
Domain Name System - אנומליות, איתור ומניעה

מאת קיריל לשצי'בר

תקציר

Domain Name System, או בקיצור DNS, הינו פרוטוקול המשמש למערכת מתן שמות למחשבים, שירותים או כל משאב רשת אחר. מטרת הפרוטוקול החשובה ביותר היא לתרגם את השמות שהם בעלי משמעות לבני אדם, לכתובות IP שהן בעלות משמעות למחשבים, ובכך מאפשר לנו לאתר בקלות כל מחשב או שירות ברחבי העולם.

ה-DNS הומצא בשנת 1983 ע"י פול מוקפטרס והמפרטים המקוריים שלו מופיעים ב-RFC 882 ו-883. והם הוחלפו מאוחר יותר (בנובמבר 1987) ע"י RFC 1034 ו-1035.

שרתי ה-DNS הראשונים נכתבו בשנים 1984 ו-1985. השרת האחרון נקרא BIND (Berkeley Internet Name Domain) ומחזיק בשמו עד היום, בגרסאות שונות.

פרוטוקול DNS לא נועד להתמודד עם בעיות אבטחה ולכן פותחו שיטות פריצה רבות במטרה להוסיף נתונים שגויים לתוך DNS Lookup tables או למטמון שלו. כל שיטה המאפשרת החדרה של נתונים שגויים לתוך המטמון של ה-DNS נקראה "הרעלת מטמון", ורוב ההתקפות על שרתי DNS ניסו לבצע את המשימה הזו.

ההגנה העיקרית כנגד התקפות אלו בפרוטוקול DNS היא Transaction ID (TID) של השאילתה: אם התשובה לא חוזרת עם אותה TID כמו השאילתה המקורית, ה-DNS מתעלם ממנה. לכן, כדי להרעיל את המטמון של שרת ה-DNS, ההאקר חייב לנחש את ה-TID הנכון של השאילתה. בגרסאות BIND קודמות היה קל לנחש את מספר ה-TID, מכיוון ששדה ה-TID היה גדל ב-1 עבור כל שאילתה חדשה. בגרסאות ה-BIND האחרונות כל TID חדש הוא מספר אקראי ולכן מקשה על הניחוש.

עם זאת, ייתכן שיש מצב שבו להאקר יש גישה לשדה TID שניתן באופן אקראי: ע"י שליטה על הנתבים הראשיים, או ע"י sniffing של התעבורה של הרשת בדרך כלשהי. במקרה כזה ההאקר יכול לייצר התקפות רעל ללא כל קושי. העבודה שלנו מתרכזת בזיהוי ומניעת התקפות כאלה. אנו מניחים כי ההאקר הוא כל יודע ולכן יודע את מספר TID של כל שאילתה העוזבת את שרת ה-DNS.

אם התקיפה על שרת ה-DNS מכניסה בהצלחה מידע שגוי לתוך מטמון ה-DNS, דבר זה נקרא "תקיפת הרעלת מטמון". כל שאילתה עבור הרשומה המורעלת, תחזיר IP שגוי למשתמש, ובכך תכוון אותו לאתר או שירות שגוי. במאמר זה אנו מציעים אלגוריתמים להתמודד עם מקרה כזה.

פרוטוקול DNS הוא אחד הפרוטוקולים הפופולריים והחיוניים ביותר ברחבי רשת אינטרנט. אנו משתמשים בפרוטוקול על בסיס קבוע על מנת לתרגם שמות של אתרים או רכיבי רשת אחרים לכתובות IP. DNS עובד ברקע התקשורת שלנו, מבלי שנבחין בכמות הבקשות הגדולה שאנו שולחים.

בשל השימוש הנרחב שלו, אנו נוטים באופן טבעי להניח התשובות הנשלחות ע"י שרת ה-DNS המקומי שלנו הן מלאות וחוקיות. למרות זאת, זה לא בהכרח מתקיים תמיד. שרתי DNS, אפילו אלו המעודכנים לתוכנה העדכנית ביותר, יכולים להיות פרוצים. הרשומות שלהם לא תמיד מכילות את המידע הנכון, והמידע השגוי הזה יכול בסופו של דבר להשלח למשתמש, שינותב לאתר הלא נכון.

