(67513) מבוא למטבעות קריפטוגרפיים | פרויקט סופי

יקיר עוז | 203330683, אביב ברו | 204319453

תקציר

רשת Lightning היא רשת מבוזרת הפועלת על גבי רשת ה-Bitcoin ומטרתה לפתור את בעיית הסילוּמיוּת (scalability) של הרשת.
בפרויקט שלנו החלטנו לממש ולנתח את הרעיון למיגור תקיפת Grifeing ברשת Lightning בעזרת הטלת קנס על מבצעי המתקפה. הרעיון מוצג ב-[1].
מתקפת Grifeing היא בעיה ידועה אשר בעזרתה תוקפים יכולים להשבית חלקים גדולים מרשת Lightning ע"י "סתימת" ערוצים ברשת ובכך פוגעים ביכולת שלה להתרחב ולשמש פתרון לבעיית הסילוּמיוּת של רשת הBlockchain, בנוסף, יכול תוקף להשבית את הפעילות של Node ספציפי ובכך להסית ממנו טרנזאקציות.
במאמר הנ"ל, הרעיון הכללי הוא הטלת קנס על מבצע מתקפת Griefing ובכך לעשותה לא כלכלית. הקנס הוא חלק מפרוטוקול חדש בשם HTLC-GP שמרחיב את פרוטוקול ה-HTLC הקיים כיום ב-Lightning Network ובעצם חלק מהחוזה שנחתם בין כל Node בשרשרת העברה כלשהי.
בעבודה זו ביקשנו לבדוק את ההשפעות של הפרוטוקול על הרשת, מה ההשפעה הכללית על הרשת, ההשפעה הפרטנית על Nodes ישרים אשר משתתפים ברשת המשתמשת בפרוטוקול זה ובנוסף נציג מתקפת DOS שמתאפשרת בעזרת פרוטוקול זה ואסטרטגית תקיפה חדשה, אשר שנקרא לה Soft Griefing ונבדוק את הכדאיות וההשפעה שלה ביחס ל-Griefing "קלאסי".

מבוא

Lightning Network

רשת ה-Lightning היא רשת אנונימית המשמשת כשכבה שנייה על גבי רשת מבוססת Blockchain כמו רשת ה-Bitcoin. מטרת הרשת היא לפתור את בעית הסילוּמיוּת ע"י שימוש ב-Payment Channels דו-כיווניים בין משתמשים ברשתת כך שתשלומים נחתמים ע"י שני המשתתפים בטרנזקציה ויכולים להתבצע מחוץ ל-Blockchain וכך להתבסס על רשת התקשורת בין המשתתפים. הרשת מאפשרת תשלום בין משתתפים להם אין channel ישיר ע"י שימוש בחוזה HTLC המאפשר לעבור בין משתמשי ביניים מבלי לסמוך עליהם, אלא על ידי חתימות קריפטוגרפיות.

Griefing Attack

נציג את מתקפת Griefing שהיא מתקפה המתאפשרת ברשת שמשתמשת בפרוטוקול HTLC. מתקפה זו היא בעיה ברשת Lightning וגורמת לה להיות פחות אטרקטיבית ולהוות פיתרון אמיתי לבעית הסילוּמיוּת. מטרת המתקפה היא לנעול כמות גדולה של כסף של משתמש הוגן ולגרום ל-Denial Of Service, בידוד של המשתמש מהרשת או נעילת channel לגמרי. ניזכר כי כאשר Node A מעוניין להעביר כסף ל-Node R, הוא צריך למצוא מסלול בינו לבין R על גבי רשת Lightning, במקרה של התמונה למטה, המסלול הוא $A \to B \to C \to D \to R$. בין כל שני Nodes עוקבים יש ליצור חוזה HTLC אשר יוודא כי אם התשלום יעבור בהצלחה, כל הNodes במסלול יקבלו את המגיע שלהם, או לחילופין, שאם התשלום לא הצליח, לא יפסידו כסף.

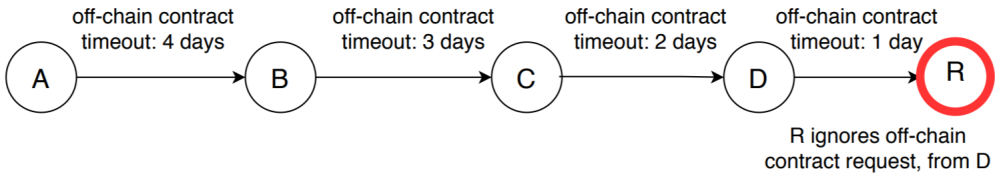
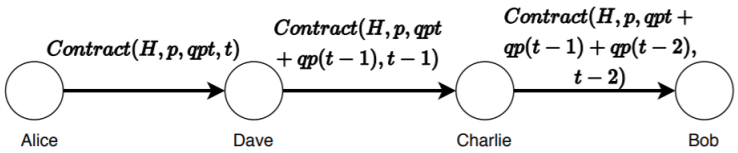


Figure 1: Griefing Attack: when node R ignores HTLC request

לכל חוזה יש תוקף (היורד לאורך המסלול) וברגע שהוא נגמר - החוזה מתבטל. במשך הזמן שבו החוזה לא ממומש אבל גם עדיין בתוקף, ישנם סכומים אשר נעולים לשימוש. כלומר יוצר ה-HTLC נועל את הכספים בשביל ה-Node אליו הוא מעביר את הכסף. תוקף המבצע Griefing רוצה לנעול כמה שיותר כספים לכמה שיותר זמן והוא עושה זאת ע"י התעלמות ממימוש חוזה HTLC, בדרך זו הוא מסוגל לנעול כספים למספר גדול של Node-ים, כאורך המסלול, ולאורך התקופה שבה החוזה שחתם עם הNode שלפניו במסלול בתוקף.
בדוגמה שלנו, R מתעלם מהחוזה וגורם לכל השרשרת לנעול
כספים ליום אחד לפחות ובמסלולים ארוכים יותר זו יכולה להיות בעיה רצינית הרבה יותר.
אחת המוטיבציות לביצוע מתקפה זו היא נעילת כספים של Node-ים, כך לגרום לטרנזאקציות ש-Node-ים רוצים לבצע לא להיות מסוגלים לעבור דרך אותם channel-ים עם כספים נעולים, ובכך להגדיל את הסיכוי שיעברו דרך התוקף, שיוכל להרוויח כסף מעמלה שהוא גובה על העברת טרנזאקציות.

HTLC-GP Protocol

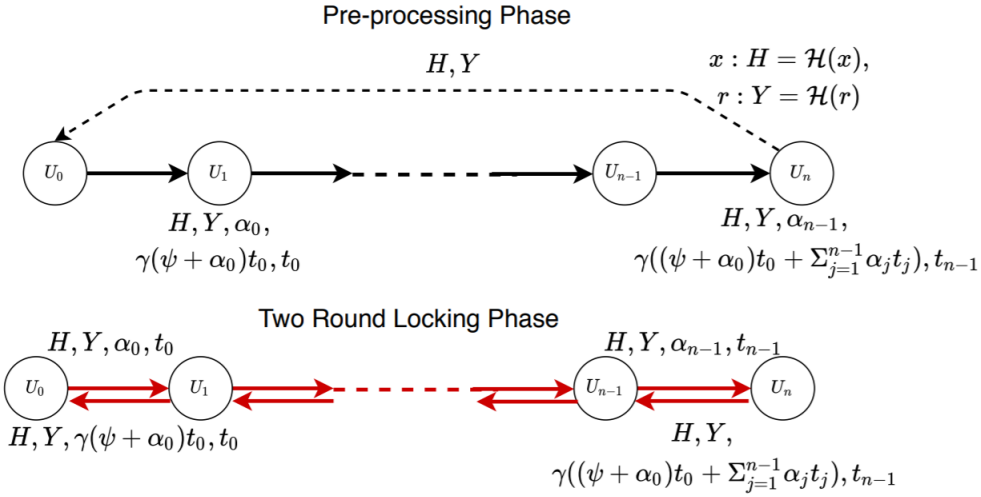
כותבי המאמר מציעים בניה חדשה של HTLC על מנת להטיל קנס על מבצע מתקפת ה-Griefing. הרעיון הוא שמקבל החוזה ינעל כספים
נוספים (בהתאם למיקומו במסלול התשלום), אם החוזה פג תוקף, מקבל החוזה משלם את הסכום הנקוב לשולח החוזה (נאכף ע"י החוזה והבלוקצ'יין).
נגדיר קבוע q שייצג את מחיר קנס ה-Griefing, כלומר כמה מקבל החוזה צריך לשלם (ביחס לסכום שננעל) על כל דקה שבה נעל כסף (ולבסוף לא העביר אותו). על התוקף לשלם לכל ה"מותקפים" שלפניו בשרשרת כלומר לכל מי שנעל את הכסף אבל בעצם לא קיבל fee.



