digital whisper
Technologic Papers

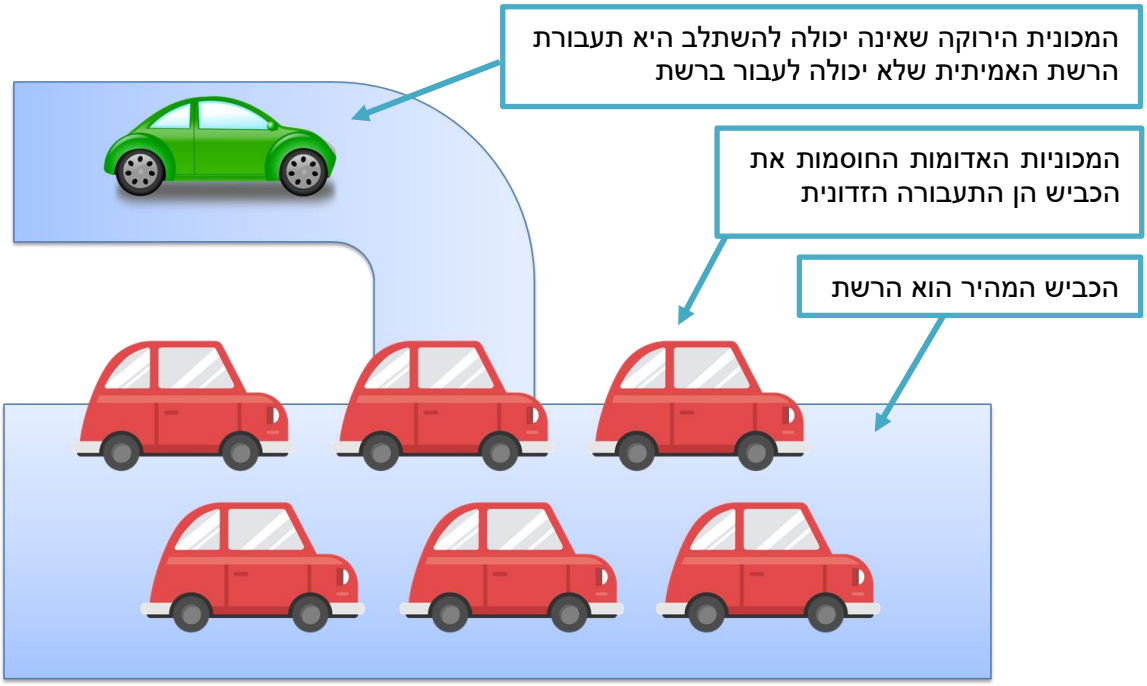
UDP and TCP DDoS Attacks

מאת תבור לביא ואלון טל זילברמן

הקדמה - מה זה התקפת DDoS

התקפת DDoS (התקפת Distributed Denial of Service) היא התקפה שמטרתה היא לשבש את יכולת התעבורה של רשת, שירות, או שרת על ידי הצפה של היעד עם תעבורה(כמו בקשות מידע או הודעות). מה שמייחד DDoS זה השימוש במספר מרובה של מחשבים פגועים כדי לשלוח את התעבורה הזדונית ממספר מקורות.

מטאפורה טובה ל-DDoS הוא פקק עצום בכביש מהיר שמונע ממכוניות להשתלב בתוך הכביש ולא מאפשר תנועה רגילה, במטאפורה הזאת הכביש המהיר הוא הרשת, הפקק הוא התעבורה הזדונית שבאה בכמויות ענק אל הרשת והמכוניות שרוצות להשתלב הן התעבורה הרגילה שאמורה להגיע לרשת ועכשיו לא יכולה בגלל שהשרת מוצף בתעבורה זדונית.



הנה דוגמא בפסודו-קוד להתקפת DDoS מסוג HTTP flood (הסבר רחב יותר על מתקפות אלה בפרק 5):

```
while true:
    s = socket()
    s.connect(target_ip)
    s.sendto(GET target)
    s.sendto(Host: fake_ip)
    s.close()
```

הפסודו-קוד הזה יוצר סוקט, מתחבר לשרת חיצוני דרך ה-IP שלו ואז שולח לו בקשת GET עם Host מזויף כדי להקשות על תהליך מציאת מקור ההתקפה. ואחרי שהוא מסיים את ההתקפה היחידה הוא סוגר את הסוקט ופותח מחדש כדי ליצור חיבורים מרובים לשרת.

בהתקפות flood לרוב יוצרים multi-threads ומבצעים כמה התקפות בו-זמנית כדי שההתקפות לא יבואו כבקשות עוקבות אלא בקשות מקבילות.

התקפות מסוג flood הן ההתקפות הבסיסיות ביותר והן הכי קלות לעצירה מכיוון שלרוב הן באות בכמויות(באופן יחסי לשיטות אחרות) קטנות וישירות יותר עם דרכי ההתקפה לעומת התקפות אחרות שלדוגמא מנצלות מימוש שגוי של שרתים כדי להפוך פקטות קטנות מאוד לפקטות תקיפה עצומות (Amplification).

במאמר זה נעסוק בעיקר ב-Amplification DDoS Attacks, תקיפת Amplification (הגברה) היא סוג של תקיפת DDoS שמנצלת שרת פגיע בכדי להגביר את כמות הפקטות או את גודל הפקטות הנשלחות לקורבן.

ישנם מספר פרוטוקולים שונים שניתן להשתמש בהם עבור התקפות אלו אך DNS amplification attacks הן הסוג הנפוץ ביותר של התקפת Amplification, שכן שרתי DNS בדרך כלל מגיבים לשאילתות עם תגובות גדולות. להרחבה בנושא DNS Amplification Attack ניתן לקרוא את המאמר הבא:
https://www.digitalwhisper.co.il/files/Zines/0x4B/DW75-4-AmplifiedDDoS.pdf

UDP mem-cached amplification

מתקפת memcached amplification היא סוג תקיפה מבוססת-UDP שמשתמשת בשרתי memcached פגיעים. התקיפה מכפילה את כמות המידע שמועבר אל המטרה עד פי 51,000.

Memcached היא מערכת שנועדה לאופטימיזציה של המערכת אשר שומרת מידע בזיכרון ועובדת איתו לפי הצורך. מערכת זו היא Cross-platform (נתמכת ע"י כל מערכות ההפעלה) והיא open-source.

ניתן לתאר את התקיפה כמו ילד שמתקשר למסעדה מזמין את כל התפריט ומבקש שיתקשרו אליו ויפרטו לו מה הוא הזמין אך נותן את מספר הטלפון של מי שהוא רוצה לתקוף.

ב-2018 התרחשה התקפה DDoS על Github, תקיפה זו הייתה בין תקיפות ה-DDoS הגדולות שהתרחשו.
למידע נוסף: https://www.cloudflare.com/learning/ddos/famous-ddos-attacks/

התקיפה בעלת 4 שלבים:

1. התוקף שותל חבילת מידע גדולה מאוד על השרת הפגיע.
2. התוקף שולח בקשת HTTP GET למידע שהוא השתיל עם IP מזויף של המחשב שהוא רוצה לתקוף.
3. השרת הפגיע שמנסה לבצע את עבודתו בתמימות שולח את חבילת המידע העצומה אל המטרה.
4. המטרה מקבלת כמות עצומה של מידע וחייבת לטפל בה בזמן שהיא לוקחת את כל קו הרשת שלה.

המטרה שנפגעה חייבת לטפל במידע המגיע משום שהשרת הפגיע פועל על UDP וב-UDP אין תהליך של תשאול לגבי קבלת המידע.

