

криיפטוגרפיה ובטחת מידע | 67392

הרצאות | פרופ' גיל שגב

כתיבה | נמרוד רק

תשפ"ב סמסטר ב'

בתחילת כל תרגול או חלק של הרצאה מופיע קישור לסרטון המקורי של חלק זה. תרגולים לפי מר ליאור רותם. תוקנו כמה שגיאות בעזרת סיקום של דוויד קיסר-شمידט ודניאל דינץ'ב.

תוכן העניינים

5	סקירה כללית של הקורס
6	הצפנה סימטרית
6	I מבוא לקריפטוגרפיה מודרנית
6	א קריפטוגרפיה קלאסית מול מודרנית
6	ב הגדרה פורמלית של הצפנה סימטרית וצפנים היסטוריים
8	ג עקרונות יסוד בקריפטוגרפיה מודרנית
8	ד סודיות מושלמת ומגבלה
11	תרגול
13	II הגדרות בטיחות חשובה ומחוללים פסאודו-אקראים
13	א גישות לבטיחות ופרמטר הבטיחות
15	ב מערכות עם הצפנות בלתי ניתנת להבחנה
15	ג מחוללים פסאודו-אקראים
17	ד שימוש במחולל פסאודו-אקראי לבניית מערכת הצפנה
18	ה א המשך הצפנה בלתי ניתנת להבחנה
20	תרגול
22	III פונקציות פסאודו-אקראיות ו-CPA
23	א ארגומנט היברידי
24	ב מתפקיד CPA
24	ג פונקציות פסאודו-אקראיות
25	ד בטיחות CPA
27	ה א צופני בлокים
28	תרגול
32	IV פונקציות פסאודו-אקראיות עמוק
32	תרגול
36	V אימות, פונקציות האש והצפנה מאומנת
36	א מבוא למערכות אימות ואמות הودעות ארוכות
40	ב Hash-and-Authenticate
42	ג הצפנה ואימות גם יחד
43	תרגול

47	אבטחת תוכנה
47	VI פגיעות תוכנה ברמה הנמוכה
47	א ניהול זיכרון במערכות הפעלה
50	ב Buffer-Overflow
51	ג הזרקת קוד
52	ד פרצחות זיכרון נוספות
54	ה פרצחות פורמט מחרוזת
55	תרגול
58	VII הגנות מפני מתקפות ברמה הנמוכה
58	א בטיחות זיכרון
61	ב קנריות מחסנית ו-ASLR
61	ג Control-Flow-Integrity
63	ד תוכנות בטוחות
66	תרגול
68	VIII זיהוי מתקפות ברמה נמוכה
68	תרגול
70	IX בטיחות רשות
70	א מבוא ל-World-Wide-Web
72	ב SQL Injection
74	ג Session וمتקפות עליו
77	ד Javascript בעד ונגד
78	תרגול
80	X ניתוח קוד סטטי
80	א מבוא לאנליזה סטטיטית של קוד
81	ב Flow-Analysis
83	ג Path-Sensitivity ו-Flow-Sensitivity
85	ד Context-Sensitivity
87	ה Information-Flow
89	תרגול
91	XI ניתוח קוד סטטי לעומק
91	תרגול
96	XII Fuzzing ו-Pen-testing
96	א מבוא להרצת סמלית
97	ב יסודות ההרצת הסמלית
99	ג הקושי החישובי בהרצת סמלית
100	ד Fuzzing
102	תרגול

סקירה כללית של הקורס

קישור

הקורס מספק מבוא לקריפטוגרפיה ולאבטחת מידע, שהם שני תחומיים שונים, אך בעלי זיקה חזקה ביניהם.

קריפטוגרפיה

נלמד על כלים ומערכות קריפטוגרפיות, לרבות כיצד להוכיח את בטיחותן וכייזד להשתמש בהן נכון ונחלק את החלק הזה לשני סוגים שונים של הצפנה :

- הצפנה סימטרית עם מפתחות סודיים (שבוע ~1-5).
- הצפנה א-סימטרית עם שילוב של מפתחות סודיים ופומביים (שבוע ~14-13).

בכל אחד מהנושאים נעמיק בשני מאפיינים חשובים שלהם - סודיות (Secrecy) ואמיות (Integrity).

אבטחת תוכנה

נלמד (שבוע ~12-6) את יסודות אבטחת התוכנה, ובכלל זה כיצד לנצל פירצאות אבטחה וכייזד להימנע מזה ולמגרנו. חלק זה כולל שלושה פרקים :

- מתקפות ומגננות ברמה נמוכה (Low Level), לדוגמה השמה עקיפה וזדונית של ערכיהם בΡΙΤΗ קוד C.
- אבטחת רשת, לדוגמה ניצול פירצאות בשפת SQL לניהול מסדי נתונים.
- בדיקה אוטומטית של קוד וניסוח וכייזד לבדוק האם קוד הוא פגיע לפרצאות אבטחה באמצעות שיטות שונות, לדוגמה ניתוח סטטי, Fuzzing והרצה סימבולית.

שבוע II | מבוא לקריפטוגרפיה מודרנית

הרצאה

חלק א' של הרצאה (קישור)

מהי קריפטוגרפיה? בעבר הייתה אמונה עתיקה שמטרתה יצרת ערוץ תקשורת סודית בין שני גורמים. בטרמינולוגיה קלאסית (שהה נשתמש במקרה כל הקורס), Alice ו-Bob רוצים לתקשר (בсадiotic) ו-Eve (הרמו ל-Evil) מאזינה לערז התקשורת שלהם. בעבר קריפטוגרפיה שומשה רק לצבאות וארגוני מודיעין גדולים. בנוסף, קריפטו' קלאסי התמקד ביצירתיות ויכולת אישית ולא בכלי מדעיים. לעומת זאת, גרמה לכך שקריפטו' עד המאה ה-20 היה מעגל אינסופי

תכnu ← שבירה ← תיקו ← שבירה ← תיקו ← ...

למי שיש זמן Aiיכשו בסMASTER הקרוב, מוזמנים/ות לקרוא [ספר מאוד מעניין](#) על שימושים של קריפטו בעת החדש, בדגש על מלחמת העולם השנייה.

לקראת סוף שנות ה-70, אירע שינוי קיצוני בקריפטוגרפיה והיא החלה להתבסס על מידע יציב ומשמשת להרבה יותר מתקשורת סודית (לדוגמה T-NFTים, לצערנו). קריפטו' זמין לכלנו כויס, ולא נחלתם של צבא אריה'ב בלבד.

במשפט אחד, קריפטוגרפיה היא העיסוק המדעי בשיטות לתכnu מערכות שעמידות בפני התנהגות עוינת. המשפט הזה מכיל מידע סמיוי - מה היא מערכת? מה היא התנהגות עוינת?

התנהגות עוינת היא כל התנהגות שמתכוון המערכת לא הגדר כהתנהגות רגילה או מכוונת. מערכת היא כל אובייקט שעבורו הגדרנו התנהגות קלט-פלט או התנהגות מסוימת לאורך זמן (לא באמצעות מוגדר היפט). מערכות אלו כוללות הצפנה וחומרה דיגיטלית אבל גם מערכות זיהוי ביומטרי, שירותים בנקאים וכיוצא'ב.

חלק ב' של הרצאה (קישור)

מערכת הצפנה סימטרית מאפשרת לשני אנשים לתקשר בצורה סודית אפילו אם גורם שלישי מازין לערז התקורת. לעומת זאת, נניח כי איבר רק מאזינה לערז, כלומר שהיא פסיבית. במערכת כזו נctrיך לשבור את הסימטריה בין אליס ובוב לבין איבר, כי אחרת כל דבר שהם יכולים לעשות גם היא יכולה. מכאן, מגיעה ההנחה הבסיסית בקריפטוגרפיה סימטרית והוא שלאליס ובוב יש מפתח סודי שרק שניהם יודעים, והוא משמש גם להצפנה וגם לפענוח. הנחה זו אינה טריויאלית שכן לעיתים קשה להגיע למצב כזה.

מערכת הצפנה סימטרית מורכבת מהצפנה סימטרית:

• פונקציית ייצור מפתחות : KeyGen (algo' הסתברותי) שפולט מפתח $K \in \mathcal{K}$ (לדוגמה מגילה באופן אחד על מחרוזות באורך 1024 ביטים), כאשר \mathcal{K} הוא אוסף כל המפתחות האפשריים.

• אלגוריתם ההצפנה : Enc (algo' דטרמיניסטי או הסתברותי) המקבל $K \in \mathcal{K}$, $m \in \mathcal{M}$ ופולט הודעה מוצפנת $c \in \mathcal{C}$ כאשר \mathcal{M} אוסף כל ההודעות האפשריות (plaintext) ו- \mathcal{C} הוא אוסף כל ההודעות המוצפנות האפשריות (ciphertext). אם Enc הסת' אז יש אוסף ההודעות מוצפנות שכל הודעה רגילה יכולה להיות מוצפנת אליהן.

- אלגוריתם הפענה : $\text{Dec}(\text{alg}, \text{דרמייניסטי})$ מקבל $c \in \mathcal{C}$ ופולט הודעה מפוענתה $m \in \mathcal{M}$.

מעיצוב זה אנו מבינים את משמעותו ה-“סימטריה” - גם הצפנה וגם הפענה מותבסים על אותו מפתח, את יכולות לבצע פעולות שונות כMOV.

עתה נשתמש בשני סימונים להשמה: ” $=$ ” עבור השמה דרמייניסטית, לדוגמה $m = \text{Dec}_k(c)$ ו-” \leftarrow ” עבור השמה הסתברותית, לדוגמה $.c \leftarrow \text{Enc}_k(m)$ או $k \leftarrow \text{KegGen}()$

יש כמה דרישות על מערכת הצפנה סימטרית:

- נכונות : $\text{Dec}_k(\text{Enc}_k(m)) = m \forall k \in \mathcal{K}, m \in \mathcal{M}$.

- סודיות (נדון בהמשך).

בנייה הסודיות של המערכת, נניח ש- $\text{KeyGen}, \text{Enc}, \text{Dec}$ ידועות בפורמי, והדבר היחיד שסודי הוא המפתח k (זהו עקרון קרכחופס, שמתבסס למעשה על המאפיות).

הצפנות ההיסטוריות

- צופן היסט (צופן קיסר).

$.k \leftarrow \{0, \dots, 25\}$ דוגם באופן אחד KegGen

$\text{Dec}(.A, \dots, Z), \mathcal{M} = \{a, \dots, z\}^\ell, \mathcal{C} = \{A, \dots, Z\}^\ell$ מסיטה כל אות k פעים קדימה (כאשר אם עברנו את Z נחזור חזרה ל- A). מסיטה חזרה כל אות.

הצפן הזה אינו בטוח כלל נכון שיש רק 26 אפשרויות למפתחות ומעבר על כל בולן בוודאי יחשוף את ההודעה המקוריות בשל המבנה התחרيري של משפטים באנגלית ואוצר המילים המוגבל למשפט באנגלית.

הנה עיקרונו חשוב ככל - אסור שאוסף המפתחות \mathcal{K} יאפשר מעבר על כל המפתחות בזמן סביר!

- צופן החלפה.

$\text{Dec}(.c[i] = k(m[i]), k, \text{Dec}(.c) \leftarrow \{a, \dots, z\}^\ell, \mathcal{C} = \{A, \dots, Z\}^\ell$ וכאן $\text{Enc}(m) = \{a, \dots, z\}^\ell$ ו- $\text{KegGen}()$ דוגם באופן אחד על פרמוטציות על $\{a, \dots, z\}^\ell$ כל אות באות המתאימה לה מהתמורה, כלומר $c[i] = k(m[i])$. כלומר כל אות במספרים k^{-1} מחליף כל אות בערך שלה בpermotziah ההופכית.

כאן, גודל אוסף המפתחות הוא $2^{88} \approx 26!$ שהוא לא פרקטית מבחינה מעבר על כל המפתחות (מה גס שיש כמו פרמוטציות שיתנו מפתחות הגיוניים).

עם זאת, בשל ההחלפה הקבועה של אותיות, היא משמרת לחלוטין את ההתפלגות של השימוש באותיות באנגלית וכך ניתן לחשב בקלות לפצח את הצפן.

- צופן ויז'נר.

$\mathcal{C} = \{A, \dots, Z\}^\ell$ ו- $\mathcal{M} = \{a, \dots, z\}^\ell$, $k = k_0 \dots k_{t-1} \leftarrow \{0, \dots, 25\}^t$ דוגם וקטור מספרים KegGen מסיט את האות i -ה- t בעדים קדימה. $\text{Dec}(k_i \bmod t, \text{Enc}(\text{alg}, \text{דרמייניסטי}))$ מסיט את האות i -ה- t לאחר מכן.

לדוגמה, $\text{Enc}_k(\text{hello}) = \text{IGOMQ}, k = 1, 2, 3$ ו $k_4 \bmod 3 = 2$ (כי צעד אחד קדימה מ- h זה i , שני צעדים מ- l זה M כי $2 \bmod 3 = 2$ ו כיווץ ב').

כאן אי אפשר לעبور על כל אוסף המפתחות ואין מתקפה סטטיסטית ישירה, אבל עם מספיק התחכਮות כן אפשר לפצח את זה.

לסיום, למדנו שאוסף המפתחות צריך להיות מאוד גדול וגם שאי אפשר לעשות פרMOVוטציה קבועה על אותן אותיות, אבל שני אלה הם תנאי הכרחיים אבל לא מספיקים.

חלק ג' של הרצאה (קישור)

לקRIPTוגרפיה המודרנית שלושה עקרונות בסיסיים בעת בניית מערכות:

1. יש לפרמל במדויק את ההגדרה של בתיות, אחרת לא יוכל לדעת האם השגנו אותה בכלל או לא. הגדרה זו דורשת שני חלקים:

- תיאור של ההתקפות מפניהם אנו מגנים, לרבות אופן הגישה של התקופים למידע בהתקפות אלו.
- תיאור של מה נחשב בעינינו כפריצת הבתיות. מצד אחד באופן ליירלי ניתן להגיד זאת כתוקף שהשיג את המפתח הסודי ומהצד השמרני כתוקף שהשיג כל מידע לא טרויואלי על הודעה, לדוגמה הבית השביעי.

חשוב לחשב על שני רכיבים אלו כי רק אז יוכל להתאים את הדרישות שלנו לישום המערכת, למשל מערכת פשוטה לא דורשת הצפנה **overkill** וזה יכול לבוא על חשבון ייעילות.

2. יש לתאר את ההנחות הבסיסיות עליהם אנו משתמשים, למשל מספר גודל לגורמי הראשוניים בזמן סביר. החשיבות של תיאור זה היא שההנחות לעיתים מוכחות כלל נכונות, ולכן חשוב לפרסם את ההנחות בפורמי כדי לתת למומחים בנושא לבחון אותן לעומק ולקבע את נכונותן (סבירות גבוהה). יתרון נוסף של כך הוא השוואת של מערכות אחרות לשניה, וכך להגיע לאייזון של הגדרת הבתיות עם ייעילות וכו'.

3. יש להוכיח שההנחות במקורו השני, לא ניתן לבצע מתקפות כפי שהוגדרו במקורו הראשון.

עתה קיבלנו כי מצד אחד הוכחנו מתמטית שהמערכת בטוחה אבל מצד שני מערכות כן נשברות - מוזר מאוד. עם זאת, פרדיגמה זאת מאפשרת לנו לנתח מדוע המערכת נשברת:

1. הגדרת הבתיות לא הייתה מספיק חזקה ולכן נרצה לחזק אותה.
2. אחת ההנחות מהמקורו השני הופרcha, למשל אם נמצא אלגוריתם יעיל לפירוק מספר לגורמים ראשוניים.
3. הוא שההנחות וההגדרות נכונות אבל שהמערכת פשוט לא מומשה כמו שצריך, ועל כךណו בחלוקת התוכנה בקורס.

תיכון כזה של מערכות משפר את השיטה הקלאליסטית של קריפטו', שכן עתה יש לנו הכוונה מדוע למערכת נשברת, ולא סתם לתקנה באופן בלתי פורמלי ואולי לא אפקטיבי.

חלק ד' של הרצאה (קישור)

נרצה לפרמל את העיקרונו שאם איבר רואה הודעה מוצפנת, הוא לא לומדת שום דבר חדש על הודעה המקורית.

הגדירה עבור מערכת הצפנה הסימטרית $(\text{KegGen}, \text{Enc}, \text{Dec})$ המ"מ המייצג את התפלגות המפתח. נגידר K עם מפתח משותף $k \leftarrow \text{KegGen}()$.

דוגמה עבור צופן קיסר, $.K \sim \text{Unif}(\{0, \dots, 26\})$

הגדירה נסמן ב- M המ"מ של התפלגות ההודעות \mathcal{M} .

דוגמה $P(M = \text{later}'' \text{Attack}) = 0.25, P(M = \text{now}'' \text{Attack}) = 0.75$

נניח שאיבר ידעת מהו M .

הגדירה K, M מגדירים התפלגות על ההודעות המוצפנות, נסמנה $.C = \text{Enc}_K(M)$

הגדירה נאמר כי $\Pi = (\text{KegGen}, \text{Enc}, \text{Dec})$ היא סודית מושלמת אם לכל התפלגות M על \mathcal{M} ולכל $m \in \mathcal{M}$ ולכל $c \in \mathcal{C}$ נקיים $P(C = c | M = m) > 0$

$$P(M = m | C = c) = P(M = m)$$

כלומר, בהינתן שהצפן הוא c , לא נודע שום דבר חדש על התפלגות M (ומכאן מהי m).

מסקנה למעשה, Π היא סודית מושלמת אם M, C ב"ת.

טענה צופן היחסטה והחלפה אין סודיות מושלמת לכל $\ell = 1$

הוכחה: (עבור צופן היחסטה, זה דומה לשאר המערכות) כדי להוכיח שמערכת אינה סודית מושלמת, נctrיך למצוא M, m, c כך שהמשווהה לא מתקיימת. לדוגמה, עבור $(\text{aa}, \text{ab}) \sim \text{Unif}(\{\text{aa}, \text{ab}\})$ שכנא אף היחסטה לא מקיימת זאת אבל ■ $P(M = \text{aa} | C = \text{AB}) = 0$. כלומר, בהינתן הצפן AB, יכולנו לשולח חלוטין את ההודעה המקורית aa שזה דבר שאנו לא רוצים شيקרה.

הגדירה מערכת One Time Pad מוגדרת באופן הבא: $(P(K = k) = \frac{1}{2^\ell}, k \leftarrow \{0, 1\}^\ell, \text{Dec}_k(c) = c \oplus k, \text{Enc}_k(m) = m \oplus k)$

טענה OTP היא מערכת נכונה.

הוכחה: \oplus (פעולת XOR) היא סימטרית ולכן $\text{Dec}_k(\text{Enc}_k(m)) = m \oplus k \oplus k = m$

משפט OTP היא סודית מושלמת עבור הודעות באורך ℓ .

הוכחה: תהי M התפלגות על \mathcal{M} , ו- $m \in \mathcal{M}$.

$$\begin{aligned}
P(C = c) &= \sum_{\omega \in \mathcal{M}} P(M = \omega) P(C = c | M = \omega) \\
C = c \iff K = c \oplus M &= \sum_{\omega \in \mathcal{M}} P(M = \omega) P(K = c \oplus \omega | M = \omega) \\
&\stackrel{\text{כג' }}{=} \sum_{\omega \in \mathcal{M}} P(M = \omega) P(K = c \oplus \omega) \\
&= \frac{1}{2^\ell} \sum_{\substack{\omega \in \mathcal{M} \\ =1}} P(M = \omega) \\
&= \frac{1}{2^\ell}
\end{aligned}$$

$$\begin{aligned}
P(M = m | C = c) &\stackrel{\text{כלבייש}}{=} \frac{P(C = c | M = m) P(M = m)}{P(C = c)} \\
&\stackrel{\text{כך מוגדרת הצפנה}}{=} \frac{P(K = c \oplus m) P(M = m)}{P(C = c)} \\
&= \frac{\frac{1}{2^\ell} P(M = m)}{\frac{1}{2^\ell}} \\
&= P(M = m)
\end{aligned}$$

■ **כלומר** $P(M = m | C = c) = P(M = m)$.

הערה ל-OTP יש כמה חסרונות:

- אורך המפתח חייב להיות זהה לאורך ההודעה, שזו בעיה כשמדבר בהודעות ארוכות (לדוגמה קובץ וידאו של 1 MB) שהוא באורך מיליון ביטים).

- OTP מצפינה הודעה אחת בלבד. עבור שתי הודעות, נניח כי c מתקיים $c = \text{Enc}_k(m), c' = \text{Enc}_k(m')$ משיג את המפתח בклות.

- אם ידועה ההודעה m לתוקף, וגם c, c' , אז $k = m \oplus c$ משיג את המפתח בклות.

הערה נרצה אם כן לעצב מערכת שמספקת את הגדרה הסודיות המשולמת אבל עשויה שימוש במפתחות קצרים יותר, האם זה אפשרי? OTP היא המערכת "חייב טובה" שמקיימת את הסודיות המשולמת, ולכן ככל מערכת אחרת שהיא סודית משולמת יהיה לה גם את אותן החסרונות, שכן התשובה היא לא.

משפט תהי Π מערכת הצפנה סימטרית עם אוסף מפתחות \mathcal{K} והודעות \mathcal{M} . אם Π סודית משולמת אז $|\mathcal{K}| \geq |\mathcal{M}|$.

הוכחה: נניח בשילילה כי $|\mathcal{K}| < |\mathcal{M}|$ ונגיע לסתירה. תהי M התפלגות האחדה על \mathcal{M} , הودעה $c \in \mathcal{C}$ ו- $m \in \mathcal{M}$.

נגיד $\text{Enc}_k(m) = k$

$$\mathcal{M}(c) = \left\{ \hat{m} : \hat{m} = \text{Dec}_{\hat{k}}(c), \hat{k} \in \mathcal{K} \right\}$$

כלומר אוסף ההודעות שניתנות כפונקציה של c עברור מפתח כלשהו. מתקיים $|\mathcal{M}(c)| \leq |\mathcal{K}|$ לפחות מההנחה $|\mathcal{M}(c)| < |\mathcal{M}|$. ככלומר, קיים $m^* \in \mathcal{M}(c)$ ש- $m^* \notin \mathcal{M}$. ככלומר אין אף מפתח שמספרנו את ההפנה $c \rightarrow m^*$. לכן

$$P(M = m^* | C = c) = 0 \neq \frac{1}{|\mathcal{M}|} = P(M = m^*)$$

בסתירה להגדרת הסודיות המושלמת. ■

תרגול

[קישור](#)

שאלה תהי Π מערכת הצפנה. הוכחו כי Π היא בעלת סודיות מושלמת אם ו惩ן לכל התפלגות M על \mathcal{M} ולכל $m_0, m_1 \in \mathcal{M}$ המתקבלות בהסת' חיובית לפי M ולכל $c \in \mathcal{C}$ מתקיים

$$P(C = c | M = m_0) = P(C = c | M = m_1)$$

כלומר ש- Π סודית מושלמת אם ובו הידועה מוצפנת c , לא נבחן בין המקרה בו זה m_0 למקרה בו זה m_1 יותר משליכנו לפני.

פתרון ⇐: תהי M התפלגות על מרחב ההודעות \mathcal{M} . נוכיח ראשית כי $\forall m \in \mathcal{M}$ המתקבלת בהסת' חיובית ולכל הצפנה $c \in \mathcal{C}$ מתקיים $P(C = c | M = m) = 0 \wedge P(C = c) = 0$ אם $P(C = c | M = m) = P(C = c)$ כרצוי. אחרת,

$$\begin{aligned} P(C = c | M = m) &\stackrel{\text{גיא}}{=} \frac{P(M = m | C = c) P(C = c)}{P(M = m)} \\ &= \frac{P(M = m) P(C = c)}{P(M = m)} \quad \text{סודיות מושלמת} \\ &= P(C = c) \end{aligned}$$

עבור $m_0, m_1 \in \mathcal{M}$ המתקבלות בהסת' חיובית ו- $c \in \mathcal{C}$ מתקיים $P(C = c | M = m_0) = P(C = c) = P(C = c | M = m_1)$ נוכיח סודיות מושלמת. תהי M התפלגות על \mathcal{M} . נוכיח ראשית כי \Rightarrow

כאשר m מתקבלת בהסת' חיוויות,

$$\begin{aligned}
 P(C = c) &\stackrel{\text{הסת' שלמה}}{=} \sum_{m' \in \mathcal{M}} P(C = c | M = m') P(M = m') \\
 &= \sum_{m' \in \mathcal{M}} P(C = c | M = m) P(M = m') \\
 &= P(C = c | M = m) \underbrace{\sum_{m' \in \mathcal{M}} P(M = m')}_{=1} \\
 &= P(C = c | M = m)
 \end{aligned}$$

$\forall m \in \mathcal{M}$ המתקבלת בהסת' חיוויות ו- $\forall c \in \mathcal{C}$ המתקבלת בהסת' חיוויות מתקיים

$$\begin{aligned}
 P(M = m | C = c) &\stackrel{\text{כיוון}}{=} \frac{P(C = c | M = m) P(M = m)}{P(C = c)} \\
 &= \frac{P(C = c | M = m) P(M = m)}{P(C = c | M = m)} \\
 &= P(M = m)
 \end{aligned}$$

שזו בדיקת הגדלת הסודיות המשולמת (עבור $M \in \mathcal{M}$ עם הסט' 0 ברור שמתקיים השוויון של הסודיות המשולמת).

שאלה הוכחו/הפריכו : לכל מערכת הצפנה II בעלת סודיות משולמת, לכל התפלגות M על \mathcal{M} ולכל זוג הצפנות $c_0, c_1 \in \mathcal{C}$ המתקבלות

$$P(C = c_0) = P(C = c_1)$$

פתרון לא נכון! תהי II מערכת הצפנה בעלת סודיות משולמת, נגדיר $(\text{KegGen}, \text{Enc}', \text{Dec}')$ Π' באופן הבא :

כלומר $b \sim \text{Ber}\left(\frac{3}{4}\right)$. האופרטור \parallel מייצג הצמדה של מחרוזות. כלומר $\text{Enc}'_k(m) = \text{Enc}_k(m) \parallel b$ מצפינים וממצידים בית לא הוגן.

מuttleם מהבית האחרון בהצפנה ומפענח את השאר עם Dec' .

תהי M התפלגות על \mathcal{M} ונסמן את המ"מ המתאים להצפנה של Enc , לביט שנדגים ב- Enc' ולהצפנה של Enc' בהתאם. $P(C' = c_0), P(C' = c_1) > 0$. $c_0 = c \parallel 0, c_1 = c \parallel 1$ ונתבונן בהצפנות II בהסת' חיוויות המתקבלת על C, B, C' .

תהי $c \in \mathcal{C}$ הצפנה המתקבלת בהסת' חיוויות על ידי II ונתבונן בהצפנות II בהסת' חיוויות לא מתקיים.

אבל B לא הוגן ולכן $P(C' = c_1) \neq P(C' = c_0)$.

נותר לחוכיח כי Π' סודית מושלמת. לכל $c \in \mathcal{C}, m \in \mathcal{M}$ מתקיים $b \in \{0, 1\}$ המתקבלת בהסת' חיובית ולכל

$$\begin{aligned} P(M = m | C' = c | b) &\stackrel{\text{ב''ג}}{=} \frac{P(C' = c | b | M = m) P(M = m)}{P(C' = c | b)} \\ &= \frac{P(C = c \wedge B = b | M = m) P(M = m)}{P(C = c \wedge B = b)} \\ \mathcal{C}, \mathcal{M} \text{ - } B &= \frac{P(C = c | M = m) P(B = b) P(M = m)}{P(C = c) P(B = b)} \\ &= \frac{P(C = c | M = m) P(M = m)}{P(C = c)} \\ \Pi' \text{ סודית מושלמת} &= \frac{P(C = c) P(M = m)}{P(C = c)} \\ &= P(M = m) \end{aligned}$$

כלומר Π' בעל סודיות מושלמת.

שאלה כמשמעותם ב-OTP עם $k = 0^\ell$ מתקיים $\text{Enc}_k(m) = k \oplus m = m$ ככלומר ההודעה נשלחת כמו שהיא שזה לא טוב. עברו המערכת Π' שזזה ל-OTP רק שהיא דוגמת איחיד על $\{0, 1\}^\ell \setminus 0^\ell$. הוכחו/הפריכו: Π' היא סודית מושלמת.

פתרון לא נכון! תהי M התפלגות איחידה על $\{0, 1\}^\ell$ ותהי $m \in \mathcal{M}$. אז עבור $C = m$

$$P(M = m | C = m) = P(K = 0^\ell) = 0 \neq \frac{1}{|\mathcal{M}|} = P(M = m)$$

בסתירה להגדרת הסודיות מושלמת.

שבוע III | הגדרות בטיחות חישובית ומחוללים פסאודו-

אקראים

הרצאה

חלק א' של הרצאה ([קישור](#))

סודיות מושלמת מוגבלת להצפנה של הودעה ארוכה כארוך המפתח ורק הודעה אחת, נרצה למצוא הגדרה מעט חלשה יותר שתהיה יותר שימושית.

סודיות חישובית מתייחסת למצבי יוטר פרקטוי, לפיו לתוקף יש איזושהי מגבלה חישובית על התוקף, ומוגדרת ע"י הצפנה שלא קביל חישובית לגłówות מידע שימושי באמצעותה על ההודעה המקורית.

הקלות מציאותיות

1. המתקפה ממנה אנו מגנים מוגבלת בכוח החישוב, לדוגמה כזו שמתמשת במוחשב העל המתקדם ביוטר כיום שרצה לפחות 2000 שנה.

2. המתקפה ממנה אנו מגנים תצליח בהסת' זניחה, כמו פעם ב-000, 10 במשמעותו, דבר שלמעשה לא פוגע באמת בביטחון ההצפנה.

הגדירה לפי הגישה הקונקרטית, סכמה היא (t, ϵ) -בטוחה אם לכל תוקף שרצ' זמן t יש הסט' ϵ לשבור את ההצנה.

דוגמא כיום משתמשים $t = 2^{60}$, $\epsilon = 2^{-60}$.

שיטה זו מאוד שימושית משום שאפשר להתאים לאמת מספקת עבור מערכת ספציפית. מכאן בא גם חסונה, שהוא שכלל שהטכולוגיה משתפרת, הפרמטרים צריכים גם הם לגדול כל הזמן, והשיטה הקונקרטית מגבילה אותנו ממבנה זה כי היא תלולה בטכנולוגיה הנוכחית. יRibים שרצים בזמן יותר מ- t יכולים אולי לשבור את ההצפנה בהסת' גובהה הרבה מ- ϵ .

הגדירה לפי הגישה האסימפטוטית, סכמה היא בטוחה אם כל יריד הסתברותי פולינומיAli שובר את ההצפנה בהסת' זניחה.

הגדירה אלג' A נקרא הסתברותי פולינומיAli אם קיים פולינום כך שלכל קלט $x \in \{0, 1\}^*$ וסדרת ערכיהם אקראיים r , זמן $t \in \{0, 1\}^*$ ומספר $p(|x|)$ נגמר לאחר ($|x|$) A נגמר לאחר ($x; r$)。

האלג' בו משתמש התוקף לא בהכרח מוגדר באופן פורמלי, ולכן כניסה פרמטר חדש שיעזר לנו לכמות את הקלט הזה.

הגדירה פרמטר הבטיחות n הוא מספר שמאפשר לנו לקבוע את רמת הבטיחות של הסכמה שלנו. באופן פורמלי, KegGen מקבל כפרמטר את n (וכך אורך הקלט הוא n) ופולט מפתח באורך n ביטים, $K_n \in \mathcal{K}$. המפתח שהוא פולט-Amor להגנו מפני מ התקפה שרצה בזמן פולינומיAli ב- n .

הערה שלל האוספים שלנו מתחלקים עתה לרמות בטיחות - $\mathcal{M} = \bigcup_{n \in \mathbb{N}} \mathcal{M}_n$ וכיוצא-ב.

הגדירה תהי $f : \mathbb{N} \rightarrow \mathbb{R}^+$. נאמר כי f זניחה אם לכל פולינום p קיים $N \in \mathbb{N}$ כך ש- $N \geq n$, $f(n) < \frac{1}{p(n)}$, כלומר שהפונקציה דועכת יותר מהר מכל פולינום מתישחו.

דוגמא $2^{-\sqrt{n}}, 2^{-n}, 2^{-n \log^2 2}$ הן פ' זנichות ואילו $\frac{1}{n^5}, \frac{1}{\log^2 n}, \frac{1}{2}$ אינן זנichות.

דוגמא חוזרת להגדירה המקורית, אבל' בטוח לפי הגישה האסימפטוטית, אלג' שרצ' בזמן $(n^{1232143})$ ישבור את ההצפנה בהסת' זניחה כלשהי.

טענה (סיגירות זנichות לסכום וכפל בפולינום) אם n_1, n_2 פ' זנichות ו- p פולינום אזי $(n_1 + n_2)p$ גם פ' זנicha.

הערה הבחירה להניח שהאלג' פולינומיAli והסת' זניחה היא אחד חלקי פולינומיאלית מבחיננתנו היא לא טריויאלית ויכלנו לבחור דברים אחרים כגון שהסת' זניחה היא אחד חלקי אקספוננציאלית וכו'.

הבחירה הזו נותנת לנו תוכנות אינטואיטיביות כי מכפלת פולינומים היא פולינום ומכפלת פולינום ופ' זניחה היא עדין זניחה שזה הינו מבחינת המציאות. ככלומר הפעלת אלג' פולינומיAli מספר פולינומיAli של פעמים עדין מכוסה תחת הבטיחות שלו והיריב עדין יפרוץ את האבטחה בהסת' זניחה.

הגדירה הצפנה תקרא בלתי ניתנות להבנה אם כל שתי הצפנות לא ניתנות להבנה, אך התוקף עדיין רואה רק הودעה אחת. כלומר,

$$\text{Enc}_k(m_0) \approx \text{Enc}_k(m_1)$$

ניסוי תהי (Π, \mathcal{A}) . נגידר את הניסוי $\text{IND}_{\Pi, \mathcal{A}}(n)$ באופן הבא:

$$1. \text{ ניצור } .k \leftarrow \text{KegGen}(1^n)$$

$$2. \mathcal{A}(1^n, m_0, m_1)$$

3. נחזיר לו את ההצפנה של אחת ההודעות בתפלגות שווה ונראה האם הוא מצליח לקבוע האם הצפנו את m_0 או m_1 . נעשה זאת

פורמלית ע"י הגרלה של $c^* \leftarrow \text{Enc}_k(m_b)$ וחשבון ושילחת

4. היריב יחזיר b' , הקביעה שלו האם זה m_0 או m_1 שהצפנה.

$$\text{IND}_{\Pi, \mathcal{A}}(n) = \begin{cases} 1 & b' = b \\ 0 & \text{אחרת} \end{cases}$$

נקבע את ערך הניסוי להיות

הגדירה Π תקרא בעל הצפנות בלתי ניתנות להבנה אם כל יריב PPT (פולינומיAli הסט') קיים פ' זניחה ϵ כך ש-

$$P(\text{IND}_{\Pi, \mathcal{A}}(n) = 1) \leq \frac{1}{2} + \nu(n)$$

כאשר חלק ההסט' הוא r של \mathcal{A} והטלת המטבע b .

הערה המשמעות האינטואיטיבית היא שהיריב בקושי יכול לבצע יותר מהסת' עיוורת ($\frac{1}{2}$) איזו הודעה הוצפנה.

טענה סודיות מושלמת גוררת בטיחות בלתי ניתנת להבנה (בתרגילים).

נרצה למצוא דרך להבטיח בטיחות חישובית ברמת בטיחות בלתי ניתנת להבנה באמצעות מטבעות קצרים.

נרצה בהינתן קלט קצר ואקראי (seed מעטה), להרchip אותו לערך ארוך יותר שנראה אקראי גם, פ' מהצורה $\{0, 1\}^*$ $\rightarrow \{0, 1\}^*$. נרצה שתהראה אקראי גם הערך ארוך פסאודו אקראי. ערך נראה אקראי אם הוא בלתי ניתן להבנה בתפלגות איחודית. נרצה לשם הפשטה להכפיל את אורך המחרוזות המתקבלת, ככלומר $\{0, 1\}^{2n} \mapsto \{0, 1\}^n$. בשל הפרשי העוצמות, לא נקבל באמצעות התפלגות איחודית שכן לא נמפה לפחות 2^n ערכאים ב- $\{0, 1\}^{2n}$. אבל עדיין נרצה שלא יוכל להבדיל ביןיהן.

הגדירה תהי $G : \{0, 1\}^* \rightarrow \{0, 1\}^*$ המחשבת בזמן פ' פולינומיAli ויהי ℓ פולינום כך שלכל קלט $s \in \{0, 1\}^\ell$ הינו $G(s) \in \{0, 1\}^{\ell(n)}$. נגידר את הטענה הוא מחולל פסאודו אקראי אם:

1. הרחבה: $n \cdot \ell(n) > n$

2. פסאודו אקראיות, לכל מבדיל PPT \mathcal{D} , קיימת \mathcal{F} זניחה כך ש-

$$\left| \frac{P_{s \leftarrow \{0,1\}^n}(\mathcal{D}(G(s)) = 1)}{P_{r \leftarrow \{0,1\}^{\ell(n)}}(\mathcal{D}(r) = 1)} - \nu(n) \right| \leq \nu(n)$$

כלומר שהפער בין החתכנותות של המבדיל על התפלגות אקראית ועל התמונה של המחולל היא זהה.

הערה הגדרת הבטיחות של מחולל פסאודו אקראי, היא שככל מבדיל הסט' פולינמיAli יבחן בהסת' זניחה בין שני העולמות $\{0,1\}^{2n}$ ו- $\{0,1\}^n$, כלומר $G(\{0,1\}^n)$ הוא טוב כאקראי.

סיכום נקבע שערך החזורה 1 מmdbil משמעו "לדעתך זה מ-G" ואילו 0 משמעו "לדעתך זה מהתפלגות אחידה".

האם קיימים באמת PRG-ים? קיימים כיום כאלה, שמתבססים על ההבנה המדעית כיום של הפעולה שלהם ועומדים על הנחות בטיחות סבירות.

נכשא לקבל אינטואציה ל-PRG. ננסה לבנות מחולל שמוסיף בית אחד.

דוגמה עבור מחרוזות $\{0,1\}^n$. האם $s = s_1 \dots s_n \in \{0,1\}^n$ נגדי $s = s_1 \dots s_n || 0$? נגדי $s = s_1 \dots s_n \in \{0,1\}^n$. האם זהה מחרוזות שנייה לבחינה מהתפלגות אקראיות? כזכור SCN, הרי שברגע שהabit האחרון הוא 1 ברור שזויה התפלגות המקורית ונקבל 1 כזה בהסת' חci וזה לא זניהם. המבדיל שלנו

$$\mathcal{D}(r) = \begin{cases} 1 & r_{n+1} = 0 \\ 0 & r_{n+1} = 1 \end{cases}$$

דוגמה נגדי עתה r_1, r_{n+1} . האם ניתנו להבחן עתה בין התפלגות אחידה לבין G? עדין כן, כי בהתפלגות רגילה ב"ת ובהסת' חci שונים זה מזה ואילו כאן הם תמיד שווים. המחולל הוא 1 בהסת' 1 ובהתפלגות האחידה יחזיר 1 בהסת' חci.

דוגמה נגדי z כאשר $s_n \in \{0,1\}^n$ $G(s) = s_1 \dots s_n || z$, האם זה מחולל פסאודו אקראי? עדין לא! בעולם של G בהסת' 1 מתקיים $r_1 \oplus \dots \oplus r_{n+1} = 0$ בעוד $r_1 \oplus \dots \oplus r_{n+1} = 1$ בעוד z קורה בהסת' חci - זהו יתרון לא הון עבור המבדיל.

קשה מאוד לבנות PRG ועצם הקיום של אחד "טהור" כזה מוכיחה ש- $NP \neq P$ שזו בעיה חישובית שטרם נפתרה. עם זאת, בהסתמך על הנחות חישוביות מסוימות אפשר למצוא פונקציות שמתנהגות מאוד קרובה ל-PRG.

טענה כל תוכנה של התפלגות אחידה מתקיימת גם עבור הפלט של PRG.

מסקנה בדיקת כל תוכנה סטטיסטית שנייה לבדיקה בזמןiesel יכולה לשמש כmdbil.

דוגמה אם G הוא PRG אז קיימת \mathcal{F} זניחה כך ש-

$$P_{s \leftarrow \{0,1\}^n} \left(G(s) < \frac{1}{4} \text{ מס' האחדות ב-} \right) \leq \nu(n)$$

הערה כדי להוכיח טענות על מערכות עם הצפנות בלתי ניתנות להבנה ו-PRG-ים, לרוב נניח בשלילה שתכונה אחת לא מתקינה ונבנה אובייקט שסותר את ההגדרה השניה.