כפי שניתן לראות בתמונה, הקנס הוא מצטבר, זה כי כל Node בשרשרת צריך לשלם ל-Node שקדם לו. בדוגמה זו דייב נועל (בנוסף לכספים עבור העברה עצמה) $qp_{Alice}t$ כספים (כאשר q זה "קצב הקנס" כמו שהצגנו למעלה - בדר"כ מוצג כמיליסאטושי לדקה, אבל לצורך הדוגמה נתייחס אליו ביחידות של מיליסאטושי ליום, t זה זמן נעילת החוזה בימים ו-p_{Alice} זה סכום העברה שמשתנה בהתאם ל-fee שמשולם ל-Node), צ'ארלי נועל $qp_{Alice}t+qp_{Dave}(t-1)$ כספים ובוב $qp_{Alice}t+qp_{Dave}(t-1)+qp_{Charlie}(t-2)$. במקרה שבו בוב יחליט לבצע Griefing ולהתעלם ממימוש החוזה שנשלח אליו,
הוא ישלם לצ'ארלי $qp_{Alice}t+qp_{Dave}(t-1)+qp_{Charlie}(t-2)$, צ'ארלי בתורו ישלם $qp_{Alice}t+qp_{Dave}(t-1)$ לדייב ובאותו אופן דייב ישלם לאליס $qp_{Alice}t$ ובכך כולם יקבלו תשלום על כך שנעלו כספים לשווא מבוב שביצע Griefing, והוא כמובן לא יקבל כלום.
כותבי המאמר הציגו 2 פרוטוקולים למימוש רעיון זה, הפרוטוקול הראשון[2], והפרוטוקול השני[1] שמוצג בגירסא הנוכחית של המאמר.
בפרוטוקול הראשון, הוצג חוזה HTLC חדש בו לפי הסיטואציה של התמונה למעלה, אם $Alice$ מעוניינת להעביר ל-$Dave$ כמות של p_{Alice}, היא נועלת p_{Alice} בחוזה, ו-$Dave$ נועל מצידו את ה-penalty (כפי שהוצג קודם). אם $Dave$ יציג x כך ש-$H(x) = H$ (כמו בחוזה

HTLC רגיל) עד זמן שנקבע ב-HTLC, הוא יקבל את כל הכסף הנעול. אם לא הציג x כזה והזמן עבר, $Alice$ תקבל את כל הכסף הנעול, וכך תפוצה על הזמן שהכסף שלה היה נעול. נשתמש בחוזים כאלו בין $Alice, Dave, Charlie$ ו-Bob ובכך אם אחד מה-Node יבצע Griefing הוא ישלם את הפיצוי כפי שהוצג קודם. הפרוטוקול אמנם פתר את Griefing attack ע"י כך שהפך את המתקפה ללא משתלמת משום שמבצע המתקפה ישלם penalty, אבל מספר בעיות, אשר נפרט עליהן בהמשך, נוספו בעקבותו.

בפרוטוקול המעודכן, כש-$Alice$ מעוניינת להעביר כסף אל Bob באותה צורה שמוצגת בתמונה, היא שולחת אל Bob הודעה על כך שהיא מעוניינת לשלם לו, הוא מחזיר לה H, Y כך ש- $H = \mathcal{H}(x), Y = H(r)$ (כאשר $\mathcal{H}$ היא פונקציית $Hash$). בשלב זה $Alice$ שולחת הודעה דרך כל המשתתפים בטרנזאקציה שיוכלו לדעת מי ה-Node שנמצא לפניהם ואחריהם, כמות הכסף שצריך לנעול, H, Y ועוד פרטים הנחוצים לביצוע הטרנזאקציה. כעת, Bob יוצר חוזה HTLC שנקרא לו Cancellation Contract עם $Charlie$ בו Bob נועל את ה-$Penalty$ הנסכם עבור שאר ה-$Nodes$ (כפי שהוצג קודם). אם Bob מציג את r או x, הכסף הנעול חוזר אליו. אם הזמן של החוזה עבר, $Charlie$ מקבל את הכסף הנעול (הפיצוי).
$Charlie$ יוצר בצורה דומה Cancellation Contract עם $Dave$ שיוצר כזה עם $Alice$.
כעת, $Alice$ שולחת אל $Dave$ חוזה $HTLC$ חדש שבו היא נועלת p_{Alice}. אם $Dave$ מציג את x הוא יכול לקבל את הכסף הנעול. אם $Alice$ מציגה את r או שתוקף החוזה פג, היא מקבלת את הכסף הנעול. לחוזה זה נקרא Forward Contract.
$Dave$ יוצר חוזה באופן דומה עם $Charlie$ שיוצר כזה עם Bob.
כעת, בוב ישחרר את x ל-$Charlie$ כדי לקבל את תשלום הכסף,
והם יבטלו את ה-Cancellation Contract (אשר מתבטל גם ע"י הצגה של x) ו-$Charlie$ ישלח כסף באופן רגיל. $Charlie$ בתורו יבצע את אותם הפעולות עם $Dave$ שיבצע עם $Alice$. וכך הכסף עבר מ-$Alice$ ל-Bob בצורה בטוחה.



בתמונה זו ניתן לראות את שלב שליחת H ו-Y ואז את שלב נעילת הכסף כך שמכיוון U_n ל-U_0 זהו חוזה ה-Cancellation Contract ומכיוון U_0 ל-U_n זהו החוזה הרגיל שמעביר את הכסף. אלמנט חשוב נוסף בפרוטוקול הוא פרמטר δ. הוא מגדיר את הזמן המקסימלי ש-Bob יחכה מרגע ששלח חוזה Cancellation ועד שקיבל את חוזה התשלום מ-$Charlie$. אם לא קיבל חוזה כזה והזמן שעבר הוא יותר מ-δ, Bob יבטל את חוזה ה-Cancellation בעזרת חשיפת r. הפרמטר חשוב משום שאחרת Bob יכול היה לנעול את ה-penalty לזמן רב.

Reverse Griefing

הגרסה הראשונה של המאמר מציג בעיה חדשה (אשר צויינה במאמר) בשם Reverse Griefing - מתקפה זו בעצם מטילה את העונש על Node ישר ובכך גורמת לו להפסיד כסף.

* בעיה זו קיימת בפרוטוקול הישן - לכן ניזכר בו מההסבר הראשוני.

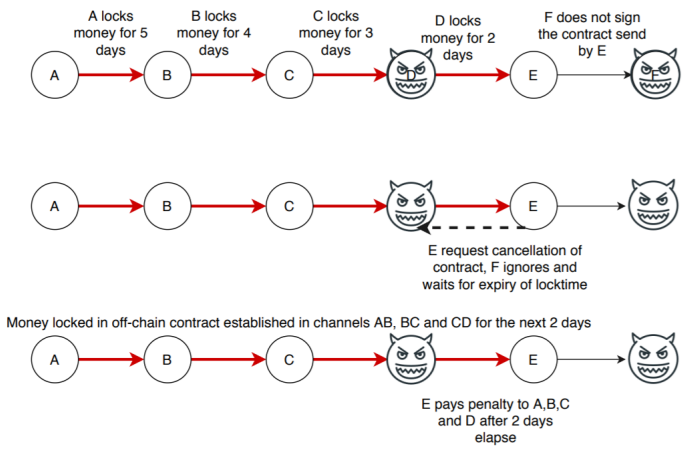


Figure 8: D and F are malicious, D reverse-griefs, E is victimized

בדוגמה זו מתבצעת העברה העוברת בתת המסלול $A \to B \to C \to D \to E \to F$ כאשר F ו-D תוקפים את E : כל החוזים בין A ל-E כבר אותחלו, כאשר כעת E מנסה לאתחל חוזה עם F ו-F לא מגיב. כעת, E מבקש לבטל את החוזה עם D אך האחרון מתעלם מהבקשה ומחכה שהחוזה שלו עם E ייפקע ו-E יצטרך לשלם קנס על לא עוול בכפו. הפרוטוקול הראשון אשר הוצע ע"י כותבי המאמר רגיש לבעיה זו.