UDP WS-Discovery amplification attack

התקפת אמפליפיקציה מבוססת WS-Discovery היא סוג התקפה על UDP שמתבססת בפרוטוקול WS-Discovery.

פרוטוקול WS-Discovery הוא פרוטוקול שפותח כשיתוף פעולה בין כמה חברות ביניהן אינטל ומייקרוסופט למטרות נוחות. הפרוטוקול משמש כדרך לזהות מכשירים על הרשת, לדוגמא החל מווינדווס 7 מחשבי ווינדווס משתמשים בפרוטוקול הזה כדי לזהות מדפסות על הרשת של המחשב והפרוטוקול הוא אבן בסיס של כל מערכת השירותים Windows Rally של מייקרוסופט שמקלה על התקנה ותמיכה במכשירי רשת כמו מחשבים, נתבים, מדפסות, מקרנים, קונסולות משחקים, מצלמות ונגנים.

הפקטה של הפרוטוקול נכתבת בשפת XML ופקטה סטנדרטית של זיהוי רשת נראית ככה והיא בגודל של 783 בתים:

```
<?xml version="1.0" encoding="UTF-8"?><s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing"><s:Header><a:Action
s:mustUnderstand="1">http://schemas.xmlsoap.org/ws/2005/04/discovery/Probe</a:Action><a:MessageID>18930069-8
80c-41b3-886c-4ae76fac6b47</a:MessageID><a:ReplyTo><a:Address>http://schemas.xmlsoap.org/ws/2004/08/addressi
ng/role/anonymous</a:Address></a:ReplyTo><a:To
s:mustUnderstand="1">urn:schemas-xmlsoap-org:ws:2005:04:discovery</a:To></s:Header><s:Body><Probe
xmlns="http://schemas.xmlsoap.org/ws/2005/04/discovery"><d:Types
xmlns:d="http://schemas.xmlsoap.org/ws/2005/04/discovery"
xmlns:dp0="http://www.onvif.org/ver10/network/wsdl">dp0:NetworkVideoTransmitter</d:Types></Probe></s:Body></
s:Envelope>
```

[מקור: https://www.akamai.com/blog/security/new-ddos-vector-observed-in-the-wild-wsd-attacks-hitting-35gbps]

אבל לאחר מחקר של חוקרים מ-Akamai נמצא כי הפקטה הזאת מלאה בהמון מלל מיותר ולאחר אופטימיזציה נמצא כי פקטה מינימלית תהיה בגודל 170 בתים ותהיה יותר שימושית בתקיפות שמייעלות את השימוש ברשת.

הדרך הכללית שתהיה משותפת לכל התקיפות הבאות כוללת רעיון בשם IP Spoofing. בעיקרון, כאשר שולחים פקטה אל השרת צריך לכלול את ה-IP השולח כדי שהשרת יוכל להחזיר תגובה ותהיה תקשורת, אך תוקפים יכולים לזייף את ה-IP השולח כדי לגרום לשרת לשלוח תגובות ענקיות בחזרה אל ה-IP המזוייף שיכול להשתייך לאדם שאותו הם רוצים לתקוף.

הדרך הראשונה שניתן לתקוף באמצעותה WS-Discovery היא שליחת פקטה מעוותת שמחזירה שגיאה מהשרת, והגודל המינימלי של פקטה כזאת שניתן לשלוח כדי לקבל שגיאה הוא 18 בתים, תגובת השגיאה היא משמעותית קטנה יותר מתגובות אחרות שניתן להשיג באמצעות WS-Discovery אבל האמפליפיקציה ביחס לגודל הפקטה המקורית עדיין עצומה.

דרך נוספת לתקוף באמצעות WS-Discovery היא שימוש בפרמטר MessageID, פרוטוקל WS-Discover הוא stateLess, כלומר, הפרוטוקול לא משמר פרטי Session ולכן יותר קל לתקוף כמה פעמים בלי צורך לאמת את זהות המכשירים. אבל למרות שהפרוטוקול stateLess, בתוך הפקטה יש פרמטר בשם MessageID שמכיל כמה בתים שאמורים "לזהות" את המכשיר השולח למקרה שבו השרת ירצה לשמור מידע על הסשן, בגלל שהפרוטוקול לכאורה stateLess אבל בפועל נותן את האפשרות לשמור מידע זה גורם לבעייתיות רצינית של חוסר קונסיסטנטיות בשמירת ערך MessageID כפרמטר מזהה ולכן שרתים מסוימים יכולים להיכשל בעיבוד הפקטה או אפילו לקבל overflow מהערך של הפרמטר.

סוג ספציפי של overflow שניתן לבצע משתמש בשיטת off by one buffer overflow, כלומר, שרתי WS-Discovery שמתוכנתים בצורה שגויה יכולים להשתמש במקומות בזכרון שלא מוקצים אל השרתים ובכך ולפגוע בשירותים אחרים.

```
#include <stdio.h>
void function_vulnerable(char *arg)
{
    char buf[128];
    int i;

    for (i=0;i<=128;i++)
        buf[i] = arg[i];
}

int main(int argc, char **argv)
{
    function_vulnerable(argv[1]);
}
```

[מקור: https://csl.com.co/en/off-by-one-explained/]

הנה דוגמא למימוש שגוי שמוביל ל-Off by one Overflow, במהלך הריצה לולאת ה-for פונה למקומות בתוך המערך buf שאורכו 128 ומשנה את ערכו אך בסופה היא פונה למקום ה-129 במערך ומציבה בו ערך, הבעיה היא שאותה כתובת בזיכרון לא שייכת ל-buf אלא לשירות אחר ושינוי ערכה יכול לפגוע בביצוע של תוכנית אחרת.

לכן, אם נמצא שרת עם מימוש לא טוב של אחסון MessageID ניתן לשלוח אליו ערך של MessageID שהשרת לא ציפה אליו ולפלוש למקומות בזיכרון שהשירות המקורי לא אמור לפנות אליהם. במקרים מסוימים חוקרים מצאו כי שליחת פקטה בגודל 257 בתים או יותר תגרום לשרתים האלו להחזיר פרמטר RelatesTo שכולל את ה-257 בתים המקוריים ובנוסף בגלל ה-overflow, ימשיך להחזיר בתים עד שימצא בית סוגר.

חוקרים הצליחו להפוך פקטה בגודל של 451 בתים לפקטת תגובה בגודל 3,445 בתים, אמפליפיקציה של כ-700%.

בנוסף, רוב השרתים ששומרים את MessageID גם מקשרים את ה-IP השולח ל-MessageID ולכן ניתן לשלוח פקטה מקורית עם MessageID ואז לשלוח פקטות מינימליות בגודל של רק 170 בתים ולקבל את אותה תגובה בכל פעם, שיטה זו יכולה להביא את האמפליפיקציה לעד 2,000%.