הערה בקורס יחסית, מערכת עם הצפנות בלתי ניתנות להבנה אומרת שכל תוקף מנבא בהסת' גדולה ממחצית באופןן זניח איזו הودעה הוצפנת עבורה.

פונקציה היא PRG אם כל מבדיל קובע שמחירות היא אקרואית טהורה או שהיא פלט של הפ' בהפרש זניח.

ברגע שזוכרים את ההפרדה זו הרבה יותר קל לעקוב אחר הוכחות גם אם הם ארוכות.

חלק ד' של הרצאה (קישור)

נשתמש ב-OTP יחד עם PRG כדי להצפין הודעות ארוכות.

יהי PRG G עם הרחבה $\mathcal{M}_n = \mathcal{C}_n = \{0,1\}^{\ell(n)}$ וכן $\mathcal{K}_n = \{0,1\}^n \cdot \ell(n)$.

$$k \leftarrow \{0,1\}^n \text{ דוגם KegGen}(1^n) .1$$

$$\text{Enc}_k(m) = m \oplus G(k) .2$$

$$\text{Dec}_k(c) = c \oplus G(k) .3$$

הערה אם $|K_n| = 2^n$, $\ell(n) = 2n$ אז גודל ההודעות שנitin להצפין הוא $|\mathcal{M}_n| = 2^{2n}$, כלומר גדול באופן קספוננציאלי.

משפט אם G הוא PRG אז למערכת יש הצפנות בלתי ניתנות להבנה.

הוכחה: נשתמש בשיטה שתuvwxyz על עצמה בקורס - רידוקציה. נניח בשלילה שהמערכת מחזירה צפנים ניתנים להבנה על ידי תוקף \mathcal{A} ונבנה באמצעות \mathcal{A} מבדיל \mathcal{D} שיפריך את היות G בסתייה להנחה. בנוסף, נקשרו את הפולינומיאליות של \mathcal{D} לו של \mathcal{A} .

מההנחה בשלילה, קיים יריד PPT \mathcal{A} ופולינום $(n) p$ כך ש-

$$P(\text{IND}_{\Pi, \mathcal{A}}(n) = 1) \geq \frac{1}{2} + \frac{1}{p(n)}$$

באופן שכיח (עבור אינסוף n -ים).

נוכיח כי קיים מבדיל PPT \mathcal{D} ופולינום $(n) q$ כך ש-

$$\left| \Pr_{s \leftarrow \{0,1\}^n}(\mathcal{D}(G(s)) = 1) - \Pr_{r \leftarrow \{0,1\}^{\ell(n)}}(\mathcal{D}(r) = 1) \right| \geq \frac{1}{q(n)}$$

נתכן את $\mathcal{D}(z)$:

1. בהינתן קלט z , נקרא ל- \mathcal{A} עם z ונקבל ממנו שתי הודעות m_0, m_1 (\mathcal{A} חושב שהוא בתוך הניסוי IND ולכן יפעל כמוותו).

2. נדגים $b \leftarrow \{0,1\}^{\ell(n)}$ כולם נשתמש ב- z כמפתח האפקטיבי.

3. נפלוט 1 אם $b = b'$, כלומר אם z היריב נכון.

זמן הריצה הוא פולינומייאלי כי כל הפעולות הן על קלטים באורך פולינומייאלי וכן \mathcal{A} פולינומייאלי ולכן \mathcal{D} הוא PPT.

ננתח את \mathcal{D} לפי מקרים על z :

1. $z \leftarrow \{0,1\}^{\ell(n)}$ (ההודעה באמת מהתפלגות איחידה). במקרה זה, הקביעה של \mathcal{A} ב"ת- b " כי הוא מקבל את c^* והוא מורכב מ- \oplus עם

z , אבל z מותפלג איחיד ולכן גם c^* ולכן כל קביעה שלו ב"ת- b ".

$$\text{לכן, } P_{z \leftarrow \{0,1\}^{\ell(n)}}(\mathcal{D}(z) = 1) = \frac{1}{2}$$

נוח לחשב על זה כך: \mathcal{D} כבר החזיר איזושהי תשובה b' שלא קשורה ל- b (שכן החלטתו ב"ת- b " כי הוא לא יודע עליו כלום כאמור). עכשו בודקים האם הערך הזה שווה ל- b , שמדובר איחיד, וחתסת' שהזיהוי היה בוודאי חצי.

2. $z = G(k) \leftarrow \{0,1\}^n$ (ההודעה מה-PRG). במקרה זה, התהליך זהה לחלווטין לניסוי $\text{IND}_{\Pi, \mathcal{A}}$ שכן הatzפה של m_0 או m_1 היא בדיקת האם הערך הזה שווה ל- b , שמדובר איחיד, וחתסת' שהזיהוי היה בוודאי חצי.

$$P_{k \leftarrow \{0,1\}^n}(\mathcal{D}(G(k)) = 1) = P(\text{IND}_{\Pi, \mathcal{A}}(n) = 1) \geq \frac{1}{2} + \frac{1}{p(n)}$$

אם נחליף את שמות k, z, s, r בהתאם, נקבל כי מצאנו מבדיל PPT \mathcal{D} המקיים

$$\left| P_{k \leftarrow \{0,1\}^n}(\mathcal{D}(G(s)) = 1) - P_{z \leftarrow \{0,1\}^{\ell(n)}}(\mathcal{D}(z) = 1) \right| \geq \frac{1}{p(n)}$$

בסתירה להגדרת ה-PRG!

■

הערה הבטיחות של II מתקינה עדין רק להודעה אחת, ולא יותר.

חלק ה' של הרצאה (קישור)

נרצה בהינתן מערכת עם הצפנות בלתי ניתנות להבחנה לקבוע איזו אינפורמציה כן אפשר ללמוד על m -(m). $\text{Enc}_k(m)$

טענה תהי II מערכת עם הצפנות בגין'ל (נמסס כבר לכתוב בלתי ניתנות להבחנה). אזי לכל יריד \mathcal{B} PPT, קיימת פ' זניחה ν כך ש-

$$P(\mathcal{B}(1^n, \text{Enc}_k(m)) = \text{LSB}(m)) \leq \frac{1}{2} + \nu(n)$$

כאשר $\nu \leftarrow \{0,1\}^{\ell(n)}$. כלומר לאף תוקף אין יתרון לא זניחה על ידיעת m בהינתן הatzפה.

הוכחה: נוכח ברiddockcia. נניח בשלילה שקיים יריב PPT \mathcal{B} ופולינום $(n) p$ כך ש-

$$P(\mathcal{B}(1^n, \text{Enc}_k(m)) = \text{LSB}(m)) > \frac{1}{2} + \frac{1}{p(n)}$$

נמצא יריב PPT \mathcal{A} ופולינום q כך ש- $P(\text{IND}_{\Pi, \mathcal{A}}(n) = 1) > \frac{1}{2} + \frac{1}{q(n)}$ בוואן שכיח, כלומר Π - \mathcal{A} אינה עם הבנ"ל.

נצריך לעשות כאן שלושה דברים: להגדיר את \mathcal{A} , לנתח את זמן הריצה שלו ולנתח את היתרונו שלו על פני ניחוש עיוור.

הרעינו הבסיסי של הגדרת \mathcal{A} הוא \mathcal{A} - \mathcal{B} יודע קבוע בהסת' לא זניחה מה הבית הראשון של m בהינתן ההצפנה, ולכן אם \mathcal{A} יבקש להבדיל בין שתי הודעות שהוא יודע שיש להן LSB שונה ($0 - m_0$ ו- $1 - m_1$), ממש יוכל להסיק בהסת' לא זניחה האם m_0 או m_1 היא שהצפנה ובכך להבחן בסתייה להיות Π עם הבנ"ל.

ובור $\{\sigma \in \{0,1\}^\ell : \text{נסמן } I_\sigma \text{ אוסף הבודעות עם LSB שווה ל-}\sigma\}$. נקבע את \mathcal{A} :

1. בהינתן קלט 1^n , נדגים $I_0 \leftarrow m_0 \leftarrow I_1 \leftarrow \dots \leftarrow m_1$ באופן אחיד ובי"ת.

2. בהינתן ההצפנה מ- Π , נפלוט $b' = \mathcal{B}(1^n, c^*, \text{נפלוט})$.

כלומר ביקשנו הודעות עם ביטים ראשונים ידועים מראש ונקבע על פי דעת \mathcal{B} את התשובה של \mathcal{A} .

ברור שהסיבוכיות של הכל היא פולינומיאלית.

מתקיים באופן שכיח

$$\begin{aligned} P(\text{IND}_{\Pi, \mathcal{A}}(n) = 1) &= P(\mathcal{B}(1^n, \text{Enc}_k(m_b)) = b) \\ &= \frac{1}{2} \sum_{m_0 \leftarrow I_0} P(\mathcal{B}(1^n, \text{Enc}_k(m_0)) = 0) + \frac{1}{2} \sum_{m_1 \leftarrow I_1} P(\mathcal{B}(1^n, \text{Enc}_k(m_1)) = 1) \\ &= \sum_{m \leftarrow \{0,1\}^\ell} P(\mathcal{B}(1^n, \text{Enc}_k(m)) = \text{LSB}(m)) \\ &\geq \frac{1}{2} + \frac{1}{p(n)} \end{aligned}$$

■ בסתייה להיות Π עם הבנ"ל!

הגדירה מערכת היא בעלת בטיחות סמנטית אם כל דבר שאפשר לחשב באופן יעיל בהינתן ההצפנה, אפשר לחשב גם בלי לקבל את ההצפנה בכלל.

פורמלית, Π נקראת בטיחות סמנטית אם לכל תוקף PPT \mathcal{A} , קיימים "סימולטור" \mathcal{S} כך שלכל התפלגות שנייה לדוגמה ביעילות על M_n ולכל פונקציות שניתנות לחישוב בזמן ייעיל f, h , קיימת פ' זניחה ν כך ש-

$$|P(\mathcal{A}(1^n, \text{Enc}_k(m), h(m)) = f(m)) - P(\mathcal{S}(1^n, h(m)) = f(m))| \leq \nu(n)$$

כאשר $.m \leftarrow M_n$ ו- $k \leftarrow \text{KegGen}(1^n)$

כלומר לכל תוקף \mathcal{A} שרוצה ללמידה מהו $(m) f$ שידוע לו נתון על ההודעה $(m) h$ וההצפנה של m , יש סימולטור S שלא מקבל את ההצפנה אבל כן את $(m) f$ שיכל לחשב את $(m) h$ כמעט טוב כמו \mathcal{A} (עד כדי הפרש זניח).

במילים אחרות, יש אלג' שלא יודע את ההצפנה שעשו את אותו הדבר שהתוקף עושה כשייש לו את \mathcal{A} ולכן ההצפנה לא עזרה בחישוב בכלל.

הערה ההבדל בין זה לבין סודיות מושלמת היא שמדובר בדברים שאפשר לחשב באופן יעיל (פולינומייאלי). למעשה, זהה ההגדרה הכי קרובות שיש בביטחוןיות חשיבות לסדריות מושלמת.

משפט II היא בטוחה סמנטית אם “ s היא עם הבן”.

בטיחות סמנטית היא הגדרה אינטואיטיבית לבטיחות של מערכת אבל לא נוכה לשימוש פורמלי ואילו מערכות עם הצפנות בלתי ניתנות להבחנה היא הגדרה נוכה לשימוש (שスクולה לבטיחות סמנטית).

תרגול

קישור

שאלה יהיו G PRG עם הרחבת ℓ . הראו כי קיימת \mathcal{P} זניחה נ כך ש-

$$\frac{1}{8} - \nu(n) \leq \underset{s \leftarrow \{0,1\}^n}{P}(G(s) \text{ with starts } 000) \leq \frac{1}{8} + \nu(n)$$

$\forall n \geq N$ עבור N כלשהו.

פתרון נניח בשילhouette שהטענה לא מתקימת. לכן קיימים פולינומים d כך ש-

$$(2) \quad \frac{1}{8} - \frac{1}{p(n)} > \underset{s \leftarrow \{0,1\}^n}{P}(G(s) \text{ with starts } 000) \quad \text{או} \quad \underset{s \leftarrow \{0,1\}^n}{P}(G(s) \text{ with starts } 000) > \frac{1}{8} + \frac{1}{p(n)} \quad (1)$$

באופן שכיח. בפרט אחד הא”ש מתקיים באופן שכיח. בה”כ (1) מתקיים באופן שכיח, ככלומר ש- G מחזיר בהסתמך לא זניחה מעל למצופה מחרוזות שמתחלילות ב-000.

נבנה מבחין \mathcal{D} PPT שסותר את הגדרת G -PRG באופן הבא:
 $\mathcal{D}(y) = \begin{cases} 1 & y_1 y_2 y_3 = 000 \\ 0 & \text{אחרת} \end{cases}$

שזה

ברור ש- \mathcal{D} פולינומייאלי. נחלק למקירם על הקלט:

$$\cdot \underset{r \leftarrow \{0,1\}^{\ell(n)}}{P}(r \text{ with starts } 000) = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8} = \{0,1\}^{\ell(n)}. 1$$

$\Pr_{s \leftarrow \{0,1\}^{\ell n}}(G(s) \text{ with starts } 000) > \frac{1}{8} + \frac{1}{p(n)}$ מהגדרת \mathcal{D} מההנחה בשלילה מתקיים $.k \leftarrow \text{KegGen}(1^n), r \leftarrow G(k)$.2
ולכן

$$\begin{aligned} \left| \Pr_{s \leftarrow \{0,1\}^n}(G(s) \text{ with starts } 000) - \Pr_{r \leftarrow \{0,1\}^{\ell(n)}}(r \text{ with starts } 000) \right| &> \left| \frac{1}{8} + \frac{1}{p(n)} - \frac{1}{8} \right| \\ &= \frac{1}{p(n)} \end{aligned}$$

בסתירה להגדרת PRG-ה.

מסקנה השתמשנו כאן בכך שהביטויים בהתפלגות אחידה מוגרים באקראי ואפשר לבדוק את התוכנה הנ"ל באופן יעיל, וכך נivial כי כל תוכונה שנייה לחשב ביעילות מקיימת עבור פלטי PRG באוטה הסת' כמו בהתפלגות אחידה עד כדי הפרש זניחה.

שאלה יהיו G_1, G_2 PRG. נגידר $G(s) = G_1(s) || G_2(s)$ הוא בהכרח?

פתרון לא בהכרח G , נביט במקרה שבו $G_1 = G_2 = H$ כאשר H PRG. ברור שהחצי הימני שווה לחצי השמאלי של הפלט של G ואילו במחuzeות מהתפלגות אחידה על $\{0,1\}^{2\ell(n)}$ תקיים תוכונה זו בהסת' זניחה.

פורמלית, נגידר מבחן PPT (r) שמחזיר 1 אם "החצי הימני של r שווה לחצי השמאלי שלו. מתקיים

$$\begin{aligned} \left| \Pr_{s \leftarrow \{0,1\}^n}(\mathcal{D}(G(s)) = 1) - \Pr_{r \leftarrow \{0,1\}^{2\ell(n)}}(\mathcal{D}(r) = 1) \right| &= \left| \Pr_{s \leftarrow \{0,1\}^n}(G_1(s) = G_2(s)) - \Pr_{r \leftarrow \{0,1\}^{2\ell(n)}}(\mathcal{D}(r) = 1) \right| \\ G_1 = H = G_2 &= \left| 1 - \Pr_{r \leftarrow \{0,1\}^{2\ell(n)}}(\mathcal{D}(r) = 1) \right| \\ r \text{ מגיע מהתפלגות אחידה} &= 1 - \frac{1}{2^{\ell(n)}} \end{aligned}$$

זה בטח לא זניחה, בסתירה להגדרת PRG-ה.

חישבו, האם עבור $G_1 \neq G_2$ כן מקבלים PRG? אם כן, נא לעדכן אותו כי אין לי מושג.

שאלה יהיו G PRG עם הרחבת ℓ . נגידר מערכת $\Pi = (\text{KegGen}, \text{Enc}, \text{Dec})$ באופן הבא:

$$\begin{aligned} .k \leftarrow \{0,1\}^n \text{ מגיריל איחיד KegGen}(1^n) &\bullet \\ .(m \in \{0,1\}^{\ell(n)}) = \mathcal{M} = \mathcal{C} \text{ (במקרה זה Enc}_k(m) = G(k) \oplus m &\bullet \\ \text{Dec}_k(m) = c \oplus G(k) &\bullet \end{aligned}$$

נתנו כי Π היא עם הבני"ל (כלומר שורדת את ניסוי IND) האם G בהכרח?

הערה הוכחה בהרצאה את הכיוון الآخر, ככלומר אם משתמשים ב-PRG כמרחיב מפתחות ל-OTP או המערכת היא עם הבני"ל.

פתרון כן! נניח בשלילה שלא, ככלומר שקיימים מבديل PPT \mathcal{D} כך ש-

$$\left| \Pr_{s \leftarrow \{0,1\}^n}(\mathcal{D}(G(s)) = 1) - \Pr_{r \leftarrow \{0,1\}^{\ell(n)}}(\mathcal{D}(r) = 1) \right| > \frac{1}{p(n)}$$

באופן שכיח. בה"כ הא"ש מתקיים באופן שכיח גם בלי הערך המוחלט (בסוף ההוכחה נראה כיצד להתאים זאת ל McKee ההפוך).

כלומר, ש- \mathcal{D} -מחזיר 1 בהסתמך יותר גבוהה כהוא מקבל קלט מ- G ולא בחתפוגות אחת.

נתקן \mathcal{A} שি�יבור את הניסוי (n) IND _{Π, \mathcal{A}} באמצעות השימוש במבדיל בכך שנגish לו את ערך ה- $G(k)$ עצמו וערך בחתפוגות שווה (שקיבלו כחוודעות מוצפנות), ומהיות מבדיל בהסתמך לא זינחה הוא ישים לב האם זהה תוצאה מ- G או לא וכן יבדיל בין ההצפנות:

1. נבקש להציג את $m_1 = 0^{\ell(n)}$ או את $m_0 \leftarrow \{0,1\}^{\ell(n)}$.

2. בהינתן ההצפנה c^* של m_b , נריץ את $\mathcal{D}(c^*)$ ונחזיר את תוצאתו.

זמן הריצה שלו פולינומיאלי כמוובן. נחלק למקרים עבור ערך החזרה:

אם $b=0$, ההצפנה היא $c^* = G(k) \oplus m_0$ והיא מתפלגת אחיד על $\{0,1\}^{\ell(n)}$ ולכן .1

$$P(\text{IND}_{\Pi, \mathcal{A}}(n) = 1 \mid b=0) = 1 - \sum_{r \in \{0,1\}^{\ell(n)}} P(\mathcal{D}(r) = 1)$$

אם $b=1$, ההצפנה היא $c^* = G(k) \oplus 0 = G(k)$.2

$$P(\text{IND}_{\Pi, \mathcal{A}}(n) = 1 \mid b=1) = \sum_{s \in \{0,1\}^n} P(\mathcal{D}(G(s)) = 1)$$

יחד נקבל כי

$$\begin{aligned} P(\text{IND}_{\Pi, \mathcal{A}}(n) = 1) &\stackrel{\text{חסות, שלמה}}{=} P(\text{IND}_{\Pi, \mathcal{A}}(n) = 1 \mid b=0)P(b=0) + P(\text{IND}_{\Pi, \mathcal{A}}(n) = 1 \mid b=1)P(b=1) \\ &= \frac{1}{2} \left(1 - \sum_{r \in \{0,1\}^{\ell(n)}} P(\mathcal{D}(r) = 1) + \sum_{s \in \{0,1\}^n} P(\mathcal{D}(G(s)) = 1) \right) \\ &= \frac{1}{2} + \frac{1}{2} \left(\sum_{s \in \{0,1\}^n} P(\mathcal{D}(G(s)) = 1) - \sum_{r \in \{0,1\}^{\ell(n)}} P(\mathcal{D}(r) = 1) \right) \\ &\geq \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{p(n)} \end{aligned}$$

בסתירה להיות Π מערכת עם הבני"ל.

אם הא"ש מתחילה ההוכחה מתקיים עם מינוס באופן שכיח, הרי ש- \mathcal{D} -מחזיר 1 בהסתמך יותר גבוהה כזו לא מ- G ולכן האלג'ריה יחויר את הביט הההפוך ממה ש- (c^*) יחויר והשאר עובד באותו האופן.

שבוע III | פונקציות פסאודו-אקראיות ו-CPA

הרצאה

חלק א' של הרצאה (**קישור**)

הגדרה נאמר כי $X = \{X_n\}_{n \in \mathbb{N}}, Y = \{Y_n\}_{n \in \mathbb{N}}$ בلتוי מוגבלות להבנה מהחינה **חישובית** אם אין אלג'רדי שיכל להבחן ביןיהם ונסיים $X \approx^c Y$

כלומר, X, Y , בני"ל חישובית אם לכל אלג'רדי PPT, קיימת ϵ זניחה כך ש

$$\left| \Pr_{x \leftarrow X_n}(\mathcal{D}(1^n, x) = 1) - \Pr_{y \leftarrow Y_n}(\mathcal{D}(1^n, y) = 1) \right| \leq \epsilon(n)$$

הערה אם X, Y , בני"ל חישובית אפשר להשתמש בהם לאותה המטרה לטירוגין.

דוגמה $Y_n = \text{Unif}\left(\{0, 1\}^{\ell(n)}\right), X_n = G(s), s \leftarrow \{0, 1\}^n$ הם בני"ל חישובית.

דוגמה $Y_n = \text{Enc}_k(101), k \leftarrow \text{KegGen}(1^n)$ וכן $X_n = \text{Enc}_k(000), k \leftarrow \text{KegGen}(1^n)$ כאשר Π בעלת הودעות בני"ל.

הגדרה נאמר כי התפלגות היא פסאודו אקראית אם היא בני"ל חישובית מהתפלגות איחידה.

משפט ידי $\text{PRG } H : \{0, 1\}^n \rightarrow \{0, 1\}^{4n}$ נגידיר $G : \{0, 1\}^n \rightarrow \{0, 1\}^n$ (מחלקים את הקלט לשני חלקים). אז $H(s_1 || s_2) = G(s_1) || G(s_2)$ נוכיח את הטענה H - \mathcal{D} מבديل \mathcal{A} ל- G .

הוכחה: נוכח ברידוקציה באמצעות ארגומנט היברידי. הרידוקציה תהיה, בהינתן מבديل \mathcal{D} ל- H , בניית מבديل \mathcal{A} ל- G .

נניח שיש ל- \mathcal{D} יתרון ϵ על פני $G(s_1) || G(s_2)$ ו- $r_1 || r_2$. נכnis פנימה עוד שחקן - הארגומנט היברידי - r_2 . נטען כי \mathcal{D} יכול להבדיל בפחות $\frac{\epsilon}{2}$ או בין $G(s_1) || G(s_2)$ והארוגומנט היברידי, או בין הארגומנט היברידי לבין $r_1 || r_2$.

פורמלית, מתקיים

$$\begin{aligned} \epsilon &\leq |P(\mathcal{D}(G(s_1) || G(s_2)) = 1) - P(\mathcal{D}(r_1 || r_2) = 1)| \\ &\leq |P(\mathcal{D}(G(s_1) || G(s_2)) = 1) - P(\mathcal{D}(G(s_1) || r_2) = 1) + P(\mathcal{D}(G(s_1) || r_2) = 1)P(\mathcal{D}(r_1 || r_2) = 1)| \\ &\stackrel{\triangle}{\leq} |P(\mathcal{D}(G(s_1) || G(s_2)) = 1) - P(\mathcal{D}(G(s_1) || r_2) = 1)| \\ &\quad + |P(\mathcal{D}(G(s_1) || r_2) = 1)P(\mathcal{D}(r_1 || r_2) = 1)| \end{aligned}$$

■

נמצא מבחין לכל אחד מהמרקמים ונוכח את הרידוקציה

- נניח כי היתרון של \mathcal{D} הוא בין $G(s_1) || r_2$ ו- $G(s_1) || G(s_2)$
- נגידיר מבديل \mathcal{A} שמקבל קלט $\mathcal{D}(G(s_1) || z), s_1 \leftarrow \{0, 1\}^n, z \in \{0, 1\}^{4n}$ ופולט

בגלו של \mathcal{D} יתרון $\frac{\epsilon}{2}$ על קלטים בדיקת מהצורה הזו, כך גם ל- \mathcal{A} . ברור שזה PPT ולכן הוא מקיים את התנאים הנדרשים, בסתירה לכך PRG G -ש.

- נניח כי יתרון של \mathcal{D} הוא בין $r_1 \parallel r_2$ ו- $G(s_1) \parallel r_2$.
- באוטו האופן, המבדיל מקבל $D(z \parallel r_2) \in \{0, 1\}^{4n}$, מגריל $z \in \{0, 1\}^{4n}$ ומחזיר r_2 . PRG G להיות

חלק ב' של הרצאה (קישור)
זכור כי בהצנות עם הבני'ל הניסוי המחשבתי היה קבלת הודעות להצפנה, החזרת הצפנה אקראית וקבלת תשובה מהיריב. בניסוי הזה, ההנחה כאן היא שהיריב מקבל רק הודעה אחת, שהיא אינה ריאלית ליריב יהי כמו הודעה, ולעתים יכול אפילו לקובע אותה בעצמו.

הגדירה נגדיר את הניסוי של Chosen Plaintext Attack IND_{Π,A}^{CPA}(n), עם מערכת הצפנה Π ויריב A. בונה אותו מעל ניסוי הבדיקה בין הצנות: נאפשר לאלג' לבקש הצפנה של כמה הודעות שהוא רוקח לפני שליחת m_0, m_1 ואחרי השליחה אך לפני מתן הבדיקה

$$\text{IND}_{\Pi,A}^{\text{CPA}}(n) = \begin{cases} 1 & b' = b \\ 0 & \text{אחרת} \end{cases} \text{ תוצאה הניסוי היא } A^{\text{Enc}_k(\cdot)}. \text{ ככלומר, יש ליריב גישה לאורקל הצפנה, שנסמן } (\cdot)$$

הערה ב글 שהיריב הוא PPT, הוא יכול לבקש מספר פולינומיאלי לכל היותר של הודעות להצפנה.

הגדירה נאמר כי ΙΙ בעלת הבני'ל תחת מתקפת CPA, או בקיצור ΙΙ היא בטוחה-CPA, אם לכל יריב A PPT קיימת פ' זניחה ϵ כך ש- $P(\text{IND}_{\Pi,A}^{\text{CPA}}(n) = 1) \leq \frac{1}{2} + \epsilon(n)$

הערה נדמה שככל הצפנה היא לא בטוחה-CPA כי היריב יכול לבקש להצפן את m_0 , אם הצפנה שהתקבלת שווה ל- c להחזיר $b' = 0$ ואחרות $b' = 1$. עם זאת, יש כאן הנחה סטנדרטית - שאלג' הצפנה הוא דטרמיניסטי (לדוגמה OTP ו-OTP (PRG), אך זהה הנחה שלא תמיד מתקיימת. לשם הגנה מפני CPA, נctrיך אלalg' הצפנה הסת').

הערה נדמה שגם הגדירה מאוד חזקה, שכן אנחנו מניחים שהינתן שהאלalg' קיבל כל הצפנה מלבד הודעה עצמה. אנחנו מניחים כאן שיריב יכול לדעת, להשפי ולקבע את ההודעות המוצפנות. זהה הנחה יחסית ריאלית ורובה המתקפות הם באמת כזו.

דוגמה בקשר על האי מידויי במלחמות העולם הראשונה, השתמשו האמריקאים ב-CPA כדי לגלוות שהאי האמור הוא אכן יעד של היפנים, ולמעשה האניגמה נפרצה באמצעות CPA.

חלק ג' של הרצאה (קישור)

הרעיוון של פ' פסאודו אקראיות, היא שהיא פ' רגילה, פסאודו אקראית, אך שהיא נראה כמו פ' אקראית למגרי. $h(x) \in \{0, 1\}^n, x \in \{0, 1\}^n$ Func_{n → ℓ} = {f : f : {0, 1}ⁿ → {0, 1}^ℓ} שזה שקול לכך שלכל f אקראית לחלוון היא פ' שנגמת אחד מ- $\{0, 1\}^\ell$ נדגם אחד וב"ת משאר ה-x-ים.

הערה מתקיים $|\text{Func}_{n \rightarrow \ell}| = |\{0, 1\}^\ell|^{|\{0, 1\}^n|} = 2^{\ell \cdot 2^n}$

שמירת טבלת הערכים של הפ' היא לא פרקטית, ולכן נצורך לשומר מפתח שייצג אותה.

הגדרה פונקציה פסאודו-אקראית היא פ' המוחשבת בזמן פולינומיאלית $F : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ (הקלט הראשון הוא המפתח) הבלתי ניתנת להבנה מפ' אקראית באמת. כלומר, לכל מבחין \mathcal{D} קיימת ν זניחה כך ש-

$$\left| P\left(\mathcal{D}^{F_k(\cdot)}(1^n) = 1\right) - P\left(\mathcal{D}^{h(\cdot)}(1^n) = 1\right) \right| \leq \nu(n)$$

כאשר $.h \leftarrow \text{Func}_{n \rightarrow \ell-1} \text{ ו } k \leftarrow \{0, 1\}^n$

או במדויק, אם מבחין PPT מקבל מאורקל ערכים של פ' מוגרת אחיד מ- $\text{Func}_{n \rightarrow \ell}$ אז מקבל מאורקל ערכים של F_k , החסת' שיענה 0 או 1 תהיה בערך שווה בשני המקרים.

הערה ההבדל המרכזי מול הגדרת PRG הוא שכאן יש גישה לאורקל, ולא רק הودעה אחת מכל מקום. זה רמז כבר על השימוש שלו בו - במקרה להשתמש בזוה כדי להרחב מפתחות, משתמש בזוה כדי להציג הרבה הודעות כך שניה מוגנים מ-CPA בסופו של דבר.

הערה אנחנו מניחים כאן שהקלט הראשון, המפתח, נדגם באופן אקראי על $\{0, 1\}^n$, עם זאת, אפשר באמצעות כל אלג' KegGen PPT לקבל את אותה התוכונה.

הוכחת הבטיחות של כל מערכת עם PRF מורכבת משני חלקים: ראשית, נבצע ניסוי מחשבתי בו נחליף את הפ' הפסאודו-האקראית לפ' אקראית באמת ונוכיח בשני שלבים:

1. נוכיח כי היריב לא מצליח להפר את הבטיחות כמשמעותו בפ' אקראית באמת
 2. נוכיח שהיריב לא מסוגל להבחן בין Π עם הפ' האקראית לבין Π עם הפ' הפסאודו-אקראית.
- nociah זאת בהנחה בשלילה ונגיעה לסתירה לכך שהפ' היא אכן פסאודו אקראית באמצעות בניית מבחין.

חלק ד' של הרצאה (קישור)

נגדיר מערכת הצפנה עמידה בפני Π_F , CPA, בהינתן F , פסאודו-אקראית.

- אלג' Dec דוגם $.k \leftarrow \{0, 1\}^n$ KegGen-
- אלג' Enc : בהינתן מפתח $k \in \{0, 1\}^\ell$ ומפתח $m \in \{0, 1\}^n$, נדגם $r \leftarrow \{0, 1\}^n$ ונחזיר $c = (r, F_k(r) \oplus m)$ (כלומר אנחנו מחזירים זוג סדור, בו אנו שוכפים בנוגע לקרט-ל-PRF).
- אלג' Dec : בהינתן מפתח $k \in \{0, 1\}^\ell$ והצפנה $c = (r, s)$ נחזיר $m = F_k(r) \oplus s$.

הערה למעשה אלג' ההצפנה והפענוח הם OTP עם PRF כמפתח, רק שעתה ה-PRF הוא הסת', ולא קבוע עבור k קבוע.

משפט אם Π_F azi CPA-PRF F בטוחה

הוכחה: נסמן Π_h אוטה המערכת רק עם h במקומם F .

יהי \mathcal{A} יריב PPT המשתף ב- \mathcal{A} - $\text{IND}_{\Pi, \mathcal{A}}^{\text{CPA}}(n)$.

$$\begin{aligned} P(\text{IND}_{\Pi_F, \mathcal{A}}^{\text{CPA}}(n) = 1) &\leq P(\text{IND}_{\Pi_h, \mathcal{A}}^{\text{CPA}}(n) = 1) - P(\text{IND}_{\Pi_h, \mathcal{A}}^{\text{CPA}}(n) = 1) + P(\text{IND}_{\Pi_h, \mathcal{A}}^{\text{CPA}}(n) = 1) \\ &\stackrel{\text{עכשו נוכיח}}{\leq} \frac{1}{2} + \left(\frac{q(n)}{2^n} + \nu(n) \right) \end{aligned}$$

נוכיח כי $(n) \text{IND}_{\Pi_h, \mathcal{A}}^{\text{CPA}}(n) = 1) - P(\text{IND}_{\Pi_h, \mathcal{A}}^{\text{CPA}}(n) = 1) \leq \nu(n)$.

נניח בשלילה שקיים \mathcal{A} עם יתרון לא זניח (יותר מ- $\frac{1}{p(n)}$ באופן שכיח, p פולינום) ובננה \mathcal{D} מבחין על F . ל- \mathcal{D} יש גישה לאורקל \mathcal{O} שהוא או

■ $\mathcal{D}^\mathcal{O}$ מוגדר באופן הבא: $.h$ או F_k

1. נקרא ל- \mathcal{A} ונקבל שתי הודעות $.m_0, m_1$

2. נדגום $b \leftarrow r$ ונחזיר ל- \mathcal{A} את $c^* = (r, \mathcal{O}(r) \oplus m_b)$ (ນבוקש מהאורקל לחשב לנו את הערך).

3. נפלוט את האם \mathcal{A} צדק, כלומר האם $b' = b$

ברור ש- \mathcal{D} הוא PPT.

• אם \mathcal{O} הוא F_k כאשר k נדגם אחד, \mathcal{A} חושב שהוא בדיק משותף בניסוי ה- IND_{Π_F} , או פורמלית $= \text{IND}_{\Pi_F, \mathcal{A}}^{\text{CPA}}(n) = 1$.

• אם \mathcal{O} הוא h כאשר h אקראיית נדגמת אחד, \mathcal{A} בדיק משותף בניסוי $\text{IND}_{\Pi_h, \mathcal{A}}^{\text{CPA}}(n) = 1$.

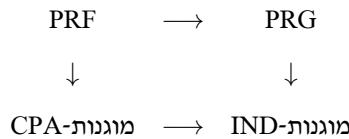
לכן היתרונו של \mathcal{A} להבדיל בין Π_F, Π_h מושרה ליתרונו של \mathcal{D} להבדיל בין $L-h$, בסתיו להיווטו PRF.

נוכיח כי $P(\text{IND}_{\Pi_h, \mathcal{A}}^{\text{CPA}}(n) = 1) \leq \frac{1}{2} + \frac{q(n)}{2^n}$ כאשר $(n) q$ מספר הביקשות להצפנה \mathcal{A} -היאורקל ביחס ל- \mathcal{A} .

הניסוי הוא כזה: \mathcal{A} מבצע קריאות לאורקל, מבקש להצפן את $r^*, h(r^*) \oplus m_b$, מקבל תשובה m'_0, m'_1 ומקבל קרייאות לאורקל b' .

לכל $i \in [q(n)]$ נסמן m_i ההודעה שהוצפנת כ- \mathcal{A} . הסיכוי שאירוע חרורה, כלומר $r_i, h(r_i) \oplus m_i$ עבورو הוא $r^* = r_i$ ש- $\text{IND}_{\Pi_h, \mathcal{A}}^{\text{CPA}}(n) = 1$ עבור אחד ספציפי ויש לנו $(n) q$ דוגמאות או מחסם האיחוד מקבלים את הביטוי הזה).

$$P(\text{IND}_{\Pi_h, \mathcal{A}}^{\text{CPA}}(n) = 1) \stackrel{\text{חסות', שלמה}}{\leq} P(\text{IND}_{\Pi_h, \mathcal{A}}^{\text{CPA}}(n) = 1 | \text{חרורה}) + P(\text{חרורה}) \leq \frac{1}{2} + \frac{q(n)}{2^n}$$



חלק ה' של הרצאה (קישור)

צופן בלוקים מיועד להחوت פרטוטזיה פסאודו-אקראי שנייתן לחשב בזמן פולינומיAli ולהשתמש בו כחצפנה. הוא מבוסס על איזושהי PRF ביסודות אך הוא עצמו אינו כזה. מדובר בפ' $\{0,1\}^\ell \rightarrow \{0,1\}^\ell$ עם קלט ראשון שהוא מפתח.

נרצה כאן במקום בטיחות אסימפטוטית, דוגמא בטיחות קונקרטית, לדוגמה עבור מפתח בגודל 256 ביטים - צופן בלוקים הוא הייררכית פרקטית, כלומר שהוא שאנו מניחים שעובד בלי הוכחה שאנו רוצים להשתמש בו במציאות. הבטיחות שלנו תספק אם המתקפה הטובה ביותר על מערכת החצפנה היא בערך 2^{2n} , כלומר ברוט פורס.

הערה לא נוכל להגיד את זה פורמלית (זהו החסרון המרכזי של בטיחות קונקרטית), אבל אינטואיטיבית זה מספיק.

צופן הבלוקים הראשון שאומץ באופן נרחב הוא DES, ב-1977, שעובד ע"י NSA ו-IBM, שימושו במפתח באורך 56 ובלוקים באורך 64. הוא מעבב הרבה ביטים בתקופה שזו יוזר, עם זאת בשנות התשעים הוא נקשר בבעיות על ידי distributed.net ולאחר מכן אף על ידי מהר ע"י EFF (ארגון שעושה עבודה בתחום אבטחת המידע, הפרטיות וזכויות אדם בתחום זהה), ולקח הרבה זמן לרשויות ולפינטק לשחרר ממנו עד שעברו לגרסה מעודכנת שלו : 3DES. הבעיה היא לא שיש מתקפה שאינה ברוט-פורס, אלא שברוט פורס הוא לא קשה מדי כמספר האפשרויות הוא 2^{56} .

ב-3DES, מחשבים $(DES_{k_1, k_2, k_3}(x) = DES_{k_1}^{-1}(DES_{k_2}(DES_{k_3}(x)))$, כאשר המפתח הוא באורך $3 \times 56 = 168$. זה אולי נראה בטיחותי, אבל בכלל Meet in the Middle ומתקפות אחרות (ראו ויקיפדיה למשמעות), הבטיחות היא רק 56×2 , שזה עדין לא מספיק טוב. עם זאת, פינטק משתמשים באופן אובייקטיבי ב-3DES במקום לעבור לאיל' דשים הרבה יותר בוחנים, בין היתר באמצעות האפשרות ל-3DES גם משאייטי. Backwards Compatibility.

ב-1997, מכון התקנים האמריקאי קיים תחרות למציאת אילג' חדש. ב-2000 הוכרז האילג' של Rijndael (איחוד שמות של בלגי ועוד מישוה) כמנצח, והאלג' שלהם הוכר כ-AES הרשמי שכולם משתמשים בו (ראוי שישמשו לפחות) - מהצבה האמריקאי ועד האתר להוחץ צרפתי של השכנה מהרחב. גודל הבלוק שלו הוא 128 ביטים ויש לו גרסאות למפתחות באורך 128/192/256. כבר המפתח הקצר ביותר מספק בטיחות מאוד גבוהה והSHIPOR מעבר לזה הוא לא כל כך דרמטי, כאשר שני אורכי המפתחות האחרים קיימים רק כי צבא ארה"ב דרוש שלוש רמות בטיחות שונות פורמלית, גם אם אין שום הבדל מוחותי ביןיהם והראשונה כבר מספק טובה להכל.

דוגמה נוכל להציג הדוגמאות כמו שראינו בעבר, בהינתן AES_k הינה $\text{Enc}_k(m; r) = (r, \text{AES}_k(r) \oplus m)$, PRF הינה $\text{AES}_k(r)$

הדוגמה הנ'ל עובדת רק על בלוק אחד - איך נעשה את זה על הדוגמאות ארוכות?

1. נאיבי: נוכל לחשב

$$\text{Enc}_k(m_1 || \dots || m_\ell; r_1 || \dots || r_\ell) = (r_1, F_k(r_1) \oplus m_1) || \dots || (r_\ell, F_k(r_\ell) \oplus m_\ell)$$

הבעיה עם שיטה זו היא שהיא מאריכה את ההודעה באופן דרמטי, לדוגמה עם AES עם מפתח באורך 128 ביטים, הכפלנו את אורך ההודעה בעת הצפנה.