בפרוטוקול של המאמר המעודכן, העובדה שאנחנו קודם יוצרים את חוזי ה-Cancellation מונעת את הבעיה משום שאם F יתעלם לגמרי מהודעות של E, הוא יצטרף לשלם לו penalty. ואם יבטל את חוזה ה-ה-Cancellation שלו עם E אז זה אומר שהוא שיחרר את x או r, ולכן E יוכל לבטל את חוזה ה-ה-Cancellation שלו עם D.

Soft Griefing

בשרשור מיילים בנוגע למאמר, הוצג הרעיון של Soft-Griefing. מתקפה זו דומה ל-Griefing הרגיל, ומנצלת חיסרון של הפרוטוקול בכך שהעונש המוטל על Node לא פרופורציונלי לכמות הזמן שהכסף היה נעול, אלא רק מתייחס להאם העברת הכסף בוצעה או שלא. רעיון המתקפה הוא שה-Node הרשע ממשיך את מהלך התשלום הרגיל (כלומר חושף את x) ברגע האחרון לפני שהחוזה פג, כלומר הוא גורם לכסף להיות נעול במשך הזמן הכי ממושך שאפשר, כך שהתשלום עדיין עובר והוא לא צריך לשלם את הקנס. באותה מידה הוא יכל גם לחשוף את r, לא לקבל קנס ולא להעביר את התשלום. Node מסוים יכול לנצל זאת וכשהעברת כספים עוברת דרכו, הוא יכול לחכות בעת העברת ה-Cancellation Contract ל-Node הבא בשרשרת ולגרום לנעילת כספים. לחילופין הוא יכול לחכות בפיתרון (שליחת x) ה-Forward Contract, ובכך לגרום לכספים רבים להיות נעולים.

כאשר הכספים עוברים דרכו הוא עלול להיות מוגבל בפרמטר δ, אך אם הוא מקבל התשלום, הוא יכול לבצע את המתקפה עד לזמן שסמוך לפקיעת החוזה.

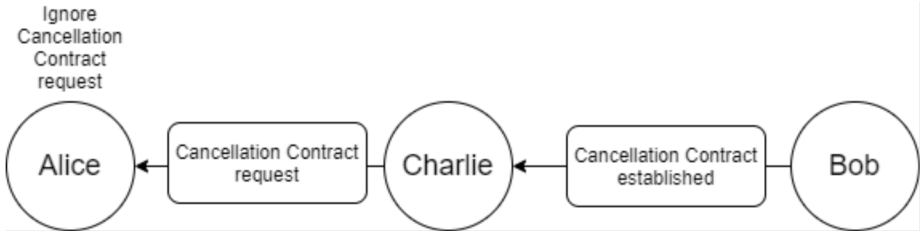
נחקור מתקפה זו בהמשך.

Dos Attack

כעת ברצונינו להציג מתקפה חדשה. נציג איך במתקפה זו התוקף לא צריך להתחייב ולנעול כסף ובכל זאת לגרום ל-Node ישר להתחייב ולנעול סכומי כסף משמעותיים.
התוקף ישלח הודעה לקורבן שהוא מעוניין להעביר אליו סכום כסף מסוים, כך שהמסלול ביניהם מכיל לפחות Node נוסף. הקורבן מעוניין לקבל את הכסף ולכן ישלח Cancellation Contract ל-Node שלפניו במסלול התשלום, שיאשר את החוזה וישלח חוזה משלו ל-Node הבא, כך עד שהחוזה יגיע לתוקף (מה-Node הראשון במסלול). כעת התוקף יבצע Griefing ויתעלם מההודעה ליצירת החוזה. בשלב זה הקורבן יזהה שלא קיבל הודעת Forward תוך זמן של δ, ויבקש מה-Node שלפניו לבטל את חוזה ה-Cancellation ביניהם, שמצידו יבטל את החוזה הבא, וכך הלאה.

נשים לב שמלבד התוקף שהוא ה-Node הראשון במסלול, כל ה-Node-ים במסלול ובניהם הקורבן נועלים סכום כסף גבוה ביחס לסכום שהתוקף הכריז שישלח (כתשלום penalty לכל אחד מהמשתתפים לפניו), לזמן מוגדר מראש - δ, והקורבן נועל את הסכום הגבוה ביותר.
התוקף לא היה צריך שום סכום התחלתי (כן צריך קיבולת מוגבלת בערוץ, אך נעילת כספים לא מתבצעת), אלא רק ערוץ עם מסלול לקורבן, ואין מניעה שיתחיל שליחות נוספת כאלו.
כלומר, הוא יכול דרך מספר Node-ים שונים שבשליטתו לבצע את המתקפה על הקורבן, שינעל עבור כל הודעה סכום גדול של כסף למשך זמן של δ. כך נוכל לגרום ל-Denial Of Service עבור אותו קורבן משום שהוא עלול לא להיות מסוגל להעביר כסף או לקבל כסף דרך אותם channel-ים שנעל בהם כסף (גם לצורך שליחה של כסף וגם לקבלה יש צורך לנעול כסף מצידו).

כמו כן, נוכל לבצע מתקפה דומה גם עבור תשלומים שלא מיועדים לאותו קורבן, אלא שהוא אחד מה-Node בדרך, כך שגם אם יחליט להתעלם מהודעות תשלום מ-Node מסוים נוכל עדיין לגרום לאפקט דומה (אם כי בצורה זו הוא לא נועל את הסכום המקסימלי של כסף) - זאת על מנת לא להכנס ל-blacklist במהירות.



בדוגמא זו, Alice מבצעת את המתקפה ולאחר שליחת ההודעה לבוב על הרצון לשלם לו, היא מתעלמם מהודעות בקשה ליצירת חוזה Cancellation.

Penalty Risk

כדי לפתור את בעית ה-Griefing, HTLC-GP מציע לנעול כמות גדולה מאד של כספים ביחס לכסף הנשלח (לפי המאמר בו צוין יחס Griefing penalty של 0.001) מה שעלול לחשוף את המשתתפים במסלול התשלומים לסיכון גבוה. חלק מהמתקפות שהצגנו מסתמכות על כך.
למשל, נעילת ה-penalty עלולה לגרום למתקפת DOS רגילה על Node ברשת להיות אטרקטיבית יותר. לאחר ש-Node מסוים נועל כסף ל-penalty, ניתן לתקוף את המכשיר ממנו ה-Node פועל (או את מערכת התקשורת בה הוא משתמש) כדי שלא יוכל להגיב בזמן, ויאלץ לשלם את ה-penalty. בנוסף, ישנו סיכון שמשתמשי הרשת צריכים לקחת בחשבון, אם ה-Node נופל באמצע מעבר של טרנזקציה, הוא עלול לשלם penalty גבוה.

העבודה שלנו

מבוא

בחרנו לממממש סימולציה של רשת Lightning עם פרוטוקול HTLC המוטמע כרגע ועם הפרוטוקול HTLC-GP החדש ולבחון את השפעות השינוי.
תחילה רצינו להתמקד באפקטיביות מתקפת ה-Reverse Griefing על פרוטוקול HTLC-GP בגירסתו הראשונה, אבל לאחר העדכון של המאמר, הפרוטוקול פותר את הבעיה (או לפחות

הופך אותה לפחות אקרקטיבית) ולכן העדפנו להתמקד בפרוטוקול החדש וביעילותו.
נראה את האפקטיביות של המתקפות Soft Griefing ו-Dos attack ואת השפעות השינוי על עומס הרשת, כמות הכסף שנגעל ומציאת מסלולי תשלום.

מימוש

בחרנו לממש את הפרוטוקול על סימולציה של רשת Lightning שאנחנו כתבנו בשפת Python [6]. בחרנו לא להשתמש בספריות קיימות משום שהן כתובות בשפות שאנחנו לא בקיעים בהן, מסובכות וגדולות מדי לצרכים שלנו. בנוסף, הסימולציות הקיימות לא מאפשרות פתיחת ערוץ בו שני Nodes מפקידים לתוכו כסף, כאשר פרוטוקול זה מחייב זאת (אחרת בהעברה הראשונה לא נוכל לנעול את הכספים עבור העונש).
בסימולציה שלנו אנחנו עושים הרבה הנחות - למשל שה-Nodes מוודאים נכונה שהחוזה שקיבלו תקין או שימוש בבלוקצ'יין "מזויף" שפשוט מספק נתונים הנחוצים לריצת הסימולציה. זאת מכיוון שאנחנו לא באים לבדוק את הבלוקצ'יין או מימוש ספציפי של Node, אלא רק את המקרים שהפרוטוקול משפיע עליהם ישירות ברשת. כלומר מימשנו התנהגות רגילה של הפרוטוקול ברשת ללא הקריפטוגרפיה ואימותה.