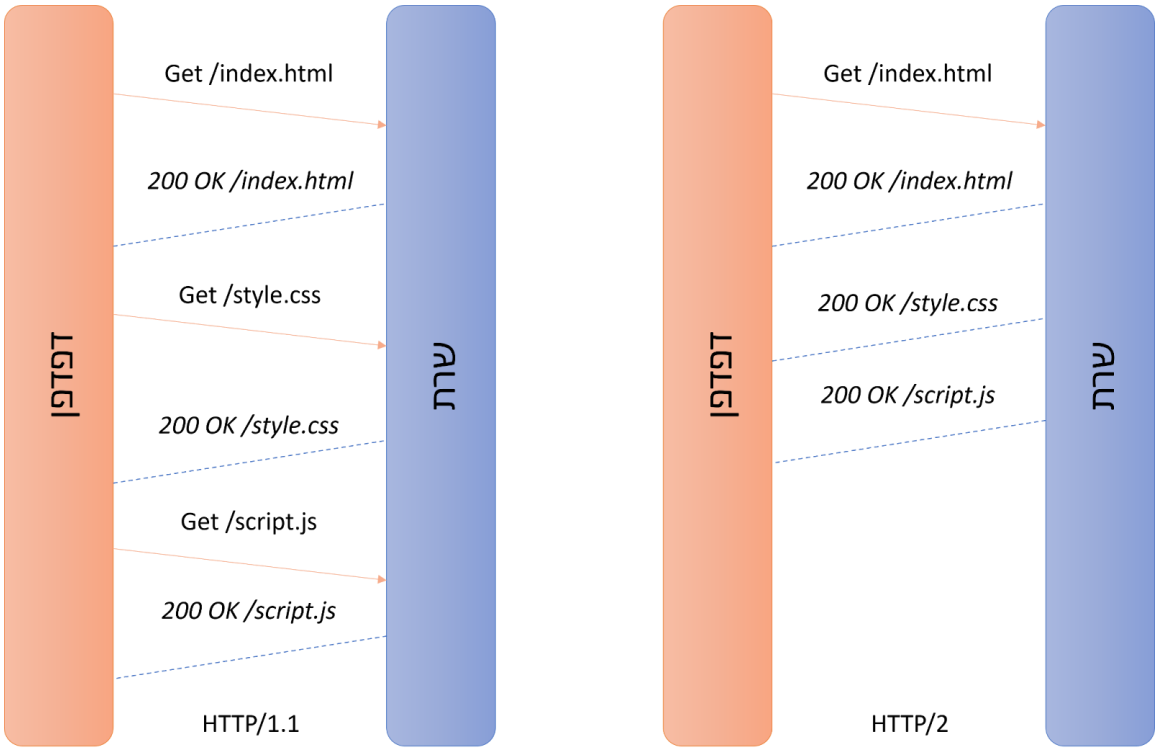
השיטה האופטימלית שנמצאה לתקיפה באמצעות WS-Discovery כוללת שילוב של כל הדרכים האלו, היא שולחת פקטה מקורית עם ערך MessageID שגורם ל-overflow בתגובה ואז משתמשת בעובדה שהשרת זוכר את ה-IP השולח כדי לשלוח פקטות מעוותות ולקבל תגובות שגיאה שכוללות את ה-overflow. נמצא כי פקטה של כ-18 בתים יכולה להוביל לתגובה של כ-2,762 בתים, כלומר אמפליפיקציה של 15,300%.

HTTP/2 Amplification DDoS attacks

התקפת HTTP/2 Proxy Amplification היא תקיפת DDoS שמכוונת לשרתים המתבססת על עדכון בפרוטוקול HTTP המאפשר לשלוח כמה בקשות במקביל באותו חיבור ודחיסה של שדות הכותרת ושליחת כמה בקשות HTTP מבלי לפתוח חיבור TCP חדש. **Hypertext Transfer Protocol** או **HTTP** הוא פרוטוקול תקשורת מעל TCP שנועד להעברת דפי HTML ואובייקטים שהם מכילים (כמו תמונות, קובצי קול, סרטוני פלאש וכו') ברשת האינטרנט וברשתות אינטרנט.

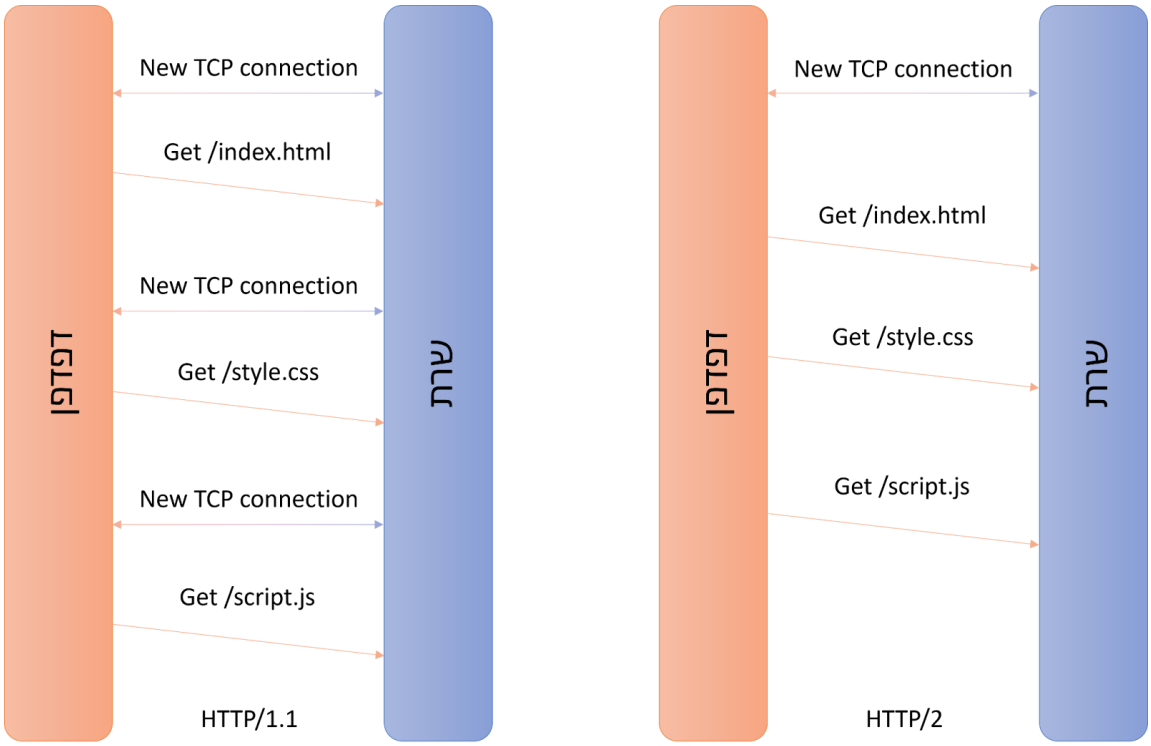
התקיפה מנצלת כמה תכונות של HTTP/2 בכדי לשלוח בקשות קטנות ולקבל חזרה יותר תגובות ושכל תגובה תכיל בתוכה כמות מידע יותר גדולה.