2. נחטיב : Counter Mode (CTR)

$$\text{Enc}_k(m_1 \dots m_\ell; r) = (r, F_k(r+1) \oplus m_1, F_k(r+2) \oplus m_2, \dots, F_k(r+\ell) \oplus m_\ell)$$

וכך נוסיף רק אורך קבוע להצפנה ולא משהו מהותי. אפשר להוכיח שהצפנה זו היא אכן בטוחה- F -CPA בהינתן $-F$ -PRF. תווואי ההוכיח הוא כזה: נבצע את הניסוי המחשבתי של CPA. נניח שככל ההודעות המוצפנת ע"י האורקל הן $r-\ell$ בלוקים. מתקיים כי $s_i = (r_i, F_k(r_i+1), \dots, F_k(r_i+\ell))$ (בזה אנו משתמשים להצפנה ההודעה ה- i) היא פסאודו אקראית וכל $s_i, s_{i'}, s_{i''}$ הן ב"ת אלא אם $j' > j$. משם משתמשים בארגומנט היברידי ומסויימים את ההוכיח.

3. (ECB) Electronic CodeBook: נצפין ($\text{Enc}_k(m_1, \dots, m_\ell) = (F_k(m_1), \dots, F_k(m_\ell))$). זה צופן מאוד גרווע, לא רק כי הוא לא עמיד בפני CPA (בדומה לצופן דטרמיניסטי רגיל), אלא שכשיש כמה הודעות אין לו אפילו הודעות הבנ"ל, לדוגמה אם מב奸 מבקש הצפנה של 0^n ו- 1^n יוכל בקלות לגלוות מי זו מי. בנוסף, הצפנה של תמונה לדוגמה אולי תשנה את הצבעים אבל תשמר את הזרה ולא תשיג את המטרה במאמה.

תרגול

קישור

ازהרה

בתרגול הזה ובזה שאחריו, חשוב לא לטבוע בחול הטובעני שהוא הוכחות חורשות של טענות. בכל פעם, כל מה שאנחנו עושים זה להניח בשילילה שימושו הוא לא בלתי ניתן להבחנה (בין אם הتلפוגיות, PRF או כל דבר אחר), בונים באמצעות המבחן מבחין חדש שסותר תכונה נתונה ומגיעים לסתירה. הבניה היא כמעט טרייזיאלית אבל לעיתים קשה לתפוס את התמונה הכללית. לכן, וודאו שבסכל טענה אתם מבינים את הלק' הדברים הכללי, שכן הוא חשוב על אחת כמה וכמה על פני היכולת לדקלם אינדקסציה על פ' הסט'. סעו לאט!

שאלה: $|s_1| \in \{|s_2|, |s_2| + 1\}$. האם $G(s_1 || s_2) = G_1(s_1) || G_2(s_2)$? נגידיך (2). עבורם $s_1, s_2 \in \{0, 1\}^*$ לכל G PRG G_1, G_2 בהכרח PRG? (זו הכללה של טענה שהוכחנו בתרגיל).

פתרונות כן. G הוא PPT כי G_1, G_2 מעתיק ממחרוזות באורך n ל- ℓ_1 ו- ℓ_2 מ- n ל- ℓ_2 .

נניח בשלילה כי G אינו PRG, כלומר קיים מבחין פולינומי $p(n)$ כך שמתקיים באופן שכיח

$$\left| \frac{P_{s \leftarrow \{0,1\}^n}}{P_{r \leftarrow \{0,1\}^{\lceil \frac{n}{2} \rceil + \ell_2(\lceil \frac{n}{2} \rceil)}}} (D(G(s)) = 1) - \frac{P_{r \leftarrow \{0,1\}^{\lceil \frac{n}{2} \rceil + \ell_2(\lceil \frac{n}{2} \rceil)}}}{P_{s \leftarrow \{0,1\}^n}} (D(r) = 1) \right| > \frac{1}{p(n)}$$

בגלל שהוא “ש” מתקיים באופן שכיח, הוא מתקיים באופן שכיח או על איזוגים, נניח בה”כ כי זה הуль זוגיים. ככלומר,

מתקיים כמעט תמיד

$$\left| \frac{P_{s \leftarrow \{0,1\}^{2n}}}{P_{r \leftarrow \{0,1\}^{\ell_1(n) + \ell_2(n)}}} (D(G(s)) = 1) - \frac{P_{r \leftarrow \{0,1\}^{\ell_1(n) + \ell_2(n)}}}{P_{s \leftarrow \{0,1\}^{2n}}} (D(r) = 1) \right| > \frac{1}{p(2n)}$$

מתקיים

$$\begin{aligned} \frac{1}{p(2n)} &< \left| \frac{P_{s \leftarrow \{0,1\}^{2n}}}{P_{r \leftarrow \{0,1\}^{\ell_1(n) + \ell_2(n)}}} (D(G(s)) = 1) - \frac{P_{r \leftarrow \{0,1\}^{\ell_1(n) + \ell_2(n)}}}{P_{s \leftarrow \{0,1\}^{2n}}} (D(r) = 1) \right| \\ &= \left| \frac{P_{s_1, s_2 \leftarrow \{0,1\}^n}}{P_{r_1 \leftarrow \{0,1\}^{\ell_1(n)}, r_2 \leftarrow \{0,1\}^{\ell_2(n)}}} (D(G_1(s_1) || G_2(s_2)) = 1) - \frac{P_{r_1 \leftarrow \{0,1\}^{\ell_1(n)}, r_2 \leftarrow \{0,1\}^{\ell_2(n)}}}{P_{s_1, s_2 \leftarrow \{0,1\}^n}} (D(r_1 || r_2) = 1) \right| \\ &\stackrel{\Delta}{\leq} \left| \frac{P_{s_1, s_2 \leftarrow \{0,1\}^n}}{P_{r_1 \leftarrow \{0,1\}^{\ell_1(n)}, s_2 \leftarrow \{0,1\}^n}} (D(G_1(s_1) || G_2(s_2)) = 1) - \frac{P_{r_1 \leftarrow \{0,1\}^{\ell_1(n)}, s_2 \leftarrow \{0,1\}^n}}{P_{s_1, s_2 \leftarrow \{0,1\}^n}} (D(r_1 || G_2(s_2)) = 1) \right|, \text{ הבחנה בין היצרך בין הארגומנט היבריידי} \\ &+ \left| \frac{P_{r_1 \leftarrow \{0,1\}^{\ell_1(n)}, s_2 \leftarrow \{0,1\}^n}}{P_{r_1 \leftarrow \{0,1\}^{\ell_1(n)}, r_2 \leftarrow \{0,1\}^{\ell_2(n)}}} (D(r_1 || G_2(s_2)) = 1) - \frac{P_{r_1 \leftarrow \{0,1\}^{\ell_1(n)}, r_2 \leftarrow \{0,1\}^{\ell_2(n)}}}{P_{r_1 \leftarrow \{0,1\}^{\ell_1(n)}, s_2 \leftarrow \{0,1\}^n}} (D(r_1 || r_2) = 1) \right|, \text{ הבחנה בין היצרך בין הארגומנט היבריידי לבין הערכאים האקראיים באמצעות} \end{aligned}$$

. $\beta > \frac{1}{2p(2n)}$ או $\alpha > \frac{1}{2p(2n)}$ וכן

• אם נגידר מבחין \mathcal{D} באופן הבא: בהינתן קלט $s, z_1 \in \{0,1\}^{\ell_1(n)}$ ומחזיר את הפלט של החישוב

ברור שזה PPT. מתקיים $D(z_1 || G_2(s_2))$

$$\begin{aligned} &\left| \frac{P_{s \leftarrow \{0,1\}^{2n}}}{P_{r_1 \leftarrow \{0,1\}^{\ell_1(n)}}} (D(G(s)) = 1) - \frac{P_{r_1 \leftarrow \{0,1\}^{\ell_1(n)}}}{P_{s \leftarrow \{0,1\}^{2n}}} (D(r_1) = 1) \right| \\ &= \left| \frac{P_{s_1, s_2 \leftarrow \{0,1\}^n}}{P_{r_1 \leftarrow \{0,1\}^{\ell_1(n)}, s_2 \leftarrow \{0,1\}^n}} (D(G_1(s_1) || G_2(s_2)) = 1) - \frac{P_{r_1 \leftarrow \{0,1\}^{\ell_1(n)}, s_2 \leftarrow \{0,1\}^n}}{P_{s_1, s_2 \leftarrow \{0,1\}^n}} (D(r_1 || G_2(s_2)) = 1) \right| \\ &> \frac{1}{2p(2n)} \end{aligned}$$

באותו הסתיירה להיות G_1 PRG.

• אם באותו האופן נגידר \mathcal{D}_2 מבחין שמקבל $r_1 \leftarrow \{0,1\}^{\ell_1(n)}$, $z_2 \leftarrow \{0,1\}^{\ell_2(n)}$ ומחזיר $D(r_1 || z_2)$.

$$\begin{aligned}
& \left| \frac{P_{s_2 \leftarrow \{0,1\}^n} (D_2(G_2(s_2)) = 1) - P_{r_2 \leftarrow \{0,1\}^{\ell_2(n)}} (D_2(r_2) = 1)}{P_{s_2 \leftarrow \{0,1\}^n} (D_2(r_1 || G_2(s_2)) = 1) - P_{r_1 \leftarrow \{0,1\}^{\ell_1(n)}, s_2 \leftarrow \{0,1\}^n} (D_2(G_1(s_1) || r_2) = 1)} \right| \\
&= \left| \frac{P_{s_2 \leftarrow \{0,1\}^n} (D_2(r_1 || G_2(s_2)) = 1) - P_{r_1 \leftarrow \{0,1\}^{\ell_1(n)}, r_2 \leftarrow \{0,1\}^{\ell_2(n)}} (D_2(G_1(s_1) || r_2) = 1)}{P_{s_2 \leftarrow \{0,1\}^n} (D_2(r_1 || G_2(s_2)) = 1) - P_{r_1 \leftarrow \{0,1\}^{\ell_1(n)}, s_2 \leftarrow \{0,1\}^n} (D_2(G_1(s_1) || r_2) = 1)} \right| \\
&> \frac{1}{2p(2n)}
\end{aligned}$$

בסתירה להיות G_2 PRG.

שאלה תהיינה Π_1, Π_2 מערכות הצפנה. ידוע כי לפחות אחת מהן היא בעלת הבנ'ל, אך לא ידוע איזו מהן. בנו מערכת הצפנה בעלת הבנ'ל על בסיסן.

פתרונות נגדיר $\Pi = (\text{KegGen}, \text{Enc}, \text{Dec})$ באופן הבא:

$$\cdot k = (k_1, k_2) \text{ ומוחזר } k_1 \leftarrow \text{KegGen}_1(1^n), k_2 \leftarrow \text{KegGen}_2(1^n) \quad .1$$

$$\cdot \text{Enc}_k(m) = \text{Enc}_{2,k_2}(\text{Enc}_{1,k_1}(m)) \text{ ונדרשה } m \text{ והודעה } k = (k_1, k_2) \quad .2$$

$$\cdot \text{Dec}_k(c) = \text{Dec}_{1,k_1}(\text{Dec}_{2,k_2}(c)) \text{ והצפנה } c \text{ ונדרשה } k = (k_1, k_2) \quad .3$$

מנוכנות Π, Π_1, Π_2 גם כן נוכנה. נוכח כי Π בעלת הבנ'ל. נניח בשלילה שקיימים יריב \mathcal{A} PPT ופולינום p כך שmotkiiim באופן שכיח

$$P(\text{IND}_{\Pi, \mathcal{A}}(n) = 1) > \frac{1}{2} + \frac{1}{p(n)}$$

נבנה אלג' הסת' \mathcal{B} PPT שישברו את הבטיחות של Π_1, Π_2 בהתאם ונגיע לסתירה לכך שאחת מהן בעלת הבנ'ל.

נגדיר \mathcal{B}_1 שמשתתף בניסוי $\text{IND}_{\Pi_1, \mathcal{B}_1}(n)$ באופן הבא:

$$\cdot \mathcal{A}(1^n) \text{ מגיריל מפתח } \mathcal{B}_1 \text{ ומוחשב } k_2 \leftarrow \text{KegGen}_2(1^n) \quad .1$$

$$\cdot \mathcal{B}_1 \text{ מעביר הלאה את ההודעות } m_0, m_1 \text{ שהוא מקבל מ-} \mathcal{A} \text{.} \quad .2$$

$$\cdot \text{בקבלת הצפנה } c^* \text{ של } \Pi_1 \text{ את אחת ההודעות } c^* \leftarrow \text{Enc}_{2,k_2}(c^*) \text{ והוא מחשב } c \text{ ומუביר ל-} \mathcal{A} \text{ את } c \quad .3$$

$$\cdot \text{כש-} \mathcal{A} \text{ מחזיר גם כן את } b' \quad .4$$

נשים לב כי בניסוי $\text{IND}_{\Pi_1, \mathcal{B}_1}(n)$, \mathcal{A} מבצע בדיקת הניסוי c הוא פשוט ($\text{Enc}_k(m_b)$) והוא פשוט (ובנוסך ההסת' ש- \mathcal{A} -מנחש נכוון היא הסת' \mathcal{B}_1 מנחש נכוון הودעה הוצפנה, ככלומר מתקיים באופן שכיח

$$P(\text{IND}_{\Pi_1, \mathcal{B}_1}(n) = 1) = P(\text{IND}_{\Pi, \mathcal{A}}(n) = 1) > \frac{1}{2} + \frac{1}{p(n)}$$

נגדיר \mathcal{B}_2 שמשתתף ב- $\text{IND}_{\Pi_2, \mathcal{B}_2}(n)$ באופן הבא:

$$\cdot \mathcal{A}(1^n) \text{ מגיריל מפתח } \mathcal{B}_2 \text{ ומוחשב } k_1 \leftarrow \text{KegGen}_1(1^n) \quad .1$$

. $m'_0 \leftarrow \text{Enc}_{1,k_1}(m_0)$, $m'_1 \leftarrow \text{Enc}_{1,k_1}(m_0)$.2. כשהוא מקבל m_0, m_1 מعتبر הלאה את \mathcal{B}_1 , m_0, m_1 מعتبر הלאה את \mathcal{A} .

.3. בקבלת הצפנה c^* של Π_2 את אחת ההודעות m'_0, m'_1 , הוא מعتبر ל- \mathcal{A} את c^* .

.4. כ- \mathcal{A} -محזיר b' , \mathcal{B}_2 מוחזיר גם כן את b' .

נשים לב כי בניסוי $\text{IND}_{\Pi_2, \mathcal{B}_2}(n)$ בדיק משותף ב- Π_2 הוא הצפנה של Π_2 הודהה מוצפנת של Π_1 , כלומר הצפנה של Π וההסתה ש- \mathcal{A} צודק שווה לו של \mathcal{B}_2 , כלומר

$$P(\text{IND}_{\Pi_2, \mathcal{B}_2}(n) = 1) = P(\text{IND}_{\Pi, \mathcal{A}}(n) = 1) > \frac{1}{2} + \frac{1}{p(n)}$$

באופן שכיח.

כלומר לא Π_1 ולא Π_2 בעלות הבנייה, בסתייה לנtru. לכן Π כן בעלת הבנייה.

טענה תהיינה $X = \{X_n\}_{n \in \mathbb{N}}, y = \{Y_n\}_{n \in \mathbb{N}}$ התפלגויות בני"ל חישובית ו- f ' חשיבה בייעילות (ודטרמיניסטית), אזי ההתפלגויות $f(X), f(Y)$ הן בני"ל חישובית.

הוכחה: נניח בשלילה שקיים פ' חשיבה בייעילות עבורה $f(X), f(Y)$ נימנות להבנה חישובית. כלומר, קיים מבחין \mathcal{A} PPT ופוליאום p כך שמתקיים באופן שכיח

$$\left| \underset{x \leftarrow X_n}{P}(\mathcal{A}(1^n, f(x)) = 1) - \underset{y \leftarrow Y_n}{P}(\mathcal{A}(1^n, f(y)) = 1) \right| > \frac{1}{p(n)}$$

בנה מבחין \mathcal{D} באופן הבא: בהינתן $(1^n, z)$, החזר את הפלט של $\mathcal{A}(1^n, f(z))$ ו- f -PPT \mathcal{A} . $\mathcal{A}(1^n, f(z))$ הוא PPT. בנוסח מתקיים

$$\underset{x \leftarrow X_n}{P}(\mathcal{D}(1^n, x) = 1) = \underset{x \leftarrow X_n}{P}(\mathcal{A}(1^n, f(x)) = 1)$$

$$\underset{y \leftarrow Y_n}{P}(\mathcal{D}(1^n, y) = 1) = \underset{y \leftarrow Y_n}{P}(\mathcal{A}(1^n, f(y)) = 1)$$

לכל $n \in \mathbb{N}$ מהגדירה, ולכן באופן שכיח מתקיים

$$\left| \underset{x \leftarrow X_n}{P}(\mathcal{D}(1^n, x) = 1) - \underset{y \leftarrow Y_n}{P}(\mathcal{D}(1^n, y) = 1) \right| > \frac{1}{p(n)}$$

ולכן \mathcal{D} הוא PPT ש מבחין עם יתרון לא זניח באופן שכיח בין שתי התפלגויות בני"ל חישובית, סתייה.

שאלה האם הטענה הנ"ל נכונה גם עבור f לא חשיבה בייעילות?

פתרון לא! יהיו PRG G שמעתיק n ל- $1+n$ ונבנה דוגמה נגדית:

$$\text{כasher } G(s) \leftarrow \{0,1\}^n \text{ נדגם אחד.}$$

• היא $r \leftarrow \{0, 1\}^{n+1}$ נדם אחד.

• נדייר לכל $.G(x) = y$ עבורו $x \in \{0, 1\}^n$ $f(y) = \mathbb{1}_{\text{Im}(G)}, y \in \{0, 1\}^{n+1}$.

בגלל ש-PRG G חן בנו'ל חשיבות (יחסית ישיר). נוכיח כי $f(X), f(Y)$ ניתנות להבנה חשובית. נדייר אלג' מבחין \mathcal{D} שמקבל בית b (הערך של f על y למעשה) ומחזיר את b . מתקיים $\forall n \in \mathbb{N}$

$$\begin{aligned} & \left| \Pr_{x \leftarrow X_n} (\mathcal{D}(1^n, f(x)) = 1) - \Pr_{y \leftarrow Y_n} (\mathcal{D}(1^n, f(y)) = 1) \right| \\ &= \left| \Pr_{s \leftarrow \{0,1\}^n} (\mathcal{D}(1^n, f(G(s))) = 1) - \Pr_{r \in \{0,1\}^{n+1}} (\mathcal{D}(1^n, f(r)) = 1) \right| \\ &= \left| \frac{\Pr_{s \leftarrow \{0,1\}^n} (\mathcal{D}(1^n, f(G(s))) = 1) - \Pr_{r \in \{0,1\}^{n+1}} (\mathcal{D}(1^n, f(r)) = 1 \mid r \in \text{Im}(G))}{\Pr_{r \in \{0,1\}^{n+1}} (r \in \text{Im}(G))} - \frac{\Pr_{r \in \{0,1\}^{n+1}} (\mathcal{D}(1^n, f(r)) = 1 \mid r \notin \text{Im}(G))}{\Pr_{r \in \{0,1\}^{n+1}} (r \notin \text{Im}(G))} \right| \\ &= \left| 1 - \Pr_{r \in \{0,1\}^{n+1}} (\mathcal{D}(1^n, f(r)) = 1 \mid r \in \text{Im}(G)) \right| \\ &= \left| 1 - \frac{\text{Im } G}{2^{n+1}} \right| \\ |\text{Im } G| \leq |\{0, 1\}^n| \geq \frac{1}{2} \end{aligned}$$

שבוע VII | פונקציות פסאודו-אקראיות לעומק

תרגול

קישור

טענה תהינה PRF F, G כך שעבור מפתחות באורך n F_k מעתיקה מ- n -ל- (n) ℓ וכך גם G_k כאשר ℓ כשלשה. אזי לכל מבחין \mathcal{D} קיימת ν זניחה כך ש-

$$\left| \Pr_{k \leftarrow \{0,1\}^n} (D^{F_k}(1^n) = 1) - \Pr_{k \leftarrow \{0,1\}^n} (D^{G_k}(1^n) = 1) \right| \leq \nu(n)$$

כמעט תמיד. כלומר, שלא ניתן להבחן חשיבות בין G_k ו- F_k .

הוכחה: עקרונית כל אלג' PPT עם גישה לאורקל לא יכול להבדיל בין F לפ' אקראית באמצעות וגם לא בין G לבין פ' אקראית ולכך גם לא בינהם ישירות.

$$\begin{aligned} & \left| \Pr_{k \leftarrow \{0,1\}^n} (D^{F_k}(1^n) = 1) - \Pr_{k \leftarrow \{0,1\}^n} (D^{G_k}(1^n) = 1) \right| \\ & \stackrel{\Delta}{\leq} \left| \Pr_{k \leftarrow \{0,1\}^n} (D^{F_k}(1^n) = 1) - \Pr_{h \leftarrow \text{Func}_{n \rightarrow \ell(n)}} (D^h(1^n) = 1) \right| \\ & \quad + \left| \Pr_{k \leftarrow \{0,1\}^n} (D^{G_k}(1^n) = 1) - \Pr_{h \leftarrow \text{Func}_{n \rightarrow \ell(n)}} (D^h(1^n) = 1) \right| \end{aligned}$$

$\text{PRF } G_k, F_k \leq \nu_1(n) + \nu_2(n)$

$\nu = \text{סגירות לחיבור של זניחות}$

■

טענה תהי F שפעלה על n -bits ומייצרת פונקציית PRF. נגידר $.k, x \in \{0,1\}^n, n \in \mathbb{N}, H_k(x) = F_{F_k(0^n)}(x) \parallel F_k(x)$ אזי H היא פונקציה א/oracle- F .

הוכחה: נמצא מבחין \mathcal{D} עם יתרון לא זניח. ננסה לנצל את התלות בין המפתחות של החצאים. נגידר באופן הבא:

1. בהינתן 1^n וגיישת א/oracle- \mathcal{O} (או h או H_k) חשב את $\alpha_0 || \beta = \mathcal{O}(0^n) \in \{0,1\}^{2n}$ באמצעות הא/oracle כאשר $\alpha_0 \in \{0,1\}^n$ ו- $\beta \in \{0,1\}^n$.

2. חשב את הערך $\alpha_1 = F_\beta(0^n) \in \{0,1\}^n$ (לא דרך הא/oracle, F ידוע לנו בלי קשר כי הוא לא חלק מהণיסוי).

$$. b' = \begin{cases} 1 & \alpha_0 = \alpha_1 \\ 0 & \alpha_0 \neq \alpha_1 \end{cases} . \text{ החזר}$$

• אם הא/oracle הוא H אז $\alpha_0 = \alpha_1 = F_k(0^n)$ ולכן $\alpha_0 = F_{F_k(0^n)}(x)$ אזי $k \leftarrow \{0,1\}^n$ עם H_k תמיד נכון.

מחזיר 1.

• אם הא/oracle הוא h אז $\alpha_0, \beta \in \text{Func}_{n \rightarrow 2n}$ מתפלגים אחיד וב"ת וולכן D מחזיר 1 בחסת $\frac{1}{2^n}$ (ההסת' שההודעה המוצפנת שווה לערך אקריאי וב"ת).

סה"כ קיבל

$$\begin{aligned} \left| \Pr_{k \leftarrow \{0,1\}^n} (D^{H_k}(1^n) = 1) - \Pr_{h \leftarrow \text{Func}_{n \rightarrow 2n}} (D^h(1^n) = 1) \right| &= \left| \Pr_{k \leftarrow \{0,1\}^n} (F_{F_k(0^n)}(0^n) = F_{F_k(0^n)}(0^n)) - \Pr_{\alpha_0, \beta \leftarrow \{0,1\}^n} (\alpha_0 = F_\beta(0^n)) \right| \\ &= \left| 1 - \frac{1}{2^n} \right| = 1 - \frac{1}{2^n} \end{aligned}$$

■ שזה בודא יתרון לא זניח של \mathcal{D} על פני H_k , כלומר H_k אינה PRF.

שאלה תהי F שפעלה על n -bits ומייצרת פונקציית PRF. נגידר $(k_1, k_2) \in \{0,1\}^{2n}$ ו- $W_{k_1, k_2}(x) = F_{F_{k_1}(0^n)}(x) \parallel F_{k_2}(x)$ אזי W היא פונקציה PRF?

ו- $x \in \{0,1\}^n$

פתרונות כן! השינוי לעומת הפעם הקודמת הוא שיעכשו המפתחות של החזאים ב"ת ומעט אחידים (עד כדי PRF).

טענה תהי $(\mathcal{D}, \text{PRF } W)$ מתקיים באופן שכיich $t_1, t_2 \in \{0, 1\}^n$ לכל $x \in \{0, 1\}^n$ $\mathcal{D}^{W_{t_1, t_2}}(x) = H_{t_1}(x) \parallel \mathcal{D}^{W_{t_1, t_2}}(x) = H_{t_2}(x)$.

הוכחה: נניח בשלילה שאינה PRF, כלומר \mathcal{D} ופולינום $p(n)$ עבורם מתקיים באופן שכיich

$$\left| \Pr_{t_1, t_2 \in \{0, 1\}^n} (\mathcal{D}^{W_{t_1, t_2}}(1^n) = 1) - \Pr_{w \leftarrow \text{Func}_{n \rightarrow 2n}} (\mathcal{D}^w(1^n) = 1) \right| > \frac{1}{p(n)}$$

ובכל מקרה $w = h \parallel f$ כאשר $h, f \in \{0, 1\}^n$ ומתקיימות אחדות ב"ת, כי כל ערך של h ב"ת באחר.

בנעה אורך היברידי שחציו השמאלי $h \leftarrow \text{Func}_{n \rightarrow n}$ וחציו הימני F_{t_2}

$$\begin{aligned} \frac{1}{p(n)} &< \left| \Pr_{t_1, t_2 \in \{0, 1\}^n} (\mathcal{D}^{H_{t_1} \parallel F_{t_2}}(1^n) = 1) - \Pr_{h, f \leftarrow \text{Func}_{n \rightarrow n}} (\mathcal{D}^{h \parallel f}(1^n) = 1) \right| \\ &\stackrel{\Delta}{\leq} \left| \Pr_{t_1, t_2 \in \{0, 1\}^n} (\mathcal{D}^{H_{t_1} \parallel F_{t_2}}(1^n) = 1) - \Pr_{h \leftarrow \text{Func}_{n \rightarrow n}, t_2 \in \{0, 1\}^n} (\mathcal{D}^{h \parallel F_{t_2}}(1^n) = 1) \right| \\ &+ \left| \Pr_{h \leftarrow \text{Func}_{n \rightarrow n}, t_2 \in \{0, 1\}^n} (\mathcal{D}^{h \parallel F_{t_2}}(1^n) = 1) - \Pr_{h, f \leftarrow \text{Func}_{n \rightarrow n}} (\mathcal{D}^{h \parallel f}(1^n) = 1) \right| \end{aligned}$$

הבחנה בין W לבין הארגומנט היברידי, הבחנה בין הארגומנט והיברידי לבן הפ' האקרואית במת' β

• אם $\alpha > \frac{1}{2p(n)}$:

1. דוגמ \mathcal{A} וMRI' את \mathcal{D} .

2. בכל פעם ש- \mathcal{D} מבקש ערך חישוב על ערך x מאורך \mathcal{O} , נחשב $\mathcal{O}'(x) \parallel F_{t_2}$ כאשר $\mathcal{O}' \in \{h, H_{t_1}\}$ ונעביר ל- \mathcal{D} (כדי

שיבידיל בין $H_{t_1} \parallel F_{t_2}$.

3. נחזיר את הפלט של \mathcal{A} .

מתקיים מכך שערך החזרה של \mathcal{A} זהה לזו של \mathcal{D} כי

$$\Pr_{t_1 \leftarrow \{0, 1\}^n} (\mathcal{A}^{H_{t_1}}(1^n) = 1) = \Pr_{t_1, t_2 \leftarrow \{0, 1\}^n} (\mathcal{D}^{H_{t_1} \parallel F_{t_2}}(1^n) = 1)$$

$$\Pr_{h \leftarrow \text{Func}_{n \rightarrow n}} (\mathcal{A}^h(1^n) = 1) = \Pr_{h \leftarrow \text{Func}_{n \rightarrow n}, t_2 \in \{0, 1\}^n} (\mathcal{D}^{h \parallel F_{t_2}}(1^n) = 1)$$

ולכן מתקיים באופן שכיich

$$\left| \Pr_{t_1 \leftarrow \{0, 1\}^n} (\mathcal{A}^{H_{t_1}}(1^n) = 1) - \Pr_{h \leftarrow \text{Func}_{n \rightarrow n}} (\mathcal{A}^h(1^n) = 1) \right| = \alpha > \frac{1}{2p(n)}$$

בסתירה להיות H_{t_1} PRF.

- אם $\beta > \frac{1}{2p(n)}$: נוכח כי F אינה PRF. נבנה \mathcal{B} שմבחן בגישת אורקל בין F_{t_2} לבין f (נדגמת אחד). הפעם נדגום $.H_{t_1}||F_{t_2}, H_{t_1}||f$ ורץ את \mathcal{D} ובכל פעם שיבקש ערך מהאורקל שלו \mathcal{O} , ניתן לו את $(x) || \mathcal{O}'(x)$ כך שיבידיל בין f מוגעים לסתירה להיות F_{t_2} האופן כמו לפני, בגלל ש- \mathcal{B} - \mathcal{M} מבצע סימולציה מושלמת של הבדיקה \mathcal{D} בין $H_{t_1}||F_{t_2}, H_{t_1}||f$, מוגעים לסתירה להיות PRF (הסטודנטית המשקיפה תוכיה זאת באופן מלא).

■

טענה תהי PRF F המקיימת n -ל- n עם מפתחות באורך n וא- H היא $H_k(x) = F_{F_k(0^n)}(x)$ לכל $k, x \in \{0,1\}^n$.

הוכחה: נניח בשלילה ש- H אינה PRF, אך קיים PPT מבחן \mathcal{D} ומולינום p כך שמתקיים באופן שכיח

$$\left| \Pr_{k \leftarrow \{0,1\}^n}(\mathcal{D}^{H_k}(1^n) = 1) - \Pr_{h \leftarrow \text{Func}_{n \rightarrow n}}(\mathcal{D}^h(1^n) = 1) \right| > \frac{1}{p(n)}$$

נשתמש בארגומנט היברידי בין H_k ל- h שיחה f מתפלגת אחד. כלומר,

$$\begin{aligned} \frac{1}{p(n)} &< \left| \Pr_{k \leftarrow \{0,1\}^n}(\mathcal{D}^{H_k}(1^n) = 1) - \Pr_{h \leftarrow \text{Func}_{n \rightarrow n}}(\mathcal{D}^h(1^n) = 1) \right| \\ &\stackrel{\Delta}{\leq} \left| \Pr_{k \leftarrow \{0,1\}^n}(\mathcal{D}^{H_k}(1^n) = 1) - \Pr_{f \leftarrow \text{Func}_{n \rightarrow n}}(\mathcal{D}^{F_{f(0^n)}}(1^n) = 1) \right| \\ &\quad + \left| \Pr_{f \leftarrow \text{Func}_{n \rightarrow n}}(\mathcal{D}^{F_{f(0^n)}}(1^n) = 1) - \Pr_{h \leftarrow \text{Func}_{n \rightarrow n}}(\mathcal{D}^h(1^n) = 1) \right| \end{aligned}$$

- אם $\alpha > \frac{1}{2p(n)}$: כלומר \mathcal{D} מצליח להבחן בין $F_{F_k(0^n)}, F_{f(0^n)}$ ביתרין לא זניח. נבנה \mathcal{A} שמבחן בהסתמך לא זניחת בגישת אורקל בין F_k ובין f וכן נגיע לסתירה כי F הוא PRF. תהיה לו גישה לאורקל $\mathcal{O}' \in \{F_k, f\}$ כדי לסמלץ עבור \mathcal{D} גישה לאורקל $\mathcal{O} \in \{F_{F_k(0^n)}, F_{f(0^n)}\}$.

בහינתו 1^n וגישה ל- $\mathcal{A}, \mathcal{O}'$, מבקש את $\mathcal{O}'(0^n), t = \mathcal{O}'(0^n)$, מרים את \mathcal{D} עם סימולץ אורקל F_t (שבהתאם ל- \mathcal{O}' יהיה או הארゴומנט ההיברידי או H_k) ומוחזיר את הערך של \mathcal{D} .

מתקיים מהיות \mathcal{A} סימולציה מושלמת של \mathcal{D} עם האורקל \mathcal{O} האמור,

$$\begin{aligned} \Pr_{k \leftarrow \{0,1\}^n}(\mathcal{A}^{F_k}(1^n) = 1) &= \Pr_{k \leftarrow \{0,1\}^n}(\mathcal{D}^{F_{F_k(0^n)}}(1^n) = 1) \\ \Pr_{f \leftarrow \text{Func}_{n \rightarrow n}}(\mathcal{A}^f(1^n) = 1) &= \Pr_{f \leftarrow \text{Func}_{n \rightarrow n}}(\mathcal{D}^{F_{f(0^n)}}(1^n) = 1) \end{aligned}$$

ולכן

$$\left| \Pr_{k \leftarrow \{0,1\}^n}(\mathcal{A}^{F_k}(1^n) = 1) - \Pr_{f \leftarrow \text{Func}_{n \rightarrow n}}(\mathcal{A}^f(1^n) = 1) \right| = \alpha > \frac{1}{2p(n)}$$

כלומר F ניתנת להבנה ביותרון לא זניח מפ' אקרואית באמת, כלומר היא אינה PRF - סתירה.

$$\text{• אם } \beta > \frac{1}{2p(n)} \text{ •}$$

$$\underset{f \leftarrow \text{Func}_{n \rightarrow n}}{P} \left(\mathcal{D}^{F_{f(0^n)}} (1^n) = 1 \right) = \underset{k \leftarrow \{0,1\}^n}{P} \left(\mathcal{A}^{F_k} (1^n) = 1 \right)$$

כלומר באופן שכיח

$$\left| \underset{k \leftarrow \{0,1\}^n}{P} \left(\mathcal{D}^{F_t} (1^n) = 1 \right) - \underset{h \leftarrow \text{Func}_{n \rightarrow n}}{P} \left(\mathcal{D}^h (1^n) = 1 \right) \right| = \beta > \frac{1}{2p(n)}$$

.PRF F להיות

נסים את הוכחת הטענה המרכזית (לרשומכם היא הטענה [כאן](#)). מהטענה השנייה, F_k היא PRF וובנוסף $F_{F_k(0^n)}$ היא PRF מוגנתו. מהטענה הראשונה, הצמדה של PRF-ים ב"ת היא PRF בעצמה. $W_{t_1,t_2} = F_{F_{t_1}(0^n)} || F_{t_2}$.

■

שבוע VII | אימות, פונקציות האש והצפנה מאומתת

הרצאה

חלק א' של הרצאה ([קישור](#))

נuzeב לעתה מערכות הצפנה ונעבור למערכות אימות.

נניח כי אליס ובוב מתקשרים ואיב שולטות על העורך ביניהם כאשר העורך לא מוצפן. נרצה לוודא שההודעה לא משתנה באמצעות, ובמקום תשלום לצ'רלי עשרה דולר כמו שאليس ביקשה, בבוב ישלים 10,000 Ci איב שינה את ההודעה לכך. ככלומר, נרצה לוודא שההודעה לא שונתה במהלך מעבר ההודעה.

הערה בהצפנה אנחנו מבטיחים סודיות ואמות אוחנו מבטיחים שלמות - שתי מטרות שונות לגמרי שלא בהכרח מבטיחות אחת את השניה.

הגדרה מערכת אימות היא $\Pi = (\text{Gen}, \text{Mac}, \text{Vrfy})$ כאשר :

• Gen הוא אלג' יצרת מפתחות שמקבל קלט 1^n ומחזיר מפתח k .

• Mac הוא אלג' ייצור תגים שמקבל קלט מפתח k ו הודעה $m \in \{0,1\}^*$ ופולט Tag $t \in \{0,1\}^*$.

• Vrfy הוא אלג' אימות שמקבל קלט מפתח k , הודעה m וTAG t ופולט b שימושתו האם ההודעה נדחתה או התקבלה.

נאמר כי היא נכונה אם $\text{Vrfy}_k(m, \text{Mac}_k(m)) = 1, \forall k, m$

כדי לתקשר באמצעות Π , ניצור מפתח k , נפעיל את ה-Mac על m , נשלח את (m, t) לצד השני, הוא יפעיל את Vrfy על (m, b) ויקבל ויחליט האם ההודעה אמיתית או לא.

נרצה למנוע מכל ייריב PPT ליצור תגים חוקיים עבור הודעות שהוא רוצה. נניח כי ליריב יש גישת אורקל למערכת האימות, ונרצה למנוע ממנו למצואו תג חוקי עבור כל הודעה חדשה שהיא.

הגדירה נגידר את הניסוי $\text{MacForge}_{\Pi, \mathcal{A}}(n)$ באופן הבא:

- ניצור מפתח באמצעות $.k \leftarrow \text{Gen}(1^n)$
- נתן ל- \mathcal{A} גישת אורקל-. Mac_k .
- \mathcal{A} יחזיר בשלב כלשהו זוג (m^*, t^*) .

$$\text{MacForge}_{\Pi, \mathcal{A}}(n) = \begin{cases} 1 & \text{Vrfy}_k(m^*, t^*) = 1 \wedge m^* \notin \mathcal{Q} \\ 0 & \text{אחרת} \end{cases}$$

נכרייז את ערך הניסוי להיות ע"י האורקל, ככלומר שהיריב צדק וגם לא ביקש לאמת את ההודעה שהחזיר.

הגדירה נאמר כי מערכת אימות Π היא בטוחה אם לכל ייריב PPT \mathcal{A} , קיימת ν זניחה כך ש-

$$P(\text{MacForge}_{\Pi, \mathcal{A}}(n) = 1) \leq \nu(n)$$

הערה הגדרה זו לא מונעת שלילה חוזרת של אותה הודעה מאומנת, לדוגמה שאليس תבקש מבוב "שלח לצ'רלי 2000 ש"ח" ואיבר תשלח לבוב את ההודעה הזו ארבע פעמים. אפשר לדוגמה להוסיף ערך חח"ע לכל העברה וכן למנוע מתקפת חוזרת כזו.

תהי $F : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ נגידר מערכת אימות Π באופן הבא:

- יצירת מפתחות: דוגום $.k \leftarrow \{0, 1\}^n$
- יצירת Tag: בהינתן $.t = F_k(m), m \in \{0, 1\}^n$ ו- $k \in \{0, 1\}^n$
- ויזוא: בהינתן n $F_k(m) = t$ $\forall k \in \{0, 1\}^n, m \in \{0, 1\}^n, t \in \{0, 1\}^n$

משפט אם F היא PRF, אז Π היא מערכת אימות בטוחה.

הוכחה: נעשה זאת ברידוקציה. נניח בשלילה שקיים \mathcal{A} ייריב PPT ו- p פולינום כך ש- $P(\text{MacForge}_{\Pi, \mathcal{A}}(n) = 1) \geq \frac{1}{p(n)}$. נגידר $\mathcal{D}^{\mathcal{O}}$ ייריב עם גישת אורקל-. $\{h, F_k\}$ כאשר h נדגם אחד ו- k נדגם אחד באופן הבא:

נקרא ל- \mathcal{A} ונגיב לכל בקשת אורך שלו (הוא חשוב שהוא מאמת הودעות) ב- $t = \mathcal{O}(m)$. נקבל בסוף מ- \mathcal{A} (m^*, t^*) וNFLot

$$\begin{cases} 1 & \mathcal{O}(m^*) = t^* \wedge m^* \notin \mathcal{Q} \\ 0 & \text{אחרת} \end{cases}$$

ראשית ברור ש- \mathcal{D} -הו PPT.

■ למעשה הרעיון כאן הוא שאם \mathcal{A} מצליח妾, כנראה שמדובר, F_k , אחרת לא היה לו מושג כי זו פ' אקראית.

• אם $\mathcal{A}, \mathcal{O} = F_k$ בדיק נמצא בניסוי (n) = 1 MacForge _{Π, \mathcal{A}} (n) = 1 וכאן MacForge _{Π, \mathcal{A}} (n) = 1.

• אם $h = \mathcal{O}$ הייתה פ' אקראית באמות, אם $Q \neq m^*$, נקודת המבט של \mathcal{A} ב"ת למחרי ב- (m^*) ולכן הוא יחש בהצלחה בהסת' $\frac{1}{2^n}$.

