את המידע שאני רוצה להעביר לחתיכות ("בלוקים"), מצפין כל חתיכה, ומעביר הלאה. אם לדוגמא אני רוצה להעביר לבנק את ההודעה "שלום בנק!" (למה לעזאזל שאני ארצה לעשות דבר כזה? אין לי מושג), אז AES יחלק את המידע לדוגמא לבלוקים של 4 תווים כל אחד ("שלום" "בנק!"), ואז יצפין אותם לטקסט אחר לגמרי ("ABCD" "WXYZ").

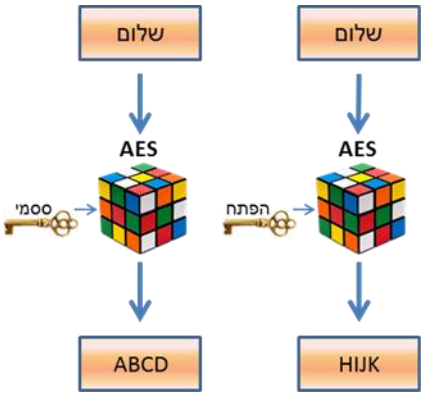


חדי העין יבחינו שבהודעה "שלום בנק!" קיים רווח בין המילים שהתעלמתי ממנו בגסות. אין לי משהו אישי נגד רווחים בין מילים, אך לשם הדוגמה הפשוטה וההמחשה בלבד, אני אמשיך לעשות זאת. כמובן שבמציאות כל תו שהוא עובר הצפנה. בנוסף, המילה "שלום" מוצפנת ל-"ABCD" רק לשם ההמחשה (וכך גם עם שאר הדוגמאות). במציאות היא תוצפן כנראה למשהו אחר לחלוטין.

שימו לב שעבור כל בלוק, AES משתמש באותה השיטה ובאותו מפתח ההצפנה. זאת אומרת, אם אני והבנק מסכימים שהמפתח בינינו הוא "ssמי", ואני אשלח לבנק "שלום שלום", מה שיקרה הוא ש-AES יחלק את המידע לבלוקים "שלום" ו-"שלום", ויצפין את שניהם. התוצאה תהיה אותו דבר - " "ABCD "ABCD"".



עם זאת, אם אני אשלח "שלום" פעם באמצעות המפתח "ססמי" ופעם אחרת באמצעות המפתח "הפתח", אני אקבל שתי תוצאות שונות. זכרו - השיטה היא אותה שיטה, אבל מפתח ההצפנה שונה, ולכן הצופן המתקבל יהיה שונה.



מה שמייחד את AES מצפנים אחרים הוא שיטת ההצפנה (זו שבדוגמא הופכת את "שלום" ל-"ABCD"). אני לא אכנס לפרטים איך ההצפנה של AES עובדת, כיוון שלשם כך יש להסביר עוד המון נושאים מתקדמים לעומק, וזה לא רלוונטי לשם הבנת BEAST. למי שמעוניין יש אחלה קומיקס שמסביר את הנושא בצורה מעולה.

למתעניינים בפרטים הטכניים (שמכאן יתחילו לצוץ יותר) - AES עובד עם בלוקים של 128 או 256 ביט (כלומר 16 או 32 בתים בהתאם). במקרה שהמידע לא מתחלק בדיוק לגודל הבלוקים, מתווסף Padding (ריפוד) לבלוק האחרון כך שיתאים בדיוק לגודל של בלוק שלם. ברוב הדוגמאות פה אני לא אקפיד להראות 16 או 32 בתים, אבל העיקרון ישאר זהה.

השורה התחתונה והחשובה שצריך לזכור מהחלק הזה, והיא זו שבסוף יוצרת את הבעיה - אם אני מכניס את אותו התוכן **plaintext** ("שלום") ואשתמש באותו מפתח **key** ("ססמי"), אני אקבל תמיד את אותו המידע המוצפן **cipher** ("ABCD").

מכאן והלאה נתחיל לראות יותר אותיות, מספרים וביטויים באנגלית בכדי להבהיר את הנקודה טוב יותר בהמשך. אל תדאגו - זה רק נראה מורכב.

מוסיפים סיבוך קל - Cipher Block Chaining

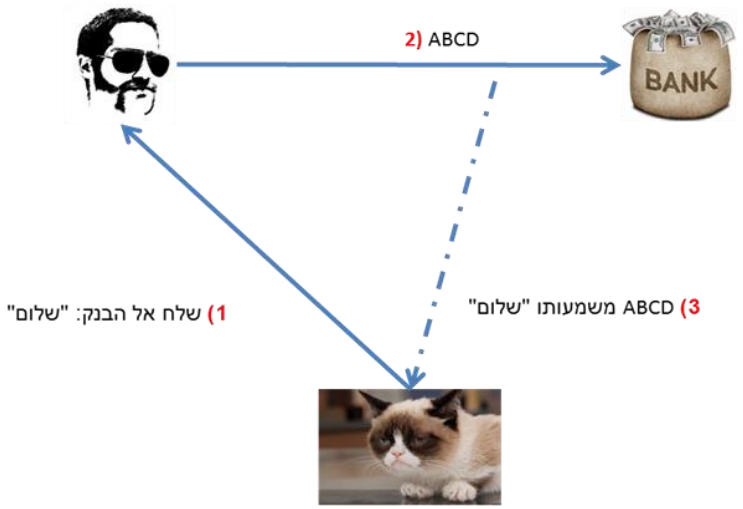
אוקיי, אז אנחנו יודעים איך Block Cipher עובד. יום אחד בבוקר, אני קם רענן במיוחד, ומקווה בלבי שהבנק יוריד לי את העמלות, יעלה לי את הריבית בחסכון, ויגיד לי שאני אדם עשיר. לשם כך אני מברך אותו בבוקר המון. ממש המון. אני שולח לו כל הזמן "שלום" "שלום" "שלום" "שלום" "שלום" בתקווה שזה ימצא חן בעיני הפקידה בבנק.

בינתיים בזמן שלמדנו על הצפנה, מיצי שלנו לא ישב על השמרים, שתה את החלב שנשפך, וניסה להבין איך הוא יכול לפרוץ לי לבנק. להזכירכם - הוא עדיין מחובר לרשת האלחוטית שלי, והוא רואה את כל מה שאני שולח לבנק. באותו בוקר של רעננות וברכות שלום, הוא רואה שאני שולח לבנק "ABCD" בלי הפסקה (זוכרים? הוא רואה את התוכן המוצפן שאני שולח לבנק).

למיצי עולה רעיון מבריק. כיוון שמדובר בשעת בוקר - ובשעת בוקר לרוב מברכים בברכת "שלום" ("בוקר טוב" זה כל כך 2001...), הוא מנחש שבטח אני מברך את הבנק ב"שלום". אז איך הוא יכול לאשש את ממצאיו? מיצי הוא שכן חכם ותחמן, והוא דופק על הדלת שלי עם תפוח מורעל ומציע לי לאכול ממנו. כיוון שאני מקפיד על הבריאות שלי ועל שכנות טובה, אני מסכים בנימוס.