- Server push: הוא פיצ'ר אופציונלי המאפשר לשרת לשלוח ללקוח משאבים או קבצים לפני שהוא מבקש ומבלי שהוא יודע, בכדי לשפר את הביצועים ולהפחית זמן ההמתנה. לדוגמה כשמבקשים אתר אינטרנט בדפדפן, הדפדפן מבקש את דף הבית, "index.html" לאחר מכן הדפדפן צריך גם את הקבצים "style.css" ו-"script.js" והוא ישלח בקשה כדי לקבל אותם. ב-HTTP/2 push אפשר להגדיר שכשהדפדפן מבקש את הדף "index.html" השרת ישלח את הקובץ "index.html" ואז ישר לאחר מכן הוא ישלח את הקבצים "style.css" ו-"script.js":

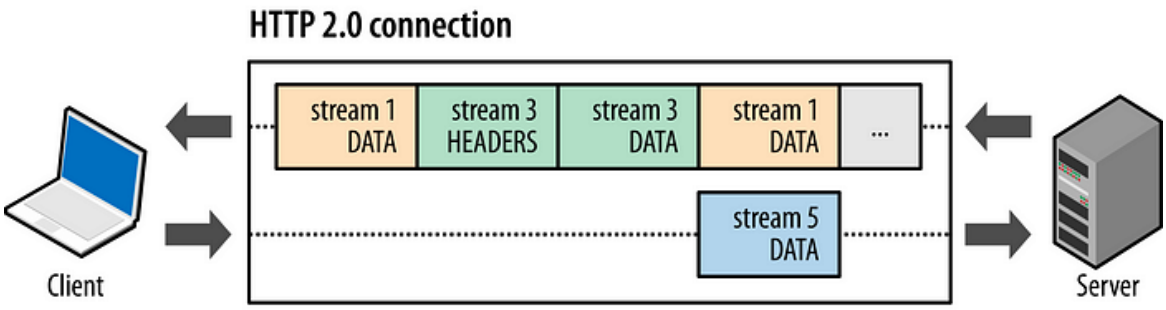


פיצ'ר זה מאפשר לשלוח בקשה של דף html ולקבל הרבה קבצים אחרים מבלי לבקש אותם. לכן כשמתכננים תקיפת HTTP/2 DDoS Amplification Attack התוקף יחפש אתר שעושה push להרבה קבצים וקבצים גדולים כדי להגדיל את יעילות התקיפה.

- Multiplexing הוא תכונה של HTTP/2 המאפשרת לשלוח כמה בקשות HTTP באותו חיבור TCP. לדוגמה כשמבקשים אתר אינטרנט בדפדפן, הדפדפן מבקשים את דף הבית, "index.html" ואת הקבצים "style.css" ו-"script.js" ב-HTTP. ב-1.1/HTTP הוא ישלח 3 בקשות ב-3 חיבורי TCP שונים וב-HTTP2 הדפדפן ישלח את 3 הבקשות באותו חיבור TCP:

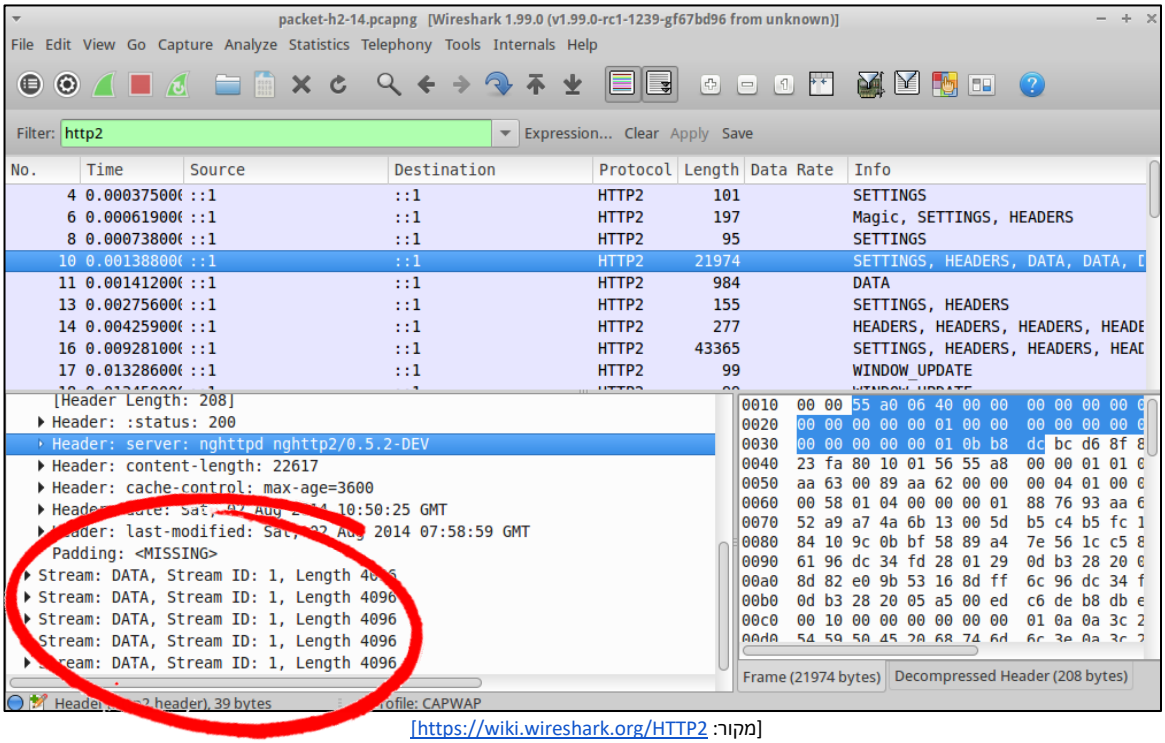


- Flow control זה תכונה ב-HTTP/2 שנועדה שלא יהיו התנגשויות בין העברות ההודעות ב-HTTP משום שהן באותו חיבור TCP. תכונה זאת מפצלת כל סוג הודעה ל-"stream" שונה. לדוגמה הודעה של העברת Header יכולה להיות ב-stream3 והודעה להעברת המידע למשל דף HTML יכולה להיות ב-steam1.



[מקור: https://medium.com/coderbyte/a-guide-to-becoming-a-full-stack-developer-in-2017-5c3c08a1600c]

פקטת **HTTP/2**, אפשר לראות את ה-**flowcontrol**:



[מקור: https://wiki.wireshark.org/HTTP2]

שלבים בתקיפה:

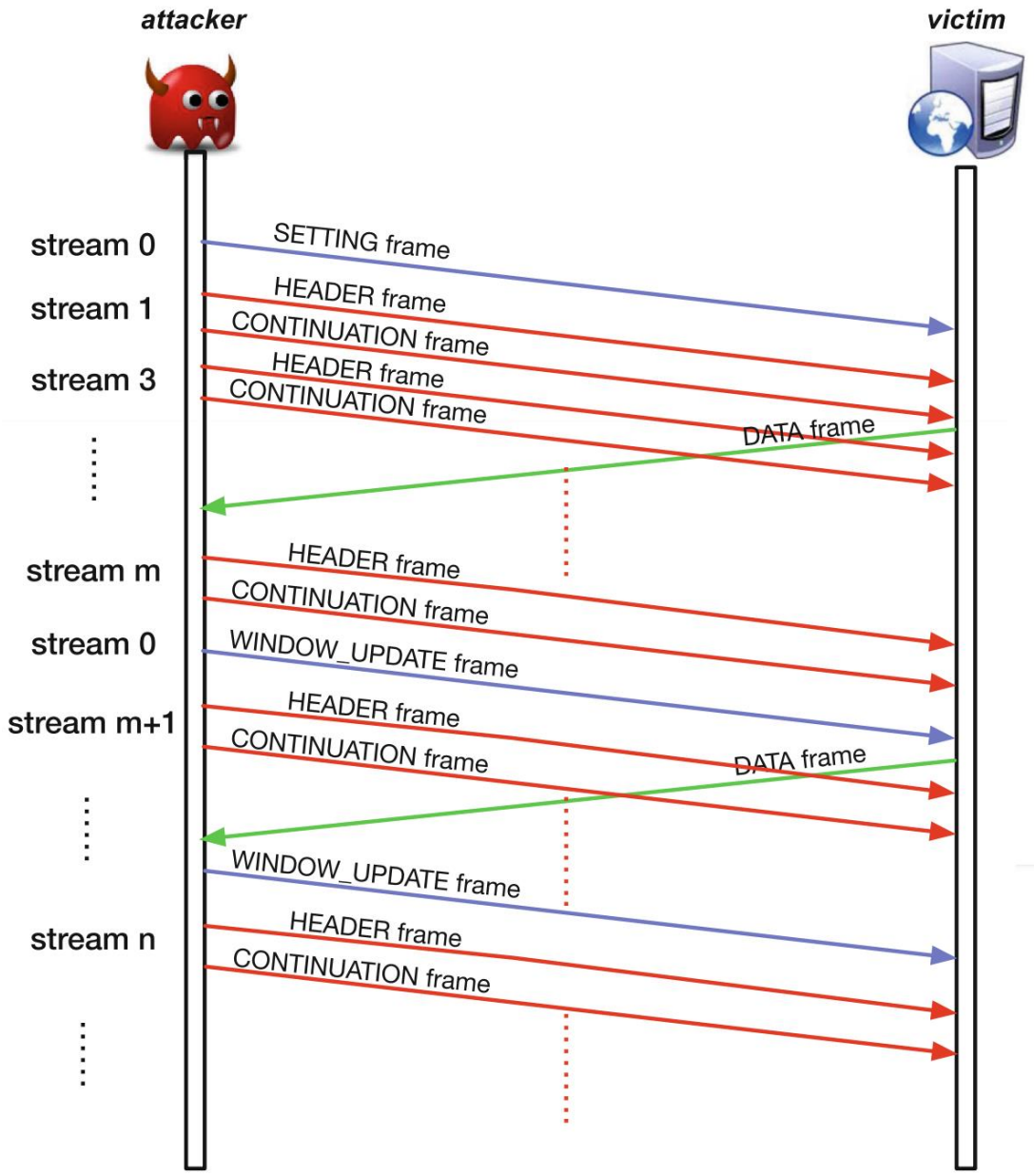
1. פתיחת חיבור TCP עם המטרה.
2. ב-stream0 נשלח *SETTINGS* Frame זאת אומרת שה *SETTINGS* Frame נשר לאורך כל החיבור. ב-SETTINGS Frame נשלח שני פרמטרים:

A. פרמטר אחד הוא *SETTINGS_MAX_CONCURRENT_STREAM*, פרמטר זה מגדיר את מספר ה-streams האפשרי להיות מקסימלי וככל שיש יותר streams כך יותר משאבים של השרת מנוצלים.

B. פרמטר נוסף שנשלח הוא *SETTINGS_INITIAL_WINDOW_SIZE*, ששולט ב-windows size כלומר גודל מקסימלי של כל פקטה שאנו נקבל. נרצה שערך זה יהיה נמוך כמה שניתן. זה גורם לתגובת השרת להיות הרבה יותר איטית.

3. התוקף בונה בקשת HTTP/2 GET ב-stream1, אשר בנויה מפריים של Header ופריים אחד או יותר של *CONTINUATION*. אנחנו נשלח בבקשת ה-HTTP/2 GET שדה גדול של ה-Header ואת רובו נשלח ב-CONTINUATION Frames ורק חלק קטן ב-Header Frame.
4. בעזרת תכונת ה-Multiplexing של HTTP/2 נשלח בקשות GET כמו בשלב 3 ומספר ה-stream יהיה תמיד אי זוגי ויעלה ב-2 (כלומר 1, 3, 5...).
5. כדי שהשרת לא יחסום אותנו נשלח הודעות *WINDOW_UPDATE* ונגדיל את ה-Windows Size בקצת.
6. כדי להגדיל את מקדם התקיפה אפשר לעשות זאת מכמה חיבורי TCP ומכמה מכשירים.

תקיפה זאת מנצלת את כל חיבורי השרת וגם מעמיסה על משאבי העיבוד של השרת:



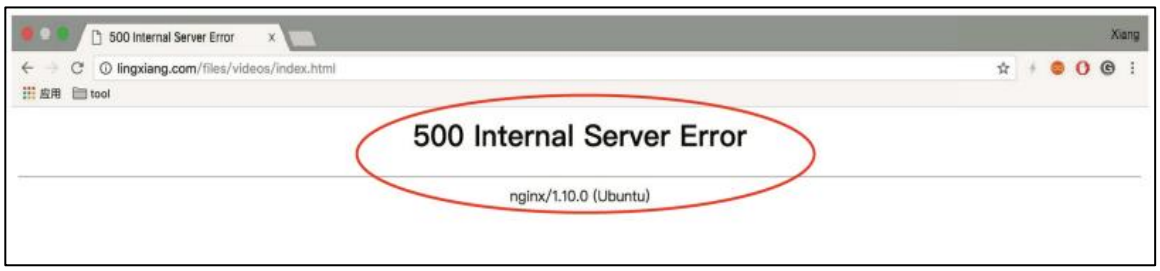
[מקור: https://nesa.zju.edu.cn/download/H2DoS%20An%20Application-Layer%20DoS%20Attack%20towards%20HTTP_2%20Protocol.pdf]

בדיקות והשוואות:

הבדיקות נעשו על המכשירים הבאים:

מכשיר	שרת	תוקף	לקוח של השרת
מערכת הפעלה	Ubuntu 16.04.1 LTS	Ubuntu 16.04.1 LTS	Mac OSX 10.11.6
מעבד	2 x Intel(R) Core(TM) i5-4590 CPU @3.30 GHz	2 x Intel(R) Core(TM) i5-4590 CPU @3.30 GHz	2.7 GHz Intel Core i5
זיכרון	4GB	4GB	8GB

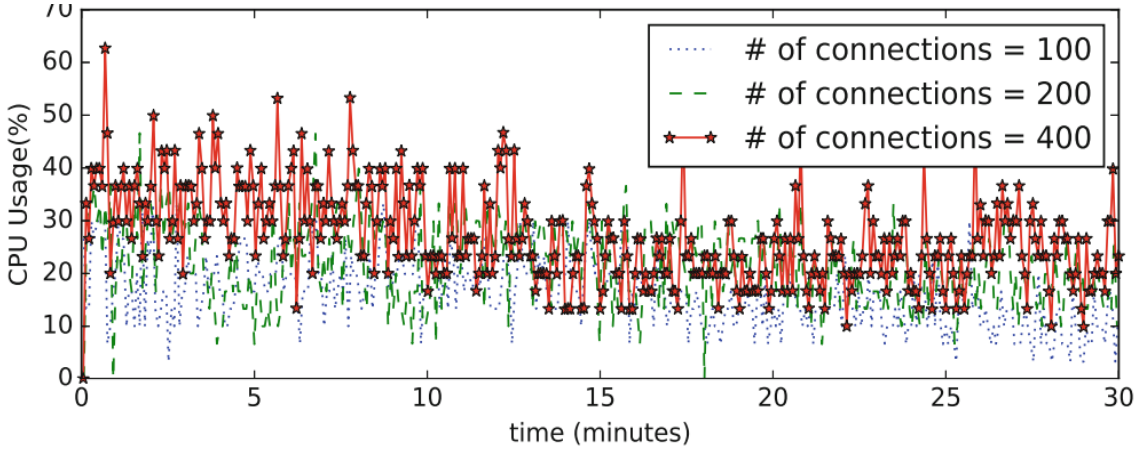
במהלך התקיפה כאשר לקוח ינסה להשתמש בשרת הוא יקבל את השגיאה הבאה:



[מקור: https://nesa.zju.edu.cn/download/H2DoS%20An%20Application-Layer%20DoS%20Attack%20towards%20HTTP_2%20Protocol.pdf]