סח'כ מתקיים באופן שכיח

$$\begin{aligned} |P(\mathcal{D}^{F_k}(1^n) = 1) - P(\mathcal{D}^h(1^n) = 1)| &= \left| P(\text{MacForge}_{\Pi, \mathcal{A}}(n) = 1) - \frac{1}{2^n} \right| \\ &\geq \frac{1}{p(n)} - \frac{1}{2^n} \end{aligned}$$

זה יתרון לא זניח על הבדלה בין פ' אקראית ל- F_k , בסתריה להיות F_k PRF.

נרצה עתה לאמת הודעות באורך פולינומייאלי - נוכל לעשות זאת באמצעות שרשור אימוטי הודעות קלאסי.

ניסיונות כושלים ליצירת מערכות אימוט בטוחות להודעות ארוכות

נניח כי יש לנו מערכת אימוט להודעות באורך קבוע $\hat{\Pi} = (\hat{\text{Gen}}, \hat{\text{Mac}}, \hat{\text{Vrfy}})$. נרצה להגדיר קבוצה Π מערכות אימוט להודעות באורך שרירותי.

.1

$$\text{Gen} = \hat{\text{Gen}}$$

$$\begin{aligned} \text{Mac}_k(m) &= (t_1, \dots, t_d), t_i = \hat{\text{Mac}}_k(m_i) \\ \text{Vrfy}_k((m_1, \dots, m_d), (t_1, \dots, t_d)) &= \begin{cases} 1 & \hat{\text{Vrfy}}_k(m_i, t_i) = 1, \forall i \\ 0 & \text{אחרת} \end{cases} \end{aligned}$$

זהו ניסיון לא מוצלח שכן אם היריב מבקש לאמת את $t = (t_1, t_2)$, מתקבל $m = (m_1, m_2)$ ואז מחזיר $(m^*, t^*) = (m_1, t_1)$ והוא ניחח בניסוי בהסת' 1. אפ"פ שיטתה זו עבדה בהצפנה, הצפנה ואמות זה לא אותו הדבר!

2. הבעיה בדוגמה הקודמת היא שלא התחשבנו במספר הבלוקים (בין היתר) ויכולנו לזייף כל רישא של הודעות שאומתו מהאורקל, נסה להתחשב במספר הבלוקים

$$\text{Gen} = \hat{\text{Gen}}$$

$$\text{Mac}_k(m) = (t_1, \dots, t_d), t_i = \hat{\text{Mac}}_k(d, m_i)$$

$$\text{Vrfy}_k((m_1, \dots, m_d), (t_1, \dots, t_d)) = \begin{cases} 1 & \hat{\text{Vrfy}}_k((d, m_i), t_i) = 1, \forall i \\ 0 & \text{אחרת} \end{cases}$$

זה עדין לא עובד, כי אנחנו לא מתחשבים בסדר עכשו - עברו $m = (m_1, m_2)$ נקבל $t = (t_1, t_2)$ והיריב יוכל להחזיר

$$(m^*, t^*) = ((m_2, m_1), (t_2, t_1))$$

3. עתה נתחשב גם במספר וגם בסדר הבלוקים

$$\text{Gen} = \hat{\text{Gen}}$$

$$\text{Mac}_k(m) = (t_1, \dots, t_d), t_i = \hat{\text{Mac}}_k(d, i, m_i)$$

$$\text{Vrfy}_k((m_1, \dots, m_d), (t_1, \dots, t_d)) = \begin{cases} 1 & \hat{\text{Vrfy}}_k((d, i, m_i), t_i) = 1, \forall i \\ 0 & \text{אחרת} \end{cases}$$

זה גם לא עובד כי אנחנו לא מתחשבים בקשר בין הודעות. לדוגמה אם נבקש מהאורקל את (t_1, t_2) עברו $m = (m_1, m_2)$ ואת (t_1, t_2) עברו $m' = (m'_1, m'_2)$ או נוכל לחתה (m^*, t^*) = $((m_1, m'_2), (t_1, t'_2))$ ולנצח שוב בהסתה.¹

ניסיונות מוצלחים למציאת אימות בטוחות להודעות ארוכות

.1

$$\text{Gen} = \hat{\text{Gen}}$$

$$\text{Mac}_k(m) = (r, t_1, \dots, t_d), t_i = \hat{\text{Mac}}_k(r, d, i, m_i), r \leftarrow \{0, 1\}^n$$

$$\text{Vrfy}_k((m_1, \dots, m_d), (r, t_1, \dots, t_d)) = \begin{cases} 1 & \hat{\text{Vrfy}}_k((r, d, i, m_i), t_i) = 1, \forall i \\ 0 & \text{אחרת} \end{cases}$$

זו אכן בטיחותית, כי היא מקשרת גם בין האיברים וההסתה להתגשות היא זניחה ($\frac{1}{2^n}$). כדי להוכיח את הנכונות שלה, צריך להראות שכל מתקפה נכנסת לאחת משלושת הקטגוריות הנ"ל (מספר הבלוקים, סדר הבלוקים וקישור בין הבלוקים) ולהוכיח ברידוקציה.

הבעיה עם מערכת זו היא שתגי האimotoות היא מאוד ארוכה.

2. כדי להימנע מtagים כל כך ארוכים, הומצא CBC-MAC המוגדר באופן הבא :

PRF F_k כאשר $t_i = F_k(t_{i-1} \oplus m_i)$ והוא $t_0 = 0^n$ (או כל ערך מסוים אחר) ו- t_d הוא $\text{Mac}_k(m) = t_d$ (לרוב AES, עם בלוקים 128 ביט) ו- d קבוע ומוסכם מראש.

הוידוא מתבצע באמצעות חישוב מחדש של t_d והשוואתו לתג שהתקבל. שימוש לב - t_d הוא באורך בלוק שלם (128 ביטים לפחות).

3. הבעיה עם הפתרון הנ"ל הוא שהוא לוקח הרבה זמן לחשב אם ההודעה ארוכה, לשם כך הומצאה שיטת Hash-and-Authenticate, לשם כך הומצאה שיטת ה-Hash-and-Authenticity. השיטה מבוססת על פ' האש שמכוכנת את ההודעה הארוכה m להודעה קצרה $H(m)$ ואז נאמת עם $H(m)$ במקומות m . הבעיה כאן היא שאנחנו צריכים להבטיח שפה מאוד למצוא' m כך ש- $\text{H}(m') = \text{H}(m)$.

חלק ב' של ההרצאה (קישור)

הגדירה נאמר כי פ' האש חסינה בפני התנשויות אם קשה מאוד למצוא' $x' \neq x$ כך ש- $\text{H}(x) = \text{H}(x')$.

הערה מבחינת היוריסטיית-קונקרטיבית, פ' האש לא מקבלת מפתח ומוחזר ערך שבתקווה לא מתנגש עם אף ערך אחר שאנו יכולים למצוא' בזמן סביר.

מבחינה אסימפטוטית, פ' האש תקבל מפתח נדגם אחד.

הגדירה (פ' האש בטוחה אסימפטוטית) פונקציית האש היא $\Phi = (\text{Gen}, H)$ כאשר :

• Gen הוא אלגוריתם ייצור מפתחות המקבל כקלט 1^n ופולט מפתח s .

• H הוא אלגוריתם חישוב שמקבל כקלט s ו- $x \in \{0, 1\}^{\ell(n)}$ ומחזיר $\{0, 1\}^{*\ell(n)}$.

הגדירה נגדיר את הניסוי $\text{HashColl}_{\Phi, \mathcal{A}}(n)$ באופן הבא :

• נדגום $s \leftarrow \text{Gen}(1^n)$ וניתן אותו ל- \mathcal{A} .

• \mathcal{A} יחזיר לנו x, x' .

$$\text{HashColl}_{\Phi, \mathcal{A}}(n) = \begin{cases} 1 & H_s(x) = H_s(x'), x \neq x' \\ 0 & \text{ארחת} \end{cases}$$

נכרי על ערך הניסוי להיות

הגדירה נאמר כי Φ בטוחה בפני התנשויות אם לכל יריב PPT קיימת פ' זניחה נ' כך ש-

$$P(\text{HashColl}_{\Phi, \mathcal{A}}(n) = 1) \leq \nu(n)$$

הערה בנויגוד לכל ההגדרות עד כה, אנחנו נתונים ליריב את המפתח והוא לא סודי כמו בפעמים הקודמות, ועודין מצפים ממנו לא להיות יכול למצוא התנשויות אלא בהסת' זניחה.

מתקפת יום ההולדת

בhinint ℓ {0,1} $^{\ell}$ → {0,1}*: $\{0,1\}^{\ell} \rightarrow \{0,1\}^*$ ב' האש בAOפנ אחד וב'ת ונמצא זוג התנגשיות בהסת' קבוצה. כמובן, לכל פ' האש שמעתיקה לפטיטים באורך ℓ יש מתקפת בזמן ריצה אקספוננציאלי-ב- ℓ . לכן כוון משתמשים בפ' האש עם $128 \geq \ell$. האלג' שביהם משתמשים כוון הם SHA-1 ו-SHA-2 ו-SHA-3, שגם בטוחות אך טרם הוחלפו מפאת קשיי שניוי מערכות והכי עדכניות. עבור פ' האש הפופולריות הרעיון המרכזי הוא פ' האש פנימית על אורך פנימי שמופעלת בצורה כזו או אחרת על קלט ארוך יותר.

Hash-and-Authenticate

בhinintן מערכת אימות $\hat{\Pi} = (\hat{\text{Gen}}, \hat{\text{Mac}}, \hat{\text{Vrfy}})$ גדרת עם מפתח, גדרת את מערכת אימות הודעות באורך שרירותי $\Pi = (\text{Gen}, \text{Mac}, \text{Vrfy})$ באפן הבא:

- יצירת מפתחות: בhinint n , נדגם $(k, s) \leftarrow \hat{\text{Gen}}(1^n)$ ו- s נפלוט
- יצירת תגים: בhinintן קלט $t = \hat{\text{Mac}}_k(H_s(m))$ ו- $m \in \{0,1\}^*$ (k, s) ו- $t \in \{0,1\}^*$ נפלוט
- וודוא: בhinintן קלט $\hat{\text{Vrfy}}_k(H_s(m), t) \in \{0,1\}^*$ (k, s) ו- $m \in \{0,1\}^*$

משפט אם $\hat{\Pi}$ היא מערכת אימות בטוחה ו- Φ חסינה בפני התנגשיות אז Π היא מערכת אימות בטוחה.

הוכחה: נוכח כרגיל ברידוקציה, רק שהפעם נניח בשלילה שיש יריב עם יתרון לא זניחה וنبנה זייפן ל- $\hat{\Pi}$ או מוצא התנשויות עבור Φ , וכן בARGUMENT הירידי לא עניין אותנו איזה מהם יתקיים, רק שאחד מהם בטוח מתקיים.

נניח בשלילה כי Π אינה בטוחה, כלומר PPT יריב \mathcal{A} מבקש את התג של הודעה m_i עבורו מתקיים $m_i \neq m^*$ אבל $H_s(m_i) = H_s(m^*)$. נגידר את מאורע ההתנגשות להיות האם במהלך הניסוי (n) , $\text{MacForge}_{\Pi, \mathcal{A}}$, מבקש את התג של הודעה m_i עבורו מתקיים $m_i \neq m^*$ אבל $H_s(m^*) \notin \{H_s(m_1), \dots, H_s(m_q)\}$.

מתקיים

$$P(\text{MacForge}_{\Pi, \mathcal{A}}(n) = 1) \leq P(\text{התנגשות}) + P(\text{MacForge}_{\Pi, \mathcal{A}}(n) = 1 \wedge \text{התנגשות})$$

נוכח כי ההסת' שכל אחד מהאיברים היא זניחה ומסגירות פ' זניחות לחיבור נקבל את הנדרש.

טענה (התנגשות) P היא זניחה ב- n .

הוכחה: נניח בשלילה שקיימים \mathcal{A} יריב PPT ופולינום p כך ש- $\frac{1}{p(n)}$ (התנגשות) P באוף שכיח.

בנה מזיף \mathcal{C} של H_s שיסמלץ את \mathcal{A} :

1. בhinintן קלט מפתח s , נדגם $(k \leftarrow \hat{\text{Gen}}(1^n), \mathcal{A})$ ונקרא ל-

2. בכל פעם ש- \mathcal{A} יבקש אימות של הودעה m , ניתן לו את $\hat{\text{Mac}}_k(H_s(m_i))$.

3. אם קיימת הודעה $m \in \{m_1, \dots, m_q\}$ כך ש- $m \neq m^*$ אבל נפלוט את (m, m^*) , אחרת \perp .

ברור שהוא PPT, ו מבחינת ערך החזרה שלו, הוא מדמה ל- \mathcal{A} . באופן מושלם את $\text{MacForge}_{\Pi, \mathcal{A}}(n)$ כאשר הוא עצמו משתמש ב- $\text{HashColl}_{\Phi, \mathcal{A}}(n)$ ולכן מתקיים

$$P(\text{HashColl}_{\Phi, \mathcal{A}}(n) = 1) = P(\text{התנשנות}) \geq \frac{1}{p(n)}$$

בסתירה להיות H_s בטוחה מפני התנשנות.

טענה אין התנשנות $\wedge 1$ זניחה. $P(\text{MacForge}_{\Pi, \mathcal{A}}(n) = 1) = 1$

הוכחה: נניח בשילhouette שקיים יריב \mathcal{A} PPT ופולינום p כך שקיימים $\frac{1}{p(n)}$ לאוון זניחה.

■ נבנה זייפן תגי אימות $\hat{\mathcal{A}}$ ל- $\hat{\Pi}$ בהתבסס על \mathcal{A} באופן הבא:

1. בהינתן הקלט 1^n נדגום $s \leftarrow \text{Gen}_H(1^n)$ ונקרא ל- \mathcal{A} .

2. בכל פעם ש- \mathcal{A} יבקש אימות על הודעה m , נחשב את $H_s(m)$ ונבקש מאורקל האימות של $\hat{\mathcal{A}}$ Tag אימות עבורו, t , ונעביר אותו ל- \mathcal{A} .

3. כ- \mathcal{A} מחזיר (m^*, t^*) נפלוט בדיקת הערך הזה גם כן.

ברור שהוא PPT. בכלל שערך החזרה הוא בדיקת ערך החזרה של סימולציה של ניסיון זיווף של \mathcal{A} את Π , מתקיים

$$\begin{aligned} P(\text{MacForge}_{\hat{\Pi}, \hat{\mathcal{A}}}(n) = 1) &= P(\text{MacForge}_{\Pi, \mathcal{A}}(n) = 1 \wedge \text{אין התנשנות}) \\ &\geq \frac{1}{p(n)} \end{aligned}$$

באופן שכיה בסתירה להיות $\hat{\Pi}$ מערכת אימות בטוחה. חשוב לשים לב כאן ש- $\hat{\mathcal{A}}$ מצליח בניסוי אם "ס" $\hat{\mathcal{A}}$ מזוייף בהצלחה הודעה וגם מאורע

■ ההtnשנות לא קורה, שזה אכן בדיקת מה שיש לנו כאן.

חלק ג' של הרצאה (קישור)

הסטודנטית המשקיפה תראה שאם יש מערכת אימות הודעות בטוחה כלשי או קיימת מערכת אימות הודעה בטוחה שבה התג חושף למגלי את תוכן ה Hodudeה שהוא אמיתי.

מערכת אימות הודעות היא בכלל לא בהכרח בטוחה בפני מתקפות זיווף, לדוגמה המערכת שבנוינו שבתוךה לפני מתקפות CPA המשתמשת $\text{Enc}_k(m; r) = (r, F_k(r) \oplus m)$ - PRF- $\text{Enc}_k(m; r) = (r, F_k(r) \oplus m)$ -

$$\text{Enc}_k(m \oplus 1^n; r) = (r, F_k(r) \oplus m \oplus 1^n)$$

ואז אפשר בקלות לזייף כל הودעה שהיא.

אם כן, נרצה להציג גם לאמת הודעות, בו בזמן - ככלומר הצפנה מאומנת. לא נציג הגדרות פורמליות עבור בטיחות של הגדרות מסווג זה, אבל כן נראה כמה נסיבות כושלים לבנייה של כלו.

ניסיונות לבניית מערכות הצפנה מאומנת

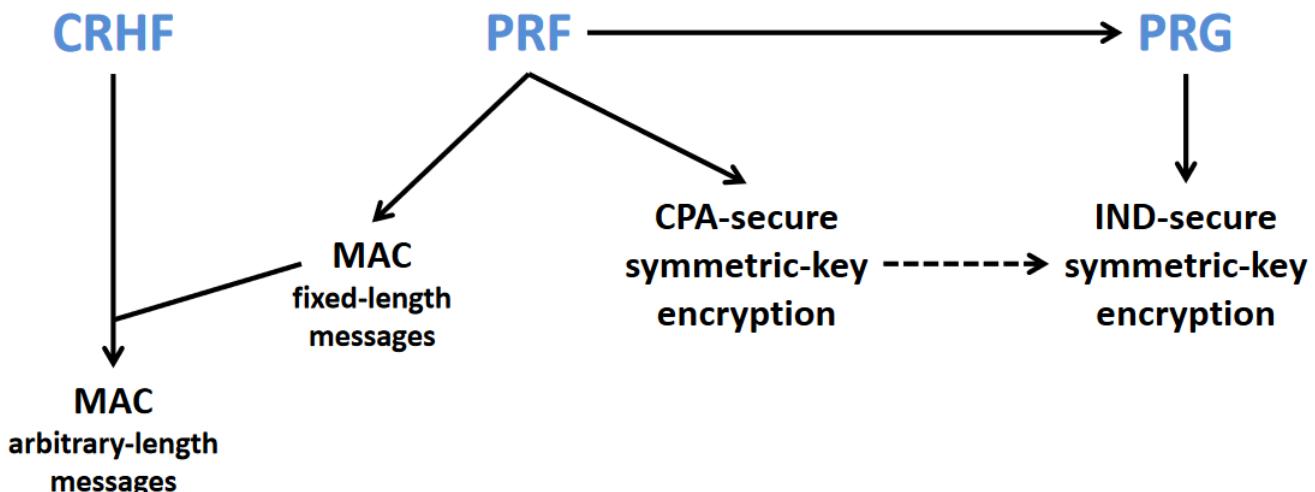
נניח כי לאليس ובוב יש מפתח סודי משותף $.k = (k_E, k_M)$

1. שיטת Encrypt-and-Authenticate: אליס תחשב $m \leftarrow \text{Dec}_{k_E}(c)$ וגם $c \leftarrow \text{Mac}_{k_M}(m)$ וbob ייחס $t \leftarrow \text{Enc}_{k_E}(m)$ במאצעות $\text{Vrfy}_{k_M}(m, t)$ את שלמות ההודעה.

הבעיה עם שיטה זו היא ש- t -מאמתת את ההודעה המקורי ולכן יכול לחושף עליה מידע (אפילו את תוכנה כפי שהסטודנטית המשקיעת בודאי התודעה כבר). נctrיך להפיק ערך אימות בלי לחושף מידע על ההודעה המקורי.

2. שיטת Encrypted-then-Authenticate: אליס תחשב $m \leftarrow \text{Dec}_{k_E}(c)$ וגם $c \leftarrow \text{Mac}_{k_M}(m)$ במאצעות $\text{Vrfy}_{k_M}(c, t)$. מבחינת בטיחות, יש לנו גם בטיחות CPA כי אם t חושף את כל ההצפנה, היא לא חשופת שום דבר על ההודעה המקורי, וגם יש לנו בטיחות מבחינות האימות באופן מיידי. באופן עקרוני, אי אפשר מבחינה חיישנית לייצר הצפנה ותג אימות חוקיים בלי גישה למפתח הסודי.

לסיכום, ראו אוior ובררו שאתם מבינים כל שני חיצים, אחרית חזו אחותה בהרצאות כי מעתה עבררים לנושא חדש לגמרי.



תרגול

[קישור](#)

שאלה תהי F מעבירה n ל- n עם מפתחות באורך n . נגיד $\Pi = (\text{KegGen}, \text{Mac}, \text{Vrfy})$ מערכת אימות הודעות באופן הבא:

. $k \leftarrow \{0, 1\}^{2n}$ 1. ייצרת מפתחות: בהינתן קלט n , האלג' דוגם

.**2. יצירת תגים: בהינתן** $\text{Mac}_k(m) = F_{F_k(0^{2n})}(m||0^n)F_k(m||1^n)$, $m \in \{0,1\}^n$ ו- $k \in \{0,1\}^{2n}$

$$\text{. וידוא: בהינתן } t \in \{0,1\}^{4n} \text{ ותגית } m \in \{0,1\}^n \text{ ו } k \in \{0,1\}^{2n}, \text{ מתקיים } \text{Vrfy}_k(m, t) = \begin{cases} 1 & t = \text{Mac}_k(m) \\ 0 & \text{אחרת} \end{cases}$$

האם Π היא בהכרח סכימת אימות הודעות בטוחה?

פרטון כו' נניח בשלילה ש- Π אינה סכמת אימיות בטוחה ונגעה לסתירה להיות $\text{PRF } F$. ממהנהה, קיים ייריב \mathcal{A} PPT ופולינום p כך שמתוקים P (MacForge $_{\Pi,\mathcal{A}}(n) = 1$) $> \frac{1}{p(n)}$ באופן שכיח. נגידר מבחן \mathcal{D} עם יתרון לא זניח בין גישת אורקל- L F_k לעומת f אקרואית

באורן הבא:

1. בהינתן קלט 1^{2n+1} , נפלוט 1 (מתעלמים מי-זוגיים).

2. בהינתן קלט 1^{2n} וגישה אורקל פ', נבקש את $a = \text{ונריצ' את } \mathcal{A}$.

3. כל פעם ש- \mathcal{A} מבקש-tag אימומות עבור הودעה $\{0,1\}^n$, $m \in \{0,1\}^n$, נבקש מהאורקל $(\mathcal{O}, m\|1^n)$ וnochoir ל- \mathcal{A} את

$$t = F_a(m||0^n) || \mathcal{O}(m||1^n)$$

וונסמן ב- \mathcal{Q} את אוסף ההודעות ש- A ביקש מהאורקל לאמת.

$$.b' = \begin{cases} 1 & t^* = F_a(m^*||0^n) \parallel \mathcal{O}(m^*||1^n) \text{ נחזיר } (m^*, t^*) \\ 0 & \text{אחרת} \end{cases} . \text{Cash-out}_{\mathcal{A}}$$

רעיון הניסוי כאן הוא ש- \mathcal{A} יצדק ביתרונו לא זניח רק אם זו F_k ובהסת' זניחה אם זה f .

• אם $\mathcal{O} = F_k$ אז \mathcal{O} עברו שיכח מטקיים באוףן MacForge _{Π, A} (n) בבדיקה את \mathcal{D} מסמלץ- A .

$$k \leftarrow \frac{P}{\binom{0,1}{2^n}} \left(D^{F_k} \left(1^{2n} \right) = 1 \right) = P \left(\text{MacForge}_{\Pi, \mathcal{A}}(n) = 1 \right) > \frac{1}{p(n)}$$

ונב $t^* = F_{f(0^{2n})}(m^*||0^n)||f(m^*||1^n)$ מוכיח $1 \in \mathcal{O}$ מכיוון $f \leftarrow \text{Func}_{2n \rightarrow 2n} : f \in \mathcal{O}$ • $m^* \notin \mathcal{Q}$

ננסמן t_2^* וולכן אם \mathcal{D} מחזיר 1 הרי ש- $t_2^* = f(m^*||1^n)$ ומם $m^* \notin \mathcal{Q}$ (זה לא אס' ס' כי זינחנו את התנאי על t_1^*). נדגם באקראי וב"ת בכל ערכי (x) f הידועים ל- \mathcal{A} בהינתן ש- $\mathcal{Q} \neq \emptyset$ (אף ערך ש- \mathcal{A} ראה לא קשור

לחוודעה m^* ו- f אקרואית באמת. לכן $t_2^* = f(m^* || 1^n)$ מתקיים בהסת' $\frac{1}{2^{2n}}$. ככלומר,

$$\begin{aligned}
& P_{f \leftarrow \text{Func}_{2n \rightarrow 2n}}(D^f(1^{2n}) = 1) = P(m^* \notin \mathcal{Q} \wedge t^* = F_{f(0^{2n})}(m^* || 0^n) || f(m^* || 1^n)) \\
& \leq P(m^* \notin \mathcal{Q} \wedge t_2^* = f(m^* || 1^n)) \\
& = P(t_2^* = f(m^* || 1^n) \mid m^* \notin \mathcal{Q}) P(m^* \notin \mathcal{Q}) \\
& \leq P(t_2^* = f(m^* || 1^n) \mid m^* \notin \mathcal{Q}) \cdot 1 \\
& = \frac{1}{2^{2n}}
\end{aligned}$$

כלומר מצאנו שמתקיים

$$\left| \underset{k \leftarrow P_{[0,1]^{2n}}}{\Pr} \left(D^{F_k} (1^{2n}) = 1 \right) - \underset{f \leftarrow \text{Func}_{2n \rightarrow 2n}}{\Pr} \left(D^f (1^{2n}) = 1 \right) \right| > \frac{1}{p(n)} - \frac{1}{2^{2n}} > \frac{1}{2p(n)}$$

באופן שכיח ולכון מתקיים באופן שכיח

$$\left| \Pr_{k \leftarrow \{0,1\}^n} (D^{F_k}(1^n) = 1) - \Pr_{f \leftarrow \text{Func}_{n \rightarrow n}} (D^f(1^n) = 1) \right| > \frac{1}{2p(\lceil \frac{n}{2} \rceil)}$$

כלומר \mathcal{D} בעל יתרון לא-זניח על F_k לעומת F במאמה, ולכן F לא PRF סתירה.

שאלה תהי $(\text{Gen}, H) = \Phi$ משפחת פ' האש חסינות בפני התנשויות המקובלות שמעתיקה $2n$ ל- n . תהי $\mathbb{N} \rightarrow \mathbb{N}$: c פ' כלשהי של פרטן הבתיות $\mathbb{N} \in n$ כך ש- $2 > c(n)$ לכל n .

לכזוז כל גודל שرك נרצה בהינתן שיש לנו פ' H' שמכוחות באופן בסיסי.

פתרון נבחר $\text{Gen}' = \text{Gen}(1^n)$. תהי y_0 מחרוזת באורך n ביטים שרירותית, לדוגמה 0^{2n} . נגידר את $(H'_s(x))$ באוף הבא עבור $x \in \{0, 1\}^{c(n) \cdot n}$.

1. נסמן $x_c, \dots, x_1 = x$ כאשר $n = c(n)$ - כל בלוק הוא באורך n ביטים.

. $i \in [c]$ נחשב $y_i = H_s(y_{i-1} || x_i)$.

$$. H'_s(x) = y_c \text{ נציג } .3$$

טענה לכל $1 \leq c \leq i \in [c]$ אם $x \neq x'$ וגם $x, x', s \leftarrow \text{Gen}(1^n)$, $n \in \mathbb{N}$ אז קיימים $H'_s(x) = H'_{s'}(x')$

$$y_{i-1} || x_i \neq y'_{i-1} || x'_i$$

ו ג

$$H_s(y_{i-1}||x_i) = H_s(y'_{i-1}||x'_i)$$

כאשר y'_{i-1} מgive מחישוב $(H'_s(x'))$.

כלומר, אם יש ב- H'_s התנשיות הרי שההתנשיות קורת לפני השלב האחרון.

הוכחה: יהו $x' \neq x$ ונניח בשלילה כי לכל i או ש- $y_{i-1}||x_i = y'_{i-1}||x'_i$ או ש- $y_{i-1}||x_i \neq y'_{i-1}||x'_i$ ווכי

באינדוקציה על c כי $(H'_s(x) \neq H'_s(x'))$

בסיס ($c = 1$): ברור מהגדרת H'

צעד ($c - 1 \rightarrow c$)

אם קיים $j \in [c - 1]$ כך $x_j \neq x'_j$ אז מה"א מתקיים $y_{c-1} \neq y'_{c-1}$ •

$$H'_s(x_1, \dots, x_{c-1}) = H'_s(x'_1, \dots, x'_{c-1})$$

בסתירה להנחה בשלילה) ולכן מתקיים $y_{c-1} \neq y'_{c-1}||x_c \neq x'_c$. מתקיים

$$H'_s(x) = H_s(y_{c-1}||x_c) \stackrel{\text{נ"ח}}{\neq} H_s(y'_{c-1}||x'_c) = H'_s(x')$$

אם לא קיים $j \in [c - 1]$ כך $x_j \neq x'_j$ הרי ש- $x_c \neq x'_c$ (כי $x \neq x'$) ולכן שוב מתקיים $y_{c-1} \neq y'_{c-1}||x_c \neq x'_c$ •
האופן מה"א מקבלים את הצד.

■

נסים עתה את ההוכחה של הבטיחות של Φ' . נניח בשלילה שקיים \mathcal{A}' אלג' PPT ו- p פולינום כך ש-

$$P(\text{HashColl}_{\Phi', \mathcal{A}'}(n) = 1) > \frac{1}{p(n)}$$

באופן שכיח. נבנה אלג' \mathcal{A} שימצא התנשיות ב- Φ :

1. בהינתן קלט של מפתח s , נרץ את \mathcal{A}' עם אותו הקלט.

2. כ- \mathcal{A}' מחויר זוג ערכים $x, x' \in \{0, 1\}^{c \cdot n}$.

(א) אם $x = x'$ או $(0^{2n}, 0^{2n})$ נחזיר $H'_s(x) \neq H'_s(x')$ (כלומר כניעה).

(ב) אחרת, מהטענה הנ"ל קיים $i \in [c]$ כך ש- $y_{i-1}||x_i = y'_{i-1}||x'_i$ וגם $y_{i-1} \neq y'_{i-1}||x'_i$ (כלומר מצאנו

התנשיות "ביניים" בפ' האש המוקנית). נחשב את ה- i -רשאון עבورو התנאי הנ"ל מתקיים ומחויר $(y_{i-1}||x_i, y'_{i-1}||x'_i)$.

מהטענה הנ"ל, \mathcal{A}' מוצא התנשיות ב- H'_s אס"ם מוצא התנשיות ב- H_s ולכן מתקיים באופן שכיח

$$P(\text{HashColl}_{\Phi, \mathcal{A}}(n) = 1) = P(\text{HashColl}_{\Phi', \mathcal{A}'}(n) = 1) > \frac{1}{p(n)}$$

שבוע VII | פגיעות תוכנה ברמה הנמוכה

הרצאה

חלק A' של הרצאה (קיישור)

עד כה רأינו צורך מדויקות לבטיחות, הנחות בטיחות והוכחה שלhn. עם זאת, ניתן לפרט מערכות אם הן ממומשות באופן פשוט. בעקבות אחר ניהול הזיכרונות של מערכת הפעלה 32 ביט של Linux. כל הזיכרונות הוא 4 ג'יגה ביבט וכתוות זיכרונות הם מספר הקסדצימלי עם 8 תווים (0-9 או A-F) כתוות גבוקות יהיו לעלה וنمוכות למטה. כל תהליך חושב שיש לו גישה לכל הזיכרונות (זיכרון וירטואלי) ומערכת הפעלה אמונה על המרת הזיכרונות הווירטואלי לזכרון פיזי.

אזור זיכרונות

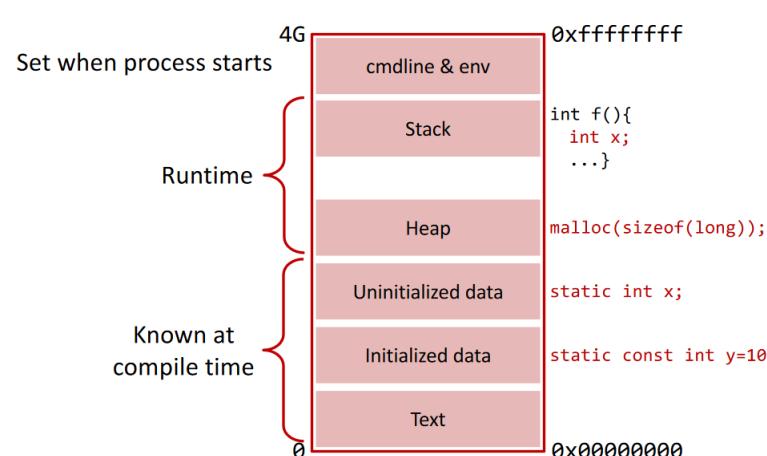
1. טקסט - קוד מכונה (אנחנו נתיחס אליו כאסמבלי).

2. מידע סטטי מאוחROL ולא מאוחROL (ידעו בזמן הקימפול).

3. מידע על הסביבה וה-Command Line.

4. מחסנית שנגמרת מכתובות כלשהי וגדלה כלפי מעלה.

5. עירימה להקצתה דינמית שמתחלת איפשהו וגדלה כלפי מעלה.



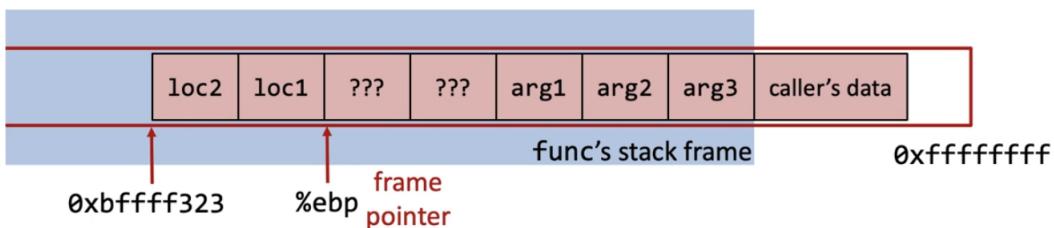
נסתכל על החלק של זמן הריצה באופן אופקי - משמאלי כתובות נמוכות (ולכן הערימה שגדלה ימינה) ומימין כתובות גבוקות (ולכן המחסנית שגדלה שמאלה).

מה קורה כשקוראים לפונקציה

```

1 void func(char *arg1, int arg2, int arg3) {
2     char loc1[4];
3     int loc2;
4     // ...
5 }
```

בקראיה לפ', נדחוף למחסנית את 3, 2, 1 בסדר יורד (קודם arg3 ובסוף arg1), נשמר מקום לשני המשתנים המקומיים ובין הארגומנטים המקומיים נשמר עוד שני מיקומים של ארבעה בתים. כדי לגשת למשתנים המקומיים או לארוגומנטים משתמשים כתובות יחסיות, שכן מבנה (המסגרת של הפ' ב-) מחסנית ידוע כבר בזמן הקומפליציה, פשוט לא איפה הוא יהיה בדיק. האוגר ebp שומר את סוף המשתנה המקומי הראשון ואז קל לדעת איפה המשתנים המקומיים. בנוסף, ידוע לנו שהארוגומנטים נמצאים ב...ebp + 8, 12, 16, ... וכמו שמננו אותם בסדר הפוך - אם היו עוד הם היו באינדקסים עליים שאחורי, ולא לפני.



עתה נראה מה קורה מ从此 של הקורא. נניח שקראנו לפ' ה'ן'ל באופן הבא

```

1 int main() {
2     // ...
3     func("Hey", 10, -3);
4     // ...
5 }
```

גם לפ', הקוראת יש %ebp שהוא מושתמש בו, וכן %esp שהוא הפוינטר לראשית המחסנית. צריך לשמר את המידע הזה איפשהו כדי שזה לא ידרס ע"י הפ' הנקראת ונאבד את המידע. לכן בקראיה לפ', נשמר בתא שמיד אחרי המשתנה המקומי האחרון, קרי ב-ebp החדש, את %ebp החדש. כשנזור מהפ', נחזיר את הערך ששמרנו לאוגר.

נצרך לדעת לאן לחזור בחזרה מהפ' הנקראת, כי זה יכול להיות בכמה מקומות שונים. קו'ו הוא אוגר שמחזיק את כתובות הפוקודה הנוכחית, וכש קופצים פשט מושנים את ערכו. בנוסף, כתובות החזרה מהקראייה ידועה כבר בזמן הקומפליציה (שורה אחת אחרי פקודת ה-call) ולאחר מכן אפשר להציג אותה במפורש בפקודות, ואז גם במקומות האחר ששמרנו לשים את %ebp בעת הקראיה לפ'. בחזרה מהפ', נציג ב-קו'ו את הכתובת ששמרנו. כך נדע לאן לחזור בפקודות והיכן המידע שלנו, כשנזור מהפ'.

סיכום קריאה לפונקציה

• הפ' הקוראת:

1. דוחפת את הארגומנטים. בסדר הפוך.
2. דוחפת את כתובת החזקה.
3. קופצת לכתובת של הפ' הנקרהת.

• הפ' הנקרהת:

1. דוחפת את פוינטר מסגרת המשתנים.
2. קובעת את פוינטר מסגרת הממחסנית להיות פוינטר הממחסנית.
3. מקופה מקום למשתנים לוקאליים.

...

4. משחררת את מקום המשתנים המקומיים (מוטביה בהקטנת פוינטר הממחסנית).
5. מחזירה את פוינטר מסגרת הממחסנית השמור למקוםו.
6. מחזירה את כתובת החזקה אל תוך קי' וקופצת אליו.

• הפ' הקוראת:

- 4 שחרור מקום הארגומנטים.

דוגמה ראו פ' והתרגום שלו לקוד אסמבלי. הسطודנית המשקיפה תבצע את הריצה של הפ' בקוד עם מחסנית דמה.

```
1 void func() { // push %ebp
2                 // mov %ebp, %esp
3         int x; // sub %esp, 4
4         x = 1; // mov %ebp-4, 1
5     }          // mov %esp, %ebp
6                 // pop %ebp
7                 // ret
```

דוגמה הנה עוד פ' ותרגומה, שימו לב לאופן בו אנחנו מגעים למשתנים המקומיים.

```
1 void func(int x) { // push %ebp
2                 // mov %ebp, %esp
3         int y; // sub %esp, 4
4         y = x; // mov %eax, (%ebp+8)
5                 // mov %ebp-4, %eax
6     }          // mov %esp, %ebp
```

```

7          // pop %ebp
8          // ret

```

חלק ב' של הרצאה (קישור)

Buffer Overflow הוא באג בקוד בשפות תוכנה לו-לבל, קרי `C + C + C + ...`, שיש להן גישה ישירה ל זיכרונו. ברוב המקרים התכנית פשוטה תקרוס כשלורה Buffer Overflow אבל אם מנצלים אותו, הוא יכול לגרום לניבת מידע, השחתתו וapeutic השגת שליטה מלאה על המערכת. זו אחת המתקפות המסוכנות והנפוצות ביותר.

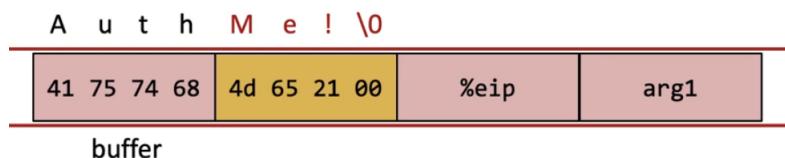
מערך (ב哀) הוא איזור זיכרונו רציף שמיוחס למשתנה כלשהו, לדוגמה מחרוזות ב-C מיוצגות ע"י באפר שנגמר בבית NULL (כלומר 0x00). האובייקט הוא שימוש בערך לגישה למיקומים מחוץ ל זיכרונו שהוקצה לו.

דוגמא כאן הקוד רקקורס, לא עשה נזק. הפ' `strcpy` מעתיקה בתים החל מהփינטר שmoveבר אליה ועד שהיא נתקלת ב-NULL. מה שקרה כאן זה שאנו נדרושים גם ארבעה בתים אחרי `buffer`, כלומר גם את `%ebp` השמור, וזה יגרום לкриיסה (ראו איור).

```

1 void func(char *arg1) {
2     char buffer[4];
3     strcpy(buffer, arg1);
4     ...
5 }
6
7 int main() {
8     char *mystr = "AuthMe!";
9     func(mystr);
10    ...
11 }

```



דוגמא הפעם הקוד הוא מסוכן, רקקורס. Unless הכתיבה דורשת את המשתנה הлокאלי `authenticated` והוא כבר לא 0, כלומר נכנס אל תוך ה-`if` אפיו שלא אמרה להיות לנו גישה. הקוד כאן לאקורס כי לא פגענו בחוקיות של שום דבר, חוץ מהשגת גישה שלא הייתה אמרה להיות לנו לחلك רגיש של הקוד.

```

1 void func(char *arg1) {
2     int authenticated = 0;

```

```

3     char buffer[4];
4     strcpy(buffer, arg1);
5     if (authenticated) {
6         // sensitive information access
7     }
8 }
9
10 int main() {
11     char *mystr = "AuthMe!";
12     func(mystr);
13     // ...
14 }
```

הערה למעשה נוכל לכתוב כמה שנרצה כל שהסטרינג יותר ארוך, עד כדי הזרקת קוד ברכזונו לתוכנה.