בחרנו לבדוק את התמודדות הפרוטוקול החדש עם עיכובים ברשת לעומת הפרוטוקול הקיים ואת התמודדותו עם המתקפות Dos attack, soft griefing שהצגנו קודם.
בפרויקט 3 חלקים מרכזיים:

- מידול רשת ה-Lightning על גבי Blockchain, כולל מידול Node הוגן ו-Node עבור כל מתקפה שרצינו לבחון.
- סימולציה שבונה רשת בעזרת המידול עם פרמטרים שונים אותם נרצה לבדוק, מריצה טרנזאקציות על גבי הרשת, אוספת נתונים על הריצה ושומרת אותם.
- עמוד UI בו ניתן לראות את נתוני הסימולציות וגרפים רלוונטים.

מידול רשת ה-Lightning:

רוב הלוגיקה הקשורה לפרוטוקול נמצאת באובייקט LightningNode. התקשורת בין האובייקטים השונים נעשית ע"י קריאה לפונקציות של כל אובייקט. למרות הנחות הנכונות על הפרמטרים שעוברים לכל פונקציה, הוספנו הרבה Assert על מנת לוודא זאת במהלך הרצת הסימולציות.
המחלקות העיקריות הן:
Blockchain: מייצג Blockchain.
LightningNode: מייצגת Node ברשת lightning.
LightningNodeSoftGriefing: מייצגת Node רשע שעושה Soft Griefing.
LightningNodeDosAttack: מייצגת Node רשע שעושה Dos Attack.
Contract_HTLC: מחלקה אבסטרקטית מייצגת חוזה HTLC רגיל (כמעט).
ContractForward: חוזה Forward כפי שהצגנו למעלה.
ContractCancellation: חוזה Cancellation כפי שהצגנו למעלה.
Channel: מייצג ערוץ פתוח ומבצע את כל הפעולות הקשורות לניהול ערוץ הנפתח ע"י שני Nodes.
Network: מייצגת את רשת הLightning בסימולציה שלנו.
MetricsCollector: אובייקט סינגלטון שאוסף נתונים על ריצת הסימולציה כדי להציגם כגרף.
FunctionCollector: אובייקט סינגלטון שאוסף פונקציות ואת מספר הבלוק שבו הן צריכות לרוץ - היינו צריכים בעצם לאפשר ל-Node "להתעכב" בשליחת הודעה בלי מיקבול, ובכל זאת לבצע פעולות אחרות בנתיים. עשינו זאת ע"י אובייקט זה שאליו שולחים פונקציות שנרצה להריץ בזמן (מספר בלוק) מאוחר יותר - יוסבר בהרחבה בהמשך.

סימולציה:

הקוד של הסימולציה נמצא בקובץ simulation.py. מהלך הסימולציה הוא יצירת רשת ב-2 טופולוגיות אפשריות (נפרט בהמשך) של Node-ים עם ה-channel-ים בניהם - לקראת הסוף הבנו שמשום מגבלות בזמן לא נוכל להריץ באופן מלא את הטופולוגיה השנייה, אך היא כן קיימת ונפרט על זה בהמשך), ניסינו לבחור פרמטרים לריצה כמו ה-capacity באותו channel, ה-fee (תשלום בסיס ואחוז מהסכום שעובר)

והכמות כסף ש-Node בוחר לשלוח, נבחרו בהתפלגות שדומה לרשת Lightning האמיתית (אחרי ניתוח של snapshot של הרשת) או שהם פרמטרים שאנחנו רוצים לבדוק על גבי מספר אפשרויות - גילינו ששונות

גבוהה מקשה עלינו לבודד משתנים, אז החלטנו להוריד את הרנדומליות שבבחירת הפרמטרים (עדיין ישנה רנדומליות בקוד, כמו בחירת ה-Nodes ששולחים כסף בריצה מסויימת).

בנוסף, ישנם מספר פרמטרים שקבענו באופן שרירותי, הסימולציה רצה במשך $15 \cdot 144$ בלוקים (15 ימים) ועם פרמטר Griefing Penalty של 0.001 כפי שנקבע והורץ במאמר.

בכל בלוק נבחר Node אקראי לשליחת כסף, Node אקראי לקבלת הכסף, ואת הכמות שנרצה להעביר, ונחפש מסלול שמסוגל להעביר את התשלום ועם תשלום fee מינימלי בעזרת אלגוריתם דומה ל-Dijkstra (שלמעשה מחפש מסלול מהמקבל לשולח, בעיקר בגלל הצורך לבדוק שיש מספיק capacity לנעול את ה-Griefing penalty), מכיוון שהתקבענו על fee אחיד, כרגע האלגוריתם מוצא את המסלול הקצר ביותר, אך מסוגל להתחשב גם ב-fee משתנה במידה ונרצה לשנות זאת. נעלה את מספר הבלוק, ונריץ את הפונקציות הרלוונטיות ב-Function Collector (נסביר בהמשך).

אם מדובר על ריצה עם Node-ים עוינים, ניצור את אותם Node-ים, ואת הקורבנות הרלוונטים למתקפה. נדאג שהתוקפים והקורבנות לא יבחרו לשליחה או קבלה ישירה של כסף (באיטרציה שהוסברה קודם). כך שהאופציה היחידה לקבלת תשלום היא בעזרת fee מתשלומים שעוברים דרכם.

בכל שלב הריצה נאסוף מטריקות באובייקט Metrics Collector שנפרט עליו בהמשך.

נפרט על כל סוגי הריצות שהרצנו, הפרמטרים שבדקנו ובאילו אפשרויות בחרנו:

- <u>כמה התקשורת ברשת עמוסה</u>: עבור פונקציות מסוימות במהלך טרנזאקציה בין Node-ים יצרנו עיכוב יזום שמשכו מתפלג באופן אחיד בין 1 לפרמטר מסוים אותו אנחנו משנים בכל ריצה (כדי לבדוק את השפעות התקשורת ברשת), כך שעבור כל פונקציה נדגום באופן אחיד כמה בלוקים נתעכב בשליחת ההודעה הבאה. בדקנו ערכי הקצה לעיכוב הבאים: 2, 6 ו-10. עבור ריצות אלו, לא הרצנו עם המתקפות, אלא הנחנו שהרשת מתפקדת היטב. כל ריצה הרצנו עם פרוטוקול HTLC הרגיל ועם פרוטוקול HTLC-GP. הפרמטר הדיפולטי עבור שאר הריצות הוא 6.
- <u>איזה סוג תקיפה</u>: עבור כל סוגי התקיפות נריץ את אותה הרשת בידיוק, עם אותה בחירה של שולחים ומקבלים (כלומר כל בחירה רנדומלית שמתבצעת תשוכפל), כאשר הריצות הן עם פרוטוקול HTLC-GP ובריצה הראשונה נבצע את המתקפה ובשנייה נרוץ ללא ביצוע המתקפה. כמו כן, נבחר את הקורבן והתוקף באופן רנדומלי (באופן זהה עבור הריצות). נבדוק את סוגי התקיפות הבאות -
  - תקיפה של קורבן אחד ע"י Node מסוג Dos attack. כפי שהוסבר בחלק הרלוונטי, ה-Node שולח בכל מספר קבוע של בלוקים הודעה על תשלום לקורבן ומתעלם מחוזה ה-Cancellation.
  - תקיפה של קורבן אחד ע"י זוג אחד או יותר Node-ים מסוג Soft Griefing. במתקפה זו נגריל את התוקף הראשון והקורבן כפי שנעשה בכל שאר המתקפות, אך ניצור את התוקף השני כך שיש לו channel רק עם הקורבן, עם capacity יחסית גבוה. כעת בכל מספר קבוע של בלוקים נשלח לכסף מהתוקף הראשון לשני. כמובן שהתשלום יעבור דרך הקורבן (הוא הקשר היחיד של התוקף השני לרשת), ולאחר העברת החוזים הרלוונטים, התוקף השני יחכה לזמן מינימלי הנדרש כדי שהתשלום יתבצע. ה"טריק" שבו אנחנו מחברים את התוקף השני לקורבן ישירות לא הכרחי בעולם האמיתי, שם אפשר פשוט לבחור לעבור דרך הקורבן בכוונה (בסימולציה לא רצינו לסבך את מציאת המסלול עם האופציה הזו).
  - תקיפה של הרשת כולה ע"י זוג אחד או יותר Node-ים מסוג Soft Griefing. במתקפה זו נגריל את התוקפים באופן רנדומלי, וכל מספר קבוע של בלוקים נבצע תשלום בין התוקפים כך שנחסום את המספר המקסימלי של כספים עבור Node-ים בדרך ובעצם נעמיס על הרשת.