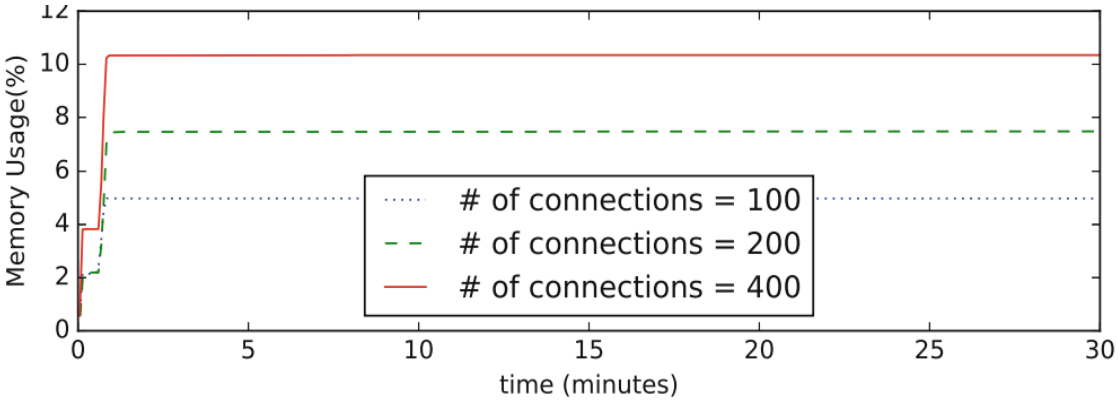
בשגיאת 500 השרת נתקל במצב לא צפוי שמונע ממנו למלא את בקשת הגולש.

טבלת של צריכת משאבי מעבד בזמן התקיפה:



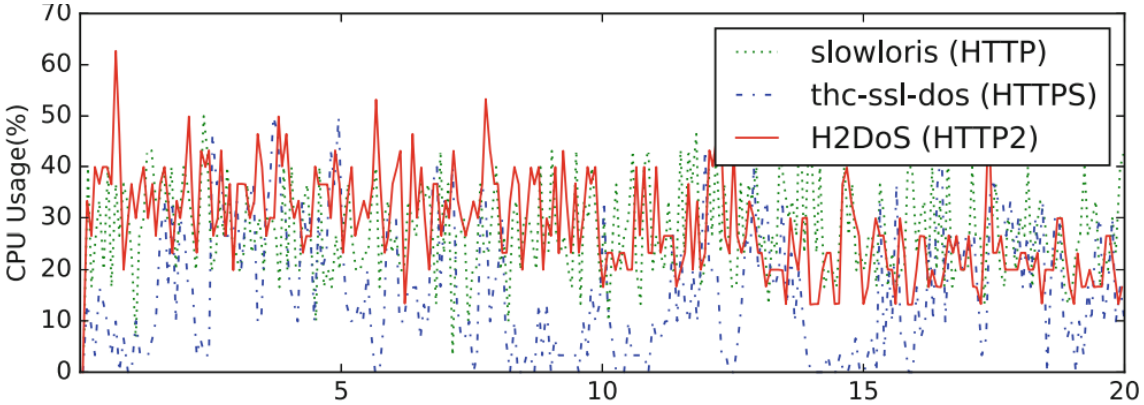
[מקור:https://nesa.zju.edu.cn/download/H2DoS%20An%20Application-Layer%20DoS%20Attack%20towards%20HTTP_2%20Protocol.pdf]

טבלת צריכת זיכרון בזמן התקיפה:



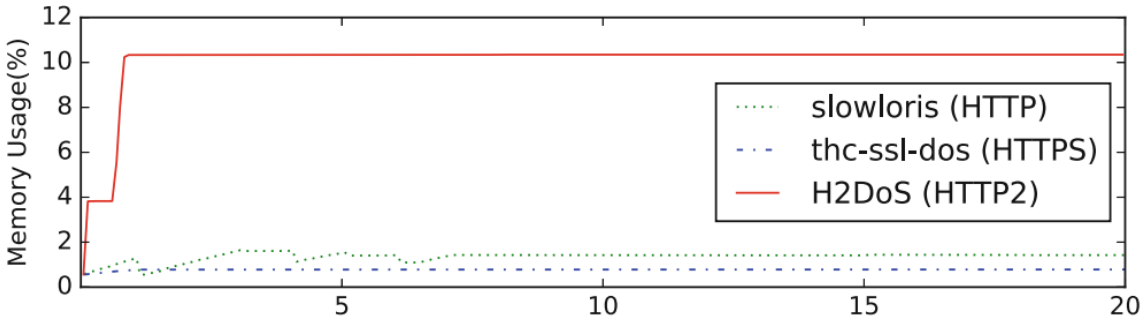
[מקור:https://nesa.zju.edu.cn/download/H2DoS%20An%20Application-Layer%20DoS%20Attack%20towards%20HTTP_2%20Protocol.pdf]

בהשוואה לתקיפת Slowloris HTTP ובהשוואה לתקיפה THC-SSL-DoS HTTPS, תקיפה זאת צורכת יותר משאבי מעבד בזמן יותר קצר:



[מקור: https://nesa.zju.edu.cn/download/H2DoS%20An%20Application-Layer%20DoS%20Attack%20towards%20HTTP_2%20Protocol.pdf]

תקיפה זו גם צורכת יותר זיכרון גם לטווח הארוך:



[מקור: https://nesa.zju.edu.cn/download/H2DoS%20An%20Application-Layer%20DoS%20Attack%20towards%20HTTP_2%20Protocol.pdf]

TCP abuse for Reflective Amplification DDoS Attacks

במחקר מ-2014 שהציגו חוקרים מאוניברסיטת רוהר בוכום על תקיפות TCP נמצאו מגוון של דרכים שונות שמאפשרות לנצל את תהליך לחיצת היד המשולשת של TCP כדי לבצע תקיפות אמפליפיקציה על קורבנות בתהליך דומה לשאר תקיפות אמפליפיקציה עם Spoofing.

תהליך לחיצת היחד המשולשת הוא תהליך של הפרוטוקול TCP שמשמש כדי לפתוח חיבור, בהתחלה המחשב שרוצה לפתוח את החיבור(לרוב הלקוח) שולח פקטת SYN למחשב השני(לרוב השרת), ואז בתגובה המחשב השני שולח פקטת SYN/ACK שמאשרת את החיבור וקבלת הפקטה הראשונה ואז המחשב המקורי שולח פקטת ACK כדי לאשר את החיבור ומתחיל לשלוח את המידע. בנוסף, אם המחשב השני לא רוצה לפתוח חיבור הוא שולח בחזרה פקטת RST שחוסמת את החיבור ועוצרת את התקשורת.

בפועל התהליך נוצר כדי למנוע Spoofing מכיוון שהתהליך דורש שפקטות האישור ישלחו מאותו מחשב בתזמון נכון עם פרמטרים שצריך לשלוח בחזרה, אבל החוקרים מצאו שיש כמות מאוד גדולה של מחשבים ברשת העולמית שמתוכנתים להחזיר כמות מאוד גדולה של פקטות בתגובה לפקטת SYN גם אם אין פקטת אישור חוזרת.

החוקרים הקימו ניסוי כדי לבדוק את כמות השרתים שממומשים לא טוב במרחב האינטרנט(כל כתובות ה-IPv4 הפומביות), הם שלחו פקטות SYN לשרתים מסוגים שונים של פרוטוקולים שמממשים את TCP ובדקו את סוג התגובות ואת גודל האמפליפיקציה (אם קיימת).