לרעל יש השפעה מעניינת - כך שמעכשיו כל מילה שמיצי צועק לעברי מהבית שלו, אני אשלח אל הבנק. בהתחלה מיצי צועק לי לשלוח אל הבנק "יא גנבים שכמותכם!" והוא רואה שאני שולח אל הבנק מידע

מוצפן בערך כזה: "GHFDGJFHFDSF". אחר כך הוא צועק לי לשלוח אל הבנק "שלום", ורואה שעובר המידע "ABCD". בינגו! אז מה בעצם מיצי עשה? הוא עדיין לא יכול לקרוא את מה שעובר ביני ובין הבנק (כי אין לו את המפתח), אבל הוא יכול לגרום לי להעביר מידע אל הבנק וכך לקשר ש"שלום" הופך ל"ABCD" באמצעות המפתח ושיטת ההצפנה שאני והבנק קבענו. מבולבלים? ככה זה עובד:



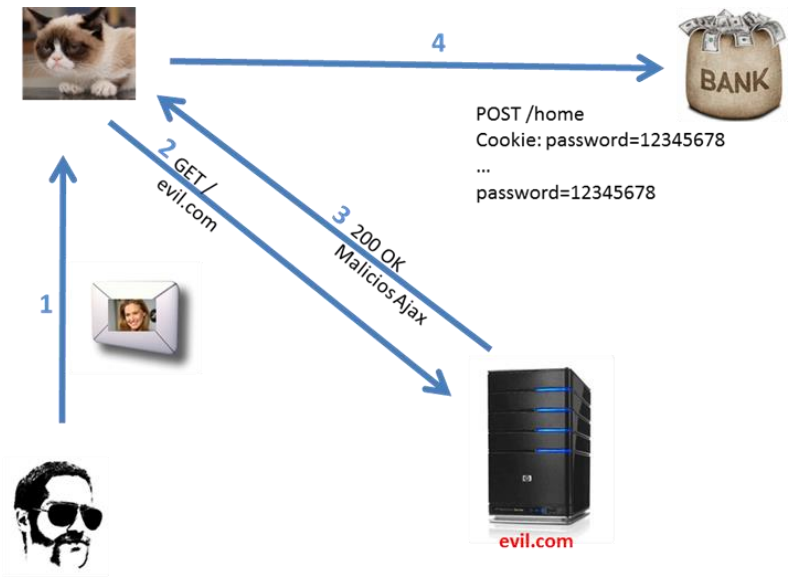
המשמעות היא שבהינתן שמיצי יכול גם להאזין למידע שלי, וגם לגרום לי לשלוח מידע אשר בשליטתו, הוא יוכל לקבל אפשרות "מנוונת" לקרוא את המידע אשר עובר ביני ובין הבנק. המידע עדיין לא חשוף לו לחלוטין, אך הוא יכול לנחש מה אני מעביר לבנק (לדוגמא: "סיסמא", "משיכת מזומן" וכו'), לגרום לי לשלוח את המידע הזה פעם אחת, ובכך להבין למה כל אחד מהטקסטים הללו הופך כאשר הוא מוצפן. לכאורה הוא עדיין צריך לנחש המון אפשרויות בכדי להשיג את הסיסמא שלי, אך בהמשך נראה איך מטרה זו הופכת לקלה באופן יחסי.

פה אני נאלץ להפרד ממיצי ומהדוגמא הזו כי כבר די כפיתי את הדוגמא על המציאות, תוך כדי שימוש בסיפורי שלגיה, חתולונובלה, ודירה להשכיר. אז איך אנחנו מתרגמים את מה שקורה כאן לפרקטיקה בעולם האינטרנט? (הפעם נניח שאני התוקף הנבזי ואני מאזין לרשת של השכן שלי).

ראשית, עלי לגרום לשכן שלי להכנס לאתר מרושע בשליטתי, **evil.com** (אחד הדומיינים אם לא ה-!). אני יכול לעשות זאת באמצעות מייל פישינג לדוגמא ("תמונות של בר רפאלי לוהטטט חובה"). השכן נפל קורבן וגלש לאתר האינטרנט שלי. אני מפנה אותו לאתר הבנק שלו באמצעות AJAX לדוגמא, וגורם לו לבצע את הפניה הבאה:

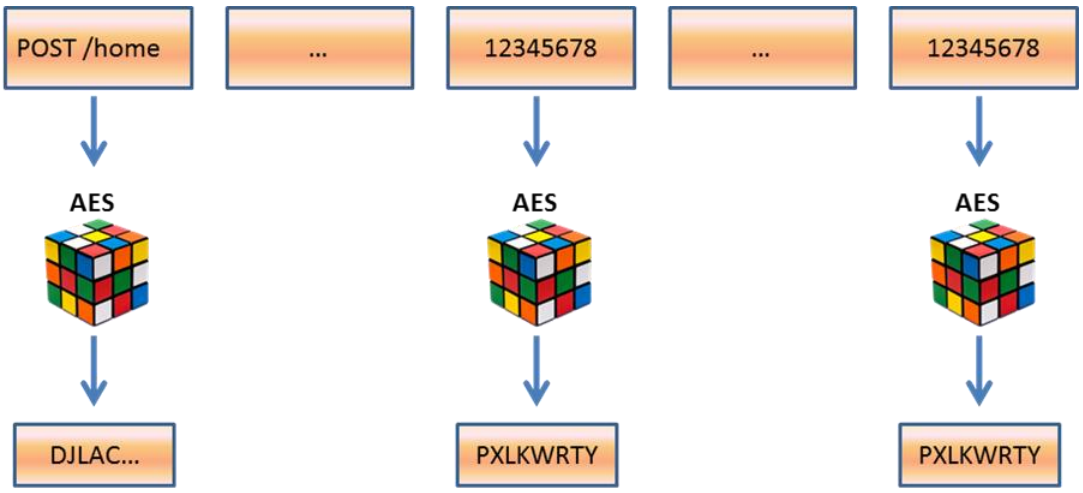
```
POST /home \r\n
Cookie: password=12345678\r\n
....
\r\n
password=12345678
```

הערה טכנית: מותר לי להפנות את השכן לאתר של הבנק שלו עם העוגיות המתאימות בלי לשבור את ה-Same Origin Policy. השכן יפנה לאתר, אבל אני עדיין לא אוכל לקרוא את העוגיות שלו מהדפדפן.



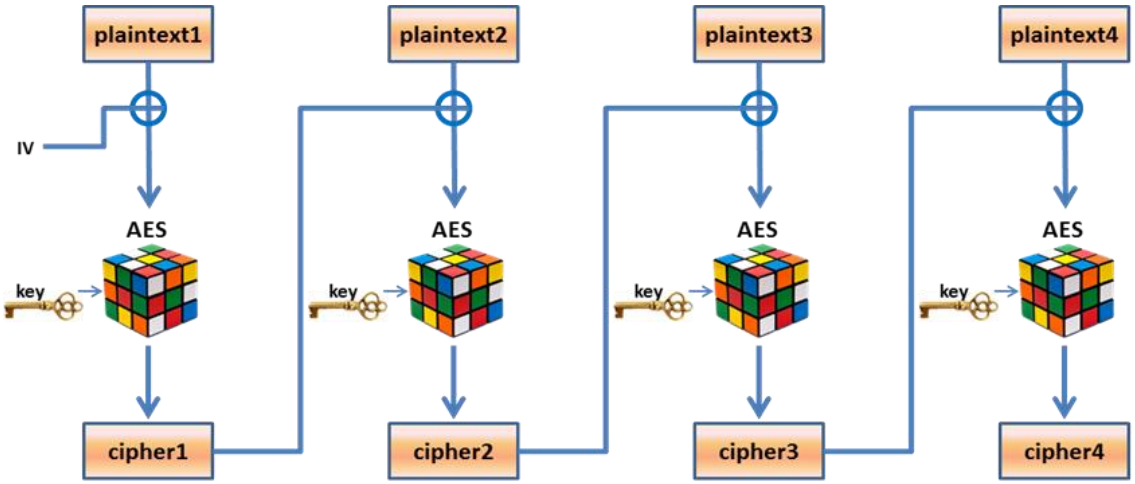
עכשיו, זוכרים שהמידע מתחלק לבלוקים? מה יקרה אם password שמופיע תחת שדה ה-Cookie ו-password שמופיע בתוכן הבקשה יהיו שניהם בדיוק מיושרים על בלוק? לפי מה שלמדנו עד עכשיו אם התוכן (plaintext) זהה, והמפתח (key) זהה, אז הצופן (cipher) יהיה זהה.