  * ניזכר כי שתי המתקפות האחרונות הן בעצם יישום של מתקפות griefing רגילות, רק בלי לבצע griefing קלאסי.
- <u>פרמטר δ</u>: הפרמטר קובע כמה זמן מקבל הכסף יחכה לקבלת ה-Forward Contract לפני שהוא יבטל את ה-Cancellation Contract שלו. נרוץ עם הפרמטר 40, 70 ו-100. עבור כל פרמטר נריץ את המתקפה Dos attack כפי שפירטנו בסעיף הקודם. בצורה זו נוכל לבדוק כמה משפיע פרמטר זה על האפשרות לבצע את המתקפה. הפרמטר הדיפולטי עבור שאר הריצות הוא 70.
- <u>טופולוגית הרשת</u>: כל אחת מהריצות שעליהן פירטנו נריץ בשתי הטופולוגיות הבאות:

- רשת בעלת יתירות. עבור רשת בעלת n Node-ים, ניצור channel בין שניים אם המספר הסידורי שלהם שונה ב-$10^i \mod (n)$ עבור $i \in [0, \lfloor \log_{10} n \rfloor + 1]$. כך ניצור רשת שלכל קודקוד יש 6 channel-ים והמרחק בין כל שני קודקודים לא גדול מידי והמסלולים בינהם לא יהיו ארוכים יתר על המידה. מקור :[4].
- רשת אמיתית. נעזרנו בקוד שצורף למאמר המעודכן ויצרנו רשת לפי snapshot של הרשת מהתאריך 21/5/2020, כולל הפרטים על ה-capacity ו-fee של כל channel.

כפי שהסברנו למעלה, בחרנו שכרגע לא להריץ את הטופולוגיה הזו, זאת מכיוון שהיא בעלת המון Nodes ומספר הטרנזאקציות שאנו שולחים כדי להריץ אותה בזמן סביר לא מתאפשר למדוד את השפעות המתקפות.

לא נפרט את כל סוגי המטריקות שאספנו כי יש הרבה (ניתן לראות את כולן בעמוד ה-UI עליו נפרט בפסקה הבאה), אך הם מאפשרים לנו לדעת מה מצב הרשת בכל שלב שנרצה ובסוף כל ריצה הן נכתבות לקובץ json.

עמוד UI :

יצרנו עמוד web שכתוב ב-javascript ומשתמש בסיפריית chartjs [7] ומציג את תוצאות הריצות בצורה נוחה המאפשרת לבחון קורולציה בין נתונים ואת השפעות המתקפות שבחרנו לבדוק. בעזרת script העברנו את קבצי ה-json של כל ריצה ל-json עימו נוח יותר לשלוף נתונים. הscript עושה ממוצע של המטריקות על ריצות עם אותם הפרמטרים.
בעמוד ניתן לבחור פרמטר אינפוט לריצה (עומס הרשת, פרמטר δ ומתקפות) בתור ציר X ובנוסף מטריקה שנאספה במהלך הריצות בתור ציר Y (לחלק מהמטריקות אין נתונים משום שלא כל הנתונים נאספים עבור כל סוגי הריצות). עבור עומס הרשת נוכל לראות בגרף את הקורלציה בניהם עבור ריצה עם פרוטוקול HTLC וריצה עם פרוטוקול HTLC-GP. עבור פרמטר δ נראה את הקורלציה בניהם עבור ריצה עם המתקפה ובלי. ועבור המתקפות נראה עמודה שמשווה את המטריקה הנבחרת בין הריצות עם המתקפה ובלי. בנוסף, נוכל לבחור את המתקפה הרלוונטית ואת טופולוגיה שהסימולציה רצה עליה.
צירפנו טבלאות עם תוצאות שאר המטריקות מתחת לגרפים, כך שאפשר לבחון אותם גם.
נרחיב על החיבור בין מידול הרשת לסימולציה. בחרנו להריץ את הסימולציה על thread יחיד כדי לפשט את הכתיבה והדיבוג. סימלצנו את התקשורת הא-סנכרונית בין ה-Node-ים ע"י כך שבמקרה הצורך כל Node רושם פונקציה שהוא מעוניין להריץ ומתי צריך להריץ אותה (לפי הגעה למספר בלוק מסוים) לסינגלטון מסוג Function Collector, כך שבכל איטרציה הסימולציה מעלה את מספר הבלוק ומריצה את כל הפונקציות שה-Collectior אסף ושצריך (כלומר מספר הבלוק הרלוונטי הגיע). כך, יכולנו לסמלץ עיכובים, נפילות של Node-ים או לבצע מתקפות. כלומר Node או הסימולציה לא מסתמכים כלל על ערך החזרה של פונקציות של מחלקת ה-Node.
כמו כן, כדי לאסוף נתונים על הריצה, השתמשנו בסינגלטון מסוג Metrics Collector וכך רשמנו נתונים מכל מקום שהיה צריך (נאסוף ממוצע, סכום, ספירה וכו').

איך להריץ את הקוד

על מנת להריץ את הקוד, צריך להתקין את הסיפריות fire [8] ואת סיפריית networkx [9] ולהריץ את הפקודה :

```
python3 simulation.py run_all
```

תוצאות הריצה יכתבו בתיקייה simulation_results לקובץ עם בשם שמכיל את תאריך הריצה, כאשר כל ריצה בשורת json משלה. ריצות שנרצה להציג ב-UI נעביר לתיקייה GoodRuns ונריץ את :

```
python3 simulation_result_parser.py main
```

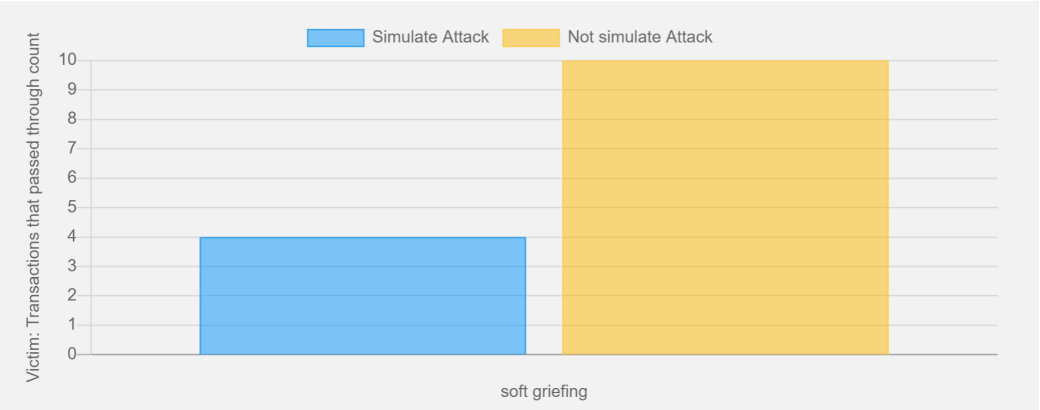
שיוצר קובץ runs.js בתיקייה simulation_gui. בתיקייה זו נמצא graph.html, אותו ניתן לפתוח אותו ולראות את תוצאות הריצות בגרפים וטבלאות. ניתן להוסיף מספר ריצות עם אותם הפרמטרים ולקבל ממוצע של כל מטריקה.