חלק ג' של הרצאה ([קישור](#))

הזרקת קוד מרכיבת משני חלקים: הזרקת הקוד (הכנסתו לזכרון) והרצתו (שינוי קי % למקומות ההזרקה).

הערות על הזרקת הקוד

- הקוד חייב להיות בשפת מקונה, ככלומר כבר מוקומפל, ולא ב-C או באסמבלי.
- אסור על הקוד להכיל בית מלא אפסים כי אחרת strcpy ודומה פשוט יפסיקו את ההעתקה באמצעות.
- הקוד חייב להשתמש בכתובות אבסולוטיות שכן אין לנו loader בזמן קומpileaza שיכל להמיר בין יחסיות לאבסולוטיות.

המטרה היא להזירק קוד shell שייתן גישה ל- Command Line; שטמו הרבה יותר קל להמשיך את המתקפה.

אנחנו לא בהכרח יודעים איפה הכתובת ראשונה של הקוד המוזרך ואנחנו לא יכולים להכניס פקודה של קפיצה לקוד שלנו ישירות. כדי להציג לקוד שלנו פשוט נדרוס את כתובות החזרה ששמנו במחסנית עם עוד Overflow. אם יש לנו גישה לקוד שאנחנו מרים איז אפשר לחשב את עומק המחסנית או את מיקום כתובות החזרה, אבל אם אין לנו, אנחנו לא יודעים איפה להזירק את כתובות החזרה שלנו.

מציאת כתובות החזרה

- לעבור על כל האפשרויות שזה לא ממש עוזר.
- לבצע ניחוש מושכל (בנחה שאין רנדומיזציה על הזיכרון, נדון בהמשך) - המחסנית מתחילה באותו מקום והמחסנית לרוב לא גדולה יותר מדי, וכך נוכל להגדיל את סיכויי ההצלחה שלנו.

3. להשתמש במגלשות דות. דות היא פקודה שלא מבצעת דבר, ונוכל לרפד את הקוד המוזרק עם הרבה קומ-ים ואז כל ניחוש של כתובות זיכרו בתוך הטווח המרופד (יחסית גדוֹל) יצליח.

חלק ד' של הרצאה (קישור)

עד כה עסקנו ב-Buffer Overflow רק במחסנית. אפשר לעשות את זה גם בערימה,

דוגמא הקוד הבא מאנצל Overflow בערימה. אנחנו מעתיקים את הסטרינג הראשון אל תוך הבאפר ואז עושים שרשור של הסטרינג השני לבאפר. לסיום קוראים לפ' עם הבאפר. הבעיה כאן היא שם סכום אורכי המחרוזות ארוך מ-MAX_LEN או נדרוס את המצביע הפ' השמורה ב-struct וכך יוכל לגרום להצחיבע לפ' כרצוננו, לדוגמה ל-.shellcode

```

1 typedef {
2     char buff[MAX_LEN];
3     int (*comp)(char *, char *);
4 } exploitable;
5
6 int foo(exploitable *s, char *one, char *two) {
7     strcpy(s->buff, one);
8     strcat(s->buff, two);
9     return s->comp(s->buffer, "badstring");
10 }
```

דוגמא לאובייקטים ב- C + יש שמצביעים למתחדות של האובייקט בהתאם לסוג שלו (בפולימורפים זה עוזר). השדה הזה חביב מההמשתמש אבל אפשר באמצעות Overflow קלاسي לדרס אותו כך שיצטבע כתובות כרצוננו.

דוגמא אפשר לנצל גם Overflow כדי לפגוע באובייקט שנמצא ליד הבאפר על הערימה.

דוגמא הערימה שומרת header קטן מוחבא מהמשתמש לפני פוינטר שמדובר באמצעות malloc, אם נעשה Overflow אל תוך header נוכל לגרום לערימה להשחת את עצמה.

דוגמא אפשר במקרה לעשות Overflow לבאפר, לעשות זאת במספר. אם len גדול מדי, קיבל Overflow או המספר יהפך ל-0, קלומר נעשה הקציה של 0 בתים. תלוי במערכת, יכול להיות שנקלט הקציה של 0 בתים על הערימה, ואז כל כתיבה לbuff תביא לכתיבה לכתובות מחוץ לשטחו.

```

1 void vulnerable() {
2     char *buff;
3     int len = packet_get_int();
4     if (len > 0) {
5         buff = malloc(len * sizeof(char ));
```

```

6         for ( int i = 0; i < len; i++) {
7             buff[ i ] = packet_get_string( NULL );
8         }
9     }
10 }
```

דוגמה חוץ מלכתוב לזכורן, אפשר גם לקרוא לזכרו של אמורים לגשת אליו, לדוגמה מידע רגיש. התכנית הבאה מקבלת מחרוזת מהמשתמש ומדפיסה אותה למסך.

היא עשויה זאת באופן הבא : היא מקבלת את אורך המחרוזת שהמשתמש עבר ווז קוראת אותו וכותבת חזרה למסך אותו, כאשר תווים שאינם אותיות/סימנים היא מחליפה ב-..

```

1 int main() {
2     char buf[100];
3     char *p;
4     int i, len;
5     while (1) {
6         p = fgets(buf, sizeof(buf), stdin);
7         len = atoi(p);
8         p = fgets(buf, sizeof(buf), stdin);
9         for (i = 0; i < len; i++) {
10             if (!iscntrl(buf[i])) putchar(buf[i]);
11             else putchar('.');
12         }
13         printf("\n");
14     }
15 }
```

אם המשתמש נותן `len` ארוך מההודעה עצמה (בהתעלם מהגבול של 100 תווים סה"כ), הוא יכול לקרוא מידע שהוא לא אמור לראות. אמנים זה נראה כמו קוד שברור שמסוכן, הוא הוביל לאחת מפרצויות האבטחה הći חמורות בהיסטוריה - SSL Heartbleed (2014). שרת SSL מקבל הודעה "heartbeat" שמעדכנות אותו שהששן הנוכחי עדין כי והוא מחזיר אותו. המשתמש שולח את אורך ההודעה וההודעה עצמה, וכך מהמימושים של השירותים האלה לא בדקו את תקינות האורך, וכך קיבלנו בדיקת הדוגמה הנ"ל. התוכן שאפשר לקרוא מעבר להודעה הכליל סיסמות ופתחות הצפנה מה שפגע קשות באיכות הפרטוקול.

דוגמה שימוש במצביע שבורר. לאחר שמשחררים זיכרון שהוקצתה על הערימה, אפשר להקצתו מחדש ויש סיכוי לא Kapoor שהמוקום שיוקצת יהיה זהה לשוחרר. כך, ניתן לקרוא את התוכן שהיה שם לפני ע"י הקצתה מחדש חלק מהזרקת קוד. בדוגמה הבא

אנחנו משתמשים ב-`p` גם אחרי שכבר שוחרר (מתוך חוסר תשומת לב בכתיבת הקוד) אבל הקוד שהזרקנו מצליח להשתלט על הפ' הנקראת ולעשות בו שימוש זמני. זהה פרצת האבטחה שהכי קל להגן מפני (פשוט לשים לב באיזה מצביע משתמשים).

```
1 struct foo {  
2     int (*cmp)(char *, char *);  
3 };  
4 struct foo *p = malloc(...);  
5 // program uses pointer  
6 free(p);  
7 // ...  
8 // injected code:  
9 q = malloc(...);  
10 *q = 0xdeadbeef;  
11 // ...  
12 p->cmp("hello", "hello");
```

חלק ה' של ההרצאה (קישור)

כדי להציג מחרוזות עם פורמטינג כלשהו (לדוגמא השמת מספר באמצעות מחרוזת) אפשר להשתמש ב-`printf` עם סימולי פורמט, לדוגמה ”%d : Number” מאפשר הדפסת מספר כדימלי בתוך המחרוזת. `printf` יש מספר בלתי מוגבל של משתנים שאפשר להעביר ל-`printf` כדי להציגם אל תוך פורמט המחרוזת.

דוגמה נשווה קוד בטוח ומולו קוד לא בטוח להדפסת פורמט.

```
1 void echo() {  
2     char buf[80];  
3     if (fgets(buf, sizeof(buf), stdin) == NULL)  
4         return;  
5     printf("%s", buf); // secure  
6     printf(buf); // insecure  
7 }
```

כאן הגרסה הבטוחה פשוט מדפסה את `buf` כמחרוזת ליטרלית ותו לא. הקוד הלא בטוח מתייחס ל-`buf` כפורמט עצמו, והמשתמש יכול לחת פורמט זמני שיגרום להדפסה של דברים שאנו לא רוצים לחושף. לדוגמא העברת המחרוזת ”%” תגרום ל-`printf` לחשב שהיא צריכה להציג מספר שנמצא במקומות הארגומנטים שלה. עם זאת, אין לה ארגומנטים ולכן היא פשוט מתחילה לקרוא את המיקום שאחורי איפה שהארגומנטים היו אמורים להיות, ככלומר מידע מהמחסנית שלא קשור בכלל אליה ושהיא לא אמורה לחושף למשתמש.

דוגמה עבור אותו קוד לא בטיחותי לפני, אפשר לקבל התנחות לא רצואה גם אם אפליו לא מנסים. לדוגמה אם המשתמש יזין את המחרוזת `printf("100% dave")` שזו כפובה בתנחות לא רצואה.

דוגמה המחרוזת "%d..." תקרה כמות ברכוננו של בתים מותוק מהחסנית.

דוגמה "...x%08x" תקרה בתים ותפרמטר אותםיפה בהקסאדיימאלי.

דוגמה “100% no way!” כתובת את המספר 3 לכתחות הזיכרון שלויה מצביע המיקום $b-12 + k$. זה בכלל ש- m % מצפה לפוינטור שהוא כתוב בו את מספר התווים שנכתבו עד כה (במקרה זה 100 הם שלושה תווים).

דוגמה ניצול מחרוזות פורמטן גם טכנית Overflow-Buffer אם אנחנו ניגשים לכתחות שאנו לא אמורים לגשת אליהם, כאשר מסגרת המחסנית של הפ' היא הבאר בנסיבות הזה.

תרגום

קישור

שאלה תהי F משפחת פ' שמעתקה n ל- n עם מפתחות באורך a . נתבונן במערכות ההפנה II המוגדרת באופן הבא:

$$\begin{aligned} & \cdot k \leftarrow \{0,1\}^n \text{ KegGen}(1^n) \\ & \cdot c = (r, F_k(r) \oplus m) \leftarrow \{0,1\}^n \text{ מג'יל Enc}_k(m) \\ & \cdot c = (c_1, c_2) \oplus F_k(c_1) \text{ כאשר } c_2 \leftarrow \text{מח'יר Dec}_k(c) \end{aligned}$$

בהרצאה ראיינו כי אם F היא PRF או II היא מערכת ההפנה CPA-בטוחה. עתה נעסק בכיוון השני.

1. נניח ש-II היא בעלת הבני".ל. האם F היא בהכרח ?PRF?
2. נניח ש-II היא CPA-בטוחה, האם F היא בהכרח ?PRF?

פתרון

1. לא! נביט ב- $k = F_k(x)$. ברור שהיא אינה PRF כי היא קבועה על k ולכן שונה מאוד מפ' אקראית באמת. מצד שני, II היא בעלת הבני"ל ואפיו בעלת סודיות מושלמת. זאת ממש שום F חנ"ל, מדובר בדיק ב-OTP שסביר להניח שהיא סודית מושלמת.

2. לא! אינטואיטיבית מבחין על F מול פ' אקראית באמת יכול לבקש ערכים ספציפיים לחישוב, בעוד ב-(n) הקלטים IND_{II,A}^{CPA} של F הם אקרים (a -ים) ולכן לחמק מהאחד אבל לא מהאחר.
 $F_k(x) = \begin{cases} 0^n & x = 0^n \\ H_k(x) & x \neq 0^n \end{cases}$ תהי PRF H_k שמעתקה n ל- n . נגידו שאלות החזרה שזו אינה PRF. נוכיה ש-II היא אכן CPA-בטוחה. תהי Σ מערכת ההפנה שעובדת כבשאלה רק עם H במקום F . ראיינו שזו CPA-בטוחה. נוכיה ברידוקציה כי גם II היא CPA-בטוחה.

יהי \mathcal{A} אלג' PPT. נגדיר ZERO_{Π} המאוורע בו אחת ההצענות שרוואה \mathcal{A} בניסוי $\text{IND}_{\Pi, \mathcal{A}}(n)$ גם מהאורקל וגם c^* עושה שימוש ב- $r = 0^n$. נגדיר ZERO_{Σ} באופן דומה.

$$P(\text{IND}_{\Sigma, \mathcal{A}}^{\text{CPA}}(n) = 1) = P(\text{IND}_{\Sigma, \mathcal{A}}^{\text{CPA}}(n) = 1 | \text{ZERO}_{\Sigma}^C) P(\text{ZERO}_{\Sigma}^C) + P(\text{IND}_{\Sigma, \mathcal{A}}^{\text{CPA}}(n) = 1 | \text{ZERO}_{\Sigma}) P(\text{ZERO}_{\Sigma})$$

ואילו Σ היה CPA-בטוחה ולכן מההגדרה קיימת ν' זניחה כך ש-

$$P(\text{IND}_{\Sigma, \mathcal{A}}^{\text{CPA}}(n) = 1 | \text{ZERO}_{\Sigma}^C) \stackrel{(*)}{\leq} \frac{1}{P(\text{ZERO}_{\Sigma}^C)} \left(\frac{1}{2} + \nu(n) \right) \leq \frac{1}{1 - \frac{q(n)}{2^n}} \left(\frac{1}{2} + \nu(n) \right) \leq \frac{1}{2} + \nu'(n)$$

(*) פתחנו את הגדרת ההסת' המותנת והציבנו את החסם על הסט' הצלחות הניסוי.

כאשר $q(n)$ חוסם את מספר ההצענות \mathcal{A} -מבקש מהאורקל ו- ν' היא ν זניחה שמורכבת מ- ν ו- (n) .

$$P(\text{IND}_{\Pi, \mathcal{A}}^{\text{CPA}}(n) = 1 | \text{ZERO}_{\Pi}^C) = P(\text{IND}_{\Sigma, \mathcal{A}}^{\text{CPA}}(n) = 1 | \text{ZERO}_{\Sigma}^C)$$

כי אם לא יצא לנו 0^n אף פעם אז התפלגות ההצענות זהה בין המערכות. לכן סה"כ קיבל

$$\begin{aligned} P(\text{IND}_{\Pi, \mathcal{A}}^{\text{CPA}}(n) = 1) &\stackrel{\text{הסת' שלמה}}{\leq} P(\text{IND}_{\Pi, \mathcal{A}}^{\text{CPA}}(n) = 1 | \text{ZERO}_{\Pi}^C) + P(\text{ZERO}_{\Pi}) \\ &\leq P(\text{IND}_{\Sigma, \mathcal{A}}^{\text{CPA}}(n) = 1) + \frac{p(n)}{2^n} \\ &\leq \frac{1}{2} + \nu'(n) + \frac{p(n)}{2^n} \\ &= \frac{1}{2} + \nu''(n) \end{aligned}$$

כאשר גם ν'' היא ν זניחה, שמורכבת מ- ν' ו- p שהוא פולינום שחוסם את מספר ההצענות \mathcal{A} -מבקש מהאורקל בניסוי IND_{Π, A}-בטוחה. ככלומר, Π היא CPA-בטוחה.

טענה סכימות ה-CBC-MAC אינה בטוחה כאשר היריב בניסוי MacForge_{Π, A}(n) יכול לבקש ערכי אימוט עבור הודעות מאורך משתנה (עבור d -ים שונים).

הערה CBC היה הסכימה בה התגים שלנו הם PRF $F_k(t_{i-1} \oplus m_i)$ ו- $t_0 = 0^n$ כאשר F_k היא

הוכחה: נגדיר יריב \mathcal{A} באופן הבא: בקש תג עבור $m_1 = 0^{2n}$ וכן $t_d = F_k(m_1) \oplus t_{d-1}$ (האחרון שמחושב) כאשר $t_1, t_2 \in \{0, 1\}^n$. נסמן את התשובות המתקבלות בהתחאה. נזכיר את הזוג $(m^*, t^*) = (t_1, t_2)$ (כלומר הצלחנו לאמת את t_1, t_2 , באורך 1, ב- $d = 1$, באמצעות התג של t_2 בהתחאה).

ברור שהוא יריב PPT. מתקיים

$$t_1 = \text{Mac}_k(0^n) = F_k(0^n \oplus 0^n) = F_k(0^n)$$

$$t_2 = \text{Mac}_k(0^{2n}) = F_k(F_k(0^{2n}))$$

כלומר ש-

$$\text{Mac}_k(t_1) = F_k(t_1) = F_k(F_k(0^n))$$

לכן t_2 הוא תג אימות חוקי של t_1 .

נותר להוכיח שבהסת' גבולה, \mathcal{A} . לא ביקש תג ל- $t_1 = t_1^*$. זה קורה אם "מ" $F_k(0^n) = 0^n$. והוא מבקש תג ל- t_1^* . וזה קורה רק אם הוא מבוקש תג ל- m^* . וזה קורה אם $F_k(0^n) = 0^n$ והוא היה מצליח בהסת' לא זניחה מעל $\frac{1}{2^n}$ (אחרת היינו בונים מבחין שבודק האם $F_k(0^n) = 0^n$ והוא היה מצליח בהסת' לא זניחה מעל $\frac{1}{2}$). כמובן, קיימת פ' זניחה נ' כך ש

$$\Pr_{k \leftarrow \{0,1\}^n}(F_k(0^n) = 0^n) \leq \frac{1}{2^n} + \nu(n)$$

כמעט תמיד. כמובן, כמעט תמיד מתקיים

$$\begin{aligned} P(\text{MacForge}_{\Pi, \mathcal{A}}(n) = 1) &= P(\text{Vrfy}_k(m^*, t^*) = 1 \wedge m^* \notin \mathcal{Q}) \\ &= P(\text{Vrfy}_k(m^*, t^* = 1) \mid m^* \notin \mathcal{Q}) P(m^* \notin \mathcal{Q}) \\ &\geq 1 - \frac{1}{2^n} - \nu(n) \end{aligned}$$

■ כאשר \mathcal{Q} אוסף ההודעות ש- \mathcal{A} קיבל בעברן תגי אימות.

טענה סכמת ה-CBC-MAC אינה בטוחה להודעות מאורך קבוע (עבור t_0 נבחר באקראי ע"י Mac וערך $d = d(n)$ קבוע מראש) כאשר הערך t_1 נקבע על ידי $t_1 = t_0 \parallel t_d$ (ולא רק).

הוכחה: נוכיח עבור $d = 1$. נגדיר \mathcal{A} באופן הבא: בקש תג אימות עבור $m = 0^n$. נסמן $t = t_0 \parallel t_1$ את התשובה. החזר

$$(m^*, t^*) = (1^n, (t_0 \oplus 1^n) \parallel t_1)$$

ברור שזו יRib PPT. בנוסף, $F_k((t_0 \oplus 1^n) \oplus 1^n) = F_k(t_0) = t_1$ ולכן t^* הוא תג אימות חוקי של m^* בהסת' 1. ברור ש- \mathcal{A} לא ביקש תג אימות עבור $m = 1^n$ כי הוא ביקש רק עbor 0^n . לכן

$$P(\text{MacForge}_{\Pi, \mathcal{A}}(n) = 1) = P(\text{Vrfy}_k(m^*, t^*) \wedge m^* \notin \mathcal{Q}) = 1$$

טענה סכמת CBC-MAC אינה בטוחה להודעות מאורך קבוע כאשר ערך האימות הוא t_d (ולא רק $t = t_1 \parallel \dots \parallel t_d$).

הוכחה: נראה זאת עבור $d = 2$, קרי מחרוזות באורך n . נגדיר יריב באופו הבא :

1. בקש-tag אימות עבור $t = t_1 \parallel t_2 = m_1 \oplus 0^{2n}$ ונסמן את התשובה.

2. בקש-tag אימות עבור $t' = t'_1 \parallel t'_2 = (t_1 \oplus 1^n) \parallel 0^n$ ונסמן את התשובה.

3. החזר $(m^*, t^*) = (0^n \parallel 1^n, t_1 \parallel t'_1)$

ברור שהוא יריב PPT. כלומר $t'_1 = F_k(F_k(0^n) \oplus 1)$ ו $t_1 = F_k(0^n)$.

$t^* = t_1 \parallel t'_1 = F_k(0^n) \parallel F_k(F_k(0^n) \oplus 1^n)$

שזהו tag אימות תקין עבור $m^* = 0^n \parallel 1^n$ בהסתמך. לא ביקש ערך אימות עבור m^* כי הוא תמיד בבקשת הודעות שנגמרות באפסים. לכן,

$$P(\text{MacForge}_{\Pi, \mathcal{A}}(n) = 1) = P(\text{Vrfy}_k(m^*, t^*) \wedge m^* \notin \mathcal{Q}) = 1$$

שבוע III | הגנות מפני מתקפות ברמה הנמוכה

הרצאה

חלק A' של הרצאה (קישור)

המכנה המשותף לכל המתקפות שראינו בהרצאה הקודמת הוא שכולםאפשרות ל透וקף לשלוט במידע של התוכנה ומכוון כך לגשת לזיכרונו שהוא לא אמור להיות יכול לגשת אליו.

הגדרה נאמר כי תוכנה היא בטוחה מבחינת זיכרון אם היא :

- משתמשת רק בדרכים הסטנדריות להקצאה/גישה לזיכרונו ;
- משתמשת בפונקציות גישה לזיכרונו שיינן לאוטו הפונטרא בלבד - מחולק לבתיות מרחבית וזמןית.

בתיות מרחבית

סימון נבייט בפונטרא כשלשה (p, b, e) כאשר :

.1. p הוא הפונטרא עצמו.

.2. b הוא כתובת הבסיס של הזיכרון המשויך לפונטרא.

.3. e החסם העליון על הזיכרון המשויך לפונטרא החל מהבסיס.

מכאן, שימוש בפונטרא לגישה לזכרו הוא חוקי אם $b \leq p \leq e - \text{sizeof}(\text{typeof}(p))$. כאשר פעולה ארכיטקטית על הפונטרא יוצרת שלשה חדשה שעבורה p משתנה אבל b, e קבועים. בשימוש ב- $\&$, נקבע ע"י גודל הטיפ המקורי של מה שאחננו מצביים עליו.

דוגמה נראה את הפורמליות הנ"ל על קוד לדוגמה.

```
1 int x; // sizeof(int) = 4
2 int *y = &x // p = &x, b=&x, e=&x + 4
3 int *z = y + 1; // p = &x + 4, b=&x, e=&x + 4
4 *y = 3; // legal because &x ≤ &x ≤ (&x + 4) - 4
5 *z = 5; // illegal because &x ≤ &x < (&x + 4) - 4
```

הצלחנו לקבוע בזמן ריצה איזו גישה היא חוקית ואיזו לא.

דוגמה נבייט בדוגמה נוספת.

```
1 struct foo {
2     char buf[4];
3     int x;
4 };
5
6 struct foo f = {"cat", 5};
7 char *y = &f.buf; // p = b = &f.buf, e = &f.buf + 4 (determined by size of buf)
8 y[3] = 's'; // legal because p = &f.buf + 3 ≤ (&f.buf + 4) - 1
9 y[4] = 'y' // illegal because p = &f.buf + 4 < (&f.buf + 4) - 1
```

דוגמה נשים לב הפעם שיש לנו כמה פונטראים מקוונים.

```
1 struct foo {
2     int x;
3     int y;
4     char *pc;
5 }
```

6

```
7 struct foo *pf = malloc(...); // defines a trio (p, b, e)
8 pf->x = 5;
9 pf->y = 256;
10 pf->pc = "before"; // pc is another trio, (p', b', e')
11 pf->pc += 3; // this doesn't change b', e' but does change p' (and is legal in this case)
12 int *px = &pf->x; // this is yet another trio
```

מתקפות שנפגעות מבטיחות מרחביות

1. לא יתכו **Buffer Overflow**-ים כי כולם הם מההגדירה גישה לא חוקית לזכורן שהייתה נפסלת בבדיקה החדש.
2. ביצוע **integer overflow** גם לא ייתכן כי גבולות הגורה החוקיים של הפונטער (*e*) לא יאפשרו גישה תועה לזכרו.
3. אין יותר מתקפות **Format String** כי הן כולן גישה לזכרו שלא משוויך לפונטער של המחרוזת (מוחוץ לתוך המחרוזת).

בטיחות זמנית

יתכן שניגשים למקומות עם פונטער שנמצא בגבולות הגורה אבל לא פעיל כרגע, לדוגמה אם עשינו לו `free`. לשם כך נדרש לוודא שהפונטער תמיד פעיל - לא לפני הקצהה, אחרי שחרור או לא מאותחים.

דוגמה גישה לפונטער ששוחרר כפי שראינו בהרצאה הקודמת (stale pointer) היא לא חוקית מבחינת הבטיחות הזמנית. בנוסף, השימוש ערך בפונטער שמצויב למקומות שלא הוקצה באופן סטנדרטי (לדוגמא סטם `int *p`) גם היא לא חוקית.

הדרך הכי טובה להימנע מפגיעה זיכרון היא שימוש בשיטות בטוחות מבחינת זיכרון, שהן רוב מה שהוא לא C ודומיו. רוב השיטות הן לא רק בטוחות מבחינת זיכרון אלא גם מבחינת טיפוסים. כמובן, שככל כל אובייקט יש סוג ופעולות על אובייקטים חייבות להיות תואמות לסוג שלו. זו דרישת יותר חזקה מבטיחות זיכרון.

דוגמה להלן דוגמה לקוד שהוא בטוח מבחינת זיכרון אבל לא מבחינת סוג.

```
1 int (*comp)(char *, char *);
2 int *p = (int *) malloc(sizeof(int));
3 *p = 1;
4 // we cast an int to a function pointer - memory safe but not type safe
5 comp = (int (*)(char *, char *))p;
6 comp("hello", "bye"); // crashes
```

הסיבה ש-C ודומותיה אינן בטוחותסוגים וגם לא זיכרנו היא ביצועים - הן הרבה יותר יעילות כי הן לא דורשות כל כך הרבה בדיקות. מעבדים בשנים האחרונות מכך הינה מובנת בחומרה לפגיעויות שונות ונראה שלשם העתיד פונה.

חלק ב' של הרצאה (קישור)

אפק"פ-Sh-C היא לא בטוחה זיכרנו, אבל אפשר להקשות על ביצועו שלחן. נסה עתה למנוע מתקפות הזורת קוד. כזכור מתקפה כזו בנזיה מהזורת הקוד והרצתו באמצעות שינוי ערך ה-`key`.

קנריות מחסנית

כיפור הקנריות שומשה בעבר לזיהוי גזים רעלים במכרות פחם, ומשם בא השימוש שלה כאן - זיהוי קוד מסוכן. נוסיף לכל מסגרת פ' בזיכרנו ערך קנרית אקראי בין 99% לבין המשנה הלוקאלי הראשוני. כך, אם תוקף דורך את הערך ב-`key` (כתובות החזרה) הוא יctrיך לעבור קודם דרך הקנרית. בבדיקה האם ערך הקנרית במסגרת הפ' שווה לזה שמ为我们 לנו בצד במקום בטוח ואם לא, לא נקפוץ לכתובות החזרה ונPsiיק את התכנית.

הערה אסור שהערך של הפ' הקנרית יהיה צפוי (צריך להיות אקראי ממספריים) כי אחרת תוקף יכול לנחש אותו ובהסתה' לא זינהה לצלוה את המתקפה. לשם וויזוא הערך צריך לשמר אותו במקומות אחר בזיכרנו ולמנוע כתיבת אליו. אפשר במקומות לשמר את הערך עצמו גם לשומר ערך שמאפשר חישוב מחדש שלו, לדוגמה מפתח הפ' פסאודו-אקראית.

גם אם אি�চשו התוקף הצליח להזיר קוד, יוכל להזכיר על כל ערך במחסנית/ערימה כבלתי-ניתן להרצאה וכן רק קוד בסגמנט הטקסט ניתן היה להרצאה, וכך נגמר למורי את המתקפה.

עם זאת, יש מתקפה שטפניה זה לא מגן - Return-to-libc. מכילה הרבה פ' סטנדרטיות כמו `printf` ודומותיה והיא נטענת לעיתים קרובות אל תוך סגמנט הטקסט. אם תוקף משנה את ה-`key` כתובות בטקסט שמצויב לפ' כמו `exec` למשל עם ארגומנט זמני (כגון `code shell`) או הצלחנו לתקן בלי להזיר שום קוד.

ASLR

Address Space Layout Randomization הוא מנגן שמשנה באקרים את מיקומי הסגמנטים בזיכרון כדי למנוע גישה צפואה של תוקף לסדריות ברירת מחדל. בכלל שמנגן `asound` כזה ידרוש יכולת למפות את כל הכתובות הלוך ושוב שזה מאוד לא עיל, מה שעשושים כוים זה פשוט להסיט את כל הכתובות (או רק את אלו של הספריות הרגילות) בהיסט אקראי. במערכות 32-ビיט השיטה לא כל כך אפקטיבית אבל בשיטות בשיטים ביטים זה כבר הרבה יותר קשה לפרוץ.

פתרונו נוסף למניעת חזרה ל-libc הוא הימנע מהכללה של כל libc אלא רק מה שצריך. גם עם כל ההגנות האלה, עדין ניתן להשתמש במתקפות בסגנון ROP (סוג מתקפות מאוד מעניין ששווה העמקה).

חלק ג' של הרצאה (קישור)

הרעין הבסיסי של CFI, או Control Flow Integrity הוא התבוננות בהתנהגות של התכנית וניסיון לראות אם היא מבצעת את מה שאנו חזו מצפים שהיא עשו, ואם לא אז יכול להיות שהיא הושחתה/נפרצת.

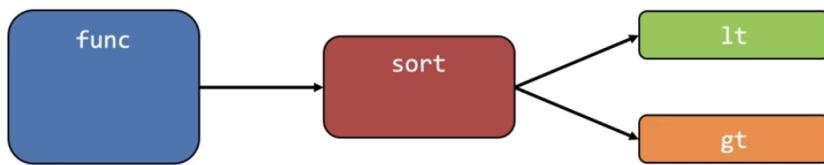
כדי להשתמש במנגנון זה, צריך להגיד מה זה התנהגות תקינה; לזהות סטיות מההתנהגות התקינה; ומונעה של מתקפה על המזווה הסטיות.

דוגמא נבין את עקרונות ה-CFI באמצעות הדוגמה הבאה: `func` קוראת לפ' שמיינית פעם בסדר עולה ופעם בסדר יורד. חשוב לשים לב ש-`sort` קוראת באופן לא ישיר ל-`order`, ככלומר שהכתובות של `order` לא נמצאות בקוד האסמלבי של `sort`, אלא לפי ערך המשתנה `order`. זאת בגיןו לקריאה ישירות, לדוגמה הקראיה של `func` ל-`sort`. בדומה, כל חזרה מפ' היא גם קראיה לא ישירה לפ' כי הכתובות שלה נמצאות בזיכרון מכיבע ולא בקוד האסמלבי.

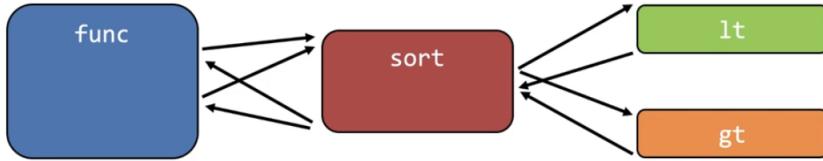
```
1 bool lt(int x, int y) {
2     return x > y;
3 }
4 bool gt(int x, int y) {
5     return x < y;
6 }
7
8 void sort(int a[], int n, fptr *order) {
9     // code of some sorting algorithm (e.g. bubble sort) that compares using order(x, y)
10 }
11
12 func(int a[], int b[], int len) {
13     sort(a, len, lt);
14     sort(b, len, gt);
15 }
```

Control Flow Graph

של התוכנית הנ"ל נראה כך Call Graph-



עתה נוסיף קשרות נפרדות לכל חזרה או קראיה לפ' ונשמר את שורות הקוד שבו מתבצעת הקראיה. באירור החצים בכוונה ממוקמים על חלקים שונים של הבלוקים כדי ליצג את השורה בה מתבצעת הקראיה/חזרה.



הסטודנטית המשקיפה תעבור על כל קשת ותבין את משמעותה, אבל לדוגמה הקשת השנייה שיצאת מ-sort היא ל-gt וממוקמת בהתאם לשורה בה היא נקראת. יש בנוסף קשת ל-*lt* כי כשיוציאו נקרא בפעם הראשונה הוא מבצע קריאות ל-*lt* באותו השורה שבה הוא עושה זאת ל-gt בהמשך.

בנויות Control Flow Graph שכזה קורא בזמן הקומpileציה.

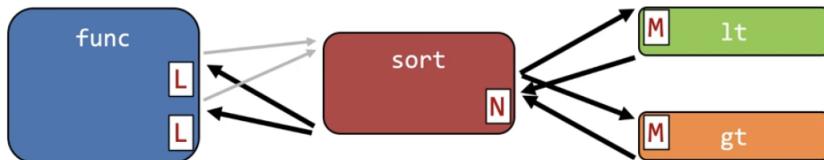
במהלך זמן היריצה, נודא שכל קפיצה היא בהתאם ל-CFG. נשים לב שצורך לעשות את זה רק על קפיצות לא ישירות כי ניתן להניע שהקוד של התכנית לא משתנה במהלך היריצה. לעומת זאת רק פועלות `jmp call` עם כתובות קפיצה לא קבועות וכן כל פעולה `.ret`. לכן אפשר להיפטר מכמה קשותות ב-G-CFG (לדוגמה הקשותות מ-`func` ל-`sort`, אבל לא האלה בכיוון ההפוך).

In-line Reference Monitor

ה-IRM אמון על שינוי הקוד של התכנית להוספת הבדיקות. נסיף ליביל לפני כל כתובות קפיצה של קריאה לא ישירה. נכניס קוד שבודק את התווית בצד השני של הקפיצה לפני הקפיצה, ואם אין התאמת נפסיק את ריצת התכנית, אחרת נקפוץ ונמשיך ברגיל.

אם ניתן לכל ה'CFG את אותו הליביל האקראי *L* אז נוכל למנוע קפיצה לפ' ספריה שאין לה ליביל, אבל זה לא מונע קפיצה לא ישירה בין פ' בתוך ה-CFG.

הפתרון הכני מדויק לגרף הנ'ל הוא הגרלת שלושה לייבלים *N*, *M*, *L* באופן הבא:



נשים לב שהlivel הוא פר קפיצה/חזרה ולכן אם קופנו מאותו המקום לשני מקומות שונים (לדוגמה ב-`sort` ל-`lt`-`gt`), נדרש שייהי למטרות הקפיצה את אותו livel (כפי שקרה כאן). בוגל ש-`sort` לא יודעת אם היא ביריצה הראשונה או השנייה שלה, נדרש שב-`sort` יהיה את אותו livel *L* בשתי החזרות מ-`sort`. מעבר לכך *N* הוא ב"ת בכל השאר ולכן ניתן לערוך אקראי שלישי שונה. כךเราจะ גם קריאה זדונית לא ישירה לפ' בתוך ה-CFG.

כדי לתזוז את ה-CFI, אפשר או להזיריק קוד עםlivel חוקי, שהוא לא אפשרי כי אנחנו מניחים שאין אפשרות להריץ קוד במחסנית/ערימה. לחולופין, אפשר לשנות את הליבלים כדי לשנות את ה-CFI לשמשו זמני, אבל זה לא אפשרי כי הקוד (ובו הליבלים) הוא-read-only ולכך בלתי ניתן לשינוי.

עם זאת, ניתן להציג את ריצת התכנית באופן שעולה בקנה אחד עם ה-CFG אבל עדין זמני.

בגלל שיטות כמו `C/C++` אין מספקות יותר מדי הבטחות בטיחות, מפתחים צריכים להשתמש בהשתמש במשמעות עצמית יחד עם בדיקה של קוד (בחמשך).

כדי לכתוב קוד בטוח, נזקוק אחר עקרונות (עיצוב שאפשר למש בדרכים שונות) וחוקים (פרקטיקות ספציפיות שמ מבוססות על עקרונות).
בפרט, נסתכל על חוקים לתכנות טוב-`B+`-`C/C++`.

אכיפת תקינות קלט

דוגמה נזכר בקוד שכבר רأינו, הבעיה בזמנו הייתה ש-`len` יכול להיות יותר גדול מאריך הבאר האמתי. השגיאה שלנו כאן היא שהנחנו שהמשתמש מענה לחוקי התכנית, אבל אין שום סיבה להניח זאת.

מכאן מגע החוק הראשון שלנו - אכיפת תקינות הקלט. ניתן בקלות לעשות סינטזיה לקלט באמצעות הכנסת השורה `strlen(str, buf);` לאחר מכן שורה מספר 7. כך לא נדפיס יותר תוים משיש בסטרינג. עם זאת חשוב בנוסח לדאוג ש-`len = min(len, strlen(buf))`. עצמו לא חורג מעבר ל-100.

```
1 int main() {
2     char buf[100];
3     char *p;
4     int i, len;
5     while (1) {
6         p = fgets(buf, sizeof(buf), stdin);
7         len = atoi(p);
8         p = fgets(buf, sizeof(buf), stdin);
9         for (i = 0; i < len; i++) {
10             if (!iscntrl(buf[i])) putchar(buf[i]);
11             else putchar('.');
12         }
13         printf("\n");
14     }
15 }
```

דוגמה כאן אנחנו מניחים אמון לא מבוסס בשימוש, כי עברו כל קלט מעבר ל-9 נקלט Overflow. בנוסף, ניתן לגשת עם אינדקס שלילי למערך שזה כਮון לא רצוי. הוספה בדיקה ש-`i` בין 0 ל-9 יפתר את הבעיה.

```
1 char digit_to_char(int i) {
2     char buf[] = "0123456789";
3     return buf[i];
```

דוגמאות אלה חלק מעיקרונו יותר כללי - תכונות רובאטי - כל רכיב בודק באופן הפסימי ביותר את מה שהוא חושב שידוע לו כי קוראים חיוניים לא בהכרח מתנהגים באותו האופן ועדיין שהתוכנה תפסיק לעברך מאשר שתפקיד קוד זמני.

שימוש בפונקציות מחרוזות בטוחות

הפ' הקלאסיות לעיבוד מחרוזות מניחות שהbabyparsים שהן מקבלות מספיק גודלים לביצוע הפעולות השונות, שזו הנחה שוגיה כמובן.

דוגמא במקום להשתמש ב-`strcpy`, מוטב להשתמש `strncpy` שמקבל גם את גודל הבאפר, וכן אם הפעולה שהוא מבקש לבצע הייתה גורמת לאוברפלאו

לא לשכח את ה-NULL Terminator

כל מחרוזת נגמרה בתו `\0`, והוא מוכנס באופן אוטומי כשמתמשים במחרוזות ליטרליים. לדוגמה "bye" הוא למעשה `e\0y\0b\0` בזיכרון. שימושם מערך למחרוזות, חשוב תמיד לזכור להקצות מספיק מקום גם ל-`\0`. שימוש בפ' מחרוזות בטוחות תופס שגיאות כאלה בזמן.

הבנת אריתמטיקה של מצביעים

ראשית, `sizeof` מחזיר את מספר הבטים שהוקצו לפוינטר מסוים ולא את מספר האובייקטים שנכנסים בהקצתה ההיא. בנוסף, הוספה של מספר לפוינטר היא בהתאם לגודל האובייקט שהוא מצביע אליו. למשל `k + sizeof(p)` לא יעביר הבה בזיכרון שהוקצת החל מ-`k`, אלא יkipoz כמה צעדים קדימה בהתאם לסוג האובייקט שהוקצתה. שימוש מוטעה בהוספה הנ'ל יכול בקלות לגרום ל-Overflow וחשוב להבין מה כל פעולה אריתמטית על פוינטרים מבצעת.

הגנה מפני פוינטרים רופפים

קריאה של `free` על פוינטר מפנה את המקום שהוא מצביע עליו למשהו אחר ומכאן שימוש שאינו בו עד שהוא מוקצת מחדש. עם זאת, פעולה השחרור לא משפיעה על ערך המצביע ולכן שימוש חוזר בקוד בפויינטר הוא מסוכן מאוד.