אנחנו נניח לדפדפן לשלוח את העוגיות המתאימות לבנק (ובמקרה הזה, סיסמא), ובגוף הבקשה נשלח את הניחוש שלנו לסיסמא. מה שזה אומר בפועל, שאני יכול כל פעם להפנות את השכן שלי לאתר הבנק ולבצע ניחושים מושכלים מה הסיסמא שלו. כשהניחוש שלי יהיה נכון, אני אראה שני בלוקים מוצפנים אשר אני לא אדע לקרוא את תוכנם, אך כיוון שהם זהים, אני אדע שהניחוש שלי נכון. יישור המידע על הבלוק לא מהווה בעיה מורכבת כיוון שאפשר לשחק לא מעט עם הפרמטרים של בקשת ה-HTTP.



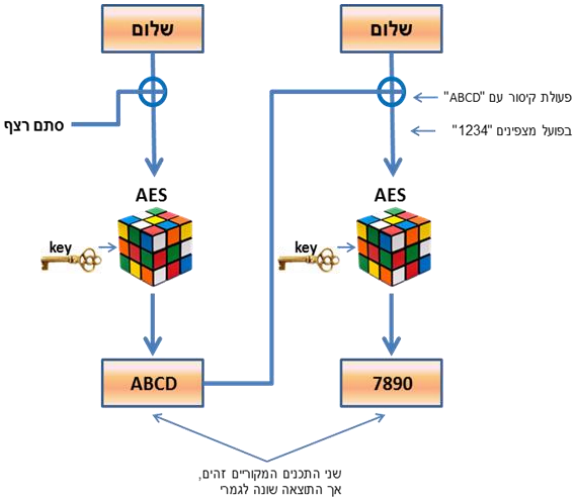
כפי שניתן לראות בדוגמא - ניחוש נכון של הסיסמא "12345678" כאשר שתי הסיסמאות מיושרות על בלוק, מוביל לשני בלוקים מוצפנים זהים.

עכשיו חכו, אין לכם מה להתרגש ולרוץ לקרוא סיסמאות עדיין, כי SSL לא באמת עובד ככה. Block Ciphers ידועים כמסוכנים אם משתמשים בהם בצורה הזו (ECB - Electronic Codebook[5]), ולכן הומצאה שיטה אחרת על גבי השיטה הנוכחית בשם Cipher Block Chaining (CBC)[6] שבה לרוב משתמשים ב-SSL. שיטה זו אומרת את הדבר הפשוט הבא: לפני שנצפין כל plaintext, נבצע עליו קודם פעולת XOR (בעברית צחה: "נקסר", פעולת XOR מסומנת עם הסימן ⊕) עם הבלוק שהוצפן לפניו. אם מדובר בבלוק הראשון, נקסר אותו עם מחרוזת רנדומלית שמוסכמת על שני הצדדים ונקרא לה מעתה בשם המסובך - Initialization Vector (IV).



אני אחזור שוב על הדברים עם דוגמא ואסביר את הרציונל. נניח שיש לנו את המחרוזת "שלום" שמוצפנת והופכת ל-"ABCD". עכשיו כשנרצה לשלוח עוד "שלום", נקסר אותו לפני כן ב-"ABCD" מה שיתן לנו (סתם לדוגמא, לא באמת) את המחרוזת "1234". עכשיו המחרוזת שאנחנו מצפינים היא "1234", ולא "שלום", מה שיוביל לצופן שונה לגמרי (שוב סתם לדוגמא: "7890").

[5] http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation#Electronic_codebook_.28ECB.29
[6] http://en.wikipedia.org/wiki/Cipher_block_chaining#Cipher-block_chaining_.28CBC.29



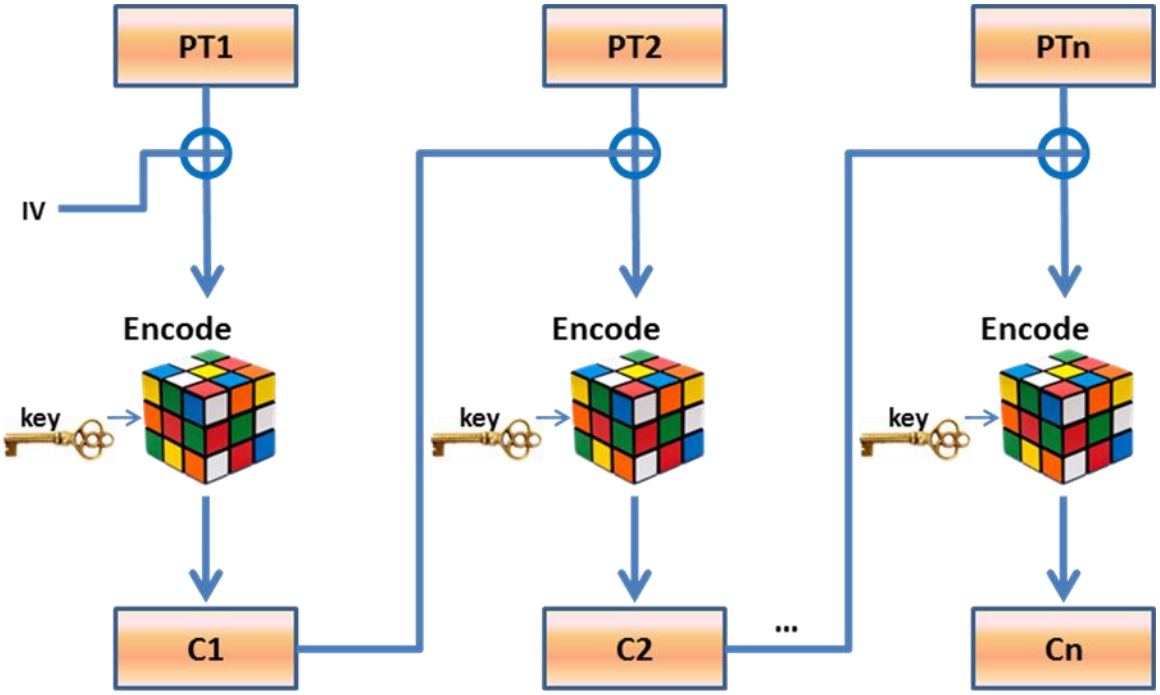
כך נוכל להמשיך הלאה ולשלוח "שלום" לבנק עד אין קץ, ובכל פעם ישלח טקסט מוצפן שונה לחלוטין. אם השכן שלי עכשיו מחליט להיות נחמד בעצמו לבנק ולשלוח המון ברכות "שלום", אני אשר מאזין לתעבורה המוצפנת, אראה כל פעם טקסט מוצפן שונה שלא יהיה קריא לי. מה השגנו בזה? אני לא יכול יותר לנחש מה עובר אצל השכן שלי. אפילו אם אני גורם לו לשלוח מידע בשליטתי. למה? כי כל פעם יעבור על הקו צופן אחר - גם אם התוכן המקורי הוא אותו התוכן.

אז בכדי לסגור את הפינה הזו נותר לנו להבין עוד מספר דברים קטנים. ראשית, מהו ה-IV ואיך הוא נקבע? ואיך כל הסיפור הזה לא דופק את ההצפנה?

אם כן, כיוון שלפני הבלוק הראשון לא מגיע שום מידע, ואנחנו לא רוצים שמידע ישלח על הקו בלי לקסר אותו באיזה רצף ביטים עסיסי - בוחרים IV בגודל מסוים בצורה רנדומלית, ומעבירים אותו בין שני הצדדים. אפשר לחשוב עליו כעל מפתח משני. ה-IV של הבלוק השני, הוא למעשה כבר תוכן הצופן של הבלוק הראשון.