תוצאות

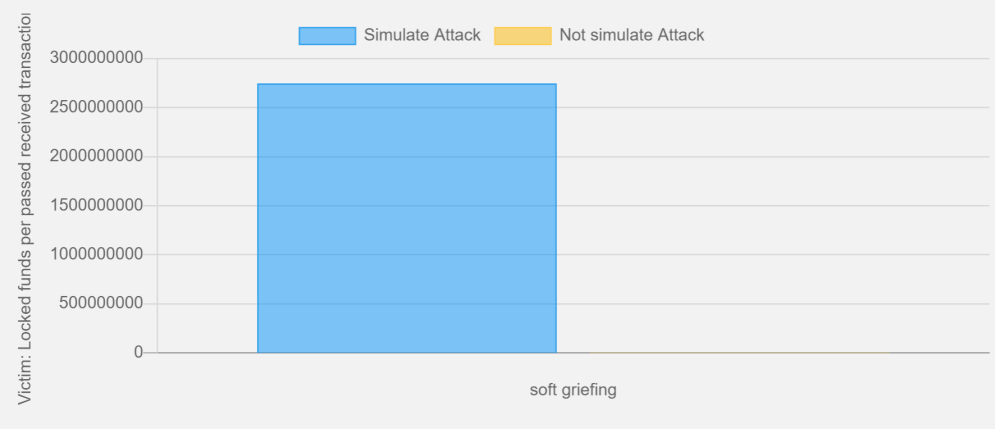
אנחנו מסתמכים בתוצאות על המטריקות שאספנו לאורך הריצה. את כל שמות המטריקות ניתן לראות ב-singleton.py. כאשר מדובר על מטריקה שנאספת מתוך Node עצמו, נוסיף

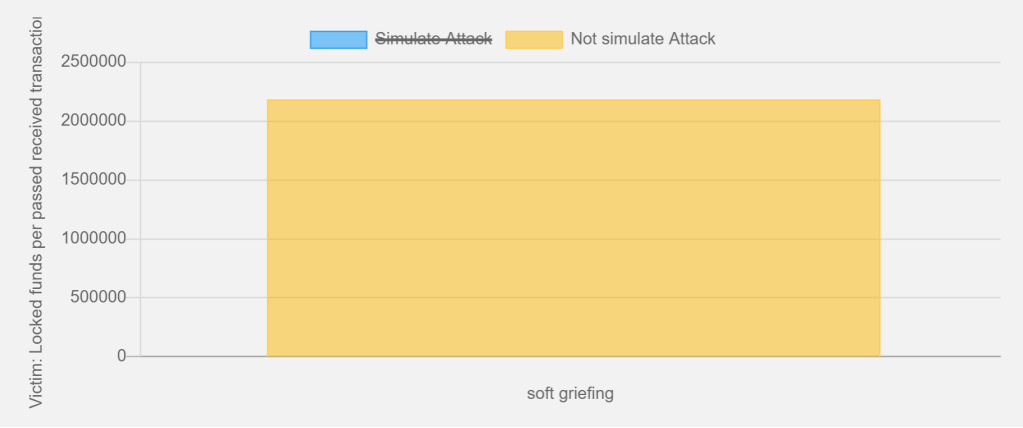
תחילית של Victim: או Attacker: בהתאם לסוג ה-Node כך נוכל לזהות מה קרה לקורבן, כמו כן, במקומות הרלוונטים, לא אספנו מטריקות שנגרמו כתוצאה מההתקפה. למשל, לא החשבנו את הטרנזקציות של תוקף Soft griefing שעברו דרך הקורבן. נראה כעת תוצאות מהגרפים שנמצאים ב-UI שבנינו.

עבור מתקפת Soft Griefing:



בגרף זה ניתן לראות כי מספר הטרנזאקציות העוברות דרך Node (ספרנו את אלו שלא קשורות למתקפה) כאשר הוא מותקף בעזרת Soft-Griefing היא משמעותית פחותה מאשר ללא המתקפה. כלומר, בעזרת מתקפה זו, הצלחנו לגרום ל-Node לקבל פחות טרנזאקציות, כלומר הוא ירוויח פחות.

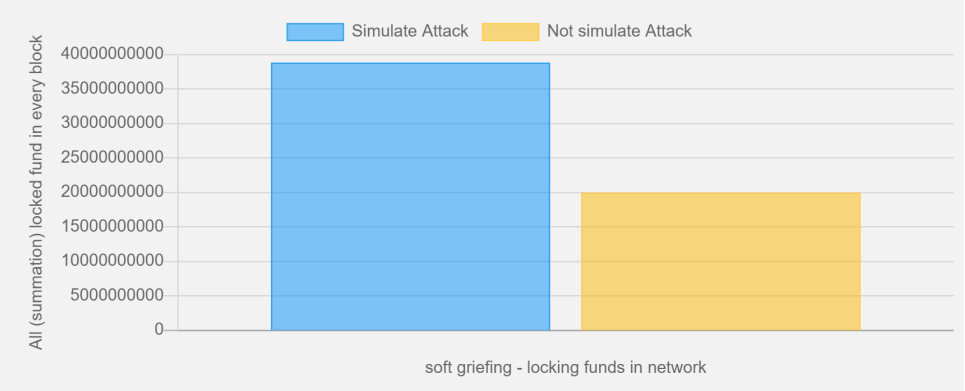




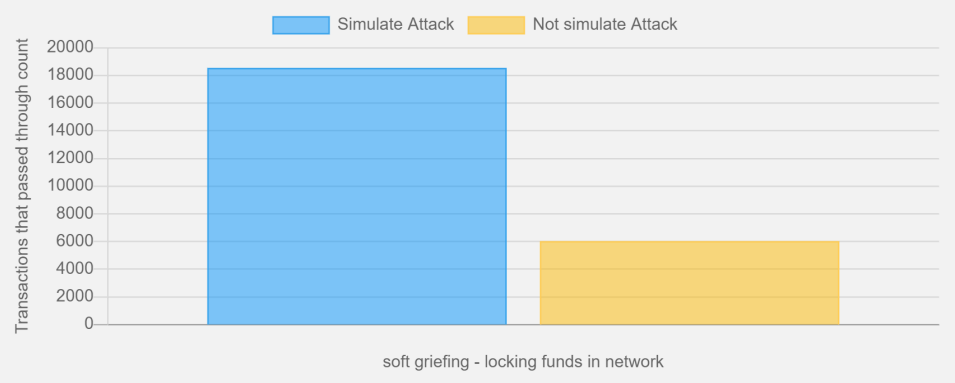
בגרפים אילו אפשר לראות את כמות הכספים הנעולים ביחס לטרנזאקציות שעברו ב-Node, עם ובלי מתקפה. גם כאן אפשר לראות שהרבה יותר כספים ננעלו (ללא התחשבות בערוץ

עם המתקיף) - כלומר הצלחנו לגרום ל-Node המותקף לנעול משמעותית יותר כספים ועל כן בעצם לחסום לו הרבה מהטרנזאקציות שיכל לקבל (אכן קיבל פחות טרנזאקציות כמו שראינו בגרף הקודם). כלומר, בדומה ל-Griefing קלאסי, ניתן להשתמש במתקפה זו כדי לתקוף Hub ובכך להסית טרנזאקציות (ובכך רווחים) אל התוקף כמו ב[5].

עבור מתקפת Lock liquidity in network:

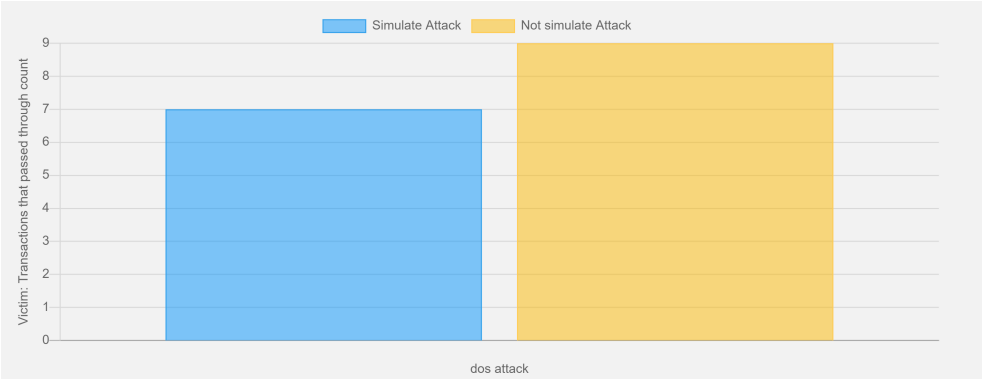


בגרף זה ניתן לראות כי אכן הצלחנו לנעול הרבה יותר כסף בעזרת מתקפה זו ובעצם לשחזר את ה"הצלחה" שראינו במאמר של אביב זוהר עבור נעילת כספים ברשת [5]. כלומר - גם מתקפה זו ניתנת למימוש, גם אם באופן מעט שונה, גם עם הפרוטוקול החדש.