דוגמא בקוד הבא קשה לזהות את השגיאה אם לא מסבים תשומת לב רבה למה שהקוד עושה. סביר מאד שהמקום שיוקצתה ל-`p` זה שיוקצת ל-`q` יהיו זהים ובמקרה כזה הערך 10 ידרס את הערך ב-`q`* ואז `3 = q *` הוא לא חוקי בכלל.

```

1 int x = 5;
2 int *p = malloc(sizeof(int));
3 free(p);
4 int **q = malloc(sizeof(int *)); // allocated in the same place p points to
5 *q = &x;
6 *p = 10;
7 **q = 3; // does *10 = 3; which is obviously invalid

```

לכן עלינו לשנות את ערכו של פוינטർ משוחזר ל-NULL תמיד, ואז אם ננסה לגשת אליו שוב התוכנה תקרוס במקומות לבצע פעולות מסוכנות.

תרגול

אין הקלטה של התרגול

שאלה תהי PRF F שמעתיקה n ל- $2n$ עם מפתחות באורך n .

1. **נדיר** (ndir) H הוכיחו כי H אינה PRF. הוכיחו כי H נדיר.

2. **נדיר** (ndir) G הוכיחו כי ההתפלגות G היא בלתי ניתנת להבנה חישובית מההתפלגות האחידה כאשר $k, x \leftarrow \{0, 1\}^n$ נדוגמים אחיד ובלתי.

3. **נדיר** (ndir) W האם W בהכרח?

פתרונות

1. בונה מבchin \mathcal{D} עם גישת אורקל- $\{\cdot\}$ כאשר $\mathcal{O} \in \{F, h\}$ באופן הבא: בקש מהאורקל את הערכים

$$\alpha_0 = \beta_1 \parallel \alpha_1 = \mathcal{O}(1^n) \text{ ו-} \alpha_0 \parallel \beta_0 = \mathcal{O}(0^n)$$

ברור ש- \mathcal{D} הוא PPT כי הוא מבצע רק שתי קרייאות.

• אם $\mathcal{O} = H_k(0^n)$ אז $\alpha_0 = \beta_1 = F_k(0^n)$ והוא תמיד ייחיד.

• אם $\mathcal{O} = h$ או α_0, β_1 מתפלגים אחיד ובלתי ולכן ההסת' ש- \mathcal{D} יחויר 1 היא $\frac{1}{2^{2n}}$

לכן

$$\left| P_{k \leftarrow \{0,1\}^n} (\mathcal{D}^{H_k}(1^n) = 1) - P_{h \leftarrow \text{Func}_{n,4n}} (\mathcal{D}^h(1^n) = 1) \right| = 1 - \frac{1}{2^{2n}}$$

שזה בבירור לא זניח.

2. נניח בשלילה שקיים מבחן \mathcal{A} ופולינום (n) כך שמתקיים באופן שכיח

$$\left| P_{k,x \leftarrow \{0,1\}^n} (\mathcal{A}(G(k||x)) = 1) - P_{r \leftarrow \{0,1\}^{4n}} (\mathcal{A}(r) = 1) \right| > \frac{1}{p(n)}$$

כלומר \mathcal{A} מקבל ערך כלשהו וצריך לקבוע מייזו התפלגות הוא הגיע.

בונה מבchin \mathcal{D} בין F לפ' אקראית באמת באופן הבא: בהינתן גישת אורקל $\mathcal{O} \in \{F_k, h\}$, נגריל

$x \leftarrow \{0, 1\}^n$, נרץ את \mathcal{A} עם $\mathcal{O}(x) \parallel \mathcal{O}(\bar{x})$ ונחזר את הערך \mathcal{A} - \mathcal{A} -מחזיר.

ברור ש- \mathcal{D} הוא PPT.

• אם $\mathcal{O} = G$ אז רץ על הקלט $G(k||x) = F_k(x) \parallel F_k(\bar{x})$

• אם $\mathcal{O} = h$ כאשר $h \leftarrow \text{Func}_{n,2n}$ אז רץ על \mathcal{A} שזיהו ערך אחד ובלתי ב- x .

לכן

$$\begin{aligned} \left| \underset{k,x \leftarrow \{0,1\}^n}{P} (\mathcal{D}^{F_k}(1^n) = 1) - \underset{h \leftarrow \text{Func}_{n,2n}}{P} (\mathcal{D}^h(1^n) = 1) \right| &= \left| \underset{k,x \leftarrow \{0,1\}^n}{P} (\mathcal{A}(G(k||x)) = 1) - \underset{r \leftarrow \{0,1\}^{4n}}{P} (\mathcal{A}(r) = 1) \right| \\ &> \frac{1}{p(n)} \end{aligned}$$

באופן שכיח בסטיירה להיות $\text{PRF } F$

W אינה בהכרח PRF . נניח שקיים $\text{PRF } F$ כלשהו ונראה כי קיימת $\text{PRF } F$ שעבורה $F_k \equiv F_{\bar{k}}$. עבור W על בסיס F כזו, ברור שהיא אינה PRF כי חצי השמאלי והימני של W שוים תמיד.

תהי $\text{PRF } H$ כלשהו שמעתקה $n+1$ ל- $(n+1)2$ עם מפתחות באורך n . נגיד $\phi(k)$ כאשר $k = k_1 \dots k_n \in \{0,1\}^n$ מוגדר באופן הבא לכל $k \in \{0,1\}^n$

- אם $k_1 = 0$ אז $\phi(k) = k_2 \dots k_n$ (השמטה של הביט הראשוני).
- אם $k_1 = 1$ אז $\phi(k) = \overline{k_2 \dots k_n}$ (כלומר השמטה והיפוך).

הערה ה@studentית המשקיעה תשים לב שלמעשה $\phi(k) = k_2 \dots k_n \oplus k_1 \dots k_1$

ברור שלכל מפתח k מתקיים $\phi(\bar{k}) = \phi(k)$ ולכן $\phi(k) = \phi(\bar{k})$ ככלומר התוכונה שחיפשו מותקינית. בנוסח, $\phi(k) \in \{0,1\}^{n+1}$ מותפלג אחד אם $k \leftarrow \{0,1\}^n$.

וכich כי F ונסים. נניח בשלילה שהיא לא, כלומר שקיים מבחין \mathcal{A} PPT ופולינום p כך שבאופן שכיח מותקינים

$$\left| \underset{k \leftarrow \{0,1\}^n}{P} (\mathcal{A}^{F_k}(1^n) = 1) - \underset{f \leftarrow \text{Func}_{n,2n}}{P} (\mathcal{A}^f(1^n) = 1) \right| > \frac{1}{p(n)}$$

נבנה אלג' מבחין \mathcal{D} בין H_k לפ' אקרואית באמצעות שמקבל פרטוטר בטיחות 1^n וגישה לארקל $\mathcal{O} \in \{H_k, f\}$ כאשר $\mathcal{D} = \{H_k, f\} \leftarrow \text{Func}_{n,2n}$ מותנהג כמו \mathcal{A} כאשר כל פעם ש- \mathcal{A} מבקש ערך מהארקל נחזיר לו את הערך מהארקל \mathcal{O} , שלנו.

ברור ש- \mathcal{D} הוא PPT

לכן

$$\begin{aligned} \mathcal{D} \text{ מותנהג כמו } \mathcal{A} &= \left| \underset{k \leftarrow \{0,1\}^n}{P} (\mathcal{D}^{H_k}(1^n) = 1) - \underset{h \leftarrow n+1,2(n+1)}{P} (\mathcal{D}^h(1^n) = 1) \right| \\ \phi(k) \text{ מותפלג אחד} &= \left| \underset{k \leftarrow \{0,1\}^{n+1}}{P} (\mathcal{A}^{H_k}(1^{n+1}) = 1) - \underset{h \leftarrow \text{Func}_{n+1,2(n+1)}}{P} (\mathcal{A}^h(1^{n+1}) = 1) \right| \\ F_k \text{ הגדרת} &= \left| \underset{k \leftarrow \{0,1\}^{n+1}}{P} (\mathcal{A}^{F_k}(1^{n+1}) = 1) - \underset{f \leftarrow \text{Func}_{n+1,2(n+1)}}{P} (\mathcal{A}^f(1^n) = 1) \right| \\ &> \frac{1}{p(n+1)} \end{aligned}$$

שבוע VIII | זיהוי מתקפות בرمה נמוכה

תרגול

[קישור](#)

[שאלות](#)

```

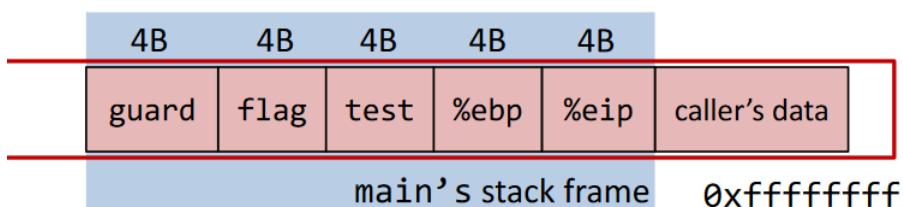
01 void main() {
02     int test = 123;
03     int flag = 1;
04     int guard = 0;
05
06     get_input();
07     flag = 1 - flag;
08     if (guard && test==123) first_goal();
09     if (flag && test==123) second_goal();
10     get_input();
11 }
```

The function `char *fgets(char *dst, int n, FILE *stream)` reads characters from `stream` and stores them starting at the location pointed to by `dst`. It stops when either `n-1` characters are read (automatically adding a terminating null byte as the `n`th character), the newline character is read (automatically replacing it with a terminating null byte), or the end-of-file is reached (automatically replacing it with a terminating null byte), whichever comes first.

```

12 void get_input(){
13     char input[20];
14
15     fgets(input, 32, stdin);
16     printf(input);
17     fgets(input, 32, stdin);
18 }
19 void first_goal(){
20     printf("Good Job!");
21 }
22 void second_goal(){
23     printf("Amazing!");
24 }
```

- תארו את תוכן המחסנית (מסגרת המחסנית) הנוצר כתוצאה מהרצת הפ' `main` כאשר הוכנסו בSHOW 05. יש להניח כי הקומpileציה מתבצעת בארכיטקטורת 32-ビט לא כולל מנגנוני אבטחה אוטומטיים או אופטימיזציות, שהארוגומנטים הניתנים לפ' מוכנסים למחסנית בסדר הפוך להגדותם ושהמשתנים המקומיים מוכנסים למחסנית לפי סדר הגדרותם (כל זה כמו שראינו בהרצאה). המחסנית מכילה את כתובות החזורה, מסגרת הפ' הקוראת ושלושת המשתנים המקומיים כבאים



- הראו כי תוקף המריץ את הפ' `main` וצופה מיידע המודפס למסך יכול לספק ערכי קלט בשורות 15 ו-17 (בכל הריצה של `get_input`). תארו את תוכנם ואורכם של הקלטים המוצעים והסבירו כיצד יגרמו להרצת `first_goal`.

אחדת או יותר של `first_goal`

חשיבותם שלם לב גם בסעיף הזה וגם בסעיף הבא שאנו קוראים ל-`get_input` קוראת ל-`fgets` פעמיים, ובכל פעם get_input מוחסנת בזיכרון הבא.

בקריאה לפ' המוחסנת נמצאת בזיכרון הבא

	20B	4B	4B	4B	4B	4B	4B	4B		
...	&fmt	input	%ebp	%eip	guard	flag	test	%ebp	%eip	caller's data
printf		get_input			main's stack frame					0xffffffff

בקראיה הראשונה ל-`get_input` ניתן לראות ב-`fgets` את הערך %ebp,%eip וכך יודפסו למסך ערכי %ebp,%eip.

בקראיה השנייה ל-`get_input` (עדין `fgets` הראשון) נספק את "A...A" כאשר ה-A-ים מספיקים כדי שהערכים שאנו שמים אחורי יפלו בדיק על המיקומים שאנו רוצים לדרוס. כתוצאה לכך כבר לא יהיה 0 וניכנס ל-`if` שלאחר מכן ואז `first_goal`.

3. הראו כי תוקף המריץ את הפ' `main` וצופה במידע המודפס למסך יכול לספק ערכי קלט בשורות 15 ו-17 (בכל הריצה של `get_input`). תארו את תוכנם ואורכם של הקלטים המוצעים והסבירו כיצד יגרמו להריצה אחת או יותר של הפ' `main`. נציג שני פתרונות.

(א) בקריאה הראשונה ל-`fgets` נספק את אותו הערך כמו בסעיף הקודם ואז יהיה לנו ערכי %ebp,%eip.

בקראיה השנייה ל-`fgets` נספק את "A...A" כאשר A הם ריפוד כנ"ל ו-s הוא ההיסט כדי ש-eip ידלג על שורה 7. כך flag נשאר 1 והפ' `second_goal` ייחד עם `first_goal`, מאותם הסיבות כבסעיף הקודם. הבעיה עם פתרון זה היא שהתוקף צריךalicahnlo להחשב את s ושהמתקפה נמנעת ע"י שימוש ב-`CFI` כי אין חץ חוזר מ-`get_input` לשורה 8 ב-`CFG`.

(ב) בקריאה הראשונה ל-`get_input` נספק כרגע את אותו הערך ואז יהיה לנו ערכי %ebp,%eip ובקראיה השנייה ל-`fgets` את אותו הערך כבסעיף 2 כך שנד:rightos את guard.

בקראיה השנייה ל-`fgets` הרשותן סתם משחו, זה לא רלוונטי ובפעם השנייה את הקלט "A...A" כאשר %ebp,%eip הם הערכים שזכרו בזמנו. כך, כשהפ' `get_input` הינו מוחסן בזירה 7 (שמנו אותם כזו מה שהוא אמר לקרים), תנסה את flag שוב ותירץ את `second_goal` (ושוב את `first_goal`). במקרה זה `CFI` לא יעזור כי שורה 7 היא אכן כתובות חוזרת אפשרית מ-`get_input` לא יעזור, אבל קנריות מוחסנת יפתרו את כל העניין כאן בנסיבות כי הן מונעות לחולוטין Buffer Overflow אל תוך כתובות החזרה.

לנוחיותכם, קוד השאלה כאן שוב כדי שלא תצטרכו לגלו הולך ושוב יותר מדי.

```

01 void main() {
02     int test = 123;
03     int flag = 1;
04     int guard = 0;
05
06     get_input();
07     flag = 1 - flag;
08     if (guard && test==123) first_goal();
09     if (flag && test==123) second_goal();
10     get_input();
11 }

```

The function `char *fgets(char *dst, int n, FILE *stream)` reads characters from `stream` and stores them starting at the location pointed to by `dst`. It stops when either `n-1` characters are read (automatically adding a terminating null byte as the `n`th character), the newline character is read (automatically replacing it with a terminating null byte), or the end-of-file is reached (automatically replacing it with a terminating null byte), whichever comes first.

```

12 void get_input(){
13     char input[20];
14
15     fgets(input, 32, stdin);
16     printf(input);
17     fgets(input, 32, stdin);
18 }
19 void first_goal(){
20     printf("Good Job!");
21 }
22 void second_goal(){
23     printf("Amazing!");
24 }

```

שבוע XII | בטיחות רשת

הרצאה

חלק א' של הרצאה ([קישור](#))

ניבור מפגיעויות בשפות ספציפיות למתקפות על הרשת ומנגנונים שונים בה.

באופן מפושט, האינטרנט הוא תקשורת בין לcptches לשרתים. הlkoo מתקשר באמצעות הדפסן ויש לו מידע לוקאלי. הרשת מתקשר באמצעות קוד של שרת ויש לו מבני נתונים.

כדי לגשת לאתר, משתמשים ב-URL-ים, לדוגמה <http://www.cs.huji.ac.il/~segev/index.html>

- `http` הוא הпрוטוקול;
- DNS ע"י `www` היא כתובת שמהרGET לכתובת IP;
- ה`path` ~segev/index.html הוא ה`path` למשאב כלשהו, במקרה זה תוכן סטטי שאנו מבקשים מהשרת. התוכן לא חייב להיות סטטי אלא לקבל פרמטרים ולהחזיר קוד ספציפי.

בכל גישה לאתר, אנחנו שולחים בקשה, לצורך עניינו או GET או POST.

- GET מבקש מידע בהינתן פרמטרים.

דוגמה הנה דוגמה לבקשת GET, נשים לב כאן כי יש לנו שדה Referer שאומר לנו מאיזה אתר הגיעו אלינו לכאן, זה מופיע רק כשהוא מבקש מידע בהינתן פרמטרים. להזכיר על קישור לאתר אחר.

<http://www.zdnet.com/worst-ddos-attack-of-all-time-hits-french-site-7000026330/>

```
GET /worst-ddos-attack-of-all-time-hits-french-site-7000026330/ HTTP/1.1
Host: www.zdnet.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Referer: http://www.reddit.com/r/security
```

- POST שלוח מידע להשמה במבני הנתונים של השרת.

דוגמה בדוגמה זו נשים לב שיש לנו רק מידע על הדפסון כמו לפנינו, אלא עכשו יש לנו גם מידע על העדכון, זה **implicitly** (כפרמטר ב-URL בשורה השנייה) והן **explicitly** (בשורה האחרונה).

```
https://piazza.com/logic/api?method=content.create&aid=hrteve7t83et

POST /logic/api?method=content.create&aid=hrteve7t83et HTTP/1.1
Host: piazza.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: https://piazza.com/class
Content-Length: 339
Cookie: piazza_session="DFwuCEFIGvEGwwHLjuCvHIGtHKECKL5%25x+x+ux%25M5%22%215%3F5%26x%26%26%7C%22%21r...
Pragma: no-cache
Cache-Control: no-cache
{"method": "content.create", "params": {"cid": "hrpng9q2ndos", "subject": "<p>Interesting.. perhaps it has to do with a change to the ...
```

כיצד מגיב השירות לבקשת כאלה? יש 4רכיבים לתגובה:

- קוד סטטוס - האם הפעולה הצליחה, ואם לא מה גורם לשגיאה.
- Headers - מטארים את המידע שהשרת החזיר.
- מידע - מה שביבנו/התגובה של השירות לבקשת שלנו.
- עוגיות - נדוע על כך בהמשך (בעיקרון המשתמש שומר בדף את הקוקי ובכל בקשה יצרף אותו כדי שהשרת ידע לו זהות את המשתמש).

דוגמה להלן דוגמה לתגובה של שרת, כאשר ניתן לראות כל רכיב שזה עתה הזכרנו.

```
HTTP/1.1 200 OK
Date: Tue, 18 Feb 2014 08:20:34 GMT
Server: Apache
Set-Cookie: session-zdnet-production=6bhqca1i0cbc1ag11sisac2p3; path=/; domain=zdnet.com
Set-Cookie: zdregion=MTI5Lj1uMTI5LjE1Mzp1czp1czpjZDjmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRm
Set-Cookie: zdregion=MTI5Lj1uMTI5LjE1Mzp1czp1czpjZDjmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRm
Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=zdnet.com
Set-Cookie: session-zdnet-production=59ob97fpinqe4bg6lde4dvq11; path=/; domain=zdnet.com
Set-Cookie: user_agent=desktop
Set-Cookie: zdnet_ad_session=f
Set-Cookie: firstpg=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
X-UA-Compatible: IE=edge,chrome=1
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 18922
Keep-Alive: timeout=70, max=146
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

כאמור אצל השרת יש מסדי נתונים, שלרוב משמשים אותם לטוח אורך והמידע בפנים יכול להיות רגיש.

כדי לגשת למסדי הנתונים, משתמשים בין היתר בשפה שנקראת (SQL) Standard Query Language (SQL).

דוגמה נניח שיש לנו את הטבלה הבאה,

Users

Name	Gender	Age	Email	Password
Dee	F	28	dee@gmail.com	j3l8g8ha
Mac	M	7	mac@gmail.com	a0u23bt
Charlie	M	32	charlie@gmail.com	0aergja
Dennis	M	28	dennis@gmail.com	1bjb9a93

נכיר ארבעה סוגים של שאלות:

- שאלת **בחירה** (SELECT): בחירת עמודה/ות בהינתן תנאים, כמו אנחנו לוקחים את כל הגילאים של משתמשים ששם הוא Dee, קלומר אייר אחד, 28.

1 **SELECT Age FROM Users Where Name= 'Dee' ;**

- שאלת **עדכון** (UPDATE): עדכו ערך בעמודה, נשים לב כאן כי כל שורה נגמר ב- ; ואחריה יכולה להיות הערה שמתעלמים ממנה לחלווטין.

1 **UPDATE Users SET Email= 'new@email.com' WHERE Age=32; -- this is a comment**

- שאלת **הכנסת** (INSERT): הכנסת אייר חדש לעמודה.

1 **INSERT INTO Users Values('Frank' , 'M' , 57, ...);**

- שאלת **מחיקה** (DROP): מחיקה עמודה/טבלה.

1 **DROP TABLE Users**

דוגמה נשתמש ב-SQL כדי לבצע login. נניח שקיבלו מהמשתמש שם משתמש וסיסמה. נבצע (ב-PHP),

```
$result = mysql_query("select * from Users
where(name='$user' and password='$pass');");
```

כאשר כוכבית מסמלת את כל השדות ולא אחד ספציפי. אם חזרו תוצאות (יותר מ-0), נזכיר שהמשתמש נכנס בהצלחה.

כיצד נצל את הקוד הזה? נניח שאנו שוכרים ורוצים להיכנס למערכת באופן לא לגיטימי. אם נתן את שם המשתמש

```
frank' OR 1 = 1); --
```

וسيסמה כלשוי לכאורה אנחנו לא אמורים להצליח להיכנס, אבל מה שקרה זה שהשרת יבצע את

```
$result = mysql_query("select * from Users  
where(name='frank' OR 1=1); -- ' and password='whocares');";
```

ואז מה שאנו עושים כאן בעצם זה מוחפשים את כל המשתמשים ששם הוא 1 או שעורם מתקיים $1=1$ שהוא כמובן מתקיים לכל משתמש ולכן יוחזרו תוצאות ונצליח להיכנס למערכת באופן לא לגיטימי.

להיכנס למערכת זה חלבי מאד, שליחת שם המשתמש

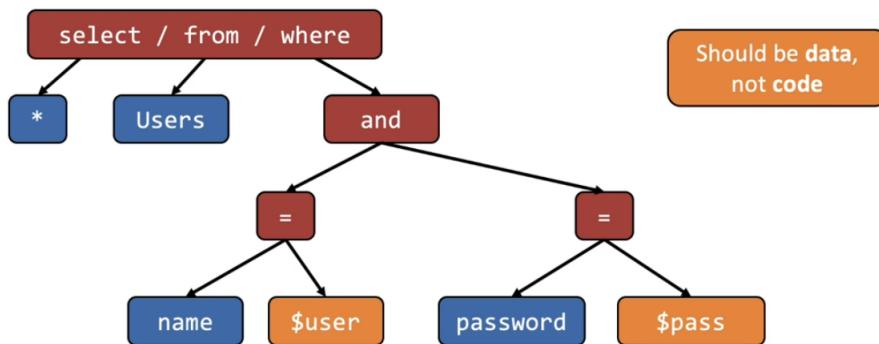
```
frank' OR 1 = 1); DROP TABLES Users; --
```

לא רק שיצליח להיכנס למערכת אלא גם ימחק את כל הטבלה אחריו - זה נזק ממשי.

אלו מתקפות מאד נפוצות וудין מהוות אחוז לא זנית ממתקפות ברשות.

אז מה הבעיה שגורמת לפגיעות הזו? אנחנו מבדיקים את הפרמטרים של המשתמש באופן ישיר וכך כאשר הסטרינג מכיל ערבות של נתונים עם קוד, אנחנו חשופים את עצמנו לפגיעות.

ניתן להסתכל על השאלה בתווך העץ הבא,



כאשר כאן ברורה מאד ההפרדה בין קוד למידע. ברגע שאין הפרדה העץ הזה משתנה בהתאם להחמת המשתמש.

מנועת SQLi

פתרון אחד, מאוד לא יעיל שאסור אף פעם להשתמש בו כי לא ככה מגנים על האינטרנט, הוא סנטיציה של התוכן באמצעות רישימה שחורה של תווים אסורים לדוגמה ; -. אבל חישרון אחד שהוא שווה לעיתים פולש אפשרויות לגיטימיות אבל במקרה זה משחק של חתול ועכבר כי תמיד ימצאו דרך לתות אינפוט מטורף שעוקף את הכל. לחłówין אפשר לעשות אסקיפינג באמצעות \ לכל דבר חשוד. גם זו שיטה בעיינית.

fail safe defaults פתרון נוסף הוא רשימה לבנה שבה ניתן קלט במקום לתקן אותו היא פשוט פסולת אותו. פתרון זה מtabסס על עקרון ה-
שוגביל פונקציונאליות לטובות בטיחות.

הצהרות מוכנות מראש

במקרים פשוטים להדביק את המידע אל תוכן הבקשה, נשמר להם מקום ובכל קרייה נשים אותם בטיקסט בלבד בלי התיחס אליהם כך. דוגמה לכך הבא,

```
$statement = $db->prepare("select * from Users  
where(name=? and password=?);");  
  
$statement->bind_param("ss", $user, $pass);  
$statement->execute();
```

אנחנו מציבים את שם המשתמש והסיסמה במקומות המידועים להם וגם מכרים אותם על סוג המשתנה שאנו מכנים, במקרה הזה פעמים סטרינג (s).

גם כאן יש מתקפות שעובדות אבל הן הרבה יותר נדירות וקשה למימוש.

חלק ג' של ההרצאה ([קישור](#))

ש่น של HTTP הוא לרוב מהצורה : הלקוח מתחבר לשרת, מבצע בקשה, השרת מגיב, הלקוח מבצע עוד בקשה וחזור חלילה על שהלכה מתנתך. עם זאת, HTTP בפני עצמו הוא חסר זיכרון לגמרי, לכן נדרש להזיכר לשרת מי אנחנו ומה אנחנו רוצחים בכל בקשה. אנחנו רואים את זה בכל מקום, דוגמה בסל קניות שזכור בין מעברים באתר מה הלקוח רוצה לקנות.

גם בטוווח הקצר, תוכנות רשות שמורות על "מצב" שמכיל תוכאות ביןיהם של השרת שהלכה שומר ובכל בקשה, הלקוח מעביר את המצב הזה לשרת. ה"מצב" ממומש באמצעות שדות חבויים וקוביז.

דוגמה נוספת שגלונו לכთובת socks.com/pay.php ולחצנו על כפתור ההזמנה עבור גרבאים כלשהם. משם עברנו למסך socks.com/order.php ולחצנו על השלמת ההזמנה.

איך זה קרה בפועל? הנה הקוד שהוגש למשתמש בדף התשלום (משמאל) וקוד השרת שմפרסר את הבקשה (מימין)

```
<html>  
<head> <title>Pay</title> </head>  
<body>  
  
<form action="submit_order" method="GET">  
The total cost is $5.50. Confirm order?  
<input type="hidden" name="price" value="5.50">  
<input type="submit" name="pay" value="yes">  
<input type="submit" name="pay" value="no">  
  
</body>  
</html>
```

```
if(pay == yes && price != NULL)  
{  
    bill_creditcard(price);  
    deliver_socks();  
}  
else  
    display_transaction_cancelled_page();
```

וכפי שניתן לראות השרת גורם להלקוח לשולח לו שוב את ה"מצב" שלו ואז השתמש בו שוב כדי לשרת בקשה נוספת.

עכשו נחשוב כתוקפים - איך אפשר לתקן דבר זהה? נוכל פשוט לשנות את header המחיר לערך מאוד נמוך ולשלם הרבה פחות מאשרנו אמורים.

כדי לפחות את הבעיה הנ"ל הומצאו מזהה שני שעובדים באופן הבא. השירות שומר את מצב הלוקה במקומות אמין והлокוח אחראי לכל השאר. השירות שומר את המצב, שולח מזהה שני לlokoch והлокוח נותן לשרת את המזהה הזה בכל בקשה נוספת. חשוב כМОון לשמר SHA-Session Identifiers (Session Identifiers) יהיו בתלי ניתנים לניחוש כדי למנוע גישה לא חוקית ל"מצב", לדוגמה String פסאודו-אקראי. עתה במקומות לחתם header של price, נעביר header של sid עם הערך שקיבלו מהשרת.

יש עדין בעיות עם המנגנון הזה, בין היתר קשה לתאם את ההעbara שלו בין הרבה דפים וצריך להתחילה כל פעם מהתחילה כל פעם שהדף נסגר (כי אז נגמר השם בהכרח).

כאן השתמש ב-Cookies שיזהו את הלוקה בכל תחילת טווח שהוא ישמור בדף וכן לא נדרש להתעסק עם זה כל כך הרבה.

דוגמא כל cookie מכיל מפתח וערך יוכל לשמר גם ערכים נוספים, רואו תגובה לתגובה השירות לבקשת כלשי, כאשר ה-Header של Set-Cookie מבקש מהлокוח לשמר את הקוקיז האלה עצמו.

```
HTTP/1.1 200 OK
Date: Tue, 18 Feb 2014 08:20:34 GMT
Server: Apache
Set-Cookie: session-zdnet-production=6bhqca110cbc1ag11sisac2p3; path=/; domain=zdnet.com
Set-Cookie: zdregion=MT15J1uMT15J1E1Mzpl1cp1cspjZDjmNW5Y7dkODU1N2Q2YzM5NGU3M2Y1ZTRm
Set-Cookie: zdregion=MT15J1uMT15J1E1Mzpl1cp1cspjZDjmNW5Y7dkODU1N2Q2YzM5NGU3M2Y1ZTRm
Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=zdnet.com
Set-Cookie: session-zdnet-production=590097f1pnqe4bg0ide4qvq11; path=/; domain=zdnet.com
Set-Cookie: user_agent=desktop
Set-Cookie: zdnet_ad_session=f
Set-Cookie: firstpg=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
X-UA-Compatible: IE=edge,chrome=1
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 18922
Keep-Alive: timeout=70, max=146
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

נשים לב כי בין היתר הועברה אפשרות main-domain שמכילה את הסיווג של כל האתרים שאלייהם צריך להעביר את הקוקיז, ולא אף אתר אחר.

בכל בקשה נוספת של הלוקות, הוא ישלח לו ב-Header Cookie את כל הקוקיז שעונאים על הדרישות של הבקשת (בתווך עדין, לאתר הנוכחי וכו').

שימושים לעוגיות

- זיהוי משתמש לאחר זיהוי ב-login בלא צורך ל-login חזרה.
- התאמת אישית של האתר למשתמש בלי מגע של המשתמש (צבע גופן מועדף וכו').
- מעקב אחר משתמש לשם פרטום מותאם. הרוב השימוש בעוגיות לאתר זה בסדר ולא חמורות. אבל אם יש עוגיות בין אתרים מעקב אחר משתמש (Cross Site Cookies) וזה יכול ליצור בעיה.

דוגמא אם אנחנו מעוניינים לרכוש ביטוח שלא דורש הצהרה רפואי ולפנינו במקרה גלשנו באתר אחר וקראנו על מחלה שיש/אין לנו, אם רשות פרסום/מעקב מצילהה עלולות על הקשר הזה וליקיר את הביטוח כי היא חושבת שיש לנו את המחלת הזה, זו הפרה בוטה של הפרטויות שלנו.

דוגמה רשות פרסום יכולה לעקוב גם בלי המנגנון המתווכם זהה. אם במהלך גישה באתר כלשהו מופיעה הודעה של רשות פרסום כלשהי, הרשות של רשות פרסום יכול לzechות את ה-referrer וכן יודע שאלשנו באתר אז לשמר קוקי שאור את כל האתרים שבו משתמש היה. כך למעשה אנחנו עוזרים לרשות פרסום לעקוב אחרינו בעל כורנו. בימינו אפשר לחסום קוקיז מצד שלישי כמו אלה.

Session Hijacking

לאחר ביצוע הזדמנות מוצלחת מול שרת, המשמש מקבל Session Cookie שבאמצעותו יוכל כניסה לאתר בפעם הבאה בלי לבצע login כל פעם מחדש. הקוקי יכול להיות מועבר או דרך ארגומנט ב-URL או דרך Header בבקשתו ו מבחינותיו אין הבדל. באמצעות גישה לקוקי אפשר לבצע פעולות רגישות באמצעות הגישה המאומנת לאתר. כיצד ניתן וניגש לקוקי?

- לפרוץ למחשב של הלוק או לשרת.
- לנחש את הקוקי (קשה אם הקוקי ממושך היטב, אקרראית).
- להסניף את התקשרות של המחשב ואם היא לא מאובטחת לגשת לקוקי.
- להרעל את ה-DNS cache - לגרום למשתמש לחשב שהוא ניגש לפיסבוק אפלו שהוא לא ואז יועבר הקוקי לאתר הזרוני.

Cross Site Request Forgery

אם התוקף הצליח לגרום למשתמש להיכנס ל-URL שמעביר כספ' מחשבון בנק אחד לאחר אחראי שהוא חקר מספיק את מנוגני הבנק, הוא הצליח לבצע מתקפה. אבל למה שהשתמש נכנס ל קישור שכזה?

נניח שהמשתמש הגיע לאתר attacker.com ושם יש תמונה שכתובתה המוצחרת היא

`http://bank.com/tranfer.cgi?amt=9999&to=attacker`

כלומר אותה כתובות הזדונית האמורה. הדף ניגש לכתובת זו באופן אוטומטי כדי להציג את התמונה וימצא את עצמו מעביר כספ' בטעות לתוקף כי הדף שולח את ה-session Cookie כי כל המהלך כאן חוקי לממרי.

מתקפה זו מוצלחת רק על משתמשים עם חשבון בשירות פגיע; המטרה של התוקף היא לגרום למשתמש לבצע בקשות חוקיות שהוא לא רוצה לבצע; והוא עושה זאת באמצעות גרים להציגו על כפתור שהוא לא רוצה (או הורדת תמונה זדונית וכו').

כיצד נן מפני מתקפה זו?

- באמצעות שדה ה-referrer, אם אנחנו רואים שהлок הפנה אותנו לא לgitימי פשוט לא לבצע את הבקשתו שלו. זו לא שיטה טובה כי זהו שדה אופציונלי ולא אמין.
- נכלול ב-URL סוד כלשהו - הקשר בין הקישור לקוקי. אפשר להשתמש בשדה סודי או לקודד את הסוד בתוך ה-URL. זה יכול להיות id-session או סטם מחרוזת אקראית שהוגירה על ידי האתר הלגייטימי שהעביר את המשתמש בבקשת הנוכחית.

במקרים מסוימים משאב סטטי או משאב שמוסג שופית בהינתן פרמטרים, יוכל להשתמש ב-Javascript כדי לבצע חישובים דינמיים אצל הלוקה במקום לעשות הכל בשפת JS. סימן את המעבר מדור ראשון לדור השני של האינטרנט.

הסקריפטים רצים על הלוקה ויכולים לעשות כל מיני דברים כמו לעקוב אחר אירועים, ליצור קוקיז ועוד.

חשוב לוודא שסקריפט של אתר אחד לא יכול לגשת לCookie של אתר אחר כי אז תוקף שהאתר שלו פתוח אצל הלוקה יוכל לגשת לבנק של המשתמש באופן זמני. דפדףים הם אלה שחייבים לוודא שסקריפט נוגע רק במקרה ששייך לו ולא שום דבר אחר.

הבדיקה הזאת מבוצעת על ידי הדפדף באמצעות Same Origin Policy שימושית כל אלמנט (Cookie, תמונה וכו') למקורו בלבד ומאפשרת לכל אלמנט לגשת רק לדברים עם מקור זהה שלו.

Cross Site Scripting

המטרה של מתקפה זו היא לעקוף את SOP, לומר אחרים לדפדף לחשב שמקורו כלשהו (זמני) הוא למעשה מקור אחר (רגעיש). מתקפות XSS גורמות לשרת הרגיש לשולח קוד זמני למשתמש וכך הדפדף יחשב שהוא מגיע מאותו מקור פגיעה כי זה באמת המצביע.

- XSS - התוקף משאיר סקריפט על השרת של הבנק. השרת בטעות יגיש ללקוח קורבן את הקוד והוא יירץ אותו מאותו מקור של הבנק. הסקריפט הזמני שנשпал אצל השרת יכול לעשות כל דבר, החל מלהגנוב Cookie ועד לביצוע פעולות רגישות. נדמה שקשה מאוד לתוקף לשתול קוד אצל השרת אבל זה בכלל לא המצביע.

דוגמה המתקפה של Samy MySpace. העלה לדף שלו סקריפט. כל משתמש שנכנס לדף שלו הריץ את הסקריפט בעל כורחו. הסקריפט הוסיף את Samy לרשימת החברים של המשתמש, שינה את הדף של המשתמש להכיל תעמולת פרו-Samy-ית והדיביק את הדף של הקורבן כך שכל מי שיבקר בדף שלו יהיה קורבן לאותו תחילה.

מספר החברים של Samy הגיע מ-73 ל-5 מיליון בעשרות שעות והוא הצליח לגרום ל-MySpace ליפול בסוף"ש באוקטובר 2005.

הערה גם בשנים האחרונות היו מתקפות XSS בטוויטר שהגרמו לזה שהמשתמש היה עוזה ליקיך ומשתף את הפוסט ומדביק עוד משתמשים.

- XSS - התוקף יגרום למשתמש לשולח את הסקריפט (פרמטר בבקשת דוגמה) לשרת של הבנק והשרת יחזיר בהמשך את הסקריפט והוא יירוץ על המחשב שלו עם מקור חומי (רגעיש).

דוגמה נניח שהמשתמש ביקר באתר זמני ולחץ על קישור לאתר הבנק שמכיל פרמטר עם סקריפט (כمحזרות), דוגמה שם נמען ההעברה.

השרת יחזיר דף שמכיל את התוכן הזה מסיבה כלשהי (דוגמה כדי להודיע שלא קיים מען כזה) אבל במקום להסתכל על זה כמוידע, הסקריפט יגיע כקוד JS להרצה והמשתמש ייריץ את הסקריפט הזמני.

הסיבה שהמתקפה הזו עובדת היא שהשרת לא מצליח לוזהות שהתגובה שלו מכילה סקריפט מוטמע בתוכה.

XSS מפני הגנות

דרך אחת להציגן מפני מתקפות כליה היא לוודא שהשרת לא מגיש דף עם סקריפט שהוא לא התכוון להציג. הבעייה היא שיש הרבה מאוד דרכים להכניס סקריפט וזה כמו הרבה דברים משחק של חתול ועכבר.

המצב אפליו יותר חמוץ כי דפדףים רבים מנסים לעזור ל-HTML שבור לעבוד אז למעשה יוצרים XSS שלא היו קיימים אחרת. פתרון נוסף הוא Whitelisting שדוחה כל דבר שנראה חשוד בלי ניסיון לתקן אותו, זה כמובן יותר מוצלח אבל גם לא פותר את כל הבעייה. חשוב להבדיל בין XSS ל-CSRF. CSRF מנצל את העבודה שבהשתמש מאמין בכל מה שהשרת נותן לו ואז גורם לשרת לשלוח סקריפט זמני שלא השרת ולא הלקוח רוצחים. CSRF לעומת זאת מנצל את העבודה שהרת מאמין לכל מה שהלקוח נותן לו.

תרגול

[קישור](#)

שאלה נתון הקוד הבא,

```

01 typedef void (*fptr)(char *);
02 void main() {
03     char name[12];
04     fptr f = forward_to_printf;
05     fptr g = boo;
06
07     printf("Enter Name:");
08     fgets(name, 12, stdin);
09     forward_to_printf(name);
10
11     printf("Enter Password:");
12     forward_to_fgets();
13
14     f("Done");
15 }
```

```

16 void forward_to_printf(char *data) {
17     printf(data);
18 }
19 void forward_to_fgets() {
20     char password[4];
21     char input[40];
22     fgets(input, 40, stdin);
23     memcpy(password, input, 8);
24 }
25 void boo(char *data) {
26     printf("Boo!");
27 }
```

main קוראת שם באורך 12 בתים, מדפסה אותו עם printf (באופן ישיר, לא עם פורמט), קוראת סיסמה באמצעות forward_to_fgets. וקוראת ל-f לסיום, שהיא מאותחלת forward_to_printf.

forward_to_fgets קוראת 40 בתים אל תוכן input ואז מעתיקה 8 בתים ל-password שהוא באפר באורך 4. תארו את תוכן המחשבנית בשורה 6. בהנחה שאין אופטימיזציות ומנגנוני אבטחה ושארגוני מוכנסים למחסנית בסדרך. הפוך לסדר הגדרותם ולוקאליים מוכנסים לפי סדר הגדרותם.