XOR היא פעולה שקולה / סימטרית, כך שפעולת הקיסור לא פוגמת במידע המוצפן, היא רק מוסיפה לו "מיסוך" שהכרחי בכדי שלא ניתן יהיה לחזות מה עובר מתחת למעטה ההצפנה. ניתן לקסר את המידע שוב באותו הערך ולקבל את המידע המקורי.

עכשיו אחרי שלמדנו כבר לא מעט, בואו נסכם וניצג את המידע בכמה תרשימים ומשוואות (כדי שזה בכל זאת יראה מדעי ומסובך):



C = Cipher

PT = Plaintext

n = Current block location

Encode = AES encoding function

$\mathbf{C_n} = \text{Encode}(PT_n \oplus C_{n-1})$

$\mathbf{C_1} = \text{Encode}(PT_1 \oplus IV)$

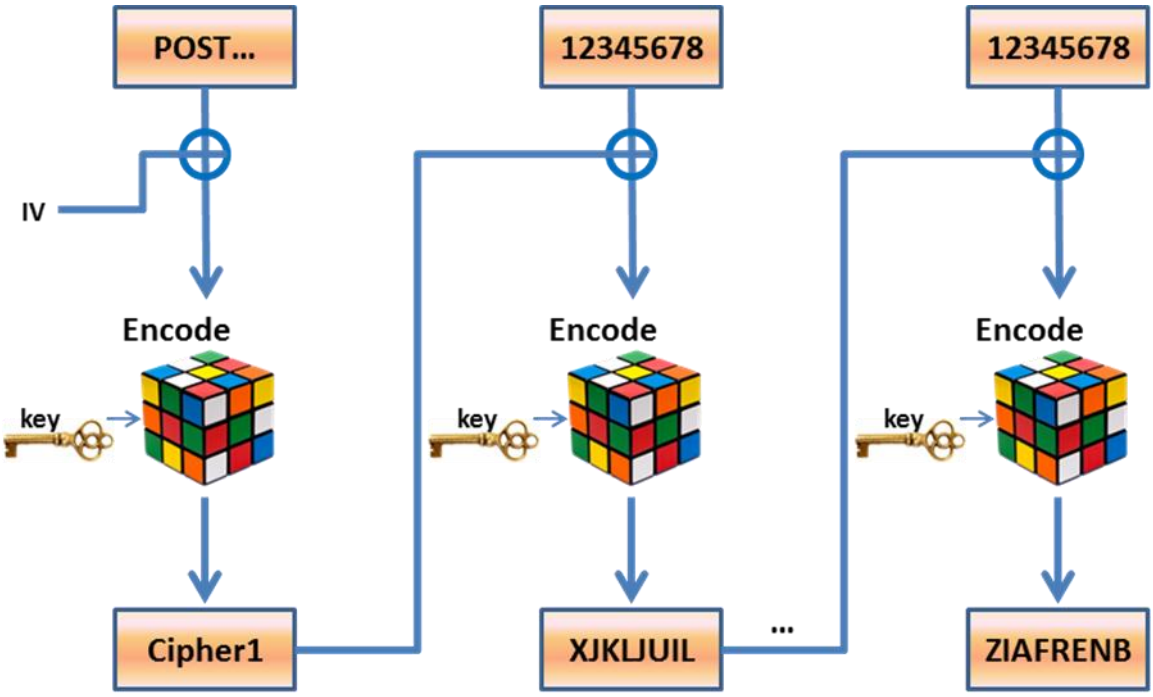
<u>טיפ של אלופים:</u> למי שרוצה להתנסות בעצמו, אני מציע לפתוח python ולשחק קצת עם הספריה pycrypto. הספריה מכילה מימוש של AES, וניתן להפעיל אותו בשני מצבים - ECB (רגיל, לא משורשר), ו-CBC. נסו להצפין מחרוזות זהות ושונות ולראות מה קורה לכל אחת מהן בכל מצב.

עכשיו אפשר להגיע לתכל'ס - BEAST

אז למה ייסרתי אתכם בחתולים, שכנים, אותות זדוניים ומשוואות עד עכשיו? הנה התשובה.

BEAST היא מתקפה שנחשבה לתיאורטית בלבד, אך הוכחה כמעשית ביותר על פרוטוקול SSL. ראשית הנחנו את היסודות התיאורטיים שעובדים מאחורי המנגנונים השונים, הבנו אפילו איך אפשר לתקוף אותם, ואיך אפשר להתגבר על המתקפה. אני מזכיר ש-SSL לעולם לא משתמש ב-Block Cipher שהוא לא Chained. BEAST היא מתקפה שמאתגרת את הקביעה הזו, ואפילו בלי להזיע יותר מדי.

אז נחזור לנקודת המוצא הטכנולוגית שלנו - אני ברשת של השכן שלי, אני מאזין לכל התעבורה המוצפנת שלו ויש לי יכולת לגרום לו לפנות אל הבנק עם איזה תוכן שאני רוצה. נניח שוב שמדובר בפניית AJAX. כיוון ש- SSL כאמור משתמש במצב CBC - אותה בקשה שלי מקודם כעת תראה אחרת:



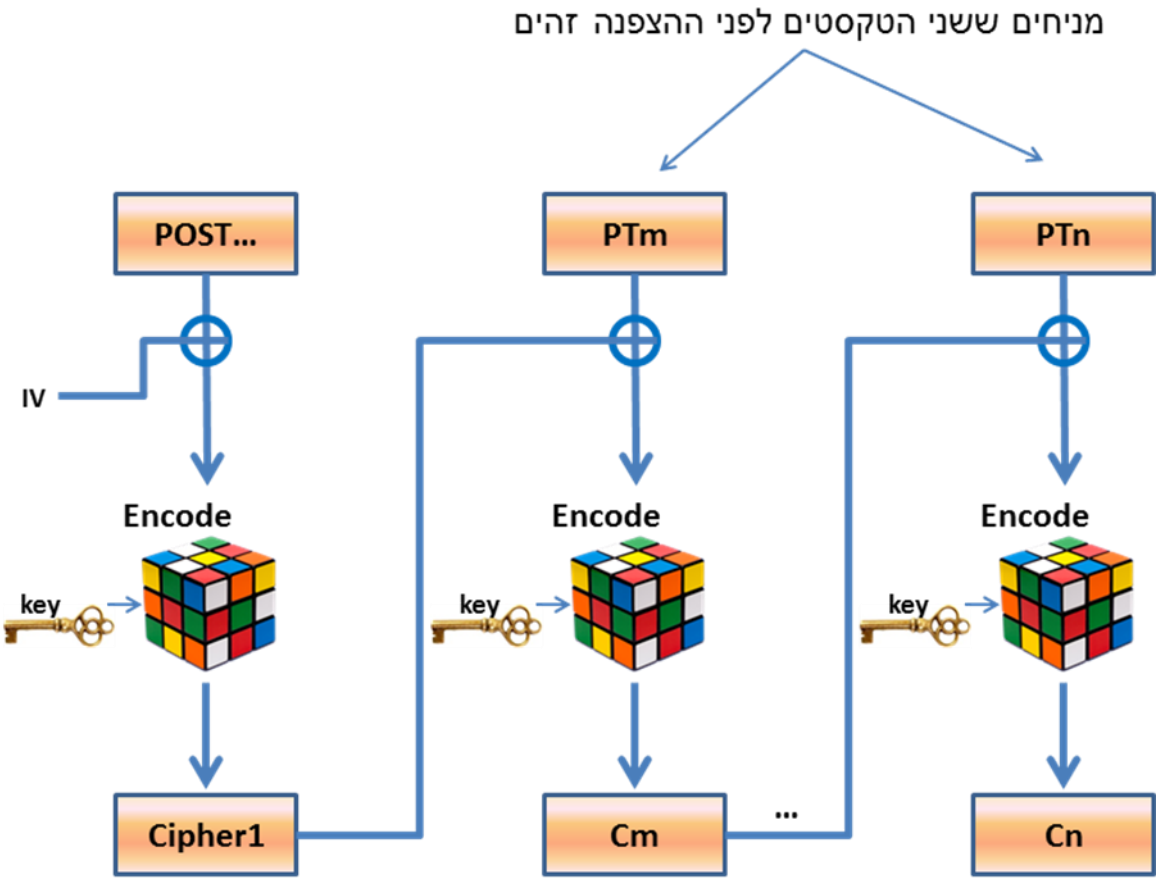
המשמעות ההרסנית מבחינתי - הצופן אשר יווצר מהבלוק password=12345678 הראשון אינו זהה לבלוק שיווצר מהבלוק password=12345678 השני.