תוצאות

בטבלה הבאה:

Table 1: Number of potential amplifiers with an amplification factor > 20 based on scans of 20 million hosts

	20 million random hosts			Estimation (IPv4)
Protocol	# Responsive	# Amplifiers	Amplifier Ratio	# Amplifiers
FTP	705,371	13,701	1 : 51	2,945,715
HTTP	715,354	1,746	1 : 409	375,390
IMAP	683,567	9	1 : 75,951	1,935
IPP	702,590	19	1 : 36,978	4,085
IRC	648,688	7	1 : 92,669	1,505
MySQL	672,336	8	1 : 84,042	1,720
NetBIOS	517,482	44	1 : 11,760	9,460
NNTP	667,598	8	1 : 83,449	1,720
POP3	689,716	7	1 : 98,530	1,505
SIP	711,210	87	1 : 8,174	18,705
SMTP	674,815	12	1 : 56,234	2,580
SSH	657,916	384	1 : 1,713	82,560
Telnet	575,067	9,315	1 : 61	2,002,725

[מקור: https://christian-rossow.de/publications/tcpamplification-woot2014.pdf]

החוקרים ביצעו את בדיקה על 20 מיליון כתובות שנבחרו באופן פסודורנדומלי וסיננו את כל התוצאות שיצרו אמפליפיקציה של לפחות 20. ניתן בבירור לראות כי פרוטוקלי FTP ו-Telnet הם הפרוטוקולים עם הכמות הגדולה ביותר של מחשבים פגיעים בפער אבל יחס האמפליפיקציה לא מרשים ביחס לשאר הפרוטוקולים, לדוגמא POP3 יוצר יחס אמפליפיקציה של כמעט 1:100,000.

בטבלה הבאה:

Table 2: Number of potential amplifiers per protocol based on scans in the entire IPv4 address space

		# Amplifiers with amplification factor					
Protocol	# Responsive	> 20	> 50	> 100	> 500	> 1,000	> 2,500
FTP	152,026,322	2,913,353	3,500	1,868	1,032	937	847
HTTP	149,521,309	427,370	15,426	6,687	1,596	649	347
NetBIOS	82,706,193	12,244	2,449	1,463	873	811	783
SIP	154,030,015	22,830	5,158	3,913	3,289	3,123	2,889
SSH	141,858,473	87,715	4,611	2,141	1,275	1,176	1,082
Telnet	126,133,112	2,120,175	16,469	7,147	2,008	1,393	994

[מקור: https://christian-rossow.de/publications/tcpamplification-woot2014.pdf]

החוקרים ביצעו את הבדיקה על כל מרחב ה-IPv4 ומיינו את כמות המחשבים הפגיעים על פי יחס האמפליפיקציה, ניתן לראות שאם מסתכלים על יחס אמפליפיקציה של לפחות 20 אז עוד פעם פרוטוקולי FTP ו-Telnet הם הכי דומיננטים אך כשמגיעים ליחסי אמפליפיקציה גדולים יותר שאר הפרוטוקולים מופיעים לדוגמא SIP, שעבורו קיימים כמעט 2900 שרתים פגיעים. עוד דבר שחשוב לציין הוא שכ-2% מכל המחשבים הפגיעים באינטרנט שמשתמשים ב-FTP או Telnet מבצעים אמפליפיקציה של לפחות 20.

בטבלה הבאה:

Table 3: Intersection of potential amplifiers

	Intersection (in %)					
Protocol	*FTP*	*HTTP*	*NetBIOS*	*SIP*	*SSH*	*Telnet*
FTP	-	4.5	0.1	0.2	0.6	19.2
HTTP	30.8	-	0.6	1.1	4.3	26.7
NetBIOS	20.9	21.9	-	53.8	14.7	22.5
SIP	21.9	21.4	28.9	-	22.5	26.0
SSH	19.8	21.1	2.1	5.9	-	21.0
Telnet	26.3	5.4	0.1	0.3	0.9	-

[מקור: https://christian-rossow.de/publications/tcpamplification-woot2014.pdf]

אפשר לראות את ההצטלבות של פרוטוקולים שונים שהופכים את אותו שרת לפגיע, לדוגמא ההצטלבות בין NetBIOS ו-SIP היא הגדולה ביותר ואומרת שמעל ל-50% מהמחשבים שניתן לבצע בהם אמפליפיקציה עם NetBIOS ניתן גם לבצע אמפליפיקציה עם SIP, בסך הכל נמצאו 4.8 מיליון שרתים פגיעים עם ששת הפרוטוקולים האלו.

בטבלה הבאה:

Table 4: Number of vulnerable hosts and average amplification factor (AF) per protocol and amplification type

	SYN/ACK		PSH		RST	
Protocol	# Ampl.	AF	# Ampl.	AF	# Ampl.	AF
FTP	2,907,279	22x	274	103x	5,577	53,927x
HTTP	421,487	60x	241	147x	3,411	432x
NetBIOS	8,863	54x	64	71x	3,087	78,042x
SIP	16,496	1,596x	2	696x	6,306	32,411x
SSH	81,256	80x	391	57x	5,889	29,705x
Telnet	2,112,706	28x	2,353	3,272x	4,242	79,625x

[מקור: https://christian-rossow.de/publications/tcpamplification-woot2014.pdf]

אפשר לראות את כמות השרתים הפגיעים על פי סוגי הפרוטוקולים, מחולק על פי סוג האמפליפיקציה, כלומר, כאשר שולחים פקטת SYN לשרת יש 3 דרכים מרכזיות שאיתן השרת יגיב, שליחת פקטת SYN/ACK מספר מרובה של פעמים עד שהוא יחליט לנטוש את הקשר(עם Threshold שליחות קבוע מראש), שליחת פקטות מידע PSH באופן ישיר מבלי לסיים את תהליך לחיצת היד המשולשת או שליחה מרובה של פקטות RST כדי לסגור את הקשר.

לכל פרוטוקול וסוג תגובה כתוב את כמות המחשבים הפגיעים ואת יחס האמפליפיקציה (AF), ניתן לראות כי הדרך הראשונה של תגובה עם SYN/ACK היא הנפוצה ביותר אבל עם יחס האמפליפיקציה הקטן ביותר(בממוצע 80 אבל מגיע ל-1,596 עם SIP). שליחת מידע באופן מיידי היא הדרך הכי נדירה אבל יחס האמפליפיקציה יותר גדול מזה של SYN/ACK. שליחת פקטות RST היא לא נפוצה אבל אפשרית ומביאה את יחס האמפליפיקציה הגדול ביותר, תוקפים יכולים לנצל את 4,242 שרתי Telnet הפגיעים כדי להגיע ליחס אמפליפיקציה של 79,625.

בנוסף, לתגובות RST יש רוחב פס גדול בהרבה מזה של SYN/ACK, זאת אומרת ש-8,863 השרתים הפגיעים ב-SYN/ACK של NetBIOS לוקחים רוחב פס של 25MB בזמן שתגובות ה-RST לקחו רוחב פס של 12GB, באופן דומה ב-FTP, למרות שיש הרבה יותר תגובות SYN/ACK (כמעט 3 מיליון) הם לקחו 3.2GB רוחב פס לעומת תגובות RST (שמהם היו רק 5,577) שלקחו 15.1GB באותה כמות זמן, יחס של 5 בין רחבי הפס.

עוד גורם חשוב בתקיפות אמפליפיקציה הוא קצב שליחת הפקטות של השרת, כלומר אם שרת יכול לשלוח כמות אדירה של פקטות בגודל בינוני באותה כמות זמן ששרת אחר יכול לשלוח רק פקטה אחת ענקית עדיף לתקוף את השרת הראשון.