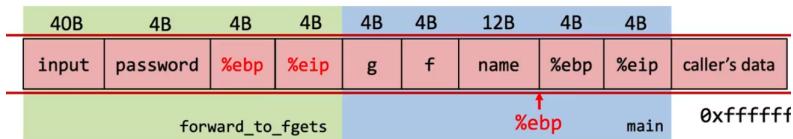


2. הראו כי תוכף המרץ את main יכול לספק קלט בשורה 8 שיאפשר לחשב את הכתובת הווירטואלית של f. נשים לב שכשאנחנו קוראים ל-forward_to_printf, אנחנו עושים את זה באופן ישיר. אם יש לנו את הכתובת הווירטואלית של %ebp (כפי שערכו בפ' main), בקהלות נוכל לחשב את f (פושט 16 – %ebp) ולכין נספק את הקלט %ebp וזה ידפיס לנו את הכתובת של %ebp (השמור במסגרת של main, לא של forward_to_printf).

3. הראו כי תוקף יכול לספק שני ערכי קלט בשורה 8 ושורה 22 כך שהתכנית תגעה לשורה 14 ותריץ את קוד הפ' 000 במקום

.forward_to_printf

בקריאה לפ' forward_to_fgets המחסנית נראית כך,



לכן בשורה 8 נספק את הקלט `k = %ebp - 4` ונקבל את ערכו של `%ebp` (של `main`) ונשמר את `4 .loc = %ebp`.

בשורה 22 נספק את הקלט `AAAAAlloc` וכן `%ebp` יוזו בארבעה בתים אחורה כך שיכងש בכתובת היחסית `-l %ebp` למה שנראה

לנו כמו `f`, למעשה אנו ניגשים ל-`g`, `g` הלא היא `foo`.

4. האם שימוש ב-canary values יעזר למנוע את המתקפה?

כו, כי לפני שנדروس את `%ebp` נצטרך לעبور קודם דרך ערך קנריית ולדרוס גם אותו.

האם שימוש ב-ASLR יעוזר?

לא, הוא לא רלוונטי כי לא חישבנו כתובות קבועה עבר או `f`, `g` בכל מקום תרוץ עדיין.

האם שימוש ב-CFI יעוזר?

כן, `000` לא מחובר בקשת לאף פ' ולא קיבל אף תווית ולכן ברור שקריאה לפ' `func` דבר לא חוקי שהיה נוצר בידי CFI.

שאלה נתון הקוד הבא,

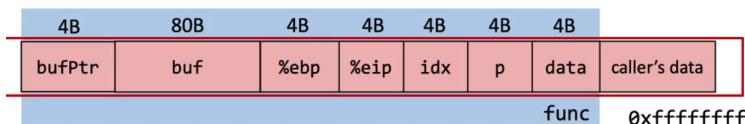
```

01 typedef struct bar {
02     int a;
03     int b;
04 } bar;
05
06 int func(int idx, char *p, char *data) {
07     bar buf[10];
08     bar *bufPtr;
09
10     if (idx < 0 || idx > 10) {
11         strncpy((char *)buf, p, sizeof(bar) * 11);
12     }
13     else {
14         for (bufPtr = buf; bufPtr <= buf + idx; bufPtr++) {
15             strncpy((char *)bufPtr, data, sizeof(bar));
16         }
17     }
18 }
```

הקוד `func` מעתיקת לכל היוטר 11 סטרוקטים מסוג `bar` אל תוך `buf` אם מתקיים התנאי הראשוני ואחרת היא מעתיקת סטרוקט

אחד כל פעם, במשך `idx + 1` איטרציות (שםו לב `for (bufPtr = buf; bufPtr <= buf + idx; bufPtr++)` במקום `<->`).

1. תראו את תוכן המחסנית בשורה מס' 9 תחת אותם הנקודות כמו תמיד.



2. נתון כי פ' המערך sleep נמצא בכתובת 0xaabbccdd. הראו כי יש שתי חולשות שונות בפ' func sleep שמאפשרות לתוקף להריץ את sleep. ככלומר מצאו שתי שלשות (idx, p, data) בклט ל-`func` שגורמות להרצאת sleep.

(א) נדרוס את קיומם איכשהו בכניסה ל-if הראשון. נשים לב כי אנחנו מעתיקים 11 כפול 8 בתים של זיכרוון, ככלומר 88 בתים, שזה בדיק מספיק כדי לדرس את קיומו. לנ' ניתן את הקלט `p` שמצוין למחרוזת AAA...AAA0xaabbccdd AA...AA כשייש לנו 84 פעמים `A`, `idx` יהיה 11 (או מספר שלילי וכיו') ו-`data` כלשהו.

התוכנה תיכנס לשורה 11 ו-`strcpy` תדרס את כל `buf`, את `ebp` %ebp ו גם את קיומו עם הפוינטור לפ' sleep. כשהפ' תסימן את ריצתה היא תחזיר לכתובת החזרה שלה, הלא היא sleep.

(ב) נדרוס את קיומו בנסיבות שורה 15. אנחנו מעתיקים 8 בתים בכל איטרציה ועושים זאת 1 + `idx` פעמים لكن עבר הקלט `p` כלשהו ודאית שהוא AAAA0xaabbccdd, לא נכנס לשורה 10 אלא 14, נרים את הלולאה 11 איטרציות ובאייטרציה الأخيرة נדרס את `ebp` %ebp וכן קיומו (בכתובת של sleep). כשהפ' תחזיר היא תמצא עצמה בסלープ פתואם.

3. האם שימוש בערכי קניית היה מונע את המתקפה?

כן, כי שתי המתקפות היו דומות גם כו את ערך הקניית ואז התוכנה הייתה מסרבת לkapo'ן בשל אי התאמה בערכיהם.

האם שימוש ב-ASLR יעזור?

כן, כי אנחנו מניחים בשתי המתקפות שהמיקום של sleep הוא קבוע וידוע לנו, אף על פי שהוא לא.

האם שימוש ב-CFI יעזור?

כן, כי התוכנה לא משתמשת בשום שלב בסלープ ולכן לא תהיה תווית תואמת עבר פ' זו ושתי המתקפות ימנעו.

שבוע X | ניתוח קוד סטטי

הרצאה

להלן א' של ההרצאה (קישור)

ניתוח קוד סטטי הוא שיטה אחת לבדיקת קוד באופן אוטומטי להגנה מפני מפגעי אבטחה. הוא מאפשר בנוסח לבדוק האם מערכת שלא אנחנו כתבנו היא בטוחה.

ראשית علينا לענות על השאלה - מה היא תכנית בטוחה? בעיקריו זו תכנית שיכולה להתנהג רק בדרכים שהמתכונת התכוון שהיא תתנהג, אבל אז צריך לענות על השאלה - מה היא הריצה הנכונה של תוכנה?

גישות לבדיקת נכונות

1. טסティング - הרצת התכנית עם קלטים שונים (שנבחרים תוקן הבנה של התכנית).

יתרונות אם התכנית כן נफלת, אפשר בקלה לברר מה השتبש ולהתקן זאת.

חסרונות יקר על הרבה קלטים, קשה לבדוק כל מסלול אפשרי של הקוד ולא מבטיח שאין קלט אחד שפיספסנו שהתוקף יכול להשتمש בו (מספיק אחד שפיספסנו והכל מיותר).

2. אודיטינג - לשכנע מישחו שהקוד שלנו הוא נכון.

יתרונות בני אדם יכולים להבין קונספטים מורכבים יותר ממכונה.

חסרונות גם יקר, קשה לכטוט כל אפשרות ולא מבטיח שלא פיספסנו שום דבר.

ניתוח קוד סטטי מנתחת את הקוד בלי להריץ אותו.

יתרונות כיסוי הרבה יותר רחב של מסלולי הריצה (לפעמים אפילו כולם, מה שכן מבטיח שהקוד באמת בטוח) ואפילו כיסוי על ספריות.

חסרונות מנתח רק תוכנות די מוגבלות, יכול ללקח הרבה זמן ויכול לפספס מפגעי אבטחה או להתריע אזעקות שונות

אנגליזה סטטיטית של קוד מאפשרת שילילה של מחלקות שלמות של פרצות אבטחה (כל הבאפר-אובפרפלואים וכו') ; אפשרותה למתכנתים להתרכו בהגנה מפני מפגעים יותר עמוקים ולא להתעסך בזוטות ; ומשתמשים בה יותר וייתר במשק.

בנוסף, אngleיזה סטטיטית גורמת למפתחים להימנע משגיאות מלכתחילה וגורמת להם לחשב על הנחות היסוד שלהם יותר לעומק.

אי אפשר להשיג אngleיזה סטטיטית מושלמת כי פתרת בעיה זו שköלה לפתרון בעיית העצירה.

עם זאת, אפשר לעשות אngleיזה סטטיטית שימושית שעלולה לא להסתיים אף פעם, להתריע התראות שווא ולפספס מפגעים.

כלומר אngleיזה סטטיטית טובה היא טריידוף בין דיקט, יכולה לנתח תוכנות גדולות ויכולת להבין מה המפגעים שהangleיזה מדוחת עליהם.

הערה גם סגנון תכנות הוא חשוב, כאשר קוד שכתוב גרוע מדי יכול פשוט להיות מסורב בידי הכלים מטעמי אבטחה. בנוסף, התראות שווא לרוב נובעות מסיבוך קוד מיותר.

חלק ב' של הרצאה ([קישור](#))

המטרה של Flow Analysis היא לעקוב אחר איך ערכיהם עוברים מקום למקום בזיכרון. הסיבה שזה עוזר לנו היא שהרבה מתקפות מtabבשות על מתן אימון בקלט לא חוקי. לכן קלט מהמשתמש ייחשב מעתה חשוד (tainted) ומידע פנימי שלנו ייחס לא חשוד (untainted).

דוגמה F כמו strcpy מניחות שהקלט שלהם הוא untainted. בדומה, שאלות SQL מניחות שהקלט הוא לא חשוד.

הרעינו המרכזי ב-FA הוא שנסמך בקוד כל משתנה אם הוא חשוד או לא חשוד. הסימון יוגדר באופן הבא : tainted אם יכול להיות שהתוכף שלו בו, ו-untainted אם התוכף בטוח לא שולט בו.

דוגמה בקוד הבא printf חייב להניח שהקלט שלו לא חשוד, אבל fgets מבונן מחזיר לנו חשוד מהמשתמש.

```
int printf(untainted char *fmt,...);  
tainted char *fgets(...);
```

אם אנחנו מוצאים את עצמנו משתמשים במשתנה tainted במקום שאמור להיות untainted - הרי לנו מפגע אבטחה!

המטרה של FA היא לענות על השאלה האם קיים קלט שגורם לזרימה של ערכים חדשים לערכים לא חדשים - זרימה כזו נקראת זרימה לא חוקית.

דוגמה הקוד הבא הוא זרימה חוקית, כי לא אכפת לנו אם ערך לא חשוב מועבר לפ' שמקבלת חשודים,

```
void f(tainted int);
untainted int a = ...;
f(a);
```

עם זאת, הקוד הבא אינו זרימה חוקית, כי אנחנו מעבירים ערך חשוב אל תוך פ' שמניחה שהקלט שלו אינו חשוב,

```
void g(untainted int);
tainted int b = ...;
g(b);
```

הערה אם נסתכל על זה בתוור סדר, כאשר untainted קטן ממש מ-tainted, וכל מעבר של ערך ממש למקום בו זכרו הוא איבר בסדר, אז זרימה לא חוקית היא סדרה לא מונוטונית עולה.

כיצד פרקטית נבעצע את האנגליזה?

1. ניתן לכל משתנה/ערך החזרה/ארגוומנט .qualifier

2. לכל statement בתכנית, ניצור תנאי מהצורה $q_1 \leq q_2$ (עם אותו הסדר הנ"ל).

דוגמה עבור $y = x$ האילוץ החדש הוא $q_x, q_y \leq q_x$ כאשר q_x -qualifier של y , q_x בהתאם.

3. נפתרור את בעיית ההשמה תחת כל האילוצים.

4. אם אין פתרון, יכול להיות שיש זרימה לא חוקית.

דוגמה ננתח את הקוד הבא,

```
int printf(untainted char *fmt, ...);
tainted char *fgets(...);

char *name = fgets(..., network_fd);
char *x = name;
printf(x);
```

ניצור x - α, β -ים \leq בהתאמאה.

האילוץ הראשון הוא α (ערך ההחזיר של fgets הוא חשוב).

האילוץ השני הוא $\alpha \leq \beta$.

האילוץ השלישי הוא $\beta \leq \text{untainted}$.

כלומר קיבלנו את המשוואה $\text{untainted} < \text{tainted} \leq \alpha \leq \beta \leq \text{untainted}$ ולכן יש פה זרימה לא חוקית וזה אינה איזעקט שווה!

ازעקות שווא בעיקרונו הן לא דבר רע כי אפשר לברר שהן אכן כאלה ופשטות להתעלם מוחן. עם זאת, יותר מדי כאלה מעקבות את הרציצה ומבלבלת אותנו. נברר כיצד למגר אוטו.

דוגמה נניח שיש לנו את הקוד הבא

```
α char *name = fgets(..., network_fd);
β char *x;
if (...) x = name;
else x = "hello!";
printf(x);
```

ננתן את הקוד. יש לנו כבר `char`-ים α, β . האילוקטים המתקבלים משוררות הקוד הם :

- $\text{.tainted} \leq \alpha$
- $\beta \leq \alpha$ יכול להיות שאחנו לא נכנסים ל-if, אבל אנחנו מסתכלים על כל המקרים ובפרט על המקרה כי גרווע וכן נסיף את האילוץ תמיד).
- (`untainted` (כלול גם את זה כדי לא לפספס אף מקרה, אפ"פ שהוא מקרה פרטי של הנ'). $\text{untainted} \leq \beta$)
- $\beta \leq \text{untainted}$

תחת אילוץ זה, יש עדין זרימה לא חוקית, `tainted` $\leq \alpha \leq \beta \leq \text{untainted}$ והתגבירנו על ה-conditional ולא הצלחנו לתת לו לבלב אותנו! העניין כאן הוא שלא לקחנו סיכוןים.

נשים לב שגם אם שתי השורות ב-if-else היו רצות תמיד (פשות בili), היינו מקבלים את אותה ההתרעה על זרימה לא חוקית כמקודם. עם זאת, הפעם, המחרוזת `hello` הלא חשודה דורשת את הערך החשוד ב-`x` וכן זו התרעת שווא - הקוד בסדר גמור.

הערה הסיבה שהייתה פה אזעקה שווא היא ש-`x` מחשוד ללא חשוד, ו-`FA` לא זיהה את השינוי הזה.

עד כה האנליה שלנו הייתה לא רגישה לזרימה, כלומר כל משתנה הוא בעל `qualifier` אחד שקובע האם המשתנה הוא חשוד תחת כל השימוש שלו במשך כל הרציצה, שהוא כמובן לא תמיד מה שאחנו רוצחים.

אנליזה רגישות זרימה (Flow-Sensitive) מכירה בכך שתוכן המשתנים משתנה לאורך זמן, ולכן לכל השמה במשתנה נחוץ `qualifier` אחר (לדוגמא `x` חשוד בשורה 23 אבל לא חשוד בשורה 105 וכו'). אפשר למשוך זאת באמצעות פיצול כל משתנה למשתנים שונים בין כל שתי השמות.

דוגמה תחת הדוגמה הקודמת בלי ה-if-ים, נפצל את `x` בין הדרישות ונקבל את הקוד הבא,

```
α char *name = fgets(..., network_fd);
β char *x1, γ *x2;
x1 = name;
x2 = "hello!";
printf(x2);
```

$\gamma = \text{untainted} - \alpha \leq \gamma$ ויש לך השמה חוקית $\alpha \leq \beta$, $\text{tainted} \leq \alpha$.
 $\alpha = \beta = \text{tainted}$ -ו.

דוגמה נביט בדוגמה קוד עם מספר conditional-ים

```
void f(int x) {
     $\alpha$  char *y;
    if (x) y = "hello!";
    else y = fgets(..., network_fd);
    if (x) printf(y);
}
```

השורה השלישית נותנת את האילוץ $\alpha \leq \text{untainted}$, הרבייהות נותנת $\text{tainted} \leq \alpha$, והחמייה נותנת $\text{untainted} < \text{tainted}$ שלזה אין פתרון כי קיבל את המשווה $\text{tainted} \leq \alpha \leq \text{untainted}$.
 עם זאת, זו כמובן אזעקה שואה כי `printf` רץ רק אם x מתקיים, ובמקרה זה y מכיל ערך לא חשוב.
 הבעה כאן היא שאפשר לפצל את המשתנים שוב ולבן flow sensitivity לא עוזרת לנו.

אנליזה וגישה מסלול (Path-Sensitive) מכירה בכך שיכולים להיות מסלולים שונים באותה הטענית. ניתן לכל statement אינדקס ואו כל path condition qualifier מכיל qualifier

דוגמה עבור הקוד הנ"ל, האינדקסים שלנו יהיו כבתמונה למטה,

```
void f(int x) {
     $\alpha$  char *y;
 $^1$ if (x)  $^2$ y = "hello!";
 $^3$ else  $^4$ y = fgets(...);
 $^4$ if (x)  $^5$ printf(y);
 $^6\}$ 
```

ואז המסלולים האפשריים הם $x = 0$ ו- $x \neq 0$.
 לאחר מכן, האילוצים שלנו הם $x = 0 \Rightarrow \text{tainted} \leq \alpha \neq 0 \Rightarrow \text{untainted} \leq \alpha$ עבור הרצת קטעים 1-2.
 $x \neq 0 \Rightarrow \text{untainted} \leq \alpha \neq 0 \Rightarrow x$ עבור הרצת קטעים 5-4. להז מובן יש השמה חוקית בכל מסלול.
 $\text{untainted} \leq \alpha \leq \text{untainted}$ ו-1-3.

גם וגישות Flow גומ וגישות Path עוזרות למנוע אזעקות שואה, ומעלה את דיק הכלי. עם זאת, חן מעלה את הסיבוכיות (ולכן את הסקילבילייטי) של הכללי כי הן דורשות את העלאת מספר האילוצים ופתרון אילוצים יותר כללי כדי לטפל ב-path condition-path condition-ים שונים, בהאחתמה.

הערה לרוב ה-conditionals לא יהיה-Calala טריוויאלים ולעתים קשה לאתר שוויון בין תנאים, ואז Path Sensitivity שוב יזרוק התרעת שואה.

דוגמה נתון הקוד הבא, וניתן **qualifier**-ים גם לערכי החזרה והารגומנטים של הפ'.

```
δ char *id(y char *x) { α char *a = fgets(...);  
    return x; β char *b = id(a);  
}
```

עתה נוכל גם לזכור זרימה בין המשתנים שהפ' הקוראת מעבירה לארגומנטים של הפ' הנקראת וערך החזרה של הנקראת למשתנים של הקוראת.

כלומר, עתה יש לנו את האילוצים:

- $\text{fgets} \in \text{tainted} \leq \alpha$.
- $\gamma \leq \alpha$ בגלל ש- a מועבר כארוגומנט.
- $\delta \leq \gamma$ כי ערך החזרה הוא פשוט x .
- $\delta \leq \beta$ עבור ההשמה בשורה האחורונה.

מוסיף עוד שתי שורות לקבלת הקוד הבא,

```
α char *a = fgets(...);  
β char *b = id(a);  
ω char *c = "hi";  
printf(c);
```

ולכן מקבל עוד שני אילוצים:

- $\text{untainted} \leq \omega$.
- $\omega \leq \text{tainted}$.

ואכן יש השמה שמקיימת את האילוצים האלה, בפרט, זהה $\omega = \text{untainted}$ ו- $\alpha = \beta = \gamma = \delta = \text{tainted}$ וכן אין זרימה לא חוקית.

עתה נחליף את הקוד הנוכחי, שמחינתנו אין שום שימוש לשינוי ועדיין אין זרימה לא חוקית,

```
α char *a = fgets(...);  
β char *b = id(a);  
ω char *c = id("hi");  
printf(c);
```

הפעם ארבעת האילוצים הראשונים עדין מתקיים, אך האילוצים הנוספים הם:

- $\gamma \leq \text{untainted}$ כי המחרוזת hi מועברת כארוגמנט לח- pi .

• $\delta \leq \gamma$ כי זהו ערך החזרה של הפ'.

• $\omega \leq \delta$ כי אנחנו מבצעים את ההשמה ה- z בשורה 3.

• $\omega \leq \text{untainted}$ כי אנחנו מדפיסים את c .

הפעם אין השמה חוקית כי מתקיימים $tainted \leq \alpha \leq \gamma \leq \delta \leq \omega \leq \text{untainted}$ ולכן קיבלנו איזוקת שווה.

המשווהה ללא הפטורון קורת בקפייה מהאילוץ $\gamma \leq \alpha$ (בשיטתה -b) לאילוץ $\delta \leq \gamma$ בקריאה $-pi$ בפעם השנייה, אפ"פ שברור שאין קשר בין הקריאה. ככלומר לא הבדנו בكونטקטסט שבו קורים הדברים, שכן אין מסלול אפשרי בין הביטויים המתאיםים לאילוצים אלו.

אנליזה רגישת קונטקטסט מבדילה בין קריאות שונות לאותה הפ'. במקומות להוסיף עוד הרבה qualifierים לכל פעם שקוראים לפ', נשנה את אופן הניתוח שלנו: נסמן אילוצים של השמות בפרמטרים/ערכי החזרה באמצעות אינדקס הקריאה לפ' (פעם ראשונה שקראננו, פעם שנייה וכיו"ב) וסימן להשמה (– אם זה ארגומנט ו+ אם זה ערך החזרה), לכן ההשמה של ערך החזרה של הקריאה ה- i היא משזה מהחזרה $q_1 \leq_{+i} q_2$ (כאשר q_1 ה-*qualifier* של ערך החזרה).

לאחר מכן, כשבודקים אם יש השמה חוקית, נודא משווהת עם כמה אילוצים רק אם יש התאמה בין האינדקסים, ועם סימנים מתאימים (קודם – ואו +). לדוגמה, את השרשור $q_{19} \leq_{+1} q_3 \leq q_4 \leq_{-1} q_1$ נבדוק אבל לא את $q_2 \leq_{-2} q_4 \leq_{+2} q_1$ (מבבל אבל הרעיון הוא שנתייחס רק למסלולים "גיאוגרפיים" מבחינת המעבר של הפרמטרים אל תוך ומוחז לפ').

דוגמא עם אותו הקוד לעיל, ניתן את התווות 1 לקריאה הראשונה $-pi$ ו-2 לקריאה השנייה. לכן אילוצים שלנו הם:

• $\text{tainted} \leq \alpha$ (השמה ב-*a*).

• $\gamma \leq_{-1} \alpha$ (השמה לפרמטר בקריאה הראשונה).

• $\delta \leq \gamma$ (מחזירים את *x* כMOV של שהוא).

• $\beta \leq_{+1} \delta$ (השמה בחזרה מהפ').

ואז לשתי השורות התחנות:

• $\gamma \leq_{-2} \text{untainted}$ (השמה לפרמטר הפ' בקריאה השנייה).

• $\gamma \leq \delta$.

• $\omega \leq_{+2} \delta$ (השמה של ערך החזרה ב-*c*).

• $\omega \leq \text{untainted}$.

ונتيיחס כאמור רק לזרימות עם אותו סימן ואותו אינדקס, ואז כבר לא נקבל את התרעת השווה.

הערה באנליזה עם רגישות לكونטקטסט, אי אפשר סתם לפתור מערכת משוואות כי יש תווות שונות וכו', שכן צריך להשתמש באלג'ריזמים מסוימים בגרפים מכונים ודבירם כאלה, שבהם לא מעסוק.

בדומה לרגישיות קודמות, הדיקט עלה אבל הסקילבילייטי נפגע כי זה שוב מעלה את הסיבוכיות. כדי קצת להקטין את הסיבוכיות, אפשר להתייחס ברגישות רק לחלק מהקריאות לפ', או לתת אינדקס מסוות לכמה קריאות לפ' או לחלופין לתת אינדקסים רק עד לעומק מסוים של קריאות לפ'.

חלק ה' של ההרצאה (קישור)

בעיה נוספת חמורה מażעקות שווה שחן בסופו של דבר לא מזיקות, הן פיספוס זרימות לא חוקיות.

דוגמה ראו הקוד הבא, שעובד עם פוינטרים, עם qualifier-ים.

```

 $\alpha$  char *a = "hi";
 $\beta$  char *p = &a;
 $\gamma$  char *q = p;
 $\omega$  char *b = fgets(...);
*q = b;
printf(*p);
```

מה שקרה פה זה שלושת השורות הראשונות יוצרות מצביעים לחרוזת, השורה הרביעית והחמישית מקבלות מחרוזת חשודה מהמשתמש ושםות במה ש- q מצביע ואז מדפיסות את זה.

האילוצים שלנו הם :

$$\cdot \text{untainted} \leq \alpha$$

$$\cdot \alpha \leq \beta$$

$$\cdot \beta \leq \gamma$$

$$\cdot \text{tainted} \leq \omega$$

$$\cdot \omega \leq \gamma$$

$$\cdot \beta \leq \text{untainted}$$

עבור ההשמה $\alpha = \beta = \text{untainted}$ ו- $\omega = \gamma$, נקבל זרימה חוקית. עם זאת, ברור שלא זרימה חוקית.

מה שפיספסנו כאן הוא ש- q והם אותם הדבר בשםות שונות, ולכן ישיהו להם qualifier-ים שונים ותייחסן זרימה "דו-כיוונית". לכן נוסיף אילוץ בכיוון החפוך, $\beta \leq \gamma$ עבור השורה השלישית. ככה נתפסת את הזרימה הלא חוקית, עם זאת נקבל עוד אזעקות שווה בגל הגישה השמרנית יותר.

זרימה הלא חוקית עכשו היא

$$\text{tainted} \leq \omega \leq \gamma \leq \beta \leq \text{untainted}$$

דוגמה מקרה נוסף שבו נפספס זרימה לא חוקית היא בהשנות אימפליסיות, ראו דוגמה.

```
void copy(tainted char *src,
          untainted char *dst,
          int len) {
    untainted int i, j;
    for (i = 0; i<len; i++) {
        for (j = 0; j<sizeof(char)*256; j++) {
            if (src[i] == (char)j)
                dst[i] = (char)j;           // legal?
        }
    }
}
```

כאן אנחנו מעתיקים שתי מחרוזות, רק שבתוך ה-for-loop החיצוני, במקומות פשוט לעשות השמה ישירה של ערכי המערך, שזה כמובן יתפס ע"י האנליזה הנוכחית, אנחנו עושים לולאה נוספת נספפת שמצוצת את j שווה לטו שאנו מעתיקים ושם אותו. כמובן שזה זהה לחלוtin סמנטי, אבל הפעם אנחנו כן מפספסים את הזרימה הלא חוקית, כי j הוא כמובן .tainted

עתה נכנסת אנליזה חדשה - Information Flow Analysis - שמנסה לאתר מעבר של מידע ממשתנה אחד לאחר באופן אימפליסיטי. נסיף בכל השמה $y = x$ אילוץ נוסף על $q_x \leq q_y$, שהוא $pc \leq q_x$, כאשר pc הוא ה-Program Counter באותו רגע. הרעיון הוא שעצם העבודה שהגענו לשורה הקוד pc , עלולה להשפיע באופן עקיף על ערכו (חשודיותו של הערך, יותר נכון).

דוגמה נתח את הקוד הבא עם אנליזות זרימות מייד.

```
tainted int src;
α int dst;
if (src == 0)
    dst = 0;
else
    dst = 1;

dst += 0;
```

כמובן שכאן dst מושפע באופן עקיף ע"י src ולכן נדרש שזו תהיה חישובית (השורה האחורונה רק מדגימה שימוש במשתנה הנגוע). האיליצים הקלאסיים שנתקבל הם :

$$\text{.untainted} \leq \alpha \cdot$$

$\cdot \text{untainted} \leq \alpha$

$\cdot \text{untainted} \leq \alpha$

ועתה אילוצי ה- pc החדשים (כאשר pc_i הוא pc בשורה ה- i) הם :

$\cdot pc_4 \leq \alpha$

$\cdot pc_6 \leq \alpha$

$\cdot pc_7 \leq \alpha$

עכשו נסוג את pc_i לפי חשודיותם - pc_3 והם untainted כי הגענו אליהם בלי תלות במשתנה חשוד (תמיד נבדוק את תנאי ה-if) ותמיד נרץ את השורה האחורית) אבל pc_4, pc_6 הם tainted כי מגיעים אליהם בהתבסט על תנאי ה-if שתלו依 ב- pc החשוד. לכן גילינו ש- α הוא חשוד ונدع להתריע מפני שימוש בו כ- untainted , שכן $\alpha = \text{tainted} = pc_4 \leq \alpha$.

עם זאת, לא נשמש תמיד **Information Flow Analysis** כי הוא מתריע בפני הרבה אזעקות שווה (לדוגמא if עם אותה ההשמה בשני המקרים). הסיבה השנייה לא להשתמש היא שהחסוביות עולה משמעותית בשל העלייה במספר המשתנים. למדנו **Flow Analysis** דהיינו יש אתגרים רבים נוספים מעבר לכך.

אתגרים באנליזה סטטיטית של קוד

- מה קורה בפעולות כמו סכימה על משתנים? אם ידוע לנו משלו על התפלגות הערכים, לעיתים ניתן לקבוע שהם גם אמ untainted . ערכו מכיל operand שהוא כן tainted .
- מה קורה עם פוינטורים לפ' ? לאילו פ' מותר לפוינטור להציבו ולאילו לא?
- האם מערכים נחשים אמינים/חשודים כאובייקט כולל, או שככל אלמנט צריך לקבל qualifier משל עצמו?
- כיצד נזהה סנטציה של ערכים (ኒקיי ערכים לא אמינים לאמינים)?
- לזהות متى משתנה חייב להיות untainted לדוגמה אינדקסים בגישה למערכים.
- כיצד להרחיב את הנition להזיהוי דליפה של מידע רגיש, לדוגמה לאסור על מעבר מידע מקורות פרטיים ל"כיריים" (מידע שמוצג למשתמש בצורה צו או אחרת) פומביים. הרבה יותר קשה להזיהות הדלפות עיקיפות של מידע סודי זהה ותחום שלם.

תרגול

קישור

שאלה נתון הקוד הבא

```

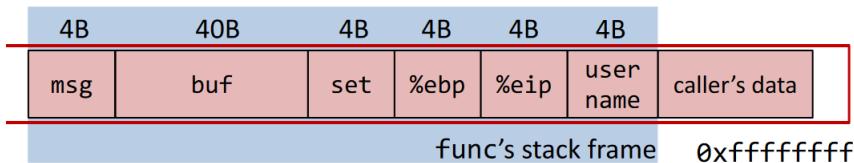
int func(char *username) {
    int set=0;
    char buf[40];
    char *msg="Good Job";

    sprintf(buf, "Welcome %s", username);
    while (set) {
        printf(buf);
    }
    return 0;
}

```

כאשר הינה פ' שמה בארגומנט הראשון את המחרוזת שהייתה מודפסת לפי הפורמט הנתון, אם היה מדובר ב-printf.
הניחו את כל ההנחות הסטנדרטיות.

1. תארו את תוכן המחסנית כתוצאה מהקריאה ל-c.func.



2. הראו כיצד ניתן לגרום לפ' להיכנס ללולאה אינסופית.

נציב ב-username את הקלט 1111..111 כך שיהיה 33 תווי 1 סה"כ, ואז המחרוזת sprintf תכתוב ל-buf בית אחד יותר מדי שידروس את set ויתן לו ערך שאינו אפס, כך שניכנס ללולאה אינסופית.

נשים לב כי set הוא ארבעה בתים ולבן הבית הראשון יהיה 1, הבית השני 0 כי זה ה-zero terminator ואז שני הבטים האחרים ישארו 0. כך set הוא לא 1, אלא 2^{17} , שהוא עדין בסדר כי זה לא 0.

3. כיצד ניתן לשנות את הפ' על מנת למנוע את המתקפה הנ'ל?

נשנה את printf ל-snprintf ונוסף בארגומנט השני 40 כדי לסמם שאחנו רוצים מחרוזת באורך לכל היותר 40 תוים (כולל ה-zero terminator).

4. הראו כיצד ניתן (בהתעלם מהשינויי הנ'ל) לתת קלט לפ' כך שיודפס Good Job בלא האינסוף (בליל שבקלט יופיע הביטוי עצמו, ואפשר להדפיס גם דברים נוספים מלבד הביטוי).

ניתן את הקלט %s111..111 וואז set ידרס לערך שאינו 0 ובולאה האינסופית, buf יתפרש כפורמט ואז printf יחפש את המחרוזת שעליו להדפיס, ימצא בדיקות msg (ראו הרכיב המרוזת הנ'ל) וידפיס אותו.

5. האם שימוש ב-ASLR או ערכי קנריית יעוז?

ASLR לא כי אנחנו לא מtabססים על שם דבר קבוע וערכי קנריית גם לא כי אנחנו לא נוגעים ב-ebp או eip.

שאלה נתונים קטעי הקוד הבאים

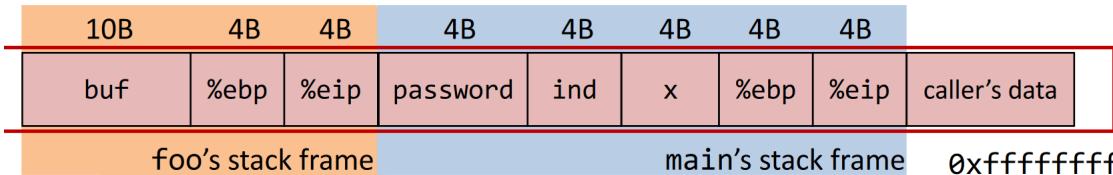
```

01 void foo() {
02     char buf[10];
03
04     gets(buf);
05     printf("%s\n", buf);
06 }
07 void bar() {
08     printf("Good Job\n");
09 }
10 typedef void (*fptr)(void);
11 fptr ptrs[3] = { NULL, foo, bar }; 20 }

12 void main() {
13     int x=0;
14     int ind=1;
15     char password[4];
16
17     gets(password);
18     printf("The address is %p\n", ptrs[ind]);
19     foo();

```

.1. תארו את תוכן מהחסנית בשורה 3 ובשורה 16 (בכחול שורה 16, בכתום הtos甫ת בשורה 3)



.2. הראו כיצד ניתן לגרום להרצת הפ' bar באמצעות הזנת שני קלטים בשורות 17 ו-1.

ל-password ניתן AAAA2...AAAAAdrs נסמנהadrস.buf ניתן bar adrস.buf נסמנהadrস.buf ניתן foo' כז' שתוודפס בשורה הבאה כתובתה של הפ' bar. כז' foo' כז' שהכתובת תדרוס את %eip, כז' בחזרה מ-foo' נקבע לכתובת של bar.

.3. הראו כיצד ניתן למנוע מתקפה זו באמצעות שינוי הפ' בשורות 17 ו-3.

נփוך את שתי קריאות fgets ל-gets ובารוגומנט אורך המחרוזת שנבקש, נשים את המספרים המתאימים (4 ו-10 בהתאמה).

.4. האם הוספת ערכי קנרי או ASLR יעוזרו?

קנריין כן כי היה תחשום את ה-ASLR אבל Buffer Overflow לא כי לא קשור.

שבוע II | ניתוח קוד סטטי לעומק

תרגול

[קישור](#)

שאלה נתון הקוד הבא

```

01 /* int printf(un tainted char *fstr, ...); */
02 /* tainted char *fgets(...); */

03 void func(FILE *fp, int set) {
04     char buf[100];
05     char *name, *output;
06
07     name = fgets(buf, sizeof(buf), fp);
08     echo(name);
09     if (set) {
10         output = echo("Untainted string");
11     }
12     else {
13         output = "One more untainted string";
14     }
15     printf(echo(output));
16 }
```

```

17 char *echo(char *str) {
18     return(str);
19 }
```

1. נתחו קטע זה ללא אף רגישות והראו שמתאפשרת התראה על זרימה לא חוקית באמצעות בניית אוסף האילוצים, והציבו העל זרימה לא חוקית הנובעת מאיילוצים אלו.

האיילוצים הם (מסודרים בסדר כרונולוגי של קריאות בקוד):

.tainted \leq name •

.name \leq str •

.str \leq echo •

.untainted \leq str •

.str \leq echo •

.echo \leq output •

.untainted \leq output •

.output \leq str •

.str \leq echo •

.echo \leq untainted •

זהירותם הלא חוקית מתאפשרת ע"י $\text{.tainted} \leq \text{name} \leq \text{str} \leq \text{echo} \leq \text{untainted}$

2. הראו כי ניתן להוציא רגישות אחת מבין context, flow, path sensitivity או זיהוי implicit flows context, flow, path sensitivity כך שלא תתקבל התראה על זרימה לא חוקית ע"י בניית האילוצים עם הרגישות והסביר מילולי על למה לא מתאפשרת התראה.

נזכיר בקצתה כי path שם לב-l-qualifier-ים גם לשורות הקוד עצמן. המשטנה, ו- context sensitivity implicit flows שומר qualifier-ים גם לשורות הקוד עצמן.

במקרה הזה, הבעה נוצרת כי יש זרימה לא חוקית שקופה דרך קריאות השונות של הפ' ולכן context sensitivity יעזר,

אוסף האילוצים במקרה זה הוא:

$\cdot \text{tainted} \leq name \bullet$
 $\cdot name \leq_{-1} str \bullet$
 $\cdot str \leq echo \bullet$
 $\cdot \text{untainted} \leq_{-2} str \bullet$
 $\cdot str \leq echo \bullet$
 $\cdot echo \leq_{+2} output \bullet$
 $\cdot \text{untainted} \leq output \bullet$
 $\cdot output \leq_{-3} str \bullet$
 $\cdot str \leq echo \bullet$
 $\cdot echo \leq_{+3} \text{untainted} \bullet$

ניתן לראות כי האילוצים לא מסתכנים לכדי זרימה לא חוקית.

שאלה נתון הקוד הבא,

```

01 /* int printf(untainted char *fstr, ...); */  

02 /* tainted char *fgets(...); */  

03 char *echo(char *msg, int set) {  

04     if (set) {  

05         return(msg);  

06     }  

07     else {  

08         return("Hello");  

09     }  

10 }

```

```

11 void func(FILE *fp, int flag) {  

12     char buf[100];  

13     char *sp, *sp1, *sp2;  

14  

15     sp = fgets(buf, sizeof(buf), fp);  

16     sp1 = echo(sp, flag);  

17     sp2 = echo(sp, flag);  

18     if (!flag) {  

19         printf(sp2);  

20     }  

21 }

```

1. נתחו את קטע הקוד ללא רגישיות והראו שמתיקת אזעקה שווה, כלומר הצביע על זרימה לא חוקית מאוסף האילוצים שתכתבו והסבירו למה היא לא באמת פרצת אבטחה.

אוסף האילוצים הוא :

$\cdot \text{tainted} \leq sp \bullet$
 $\cdot sp \leq msg \bullet$
 $\cdot flag \leq set \bullet$

- $.msg \leq echo$ •
- $.untainted \leq echo$ •
- $.echo \leq sp1$ •
- (חמשת האילוצים חנ"ל, רק עם $sp2$ הפעם).
- $.sp2 \leq untainted$ •

ازעקה השווה מתקבלת ע"י הזרימה הלא חוקית $tainted \leq sp \leq msg \leq echo \leq sp2 \leq untainted$. זו אינה באמות בעיה משום שנדפיס את $sp2$ רק אם $flag$ כבוי, ובמקרה כזה ה'echo' תמיד החזירה את המחרוזת `Hello` שהיא $untainted$.

2. הראו כי ניתן להוסיף וגישה אחת מבין הרגישויות שלMANDO כך שלא תהיה אזעקה שווה.

הבעיה כאן היא שהتعلמנו מה-if-ים השונים, אך נוסיף path sensitivity. מספר את ה-conditionals באופן הבא,

```

01 /* int printf(untainted char *fstr, ...); */
02 /* tainted char *fgets(...); */

03 char *echo(char *msg, int set) {
04     if (set) {
05         return(msg);
06     }
07     else {
08         return("Hello");
09     }
10 } void func(FILE *fp, int flag) {
11     char buf[100];
12     char *sp, *sp1, *sp2;
13
14     sp = fgets(buf, sizeof(buf), fp);
15     sp1 = echo(sp, flag);
16     sp2 = echo(sp, flag);
17     if (!flag) {
18         printf(sp2);
19     }
20 }
21 }
```

ונשים לב כי המסלולים האפשריים הם $5 - 1 - 2 - 1 - 2 - 4$ או $5 - 1 - 3 - 1 - 3 - 4$ ואף אחד מהם לא מספק זרימה לא חוקית (משמעותי חוסר כוח לא כתבתי את האילוצים מחדש, אבל בעיקר מפרידים את אוסף האילוצים לשני אוספי אילוצים עם חפיפה כלשהי, ורואים שהוא "שאפשרו את הזרימה החוקית לפני נמצאים בשני אוספים שונים ולכך אין אפשרות לבנות מהם זרימה לא חוקית").