הסיבה לכך היא חוסר המחשבה בנוגע לבעיות אבטחה כשהפרוטוקול תוכנן לראשונה, לפני כ-30 שנה.

במאמר זה, בפרק 3, אנו מציגים חלק מהעבודה שנעשתה במשך השנים במטרה למנוע ולמזער את ההתקפות על שרתי ה-DNS. אנו מסבירים את ההתקפות ואת הפתרונות המוצעים כדי לפתור אותן. הרבה מההתקפות המוצגות נחסמו לחלוטין או מוזערו במטרה להקטין את הסיכוי לתקיפה מוצלחת, ע"י גרסאות חדשות של שרתי ה-DNS. עם זאת, אפשר להניח שחלק מההתקפות לא נמצאו או לא נחשפו לציבור. התקפות zero-day אלו עשויות להיות מיושמות ויכולות להיות בעלות השפעות הרסניות ביום שבו ישתמשו בהן.

כל ההתקפות המוצגות מניחות כי ההאקר הוא לא כל יודע, כלומר הוא לא יכול לראות את התנועה ברשת. במקום זה, ההאקר צריך להסתמך על ניחושים או ביצוע חיפושים נרחבים. הניחוש והחיפוש המאסיבי יכולים להפעיל הזעקות במכשירי הניטור ברשת.

בעבודה זו, אנו שוברים את ההנחה הזו ומסתכלים על המקרה שבו ההאקר יכול לבחון כל תנועה העוזבת או נכנסת לשרת ה-DNS. במקרה כזה, ההאקר יכול לייצר בקלות תשובה חוקית ולשלוח אותה לשרת ה-DNS, תוך עקיפת מכשירי הניטור של הרשת.

בפרק 4 אנו מתארים אלגוריתם שפיתחנו שמזהה חבילות עם זמן הגעה חריג ביחס לחבילות אחרות עם מאפיינים זהים (לדוגמא: תשובות של סוג A משרת DNS מוסמך מסויים). החבילות עם זמן ההגעה החריג מטופלות כחבילות חריגות ומעוכבות לזמן מסויים. לכן האלגוריתם נקרא "Delay Fast Packets algorithm", או בקיצור: "אלגוריתם DFP". במקרה של התקפה אמיתית, בזמן העיכוב נשלחת תשובה חוקית משרת Authoritative DNS אמיתי ואלגוריתם DFP מזהה את המקרה של תגובות מרובות עבור שאילתה אחת ולכן מסיק כי שרת ה-DNS מותקף. אלגוריתם ה-DFP זורק את 2 התשובות ומאפשר לשרת ה-DNS לטפל בשאילתה חדשה, כאילו התרחש אובדן חבילות רגיל.

אנו מאמינים כי אלגוריתם ה-DFP, משולב יחד עם עוד טכניקות של מניעת זיהוי התקפות שכבר מיושמות על שרתי ה-DNS, יורידו באופן דרסטי את הסיכויים להתקפה מוצלחת. כעת ההאקר לא רק צריך לשלוח את המידע הנכון, הוא גם צריך להתאים את התשובה תוך פרק זמן שהוא רק יכול לנחש.

בסוף העבודה, בפרקים 5 ו-6, אנו מציגים בפירוט את הניסויים שביצענו על תנועה אמיתית שנתפסה בשרת DNS HUI ומתארים את נקודות החוזקה והחולשה של אלגוריתם ה-DFP.