זה כאמור תוצר של מצב הפעולה CBC שהוסבר בהרחבה. אז איך בכל זאת אפשר לתקוף את הבעיה הזו? בואו נעיין לרגע במשוואות. קבענו כי:

$C_n = Encode(PT_n \oplus C_{n-1})$

אבל - וזהו אבל קריטי ביותר - שימו לב שאנחנו מאזינים לקו! כלומר - אנחנו יודעים מה הערך של כל הבלוקים המוצפנים שנשלחו על הקו. אז מה אפשר לעשות? בואו נניח שאנחנו מנחשים שהבלוק המעניין הוא באמת password=12345678. או אם נתרגם את זה למשוואה, נניח שמידע לא מוצפן PT_m (לדוגמא, הבלוק הראשון שמופיעה בו הסיסמא) הוא למעשה זהה למידע לא מוצפן PT_n (לדוגמא הבלוק השני שמופיעה בו הסיסמא - בתוכן).

$PT_m = PT_n$



עכשיו, אנחנו יודעים באמצעות מה קיסרו את הבלוק הראשון PT_m, כי גם זה בלוק מוצפן שעבר על הקו! C_{m-1}. זאת אומרת ש:

$C_m = Encode(PT_m \oplus C_{m-1})$

אז אם אני רוצה שהבלוק המוצפן PT_n יהיה זהה לבלוק המוצפן PT_m, אני צריך להזין לפונקציית ההצפנה את הערך $C_{m-1} \oplus PT_m$. אז איך אני עושה את זה? ככה:

$$PT_n = C_{m-1} \oplus C_{n-1} \oplus PT_m$$

בואו נבין מה המשמעות של המשוואה הזו ונתרגם אותה לפרקטיקה (תרשים מגיע בקרוב למי שמתייאש). C_{n-1} ידוע לי כי הוא הבלוק המוצפן שנשלח לפני שניה על הקו. C_{m-1} הוא גם בלוק מוצפן שידוע לי, כי גם הוא עבר פעם על הקו. PT_n הוא הניחוש המושכל שלי. זה אומר שהבלוק הבא של המידע שאני רוצה לשלוח הוא "ניחוש קסור הבלוק הקודם קסור הבלוק המוצפן שלפני הטקסט המנוחש". בעברית: אני למעשה עומד להצפין מידע בינארי שזהה למידע בינארי שכבר הוצפן פעם. ומה אמרנו שקורה במקרה כזה ב-AES? יוצא אותו פלט. למה זה עובד? כי:

$$\mathbf{C_m} = Encode(C_{m-1} \oplus PT_m)$$

$$\mathbf{PT_n} = C_{m-1} \oplus C_{n-1} \oplus PT_m$$

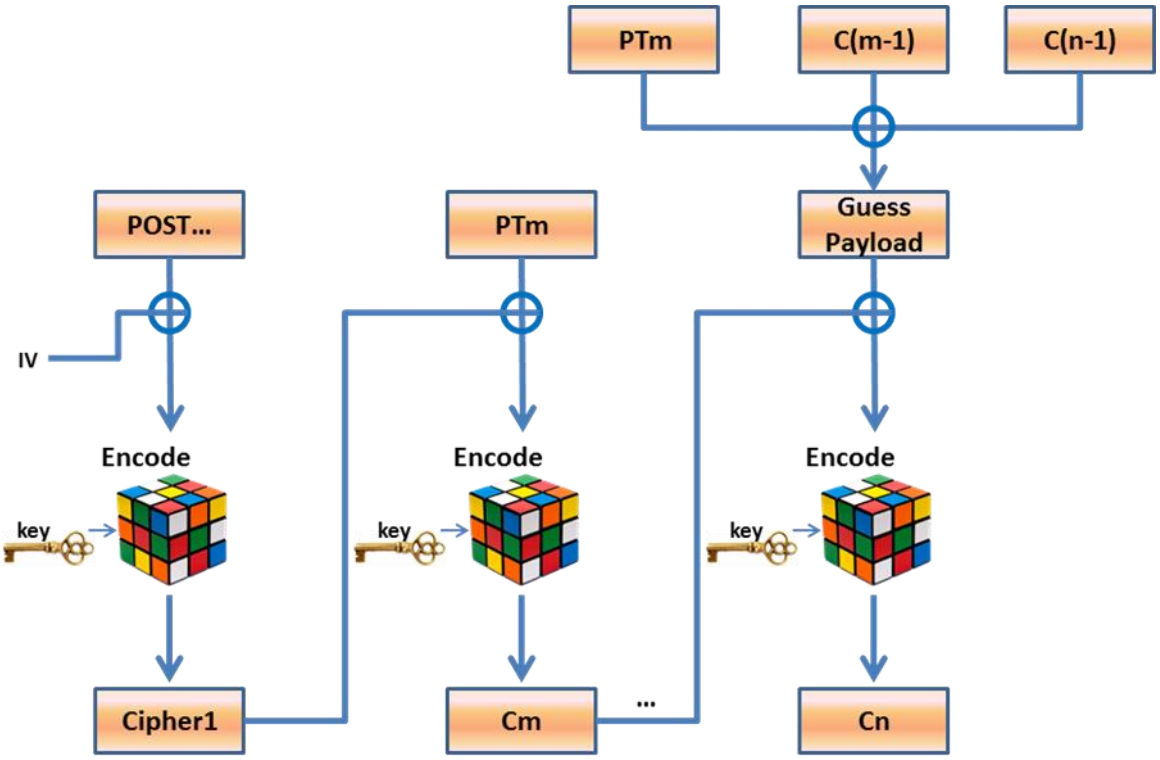
$$\mathbf{C_n} = Encode(C_{n-1} \oplus PT_n)$$

$$\mathbf{C_n} = Encode(C_{n-1} \oplus C_{m-1} \oplus C_{n-1} \oplus PT_m)$$

ופה הטריק הנחמד. חדי העין ישימו לב שאנחנו מקסרים פעמיים בערך C_{n-1}. כאמור, XOR היא פעולה סימטרית, ולכן XOR של אותו ערך פעמיים מתאפס. דבר זה מוביל למשוואה הבאה:

$$C_n = Encode(C_{m-1} \oplus PT_m) = C_m$$

ולכל מי שקץ במילים ומשוואות - הנה תמונה שאמורה להסביר הכל:

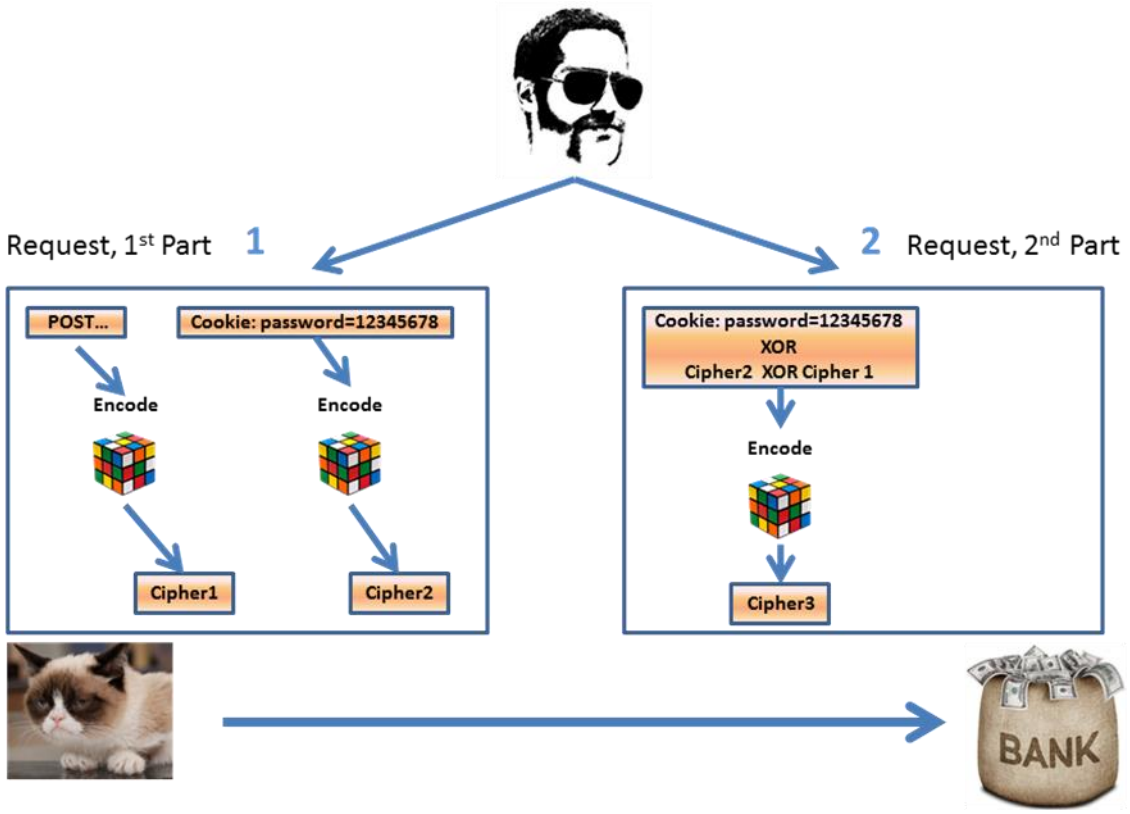


פה ניתן לראות כבר שבהינתן שהניחוש שלנו נכון - הבלוק המוצפן C_n זהה לבלוק המוצפן C_m. כל מה שהיינו צריכים זה לבטל במשוואה את האפקטים של ה-XOR, ורידדנו בכך את הבעיה לאותה בעיה אשר קיימת ב-Block Ciphers שאינם Chained.

כאמור, בעיה זו התגלתה כבר ב-2002, אך לא נמצאו לה כל כך יישומים פרקטים. עד BEAST.

אז איך מיישמים את כל מה שהצגנו כרגע? נחזור אל הדוגמא עם ה-AJAX. נניח כרגע לצורך הדוגמא שאני מסוגל לגרום לשכן שלי לשלוח את ה-Header-ים ואת גוף בקשת ה-POST בנפרד. כלומר - אותו חיבור, אותו רצף וצופן, רק שיש לי יכולת לעצור אחרי שליחת ה-Header-ים, להתבונן רגע בצופן, ולהמשיך הלאה לשלוח את גוף ההודעה. במצב כזה אני מסוגל בדיוק לבצע את המתקפה התיאורטית המוצעת לעיל.

אני יודע פחות או יותר באיזה בלוק מוצפן אמורה לעבור הסיסמא. ואני יכול ליישר אותה על בלוק בהתאם לצורך (לדוגמא: על ידי שינוי פרמטרים ב-POST). אני יודע מה הבלוק המוצפן האחרון שנשלח, ויש לי ניחוש מסוים לסיסמא שגם אותו אני מיישר על בלוק. בהנחה שאני צודק - הבלוק המוצפן שאני אראה עובר מהשכן שלי לכיוון הבנק יהיה זהה לבלוק שבו עברה הסיסמא.



יתרה מכך, יש עוד פרט שחשוב להתעכב עליו ומייעל מאוד את התהליך:

את הניחוש אני אבצע כל פעם עבור תו אחד. זאת אומרת, אני איישר את הבלוק של הסיסמא על המחרוזת "Cookie: password=X ", וכשאשלח בעצמי ניחוש, אשלח את המחרוזת " Cookie: password=1". אם צדקתי, אני אראה שני בלוקים מוצפנים זהים. אם טעיתי, אני אראה שני בלוקים מוצפנים שונים לחלוטין. בשיטה זו, כל פעם שאני צודק, אני מחסיר תו אחד מההתחלה, ומוסיף תו אחד של ניחוש בסוף, וכך אני מנחש את כל הסיסמא. הניחוש הבא יהיה "ookie: password=1X", זה שאחריו "okie: password=12X" וכן הלאה.

דוגמא:

POST /ABCDEFGH \r\n … Cookie: password=12345678\r\n …. \r\n\r\n password=X	POST /ABCDEFG \r\n … Cookie: password=12345678\r\n …. \r\n\r\n password=1X	POST /ABCDEF \r\n … Cookie: password=12345678\r\n …. \r\n\r\n password=12X

פעולה זו הופכת את הניחוש שלי מ-Bruteforce על כל הסיסמאות האפשריות (מה שבפועל יקח המון זמן), לניחוש מושכל על תו אחד כל פעם. בנוסף, במציאות סיסמאות לא באמת עוברות בצורה גלויה, ולכן לרוב נרצה לחלץ Token כלשהו כמו עוגיה או CSRF Token ובכך להשיג גישה לחשבון של הנתקף.

רוב המזהים הללו לרוב מכילים אותיות ומספרים בלבד (בסגנון - Base64), כך שעבור כל תו נצטרך בתיאוריה מקסימום של 64 ניחושים.

כל מה שהמתקפה מצריכה ממני למעשה היא יכולת האזנה לפקטות המוצפנות שעוברות בין השכן שלי אל הבנק, ויכולת הזרקה של מידע בינארי. אלא שאם הדרישה הראשונה היא יחסית סבירה (מי שחושב שה-NSA, הרוסים, הסינים, אדוארד סנואודן, גוגל ואמא שלו לא מצוטטים לו לקו - תמים), הדרישה השניה היא לא טריוויאלית, ואפילו לא טריוויאלית בכלל.

למה הדרישה הזו בעייתית? ראשית - כי AJAX, או כל טכנולוגיה אחרת לא באמת מפצלת את הבקשות לנוחיותנו ומאפשרת לשנות אותם תוך כדי. שנית, כי הדפדפן בגדול לא מעוניין שתהיה לנו יכולת להזריק סתם ככה מידע בינארי לקו. הייתה אפשרות לבצע פעולה כזו באמצעות טכנולוגיית WebSockets, אבל היא שונתה עקב בעיית אבטחה אחרת[7]. כיום המידע ה"בינארי" שנשלח באמצעות WebSockets, עובר קיסור עם מחרוזת באורך 32 ביט טרם שליחתו על הקו, כך שהמידע אשר נשלח (ולצורך העניין, מוצפן) אינו זהה בתוכנו על הקו למה שהיינו רוצים.