שאלה נתון קטע קוד הבא,

```

01 /* int printf(untainted char *fstr, ...); */
02 /* tainted char *fgets(...); */

03 char *echo(char *name) {
04     return(name);
05 }

06 void foo(FILE *fp1, FILE *fp2, int set) {
07     char buf1[100], buf2[100];
08     char *msg1, *msg2, *str1, *str2;
09
10     msg1 = fgets(buf1, sizeof(buf1), fp1);
11     if (set) {
12         msg2 = "Good Luck";
13         printf("%s", msg1);
14     }
15     else {
16         msg2 = fgets(buf2, sizeof(buf2), fp2);
17     }
18     str1 = echo(msg1);
19     str2 = echo(msg2);
20     if (set) printf(str2);
21 }

```

1. נתחו את קטע הקוד ללא רגישיות והציבו על קיומן של שתי זרימות לא חוקיות והסבירו למה הן לא מקבילות לזרימות אמיתיות בתכנית.

אוסף האילוצים הוא :

$$\cdot \text{.tainted} \leq msg1$$

$$\cdot \text{.untainted} \leq msg2$$

- (אין כאן אילוץ על printf בשורה 13 כי אנחנו דורשים רק שהפרמטר הראשון יהיה **untainted** וכן אין ה-**tainted**-הארוגומנט השני).

$$\cdot \text{.tainted} \leq msg2$$

$$\cdot msg1 \leq name$$

$$\cdot name \leq echo$$

$$\cdot echo \leq str1$$

$$\cdot msg2 \leq name$$

$$\cdot name \leq echo$$

$$\cdot echo \leq str2$$

$$\cdot str2 \leq \text{untainted}$$

tainted $\leq msg1 \leq name \leq echo \leq str2 \leq untainted$

tainted $\leq msg2 \leq name \leq echo \leq str2 \leq untainted$

כאשר אף אחת מהן לא קורת במצבות כי הראשונה קופצת בין קריאות לפ' echo והשניה לא אמיתית כי הא"ש הראשון רלוונטי רק אם set כבוי ובמקרה כזה הא"ש האחרון לא מתקיים (לא ננכדים ל-if).

2. האם ניתן להוסיף רגישות אחת מבין context, path, flow כך שלא תתקבל אזעקה שווא?

לא. flow בכלל לא הקשור context ימנع את הזרימה הראשונה אבל לא את השנייה, ו-path ימנע את השנייה אבל לא את הראשונה (ובע מחסיבות לאי-קיום הזרימות שהזכרתי לעיל). הסטודנטית המשקיפה כתוב את כל האילוצים מחדש שכן עדין יש זרימות לא חוקיות בכל המקרים.

3. הראו כי ניתן להוסיף שתי רגישות כך שלא יתקבלו אזעקות שווא.

נוסיף path ו-context ואז האילוצים עברו $set \neq msg2 \leq tainted$ והוא הכל חוץ מ- i וגם הוספה $i \pm$ על קריאות הפ' ואז לא מקבלים אזעקה, ובעור $0 = set = str \leq untainted$ מורידים את $i \pm$ ומושיפים $i \pm$ ולא מקבלים אזעקה.

שבוע Fuzzing ו-Pen-testing | XIII

הרצאה

חלק א' של הרצאה (קישור)

בחרצאה הקודמת רأינו אנליזה סטטיטית שמאפשרת לבדוק את כל הרכזות האפשריות של התוכנה והוא מאוד שימושית, אבל במצבות בעיקר מנסים להימנע מازעקות שווא וזה גורם להם לפפס באגים/פרצחות אבטחה.

הרצאה סמלית היא גישת ניתנים בין טסטינג ואנליזה סטטיטית שמנסה לשמור את היתרונות שלהם ולמגר את ההשלכות שלהם. טסטינג לא נותן לנו אזעקות שווא כי כל בדיקה מתאימה לקלט אחד בלבד, זהה גם החסרונו שלה. פורמלית, אנחנו מקבלים completeness אבל לא assert ובדיקה אחרת מעמידים רק על המסלולים שאנו בודקים ולא כל מסלול. זו בעיה כי התוקף צריך רק מסלול אחדsoundness. כל הרצאה סימבולית מבלילה טסטינג.

בחרצאה סמלית, ניתן למשתנים ערכים סמליים, לדוגמה $\alpha = y$ ואז כשנעשה assert נוכל לבדוק מקרים יותר כלליים מאשר שציב. בנוסף, תלות של מסלול הרצאה בערך סמלי מוביל אותנו לפיצול של מסלול, שגם אליו יכול לעקוב.

דוגמא נתון קטע הקוד הבא עם השמות סמליות במשתנים,

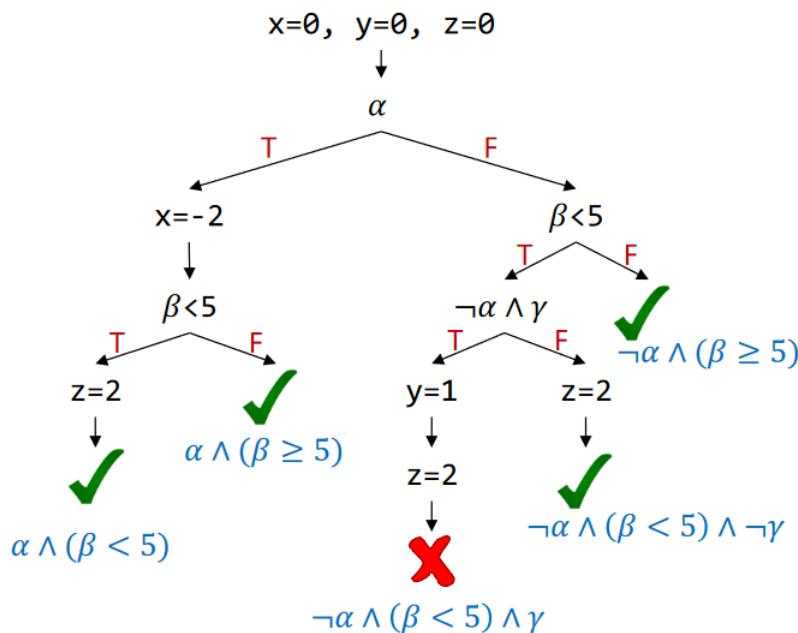
```

1. int a =  $\alpha$ , b =  $\beta$ , c =  $\gamma$ ;
2.                               // symbolic
3. int x = 0, y = 0, z = 0;
4. if (a) {
5.     x = -2;
6. }
7. if (b < 5) {
8.     if (!a && c) { y = 1; }
9. z = 2;
10. }
11. assert(x+y+z != 3)

```

אנחנו מנסים לבדוק האם קיימת ריצה עם השמה בערכים הסמליים שתקיים $x + y + z == 3$.

נזכיר עץ שבו הקודקודים יהיו תנאים על המשתנים הסמליים ב-*conditinals*,



התנאי בתחילת נקרא תנאי המסלול. מצאנו מסלול שבו אין מתקיים התנאי הלא רצוי. נשים לב כי כיסינו את כל המסלולים האפשריים.

הערה בኒוגד לטסTING, קיבלנו את כל המסלולים האפשריים בהשمات המשתנים הסמליים.

הרצת סימבולית לא נותנת אזעקות שווה, אבל היא יכולה לא להסתיים ודורשת קושי חישובי מאד גדול.

בעבר הקושי החישובי היה מכשול משמעותי בכך לשימוש בהרצת סימבולית ונדרשים פתרנים שיקבעו האם מסלולים הם אפשריים בכלל ואילו assert-ים יכולים להיכשל.

בשני העשורים האחרונים הייתה התקדמות טכנולוגית והן פיתוחית (לפתרונות) וכיום הרצת סימבולית מצליחה אכן למצוא מפגעים.

חלק ב' של ההרצאה ([קישור](#))

באופן פורמלי, הרצת סמלית מוסיפה אפשרות לביטויים בשפת התכנות (מלבד מספרים, סכומים, א"ש וכו') - משתנה סמלי (SMSML ערך לא ידוע). מכאן גם שביטויים נוספים יכולים להכיל משתנים סמליים, לדוגמה $5 + \alpha$.

נכис מעתנים סמליים במקומות שבהם מקבלים קלטיים מהמשתמש/תוקף.

דוגמה נתון קטע הקוד הבא,

```
x = read();
y = 5 + x;
z = 7 + y;
a[z] = 1;
```

כל נראה שמעבר קלטים מסוימים נקבל overflow. נניח בritchת קונקרטית שננתנו ל- x את הערך 5, ושה- a מערך עם ארבע איברים, עתה ניגש ל-[17] a שווה אובייקט, שהוא פרצת אבטחה.

בritchת סמלית, נשים ב- x את α , ואז ניגשים ל-[17] $z = 12 + \alpha$, $y = 5 + \alpha$, $a[12 + \alpha] = 1$. כאן אנחנו יודעים אפילו מעבר אילו ערכי α מקבלים אובייקט.

דוגמה נתון קטע הקוד הבא,

```
1. x = read();
2. if (x>5) {
3.   y = 6;
4.   if (x<10)
5.     y = 5;
6. } else y = 0;
```

נסמן ב- π את תנאי המסלול שנדרש כדי להגיע לשורה כלשהי.

- תנאי המסלול של שורה 3 הוא $\pi = (\alpha > 5)$.
- לשורה 5 זהו $\pi = (\alpha > 5) \wedge (\alpha < 3)$.
- לשורה 6 זהו $\pi = (\alpha \leq 5)$.

נשים לב כי תנאי המסלול מכילים משתנים סמליים. נוכל להשתמש בפתרונות כדי לראות אילו מסלולים הם פיזיבליים, ובמקרה הזה כל המסלולים הם פיזיבליים חוץ משורה 5.

דוגמה נתון קטע הקוד הבא עם תנאי המסלולים לידיו

1. x = read();	$\pi = \text{true}$
2. y = 5 + x;	$\pi = \text{true}$
3. z = 7 + y;	$\pi = \text{true}$
4. if(z < 0)	$\pi = \text{true}$
5. abort();	$\pi = (12 + \alpha < 0)$
6. if(z >= 4);	$\pi = \neg(12 + \alpha < 0)$
7. abort();	$\pi = \neg(12 + \alpha < 0) \wedge (12 + \alpha \geq 4)$
8. a[z] = 1;	$\pi = \neg(12 + \alpha < 0) \wedge \neg(12 + \alpha \geq 4)$

אם ניתן להגיע לשורה 5 או 7 (אם אחד המסלולים פיזיבלי) מצאנו מפגע בティוחות.

ריצה מתפצלת

כש מגיעים לפיצול בהרצה, נבדוק האם קיימות שימושות שנקננות למסלול ונשתמש באלו' שבודק את כל הנסיבות האפשריים באופן שיטתי.

1. אתחול: $0 = pc$ מספר השורה שעתה נרי' סמלולית, $\emptyset = \pi$ תנאי המסלול הנוכחי ו- $\emptyset = \sigma$ המצב הסמלולי.
2. נכניס את (pc, π, σ) לרשימת המשימות.
3. כל עוד רשימת המשימות לא ריקה:
 - (א) נוציא מהרשימה משימה (pc, π, σ) ונריץ אותה.
 - (ב) אם יש חסד לפיצול ב-(pc_0, π_0, σ_0) כאשר התנאי הוא p :
 - i. אם $p \wedge \pi_0$ מסופק ע"י איזושהי השמה, נוסיף לרשימת המשימות את $\{p = \text{True}\}$ כאשר $pc_1 = (pc_1, \pi_0 \wedge p), \sigma_0 \cup \{p = \text{True}\}$.
 - השורה הראשונה לאחר הפיצול כ- p מתקיים.
 - ii. אם $p \neg \wedge \pi_0$ מסופק ע"י השמה כלשהי, נוסיף לרשימת המשימות את $\{p = \text{False}\}$ כאשר $pc_2 = (pc_2, \pi_0 \wedge \neg p), \sigma_0 \cup \{p = \text{False}\}$.
 - השורה הראשונה לאחר הפיצול כ- p לא מתקיים.

הערה כבר עכשו אפשר לראות שזה מאוד דומה ל-BFS.

בשלב כלשהו נגיעה לגבולות הקוד, כאשר הפ' תקרה לספריה, תריש קוד אסמליל וכו'. אפשר לבחור להריץ גם את קוד הספריה באופן סימבולי. בגלל שספריות יכולות להיות מורכבות לעיתים, ניתן להריץ סמלית גרסה פשוטה של libc (לדוגמה newlib), שלא ייעילה אבל קללה להרצה סימבוליית.

בהרצה סמלית דינמית מרים סמלית ובאותו הזמן מרים קונקרטית עם ערכיהם שמספקים את תנאי המסלול הנוכחי. בכל פעם משתמשים על מסלול אחר כאשר לאחר סיום מסלול אחד הופכים ערך כלשהו בזיכרון ורואים אם מגיעים למסלול אחר. כך יש גם ערכים קונקרטיים שאפשר להשתמש בהם כי אנחנו באמת מרים את הקוד.

הkonkrطيות זו נותנת לנו את היכולת לzechot מסלולים קונקרטיים שmaguiim למפגעים. בנוסף, אפשר באמצעות `syscall`-ים כי אנחנו מרים קוד, אבל שם מ Abedים את הסמלוליות של התוצאה.

חלק ג' של ההרצאה (קישור)

בהרצה סמלית אנחנו מכוונים לקבל הבטחה חד משמעית בנוגע לבטיחות הקוד, אבל הבעיה היא שפרקטיית לא נוכל לחכות עד שההרצה מסיימת, כי יכול להיות שמספר המסלולים האפשריים הוא אקספוננציאלי במספר המשתנים הסמליליים או בשורות הקוד.

דוגמא קוד שעושה 3 if-else על משתנים סמלולים מוביל ל- 2^{32} מסלולים אפשריים.

דוגמא לולאות (while על α) פותחות 1 – 2^{32} מסלולים.

היתרון של אגלוזה סטטיט היא שהיא נגמרה תמיד לאחר בדיקת כל המסלולים האפשריים, כשהיא למעשה משערכת הרבה הרצות ופתרונות - החלק שגורם לאזעקות שווה.

נרצה להכוון את ההרצת סמלית למסלולים בעייתיים. אפשר להסתכל על הרצת סמלית כגרף מכון, כאשר הקודדים הם מוצבי התכנית ויש צלע בין שני קודקודים אם ניתן לעبور אחד לשני. תחת סימונים אלה, אלג' הרצת סמלית הוא פשוט BFS על הגרף שונצ'ר.

חיפוש באמצעות BFS נותן לנו חיפוי רוחבי והוא לא מתרכו רק בנישה אחת במשך רוב ריצתו (בנהחה שהיא מוגבלת), בעוד DFS מגיע למסלולים עמוקים הרבה יותר, אבל יכול להיתקע על מסלולים שבכל אחד מהם בעייתיים. לכן תמיד צריכה להיות איזושהי הנחיה מהותית שמודעת לקוד עצמו. כאמור, נדרש לטעוף מסלולים מסוימים.

אם אין לנו דרך חכמה לטעוף, אפשר להכניס אקראיות אל תוך הרצתה. לדוגמה, אם לא הגיעו במפגע, להתחיל את הרצתה מהתחלת באקראיות, או לבחור את המסלול הבא כל פעם באקראיות.

כמובן שנדרש להיות יכולים לשחזר את האקראיות הזו, וכך משתמש ב-PRF עם מפתח שנשמר במקרה שנדרצה לשחזר את התוצאות.

דוגמה לעתים אקראיות לא תעזר לנו בכלל. בקורס הבא מספר המסלולים האפשריים הוא 2^{100} ומספר המסלולים שבו 75 הוא $counter = 75^{2^{100}}$, כלומר בהסתמך 2^{78} ניפול על מסלול זה אם משתמש באקראיות - ככלומר פספסנו מפגע רציני.

```
int counter = 0;
for (i = 0; i<100; i++) {
    if (input[i] == 'B') {
        counter++;
    }
}
assert(counter != 75);
```

מערכות נפוצות להרצת סמלית

- SAGE - מבצעת הרצת דינאמית ומוקדמת במציאת אגמים בפרסרים של קבצים (شكل להשתמש בהרצת סמלית עבורה). השיטה עברה על מיליוני מסלולים וגילתה מאות באגים.
- KLEE - מרייך סמלואית bitcode של LLVM. היא משתמשת במרייך הסמליא שראינו באמצעות fork כדי לנהל מספר תהליכיים. היא משתמשת בכמה שיטות חיפוש שונות וכן מסמכלצת `syscall`-ים.
- Cloud9, Mayhem

חלק ד' של ההרצאה (קיישור)

בטסטינג הבעה היא שיכול להיות שפיספסנו קלט אחד בעייתי אבל זו שיטה די טובה מבחינות אחרות. Fuzzing נותן ערכאים אקריםים כלליים ומכוון לזיהוי בעיות שטחיות למדוי, בעיקר קריסות אקספشنים וכו', שוללות להיות תשתיית למתפקידים נוספים.

Fuzzing מנסה לשבש את הריצה של התכנית באמצעות ערכים אקרים.

Fuzzing שיטות

- קופסה שחורה - לא ידוע לנו שום דבר על התכנית או הקלטים שלה. שיטה זו היא קלה לשימוש אבל בודקת רק מצבים מאוד שטחיים.
- Fuzzing מבוסס על דקdock - הכללי יוצר קלטים שימושיים ל"דקdock". שיטה זו דורשת יותר ממאנץ' לשימוש אבל יכולה להגיע עמוק יותר.
- קופסה לבנה - כלי שיוצר קלטים באופן מודע חלקית לקוד של התכנית. שיטה זו קלה לשימוש אבל יקרה חישובית.

Fuzzing קלטים ב-

- מותציה - לחת ערך חוקי ולשנות אותו (ערך חוקי מושג ע"י בן אדם או פתרן SMT, או דקdock ידוע).
- גנרטיבי - יצירה קלטים "from scratch".
- תערובת של השניים - יצירה קלטים גנרטיבית, מטזית, עוד יצירה גנרטיבית, יצירה מותציות עם דקdock.

דוגמא Fuzzer Radamsa הוא קופסה שחורה המבוסס על מותציות.

```
% echo "1 + (2 + (3 + 4))" | radamsa --seed 12 -n 4
5!++ (3 + -5)
1 + (3 + 41907596644)
1 + (-4 + (3 + 4))
1 + (2 + (3 + 4)
```

הכלி מקבל כקלט ערך תיקין, seed ומספר מותציות רצויות ומחזיר ערכים אקרים בדומה לדקdock.

דוגמא Fuzzer Blab הוא COPFA שורש הקשר שմבוסס על ביטויים רגולריים.

דוגמא Fuzzer Americal Fuzzy Lop הוא COPFA שורש הקשר שמייצרת מותציות. הוא מרים פעם אחת, עושה מותציות לקלטים ואם נכנס למסלול אחר שומר את המותציה.

נניח שעשינו Fuzzing וחטנית קרסה. כיצד ניתן לסייע להקרישה? הקלטים של Fuzzing הם בכלל מסווגים ולכן נרצה לפשט אותם. בנוסף, פישוט קלטים כאלה יכול לגרום לנו אם שתי קרישות נבעו מאותו הבאג (לדוגמא אם שניהם פושטו לאותו קלט מפושט). בנוסף, נרצה לדעת האם קרישת צו היא באמת מפגע אבטחה או לא.

דוגמא ביצוע Dereference על NULL אינה מפגע אבטחה אבל Buffer Overflow כמובן שכן.

הבעיה היא שקשה מאוד לזהות מה הסיבה לקריסה, כי יכול להיות ש-buffer overflow השפיע מאוחר הרבה יותר מביצוע האוברפלואו עצמו וקשר בין שניים זה לא פשוט.

לשם כן אפשר להשתמש ב-Address Sanitizer, שמנגישים קוד שקורס מיד כשייש גישה לא חוקית, ואז אם קורסנו במהלך נדע לבדוק מה הגורם. Fuzzing

תרגול

קישור

שאלה נתון קטע הקוד הבא,

```

01 /* int printf(un tainted char *fstr, ...); */
02 /* tainted char *fgets(...); */

03 char *foo(int flag, char *msg1, char *msg2) {
04     if (flag) {
05         return(msg1);
06     }
07     else {
08         return(msg2);
09     }
10 }

11 void func(int action) {
12     char buf[100];
13     char *name, *temp;
14
15     name = fgets(buf, sizeof(buf), stdin);
16     foo(action, name, "Alice");
17     temp = foo(action, "Alice", "Bob");
18     if (action) {
19         name = "Eve";
20     }
21     printf(temp);
22 }
```

. נתו את קטע הקוד ללא רישיונות והראו שמתקובלת איזיקת שווה.

הailozimim hem :

.tainted \leq name •

.action \leq flag •

.name \leq msg1 •

.untainted \leq msg2 •

.msg1 \leq foo •

.msg2 \leq foo •

.action \leq flag •

.untainted $\leq msg1$ •

.untainted $\leq msg2$ •

.msg1 $\leq foo$ •

.msg2 $\leq foo$ •

.foo $\leq temp$ •

.untainted $\leq name$ •

.temp $\leq untainted$ •

ואזעקה השווה מתקובלת מ-untainted $\leq name \leq msg1 \leq foo \leq temp \leq untainted$

2. הראו כי ניתן להוסיף וגישה אחת ולהימנע מازעקה השווה.

הבעיה היא שאנו קוראים בין קריאות לפ', לכן נסיף context ומונע מהבעיה כי האילוצים לא יאפשרו לקפוץ בין אינדקסים (הא"ש שסוטרים הם השני שהוא 1 – והאחד לפני האחרון שהוא 2).
+).

שאלה נתון קטע הקוד הבא,

```
01 /* int printf(untainted char *fstr, ...); */
02 /* tainted char *fgets(...); */

03 char *simple(char *name) {
04     printf("Name: %s", name);
05     return(name);
06 }
07
08 void foo(FILE *networkFP, int set) {
09     char buf[100];
10     char *str, *str1;
11
12     str = fgets(buf, sizeof(buf), networkFP);
13     if (set) {
14         str1 = simple(str);
15         printf("%s", str1);
16     }
17     str1 = simple("Alice");
18     printf(str1);
19 }
```

1. נתחו את קטע הקוד הבא בלי רגישיות והראו כי מתקובלת אזעקה שווה.

.tainted $\leq str$ •

.str $\leq name$ •

.name $\leq simple$ •

.simple $\leq str1$ •

.untainted $\leq name$ •

.name \leq *simple* •

.simple \leq *str1* •

.str1 \leq *untainted* •

tainted \leq *str* \leq *name* \leq *simple* \leq *str1* \leq *untainted*

הזעקה מתקבלת עבור הזרימה הלא חוקית *context, path, flow* שתמנע את אזעקה השווא? 2.

הבעיה היא שדרסנו את ערך המשתנה *str1* עם ערך בסדר רק שעשינו זאת באמצעות קריאה לפ' שנקרה בפעם אחרת עם פרמטר חשוב לנו כמו *flow* וגם *context* ולכן אחת לא תספק.

לא קשור ולכון הוא לא יעוזר, רק *flow* יאפשר את הזרימה הלא חוקית *path*

tainted \leq *str* \leq *name* \leq *simple* \leq *str1B* \leq *untainted*

ורק *context* יאפשר את הזרימה הלא חוקית

tainted \leq *str* \leq_{-1} *name* \leq *simple* \leq_{+1} *str1* \leq *untainted*

3. הראו כי ניתן להוסיף שתי רגישיות כך שלא תתקבל אזעקה שווה.

נוסיף *flow* ו-*context* כאמור ונשים לב כי הא"ש העליון מהסעיף הקודם לא מתקיים כי הא"ש השני והאחד לפני האחרון מקבלים +1 – בהתאם ולכון זו לא זרימה והא"ש התיכון לא מתקיים כי הא"ש אחד לפני האחרון הוא על *str1A* והאחרון הוא על *str1B* ולכן גם שם אין זרימה לא חוקית.

שאלה נתון קטע הקוד הבא,

```
01 /* int printf(untainted char *fstr, ...); */
02 /* tainted char *fgets(...); */
03 void *func(FILE *fp) {
04     char buf[100];
05     char *name, *output;
06
07     name = fgets(buf, sizeof(buf), fp);
08     if (!strcmp(name, "Alice")) {
09         output = "Hello Alice";
10     }
11     else {
12         output = "You are not Alice";
13     }
14     printf(output);
15 }
```

1. נתחו את קטע הקוד הבא בלי אף רגישות והראו כי קיימת השמה של ערכי **tainted** ו-**untainted** שמספקת את אוסף האילוצים.

הailozim הם :

$$\cdot .tainted \leq name$$

$$\cdot .untainted \leq output$$

$$\cdot .untainted \leq output$$

$$\cdot .output \leq untainted$$

במקרה זה ההשמה **output** = **untainted**, **name** = **tainted**.

2. האם הוספת אחת מבין implicit flows context, flow, path או implicit flows name מושפע מ-**name** אבל ההשפעה קוררת בתוך ה-**if**, לכן implicit flows ייעזר. האילוצים כאן הוא ש-**output** מקבל ערך שמוספע מ-**name** אבל ההשפעה קוררת בתוך ה-**if**.

(המשמעותיים) שנוספים הם :

$$\cdot .pc_9 = tainted$$

$$\cdot .pc_{12} = tainted$$

$$\cdot .pc_7 \leq name$$

$$\cdot .pc_9 \leq output$$

$$\cdot .pc_{12} \leq output$$

ואז יש זרימה לא חוקית **tainted** = $pc_9 \leq output \leq untainted$

שבוע 13 | חזרה

תרגול

שאלה תהי G פ' ניתנת לחישוב ביעילות שמעתיקה n ל- $2n$. תהי F שמקבלת מפתחות בגודל n ומעתיקה $2n$ ל- $2n$ (ניתנת לחישוב ביעילות).

נדיר H משפחת פ' שמקבלת מפתחות בגודל n המוגדרת ע"י

. נניח כי H ה-PRG, האם G בהכרח ?

היא אינה בהכרח PRG, נסתכל על $|G(s)| = s|0^{|s|}}$ (העתקת הזוזות בסופו של דבר). ברור שזו אינה PRG כי הבית האחרון

שלה הוא 0, בהסתמך. תהי F PRF ונוכיח כי H היא

ונניח בשילוח ש- H אינה PRF, לכן קיים מבחין \mathcal{D} ופולינום $p(n)$ כך שמתקיים באופן שכיה

$$\left| \Pr_{k \leftarrow \{0,1\}^n} (\mathcal{D}^{H_k}(1^n) = 1) - \Pr_{h \leftarrow \text{Func}_{n,2n}} (\mathcal{D}^h(1^n) = 1) \right| > \frac{1}{p(n)}$$

נדיר \mathcal{A} מבוחן ל- F -אוון הבא: בהינתן גישה לאורקל $\mathcal{O} \in \{F_k, f\}$, מרים את \mathcal{D} ועונה לכל בקשת אורקל עם $(\mathcal{O}(G(x)), F_k(0^n))$ ומחזיר את הערך ש- \mathcal{D} -מחזר.

ברור שזהו יריב PPT.

- אם $\mathcal{O} = F_k$: קיבלנו סימולציה מדויקת של \mathcal{D} עם H , שכן

$$\mathcal{O}(G(x)) = F_k(G(x)) = H_k(x)$$

• אם $\mathcal{O} = f$: מתקיים כי $f(G(x))$ היא פ' שמתפלגת אחיד על $\text{Func}_{n,2n}$ שכן $G(x)$ חח'ע, אך $f(g(x)) = h(x)$ כאשר h פ' אקרואית באמת.

לכן קיבלנו (אפשר לכתוב את הנוסחאות במפורש אבל עשוינו זאת זה מספיק פעמים) של- \mathcal{A} . יש יתרון לא זניח על F בסתירה להיות PRF F .

2. נניח כי F היא PRG ו- G היא PRF, האם H היא בהכרח?

היא אינה בהכרח PRF. אם ל- G יש התגשות שדועה לנו, לדוגמה $G(0^n) = G(1^n)$, אז H אינה כיו PRF כי שזה מספיק ל מבחין שמצילich ביתרונו לא זניח (ל-PRG F מותר תהה PRF F כלשהו ונראה שקיים G כנ"ל). הוי W , בתרגיל 1 הוכחנו כי

$$G(s) = \begin{cases} W(1^{|s|}) & s = 0^{|s|} \\ W(s) & \text{אחרת} \end{cases}$$

הוא גם PRG.

שאלה תהי PRF F שעמיה n ל- n עם מפתחות בגודל n .

1. נגידיר $(0^n), G(s) = F_s(0^n) || F_{F_s(0^n)}(0^n)$ הוכיחו כי איןנו PRG לכל F .

נדיר מבוחן \mathcal{D} בין G לערכיהם אקרואים, שהניתן קלט $\alpha || \beta \in \{0,1\}^{2n}$, מחזיר 1 אם "ס" ($\alpha || \beta = F_\alpha(0^n) || F_\beta(0^n)$) ואם "ז" ($\alpha || \beta = G(s)$ כאשר s קבוע ו- β אחיד). לכן G אינו PRG כי יש לו יתרון ברור (הסטודנטית המשקיפה תפרמל זאת).

PPT.

אם $\alpha || \beta = G(s)$ תמיד יחזיר 1 ואם $\alpha || \beta = r \leftarrow \{0,1\}^{2n}$ מתפלגים אחיד וב"ת ולכן $(\alpha || \beta = G(s) \text{ או } \alpha || \beta = r)$ ביחסת' $\alpha || \beta = G(s)$ (כאשר α קבוע ו- β אחיד).

2. נגידיר $(1^n), H(s) = F_s(0^n) || (F_{F_s(0^n)}(0^n) \oplus F_s(1^n))$ הוא PRG לכל F .

ראשית ברור שזהו אלג' PPT. נניח בשילילה שקיים יריב \mathcal{A} עם יתרון לא זניח על H לעומת ערך אקרואי.

נדיר מבוחן \mathcal{D} ל- F -אוון הבא: בהינתן גישת אורקל $\mathcal{O} \in \{F, f\}$, בקש מהאורקל את (1^n) והחזיר את הפלט של \mathcal{A} על הקלט $(F_\alpha(0^n) \oplus \beta)$. ברור שזהו יריב PPT.

• אם $\mathcal{O} = F_k$: מורץ עם הקלט $(F_{F_k(0^n)}(0^n) \oplus F_k(1^n)) = H(k)$.

• אם $\mathcal{O} = f$ כאשר f אקראית בامتה : α, β מתפלגים אחיד וב"ת ולכן $\beta \oplus \alpha$ -ב"ת, כלומר \mathcal{D} מרץ

את \mathcal{A} עם קלט אקראי.

לכן,

$$\begin{aligned} & \left| \underset{s \leftarrow \{0,1\}^n}{P} (\mathcal{D}^{F_s}(1^n) = 1) - \underset{f \leftarrow \text{Func}_{n,2n}}{P} (\mathcal{D}^f(1^n) = 1) \right| \\ &= \left| \underset{s \leftarrow \{0,1\}^n}{P} (\mathcal{A}(H(s)) = 1) - \underset{r \leftarrow \{0,1\}^{2n}}{P} (\mathcal{A}(r) = 1) \right| \\ &> \frac{1}{p(n)} \end{aligned}$$

באופן שכיח, בסתיויה להיות F PRF.

?PRG האם $W(s) = F_s(0^n) \parallel F_{F_s(1^n)}(1^n)$.3 נגידר

W הוא בהכרח PRG. נניח בשלילה שהוא אינו, ולכן קיימים יריב \mathcal{A} PPT ו- $p(n)$ כך שמתקיים באופן שכיח

$$\left| \underset{s \leftarrow \{0,1\}^n}{P} (\mathcal{A}(W(s)) = 1) - \underset{r \leftarrow \{0,1\}^{2n}}{P} (\mathcal{A}(r) = 1) \right| > \frac{1}{p(n)}$$

נשתמש בארגומנט היברידי,

$$\begin{aligned} \frac{1}{p(n)} &< \left| \underset{s \leftarrow \{0,1\}^n}{P} (\mathcal{A}(W(s)) = 1) - \underset{r \leftarrow \{0,1\}^{2n}}{P} (\mathcal{A}(r) = 1) \right| \\ &= \left| \underset{s \leftarrow \{0,1\}^n}{P} (\mathcal{A}(F_s(0^n) \parallel F_{F_s(1^n)}(1^n)) = 1) - \underset{r \leftarrow \{0,1\}^{2n}}{P} (\mathcal{A}(r_0 \parallel r_1) = 1) \right| \\ &\stackrel{\Delta}{\leq} \left| \underset{s \leftarrow \{0,1\}^n}{P} (\mathcal{A}(F_s(0^n) \parallel F_{F_s(1^n)}(1^n)) = 1) - \underset{r_0, t \leftarrow \{0,1\}^n}{P} (\mathcal{A}(r_0 \parallel F_t(1^n)) = 1) \right| \\ &\quad + \left| \underset{r_0, t \leftarrow \{0,1\}^n}{P} (\mathcal{A}(r_0 \parallel F_t(1^n)) = 1) - \underset{r \leftarrow \{0,1\}^{2n}}{P} (\mathcal{A}(r_0 \parallel r_1) = 1) \right| \end{aligned}$$

לכן או שהאיבר הראשון גדול מ- $\frac{1}{2p(n)}$ באופן שכיח או השני. נחלק למקרים :

• אם הביטויו הראשוני מתקיים לא זניח באופן שכיח : נגידר יריב \mathcal{D} המבחן בין F לפ' אקראית באופן הבא : בהינתן גישת

אורקל $r_0 \parallel F_t(1^n)$, בקש את הערכים $\mathcal{O} \in \{F_s, f\}$, $t = \mathcal{O}(1^n)$, $r = \mathcal{O}(0^n)$, $r_0 = r$ ווחזור את הפלט של \mathcal{A} על הקלט

יש גישה ל- F כרגיל).

ברור שהוא יריב PPT.

אם $\mathcal{D} : \mathcal{O} = F_s$ מרים את \mathcal{A} על $F_s(0^n) \parallel F_{F_s(1^n)}(1^n)$ –

אם $\mathcal{D} : \mathcal{O} = f$ מארח את \mathcal{A} על $F_s(0^n) \parallel F_{F_s(1^n)}(1^n)$ –

$$\begin{aligned}
& \left| P_{s \leftarrow \{0,1\}^n} (\mathcal{D}^{F_s}(1^n) = 1) - P_{f \leftarrow \text{Func}_{n,n}} (\mathcal{D}^f(1^n) = 1) \right| \\
&= \left| P_{s \leftarrow \{0,1\}^n} (\mathcal{A}(F_s(0^n) || F_{F_s(1^n)}(1^n)) = 1) - P_{r_0, t \leftarrow \{0,1\}^n} (\mathcal{A}(r_0 || F_t(1^n)) = 1) \right| \\
&> \frac{1}{2p(n)}
\end{aligned}$$

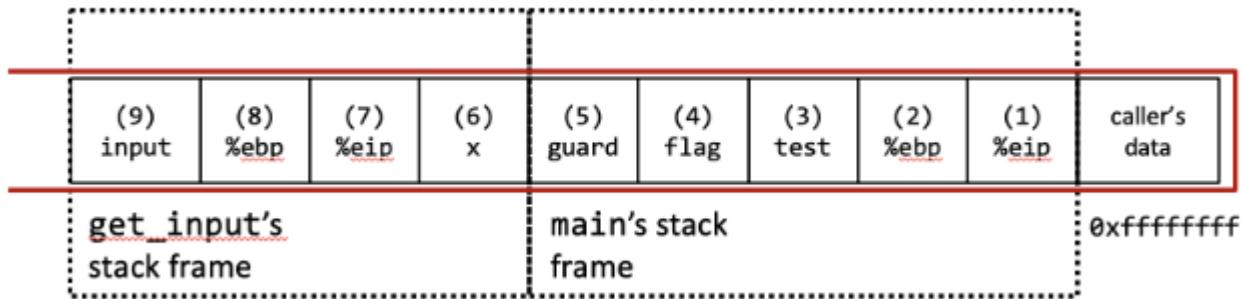
באופן שכיח, ככלומר מצאנו יריב שמנצח את F_s ביתרונו לא זניח, בסתירה להיוותה PRF.

- אם הביטוי השני מותקיים לא זניח באופן שכיח: נגידיר יריב \mathcal{D} המבחן בין F לפ' אקרואית באמצעות באופן הבא: בהינתן גישת $r_0 || r_1$, מבקש את הערך $\mathcal{O}(1^n)$ ודוגם אחד $r_1 \leftarrow \{0,1\}^n$ ומוחזר את הפלט של \mathcal{A} על $r_0 || r_1$.
 אורקל \mathcal{D} מרים את \mathcal{A} על $r_0 || F_t(0^n)$ כאשר r_0 ו- t -מתפלגים אחידים – אם $\mathcal{O} = F_s$ – או אז \mathcal{D} מרים את \mathcal{A} על $r_0 || r_1$ אקרואיטים – אם $\mathcal{O} = f_s$ –
 לכן באותו האופן כמו לפני, מצאנו יריב שמנצח ביתרונו לא זניח על פni F_s בסתירה להיוותה PRF.

שאלה נתון קטע הקוד הבא,

<pre> 01 void main() { 02 int test = 128; 03 int flag = 1; 04 int guard = 0; 05 06 get_input(0); 07 flag = 1 - flag; 08 if (flag == -127) first_goal(); 09 if (flag == 1 && test == 1) second_goal(); 10 get_input(0); 11 } </pre>	<pre> 12 void get_input(int x){ 13 char input[20]; 14 15 fgets(input, 32, stdin); 16 printf(input); 17 fgets(input, 32, stdin); 18 } 19 void first_goal(){ 20 printf("Good Job!"); 21 } 22 void second_goal(){ 23 printf("Amazing!"); 24 } </pre>
--	---

- תארו את תוכן המחסנית בתוך קרייאתת הראשונה של `main` ל-`get_input` (כלומר בשורה 14), כאשר אין שום אופטימיזציות וכל ההנחות הרגילות חלות.



2. הראו כי תוקף המרץ את main וצופה במידע המודפס למשך יכול לספק קלט בשורות 15 ו-17 כך שהfirst_goal פעם

אחד או יותר.

ניתן לקבל ל-15 את `d%k%p%d%d%d` כז שבשורה 16 יודפסו 5 מספרים שותופים את כל תוכן הבאר `input`, ואז הפযינטרים שתוכנם `eip`, `%eip`.

בשורה 17 ניתן את הקלט `k` ב-`A...A(ebp+4)eip` כאשר יש 20 א-ים כדי למלא את הבאר `input`, `ebp` הוא ערך שהודפס לנו ל-`%ebp` בשורה 16 (הפযינטර הראשון) ו-`k` בו בהתאם.

עכשו הfp' תחזיר כרגע ל-`main` לאויה הנקודה, רק שעתה המחשב חושב שהמשתנה המקומי הראשון שלו הוא אחד אחורה שהוא באמת, ככלומר גישה ל-`flag` היא עצם `test`. בשורה 7 נבצע `flag=1-flag` כאשר למשה החישוב יהיה `first_goal` ואכן `flag=-127` וכן `test=1-test`.

3. האם שימוש ב-CFI היה מונע את המתקפה שהצתתם?

לא, כי המתקפה לא שינה קריאה לא ישירה לפ' (רק `k` השונה).

4. הראו כי תוקף המרץ את main וראה מה מודפס למשך יכול לספק קלטים בשורות 15 ו-17 (בשתי ריצות `get_input`) כך שתורץ `first_goal` פעם אחד או יותר וגם תורץ `second_goal` פעם אחד או יותר.

בריצה הראונה של `get_input` נספק את אותם הקלטים כנ"ל וכך נקבל ריצה של `first_goal` כפי שכבר ראיינו, כאן לא ירוץ `second_goal` כי התנאי ב-9 לא מתקיים (`flag` שהוא בעצם `test` הוא -127).

בריצה השנייה של `get_input` נספק את הקלטים `eip`, `ebp`, `main` כבשיער 2 ויש 20 א-ים כדי לדלג מעל הבאר. המטרה של חיסור 4 בתים היא שוב להסיט את המחסנית באופן מטעה, רק הפעם לצד השני. ערך ה-`eip` כפי שקיבלו אותו בקריאה הקודמת ל-`get_input` יגרום לכך שהזרה מה-`get_input` השנייה, נחזיר לשורה 7 ב-`main`. לאחר מכן, שורה 7 תחשב `flag=1-flag` ערכו בעצם ערכו של `guard`, ככלומר נקבל `flag=1` וגישה ל-`test` בעצם גשת ל-`flag` ואז אכן מתקיים ה-`if` ונರיץ את `second_goal`.