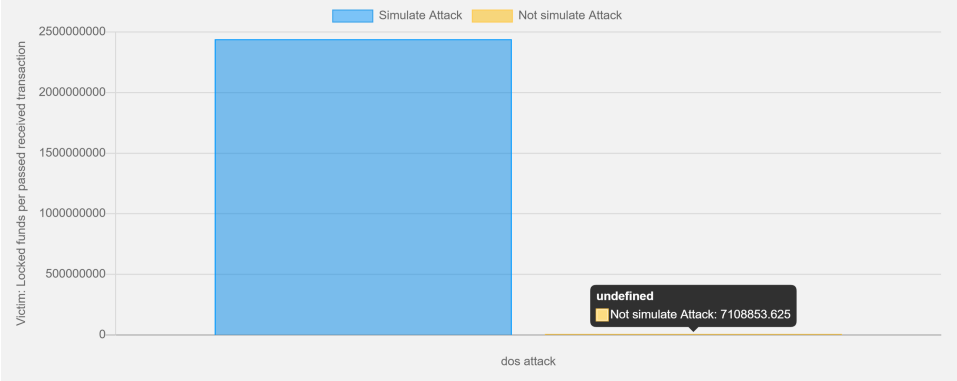


בגרף זה ניתן לראות כי עם המתקפה מתבצעות הרבה יותר טרנזאקציות, זאת מכיוון שאנחנו שולחים המון טרנזאקציות בין התוקפים והפרדה שלהן מבחינת איסוף המידע היה דורש שינוי תשתיתי שלא היה לנו זמן לממש, בנוסף לזה, הרבה מטריקות מנורמלות לפי ערך זה ולכן בעיתיות עבור המתקפה הספציפית הזו (בשאר המתקפות אנחנו כן מבדילים בין טרנזאקציות של המתקפה לבין "רגילות"). אך מה שבאמת חשוב הוא כמות הכספים שנגעלו ואכן הצלחנו לנעול כמו ניכרת של כספים.

עבור מתקפת Dos:

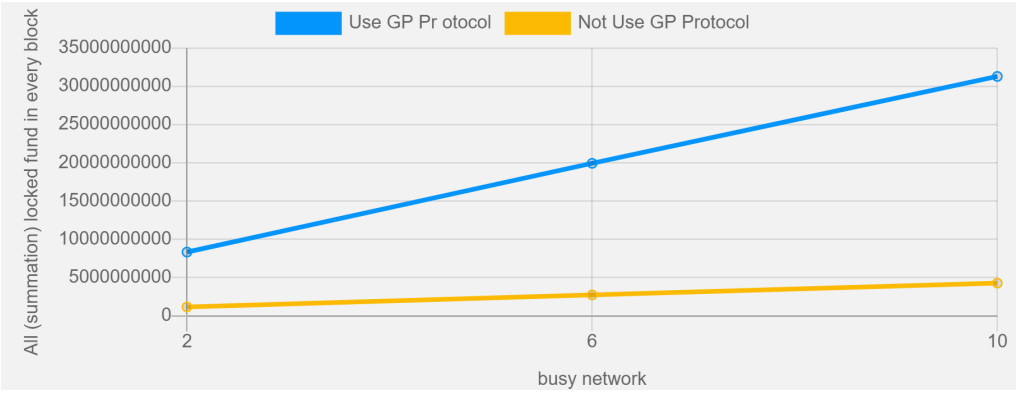


בגרף זה ניתן לראות בדומה למתקפה המקורית כי המותקף מקבל פחות טרנזאקציות ועל כן מרוויח פחות.

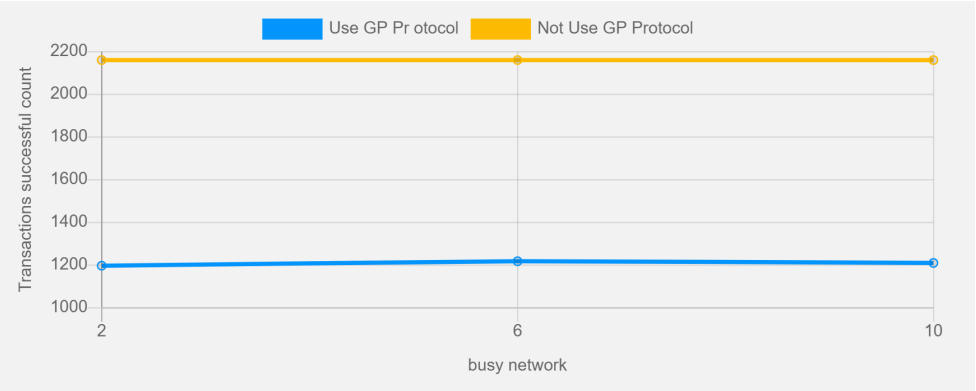


פה רואים כי כאשר ה-Node מותקף הוא נועל הרבה יותר כספים - כלומר יש לנו פה תוצאות דומות ל-Soft Griefing.
ניזכר כי במתקפה זו התוקף אינו נועל כספים! אלא רק שולח טרנזאקציות למותקף ומסרב להיענות לבקשה של לאשר את חוזה ה-Cancellation - אך נזכור כי כאן המותקף יבטל את הטרנזאקציה כתלות ב-δ שהזכרנו למעלה. אך מכיוון שאנחנו לא נועלים פה כסף, ניתן לבצע טרנזאקציות מרובות.
בהמשך נחקור את המתקפה הזו יותק לעומק.

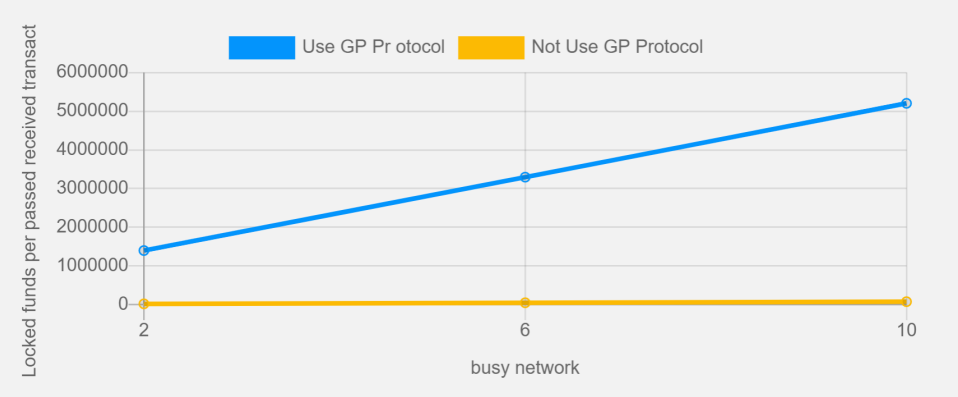
<u>השפעת פרמטר ה-Busy Network על הרשת:</u>
ניזכר כי כאשר אנחנו פותרים טרנזאקציות בסימולציה שלנו, אנחנו מגרילים עבור כל Node את מספר הבלוקים שיחכה לפני שיודיע לNode הבא על פתירת הטרנזאקציה מ-1 עד פרמטר שלו נקרא Busy Network. מאפיין של הפרוטוקול הוא שהוא דורש העברת הודעות גדולה יותר בין המשתתפים בטרנזקציה, ולכן עיכוב של Node-ים, בין אם בגלל בעיות תקשורת או בעיות זמינות, עלול להשפיע על הרשת. התבוננו על איך פרמטר זה משפיע, כאשר המחשבה הייתה שבעיות בתקשורת ייגרמו לנעילת כספים משמעותית הרבה יותר כאשר נשתמש בפרוטוקול החדש.



אכן ניתן לראות שלא רק שאנחנו נועלים הרבה יותר כספים עם הפרוטוקול החדש, אלא הכמות גדלה בקצב הרבה יותר מהיר מאשר עם הפרוטוקול הרגיל.