בפוסט על המתקפה מאת Thai Duong, הוא מונה מספר דרכים שבהם הם ניסו לשלוח מידע בינארי על הקו (דוגמאת WebSockets, Flash וכד'). בסופו של דבר הם מימשו את המתקפה שלהם עם Embedded Java Applet, וחולשת Same-Origin Policy (שאמנם לא חשפה עוגיה, אך כן העבירה אותה בבקשת Cross-Origin). כלומר, האתר הזדוני שלהם מטעין לדפדפן אפליקציית Java, שבתורה יוצרת קשר עם השרת ושולחת מידע בינארי בהתאם. חולשת ה-SOP נסגרה, אך עיקרון הפעולה של AES ו-SSL נותרו דומים.

וזהו כל התורה.

עברנו על הרבה חומר - הבנו מה יכול להיות בעייתי בצפני בלוק, איך אפשר לתמרן אותם גם במצב CBC, ואיך כל זה מתחבר פרקטית לעולם האינטרנט. נעים מאוד - BEAST.

[7] http://w2spconf.com/2011/papers/websocket.pdf

תגובה למתקפה, דרכי מניעה, ונקודות למחשבה

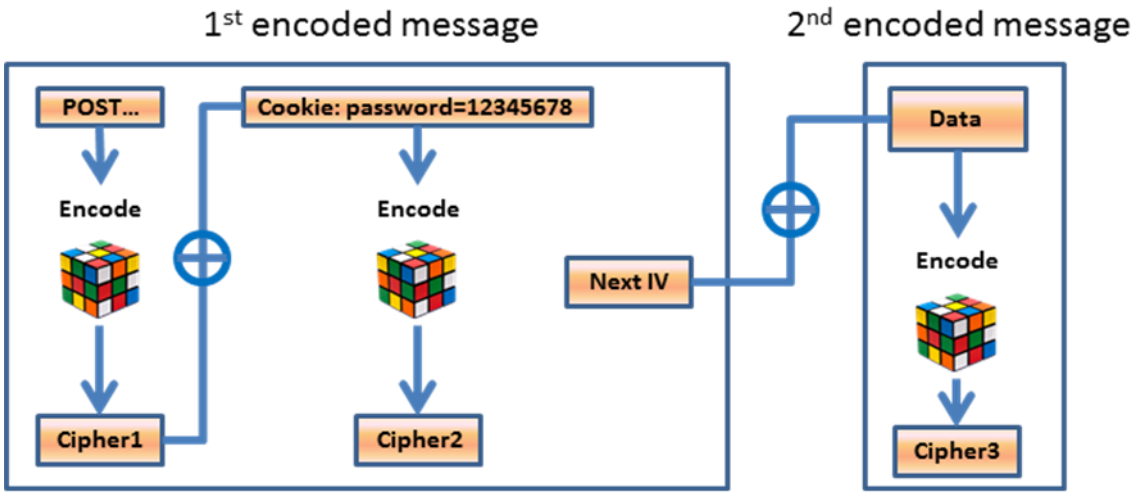
אם כן, לאחר שלמדנו על מתקפת BEAST, בואו נבין מה נעשה בכדי למגר את המתקפה.

בראש ובראשונה - ההמלצה המיידית של רוב חברות האבטחה הייתה לעבור לשימוש בצופן RC4 בלבד[8]. צופן זה הוא אינו "צופן בלוקים" ופועל בשיטה אחרת, ולכן נחשב באותה עת כמוגן יותר. אמנם גם בו כבר התגלו חולשות שונות, אך הבעיה שהתגלתה בשיטת העבודה הנוכחית הייתה משמעותית יותר, והמעבר אליו היווה תרופה ראשונית למכה.

בנוסף, בעת גילוי האפשרות למתקפה, פנו החוקרים ליצרניות הדפדפנים השונים בכדי שימצאו פתרון לבעיה. דבר הוביל לדבר, ואפילו ארגוני התקינה של הפרוטוקולים הנ"ל נכנסו לעניינים. המתקפה מוגרה במספר דרכים:

הדרך הראשונה, היא הדרך שנכנסה לתקן TLSv1.1[9]. בתוך כל הודעת SSL מוכל IV רנדומלי חדש, שישמש בתור הגורם המקסר לבלוק הבא. כלומר - "הודעת SSL" היא רצף מסוים של מספר בלוקים מוצפנים, ובסוף הרצף הזה ישלח IV רנדומלי חדש שישמש כמקסר לבלוק הבא. כיוון שה-IV הוא רנדומלי ואנחנו לא יכולים לחזות אותו (הוא עובר בתוך שכבת ההצפנה), אין לנו למעשה חיזוי של ה"בלוק המקסר", ולכן לא נוכל להוביל לביטול שלו ובכך לניחוש plaintext.

הרעיון יחסית פשוט, ומצריך שינוי במימוש של TLS אצל כל מי שרוצה לדבר בתקן המעודכן. זה נראה כך:



[8] דוגמא להמלצה של Microsoft למעבר לצופן RC4. http://blogs.msdn.com/b/kaushal/archive/2011/10/03/taming-the-beast-browser-exploit-against-ssl-tls.aspx

[9] http://www.ietf.org/rfc/rfc4346.txt 1.1 Differences from TLS 1.0, [CBCATT]

אמנם לא ניתן לראות בבירור בתרשים, אך ה-Next IV גם הוא מן הסתם עובר הצפנה ולא נשלח סתם ככה באוויר.

נוסף על כך, חשוב לציין שפרוטוקול TLSv1.1 לא אומץ כיוון שניתן היה לבצע לו מתקפת "שנמוך" (downgrade) ולמעשה די בקלות לגרום ללקוח והשרת לדבר חזרה בפרוטוקול TLSv1.0[10], שהוא כידוע - פגיע. משום כך, השדרוג המשמעותי באמת הוא ל-TLSv1.2, שבו לא ניתן לבצע מתקפת שנמוך, ומן הסתם מכיל את השינויים שבוצעו ב-TLSv1.1. שדרוג זה נחשב בעייתי כיוון שעבור חלק מהדפדפנים והאתרים הוא לא תואם לאחור, מה שהוביל לבעיות אצל לא מעט משתמשים ושירותים.

נקודה למחשבה - יכול להיות מעניין להבין מה קורה במקרים של פרגמנטציה של הודעות SSL (אני יודע שיש סוג של תמיכה בזה בתקן[11], אך לא טרחתי לבדוק לעומק), והאם אפשר בצורה כלשהי להזריק מידע "באמצע הודעה", או לגרום לכך שתחתך באמצע ולהמשיך מאותה נקודה.