בטבלה הבאה:

Table 5: Number of packets transmitted by the amplifiers within 10, 30, and 60 seconds after the first response

	SYN/ACK			PSH			RST		
Protocol	< 10	< 30	< 60	< 10	< 30	< 60	< 10	< 30	< 60
FTP	2	5	10	5	10	14	561	1,584	3,055
HTTP	2	6	11	5	10	16	140	224	264
NetBIOS	8	17	22	5	6	8	976	2,748	5,291
SIP	2	6	12	1	1	1	562	1,360	2,497
SSH	3	6	11	6	9	10	595	1,394	2,523
Telnet	2	5	10	52	154	277	996	2,345	4,254

[מקור: https://christian-rossow.de/publications/tcpamplification-woot2014.pdf]

ניתן לראות את כמות הפקטות שנשלחו במסגרות זמן של 10,30 ו-60 שניות מחולק על פי פרוטוקול וסוג תגובה של השרת, אפשר להסיק כי לתגובות SYN/ACK ו-PSH יש קצב שליחה נמוך מאוד שכנראה מומש בכוונה מצד השרת כדי למנוע תקיפות אמפליפיקציה אבל בתגובות RST יש כמויות גדולות מאוד של תגובות במסגרת זמן יחסית קטנה בכל הפרוטוקלים.

זאת כמובן בעיה מאוד גדולה לרשת מכיוון שבטבלאות קודמות מצאנו שכמעט בכל גורם RST יותר יעיל לתקיפה מכוונת.

כיצד ניתן למנוע ולזהות תקיפות DDoS

זיהוי תקיפות DDoS הוא די פשוט, מכיוון שמטרת ההתקפה היא לגרום להתנהגות יוצאת מן הכלל של המערכת (הקרסה או האטה של השרתים), אז זיהוי הוא די פשוט:

1. אם כתובת IP מסוימת שולחת המון בקשות במסגרת זמן קטנה, והבקשות לא מתממשות אז ככל הנראה זאת תקיפה שמשתמשת ב-IP Spoofing.
2. אם יש עלייה פתאומית בכמות הבקשות הכוללת לשרת ללא סיבה הגיונית אז יש סיבה לחשוד לתקיפת DDoS.

דרך הזיהוי בפועל מתחלקת לאחת משתי דרכים מרכזיות, זיהוי פנימי או זיהוי חיצוני זיהוי פנימי מתרחש בתוך הרשת בזמן התנועה, כלומר, כאשר פקטה נכנסת למערכת ומעובדת היא מיד עוברת תהליך אבחון לבדיקה האם היא מהווה התקפה. שיטה זו הייתה יותר פופולרית בעבר כאשר תקיפות היו יחסית קטנות בגודלן ולא יכלו לעשות עומס יתר על המערכות הפנימיות, אבל כיום כשההתקפות יכולות להיות בגדלים של טרהבייטים בכל פעם יש צורך בדרכים אחרות כמו מכשירים מיוחדים שמטרתם היחידה היא לערוך את תהליך הבדיקה, אבל מכשירים כאלה לרוב גם מתקשים אל מול התקפות גדולות ואורך חייהם מוגבל מאוד, לכן השיטה המודרנית היא להשתמש בעיקר בזיהוי חיצוני ולתת עבודות שדורשות כוח עיבוד מיוחד למכשירים האלו.

זיהוי חיצוני מתרחש במכשירים חיצוניים למערכת שמקבלים פרטים מלאים על התקשורת דרך פרוטוקולי ניתוח רשת כמו NetFlow, J-Flow, sFlow, IPFIX שמייצאים מידע על כל התקשורת ברשת בצורה נוחה ומהירה למכשירים חיצוניים.

אם המכשירים החיצוניים האלה מזהים התקפה אז הם יכולים באופן אוטומטי להפעיל אמצעיי נגד או לשלוח התראות לאחראיים.

על מנת למנוע התקפות DDoS יש מגוון רחב של דרכים שפורטו כבר במאמר אחר במסגרת המגזין, נציג שתי דרכים מרכזיות כאן:

1. קניית שירות ספקים, יש בכל העולם וגם בארץ חברות שמטרתן היא לעזור לחברות עם שרתי אינטרנט למגר ולמנוע התקפות שונות כולל התקפות DDoS באמצעות כלים מיוחדים בחלק שלהם. כלומר מוסיפים שלב לתקשורת השרת שעובר דרך הספק ומוודא שלא תהיה התקפה ובכך מונעים את ההתקפה ללא יותר מדי בעיות או אחריות על הנושא.

2. התקנת ציוד פנימי, במקום להסתמך על ספק חיצוני ניתן להתקין שירותים למניעת התקפות בשם החברה עצמה ובכך לקבל יותר שליטה על דרך המניעה, בשיטה הזאת יש צורך לתחזק את המערכות האלו אבל הן יכולות להיות יותר מותאמות לצורכי החברה.

סיכום

במאמר זה הראנו כמה דרכים מודרניות לביצוע מתקפות DDoS, בעיקר מסוג התקפות רפלקציה שמשתמשות בשרתים גדולים פגיעים כדי לבצע את העבודה בשבילנו. כיום תקיפות DDoS הן מאוד נפוצות בכל התחומים בין אם זה התקפות על חברות הייטק גדולות עד להתקפות על שרתי משחקי מחשב, לכן חשוב להבין ולדעת איך להתגונן מפני ההתקפות האלה.

על המחברים

תבור לביא בן 17 מבאר יעקב ו**אלון טל זילברמן**, בן 17 מחולון, תלמידים בתוכנית אודיסיאה באוניברסיטת תל אביב במסלול סייבר.

בחרנו לכתוב ולחקור על DDoS כי זו אחת ממתקפות הסייבר הנפוצות ביותר כיום ורצינו להעמיק את הידע שלנו בנושא. ברצוננו להוקיר תודה ל**ד"ר שלומי בוטנרו** ומר **אורן רנרד** על התמיכה וההנחיה במהלך העבודה על המאמר.

ביבליוגרפיה

- https://www.imperva.com/learn/ddos/udp-flood/
- https://www.wallarm.com/what/udp-flood-attack
- https://www.cloudflare.com/learning/ddos/ping-icmp-flood-ddos-attack
- https://www.imperva.com/learn/ddos/ping-icmp-flood/
- https://www.imperva.com/learn/ddos/syn-flood/
- https://www.techtarget.com/searchsecurity/definition/SYN-flooding
- https://www.imperva.com/learn/ddos/http-flood/
- https://www.myrasecurity.com/en/knowledge-hub/http-flood-attack/
- https://www.firewall.cx/general-topics-reviews/network-protocol-analyzers/1224-performing-tcp-syn-flood-attack-and-detecting-it-with-wireshark.html
- https://www.radware.com/security/ddos-threats-attacks/threat-advisories-attack-reports/hackers-arsenal-http-flood-tools/
- https://www.neuralnine.com/code-a-ddos-script-in-python/
- https://www.cisa.gov/news-events/alerts/2014/01/17/udp-based-amplification-attacks
- https://ieeexplore.ieee.org/document/8090411
- https://www.akamai.com/blog/security/new-ddos-vector-observed-in-the-wild-wsd-attacks-hitting-35gbps
- https://christian-rossow.de/publications/tcpamplification-woot2014.pdf
- https://github.com/649/Memcrashed-DDoS-Exploit
- https://nesa.zju.edu.cn/download/H2DoS%20An%20Application-Layer%20DoS%20Attack%20towards%20HTTP_2%20Protocol.pdf
- https://www.kentik.com/kentipedia/ddos-detection/