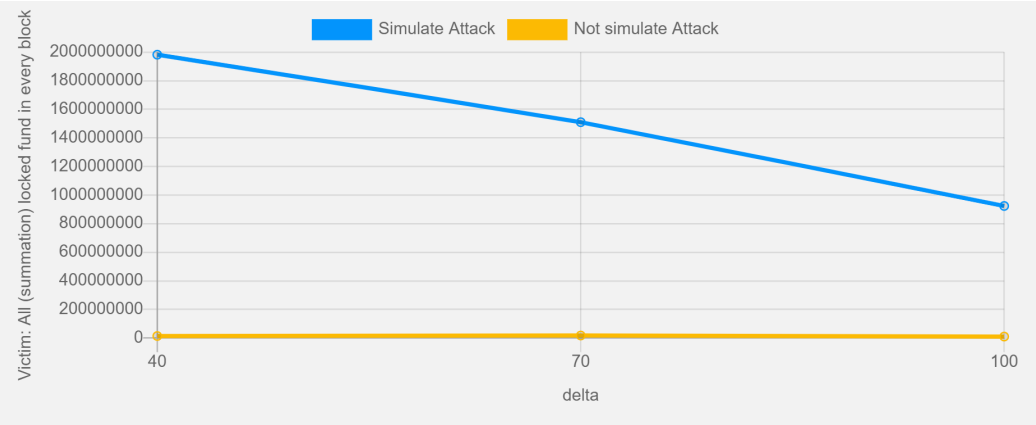


בגרף זה רואים כי בפרוטוקול HTLC-GP אנחנו מצליחים פחות טרנזקציות מאשר בפרוטוקול המקורי. השפעת פרמטר ה-Busy network לא מוחלטת ואף שונה ממה שציפינו, אבל את ההבדלים בין הפרוטוקולים ניתן לראות בבירור. הדבר מדגיש את ההבדל שראינו בגרף הקודם, משום שעל אף שפחות טרנזקציות הצליחו, נעלנו בסכ"ה יותר כספים. נוכל להסתכל גם על כמות הנעילה ביחס לטרנזקציה שעברה ב-Node בהצלחה:



ופה ההבדלים אף גדולים יותר.

<u>השפעת פרמטר δ על מתקפת Dos</u>:



כאן אנחנו מראים את ההשפעה של δ על כמות הכספים שנעלים אצל הקורבן. ניתן לראות עלייה שמתאזנת בין 70 ל-100. כלומר נראה ש-δ משפיעה על האפקטיביות של המתקפה אבל לא באופן חד משמעי.

בניגוד למה שהיינו מצפים, נראה שעברו יותר טרנזאקציות דרך הקורבן כאשר ה-δ גדלה. במקרה זה התוצאות לא אידיאליות, אך עם זאת שהראינו כי אכן המתקפה יעילה, גם עבור δ נמוכה, נקבל כי לדעתנו יש לגלות לגביה עניין רב בהקשר של מימוש פרוטוקול זה.
בנוסף, נציין שלא הפעלנו אסטרטגיה מורכבת מצד ה-Node המתקיף באשר למתי לבצע את המתקפה. אנו סבורים שניתן לחקור יותר את הנושא ואולי לנצל בצורה טובה יותר δ גבוה.

מה לא הלך כמתוכנן

הפרויקט שינה כיוון מספר רב של פעמים, בין אם בגלל עידכון המאמר ובין אם בגלל שינויים בדרכים שרצינו לבדוק את השאלות שהעלנו. השתדלנו לחשוב על הפרטים ולבנות את הסימולציה כך שיהיה קל לבצע שינויים, אך עדיין אנחנו לא מרוצים לגמרי מהדרך שהקוד שלנו בנוי ויש מספר דברים שהיינו משנים אם היינו מתחילים מחדש, או שהיה לנו עוד זמן.
הרבה מהפרמטרים שאנחנו בודקים הם סטטים. בהתחלה ניסינו לדגום מהתפלגות שדומה להתפלגות של snapshot של רשת Lightning אבל זה יצר שונות גדולה בין הריצות, ולכן היינו רוצים להריץ הרבה ריצות כדי להוריד את השונות. לצערנו לא היה לנו הרבה זמן, ולכן נאלצנו לוותר על כך. בנוסף, וויתרנו גם על הריצה עם רשת שנוצרה מ snapshot של רשת Lightning בעיקר בגלל חוסר בזמן.

Future Work

- בסימולציה שבנינו ניסינו להעביר טרנזקציה דרך מסלול אחד, ואם לא הצלחנו לא ניסינו לפצל אותה לתת תשלומים דרך מסלולים שונים. אנחנו חושבים שיש ערך לנסות לבדוק את הסימולציה בעזרת מנגנון של פיצול טרנזקציות, מאחר וזה מה שמפרסמי המאמר הציעו כששאלנו אותם על הנושא. מעניין לראות את השפעת הפיצול על ה-fee שנצטרך לשלם. מצד אחד, יכול להיות שנגלה מסלולים עם fee נמוך שלא יכולנו להשתמש בגלל capacity נמוך, אך מצד שני, כנראה שנצטרך לשלם יותר פעמים את ה-base fee וכך להסתכם בלשלם יותר. בנוסף, פיתרון זה עשוי לעזור כאשר לא מוצאים מסלולים.
- האופן שבו בנינו את סימולציה והאובייקטים מאפשר להמשיך לחקור את HTLC-GP, ואף פרוטוקולים אחרים עם שינוים מסוימים בקוד. ניתן ליצור Node-ים אחרים שמתנהגים באופן שונה ממה שהגדרנו ולבחון את ההתנהגות שלהם בסימולציה. בנוסף, ניתן להוסיף מטריקות שאנחנו לא חשבנו עליהם שיתווספו ל-UI שבנינו באופן אוטומטי וכך לחקור אספקטים שונים ממה שאנחנו בחרנו. כמו כן, אפשר לבחון מספר גדול יותר של פרמטרים ממה שאנחנו בחרנו, כמו למשל Griefing Penalty.
- נרצה לתרגם את העבודה לאנגלית (החלטנו לכתוב בעברית בגלל חוסר בזמן) על מנת שיגיע לתפוצה רחבה יותר (כולל כותבי המאמר).

סיכום

כפי שהוצג בתוצאות, פרוטוקול HTLC-GP דורש נעילה גדולה של כספים ביחס לכסף הנשלח, מה שעשוי להעמיס על הרשת, לגרום לתשלום fee גבוה או אי מציאת מסלול. בנוסף, ישנם השלכות אינדיבידואליות נוספות לכסף נעול של Node כמו אי-יכולת לשלם או להשקיע את הכסף, או אי יכולת לקבל כסף מאדם אחר (אין מספיק כסף לנעול).
הפרוטוקול גורם ל-Node-ים ברשת לסכן הרבה מהכספים שלהם כדי לבצע תשלומים רגילים עבור משתמשים אחרים, ואולי להרוויח מה-fee. כך שבשילוב עם המתקפות שהראנו, Node-ים עשויים לא לרצות להעביר תשלומים או שיבחרו להעלות את ה-fee שלהם על מנת שהסיכון ישתלם להם.
בנוסף, האפקטיביות של המתקפות שהראנו עשויה לגרום ל-Node-ים תמימים לאבד כספים, או לא להיות מסוגלים לתפקד. ובמקרה של Dos attack, אפילו ללא השקעה ראשונית מצד המתקיף (מלבד channel קיים).
מתקפת ה-Soft-Griefing דומה מאוד למתקפת Griefing רגילה וכמו שהראינו - גם אפקטיבית. בנוסף, מתקפה חדשה שהצגנו ומגרעות נוספות הופכות את הפרוטוקול, לפחות כרגע, לפי דעתנו ללא משתלם למימוש. עם זאת, נציין כי הפרוטוקול אלגנטי ולדעתנו עם שיפורים

נוספים אשר ימזערו את הדברים שהצגנו כאן (אולי בעזרת פיצ'רים עתידיים של לייטנינג, כמו חתימות מסוג שונה) יאפשרו מימוש שלו ומיזעור מתקפת Griefing.

מקורות

[1] https://arxiv.org/pdf/2005.09327.pdf
[2] https://arxiv.org/pdf/2005.09327v1.pdf
[3] https://www.mail-archive.com/lightning-dev@lists.linuxfoundation.org/msg01983.html
[4] https://hackernoon.com/simulating-a-decentralized-lightning-network-with-10-million-users-9a8b5930fa7a
[5] https://arxiv.org/pdf/2002.06564.pdf
[6] https://github.com/avivbaru/cryptocurrencies_final_project
[7] https://www.chartjs.org/
[8] https://github.com/google/python-fire
[9] https://networkx.github.io/