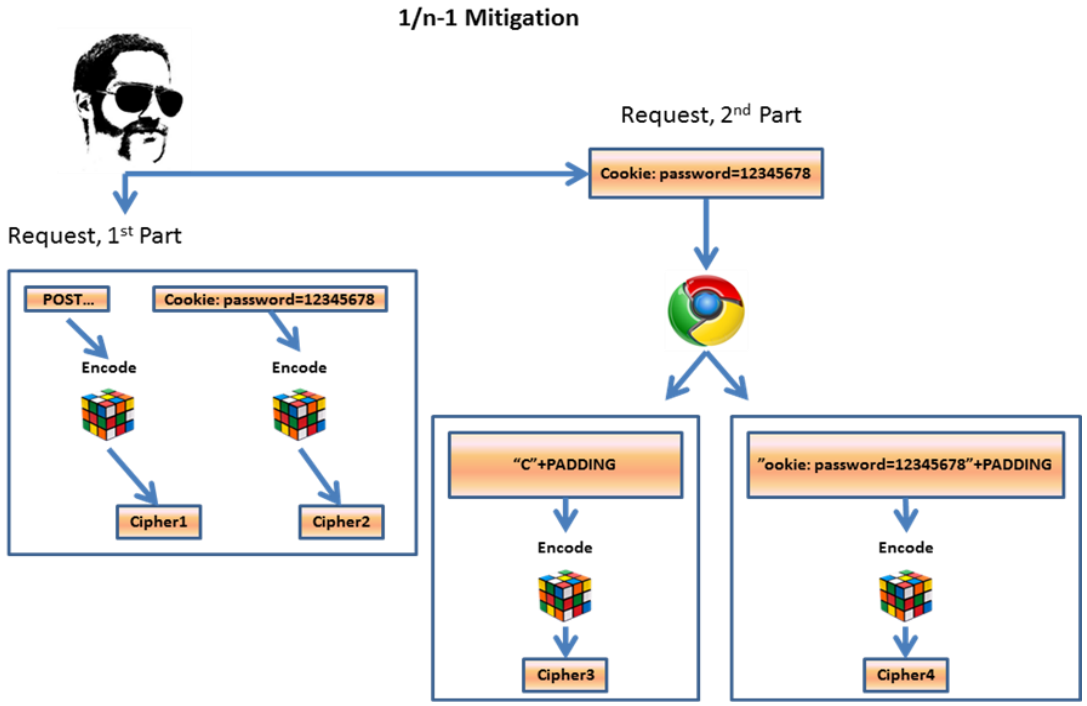
הדרכים הנוספות שבהן ניתן למגר את המתקפה הן לגרום לבלוק המוצפן הראשון בכל הודעה להכיל תוכן שאינו בהכרח בשליטת המשתמש. המשמעות היא שכאשר נרצה להזריק תוכן בינארי בשליטתנו, הבלוק הראשון יכיל מידע "זבל" שלמעשה יגרום לתוכן המוצפן להיות שונה ממה שאנחנו מצפים לו, ובכך לייצר מצב שהבלוק המוצפן שמכיל את הניחוש שלנו, יוביל לתוצאה שונה ממה שצפה לה מאשר הבלוק עם הסיסמא.

שיטות אלו הן שיטות אשר לא משנות את תקן TLS (לשנות תקן זה עסק בעייתי מאוד), אלא רק מעט מהמימוש שלו. בחלקן, הן אפילו תואמות לאחור עם המימושים הקיימים אשר פגיעים ל-BEAST.

הגישה אשר אומצה על ידי OpenSSL (עוד לפני גילוי BEAST, אך לא הופעלה), היא לשלוח את הבלוק הראשון עם מידע ריק (באנגלית צחה: "0\n"). כזכור, צפני בלוק מוסיפים ריפוד למידע אשר אינו מכיל מספיק תווים לגודל הבלוק. אם כן, הבלוק הראשון בכל הודעה הוא בלוק שמכיל רק ריפוד. קל מאוד להצפין אותו, וקל מאוד לפענח אותו. משמעות הפענוח למעשה לא מכילה מידע כלשהו, כך שתקינות ורצף המידע נשארים. גישה זו נוסתה על ידי דפדפנים שונים, אך עקב בעיות תאימות, אומצה גישה אחרת בדפדפנים, דומה מאוד.

הגישה השניה לוקחת את הבלוק הראשון אשר אמור להיות מוצפן, ומפצלת אותו על גבי שני בלוקים (באנגלית צחה: "1\n-1")[12]. בבלוק הראשון מוצפן התו הראשון כולל ריפוד, ובבלוק השני כל שאר התווים כולל ריפוד. שיטה זו מונעת מצבים שונים בהם הדפדפן מניח שהתקבל התו null, ובכך מפסיק את רצף המידע.

[10] http://www.educatedguesswork.org/2012/07/problems_with_secure_upgrade_t.html
[11] http://tools.ietf.org/html/rfc5246#section-6.2.1
[12] https://www.imperialviolet.org/2012/01/15/beastfollowup.html



גם פה מעניין להבין אם יש דרכים שבאמצעותן אפשר להזריק בלוק "באמצע", ולא בהכרח "בהתחלה", כך שהבלוק הראשון עדיין יעבור שינוי, אך הבלוקים האחרים יהיו ידועים וניתן יהיה לנחש באמצעותם תוכן מוצפן.

לסיכום, מה אפשר לעשות עם זה?

נכון להיום, נראה כי מתקפת BEAST נמנעה לחלוטין וניתן לקנות שטויות מ-ebay בראש שקט בלי לחשוש מגניבת זהות או כרטיס אשראי. עם זאת, חשוב להבין שעקרונות הפעולה של Block Ciphers נשארו זהים, כך שהמתקפה התיאורטית עדיין אפשרית. כל מה שצריך הוא יכולת האזנה ללקוח, ויכולת לגרום לאותו לקוח להזריק מידע בינארי לקו. אמנם הזרקה בינארית היא לא פעולה טריוויאלית, וכבר יש מודעות להשלכותיה בעולם, אך עם זאת, ייתכן שעדיין יש או יתווספו טכנולוגיות אשר מאפשרות לבצע פעולה זו.

בנוסף, התקיפה המדוברת מתייחסת אך ורק לדפדפני אינטרנט. כאמור, גם ביישומים אחרים אשר משתמשים ב-SSL, או ב-CBC לצורך העניין ניתן למצוא את הבעיות הללו ולנצל אותן שם.

לבסוף חשוב לזכור שתשתית האינטרנט לא משתדרגת בן רגע, וייתכן שיש עדיין לא מעט אתרים שעובדים עם TLSv1.0, ולא שדרגו עדיין ל-TLSv1.2. כמו כן, תמיד קיימת אפשרות שמישהו טעה במימוש התקן...

סוף דבר

אז, בסופו של דבר עבר פה הרבה מאוד טקסט ותוכן. עם זאת, אני מקווה מאוד שהמאמר היה ברור ומובן. ניסיתי כמה שיותר להוסיף אליו תרשימים, צבעים וחלקים פרקטיים בכדי ללמוד מתוך הדוגמאות. אני מקווה גם שצחקתם פה ושם תוך כדי קריאה ונהנתם.

כמובן שאי אפשר לסיים בלי תודות:

ברצוני להודות בראש ובראשונה לצוות DigitalWhisper - על העבודה המעולה שהם עושים, ועל הזכות לפרסם את המאמר. בנוסף, אני מודה לחברי משכבר הימים, איציק, שנתן הערות בונות מצוינות טרם פרסום המאמר.

אשמח מאוד לקבל משובים על המאמר, פניות, הערות שאלות בנושא, רעיונות וכל מה שצץ בראשכם.

תודה רבה על הקריאה,

יוחאי אייזנרייך / Yochai Eisenrich

echelonh@gmail.com

ביבליוגרפיה / קריאה נוספת:

על אף שחלק מהלינקים כבר הופיעו במאמר, ראיתי לנכון לאגד את רשימת המקורות שעליהם הסתמכתי בכתיבה.

תיאור מתקפת BEAST בבלוג של Thai Duong:

http://vnhacker.blogspot.co.il/2011/09/beast.html

סיכום מתקפת BEAST באתר נוסף באנגלית:

http://www.educatedguesswork.org/2011/09/security_impact_of_the_rizzodu.html

מאמר ב-MSDN המסביר בצורה טובה על BEAST, כולל תרשימים:

http://blogs.msdn.com/b/kaushal/archive/2011/10/03/taming-the-beast-browser-exploit-against-ssl-tls.aspx

המתקפה כפי שקוטלגה בNational Vulnerability Database:

http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2011-3389

קומיקס המתאר את צורת הפעולה של AES:

http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html

עמוד וויקיפדיה האנגלית אשר מסביר על Cipher-Block Chaining:

http://en.wikipedia.org/wiki/Cipher_block_chaining#Cipher-block_chaining_.28CBC.29

תקן TLSv1.2:

http://tools.ietf.org/html/rfc5246

Transport-Layer Security בוויקיפדיה:

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