חלק I: בטיחות DNS

פרק 1: הקדמה:

פרוטוקול DNS נועד להקל על חיינו, כפי שתואר ב-RFC 1034 [1] וב-RFC 1035 [2]. כפי שאנו לא זוכרים את כל מספרי הטלפון בספר הטלפונים, אנחנו לא זוכר את כל כתובות ה-IP של כל האתרים. לכן ה-DNS דומה מאוד לשירות ספר הטלפונים: המשתמש נותן את שם האתר, לדוגמה: "google.com", והוא מחזיר לו: '209.85.229.106' שאליו המחשב של המשתמש יכול לשלוח את השאילתה לדף.

תפקיד זה של ה-DNS מעמיד אותו במקום רגיש: המשתמש חייב לסמוך על שרת ה-DNS שיחזיר לו את התוצאה הנכונה עבור בקשתו. אם מסיבה כלשהי שרת ה-DNS עושה טעות ומחזיר כתובת IP שגויה ללקוח, למשתמש תהיה גישה לאתר אחר בזמן שהוא מניח שהוא נכנס ל-"google.com" האמיתי.

טעות זו יכולה להתרחש בגלל מערכת המטמון ב-DNS, שמשמשת את שרתי ה-DNS להאצת התהליך של הבקשות. האקרים מחפשים הזדמנויות למקם רשומות פגומות למטמון של DNS. ברגע שההאקר הצליח להשתיל כאלו רשומות (הרעלת המטמון) ב-DNS, כל משתמש שמחפש את הרשומה הזו (המורעלת) יקבל את האתר הזדוני.

ישנן שתי דרכים עיקריות לתקוף שרת DNS. הדרך הראשונה היא על ידי מציאת באגים באופן שבו פרוטוקול ה-DNS מיושם בגרסה מסויימת של Berkeley Internet Domain Name (BIND) [3]. כדי למנוע סוג זה של התקפות, גרסאות BIND מעודכנות כל הזמן כדי לתקן את הבאגים ביישום ה-DNS שלהם. הדרך השנייה לתקוף שרת DNS היא ע"י מציאת פגמים בפרוטוקול.

במחקר שלנו נתרכז במציאת פתרון לסוג השני של ההתקפות. גישה זו מאפשרת לנו למצוא פתרון יסודי יותר לחוסר הבטיחות ב-DNS.

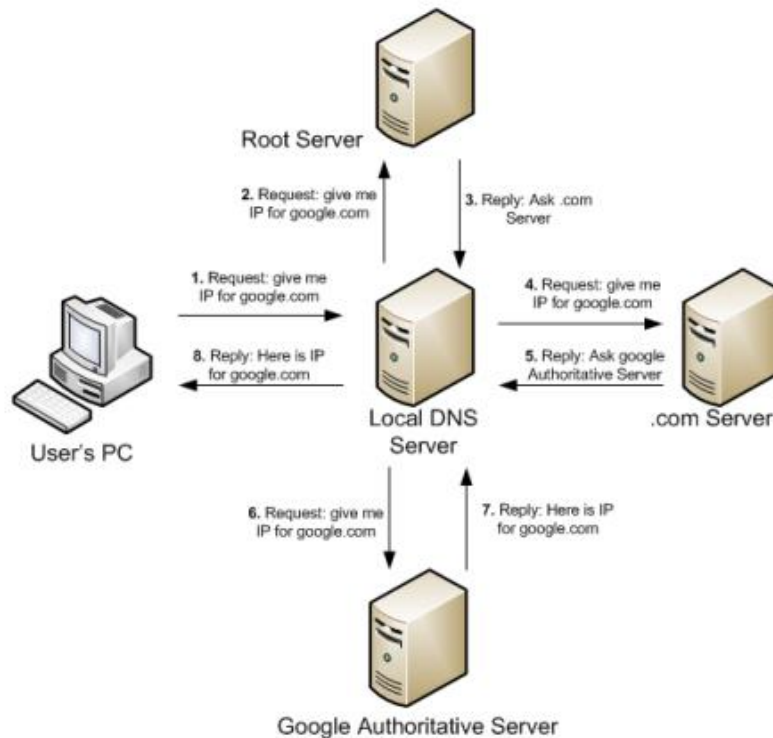
ההשלכות האפשריות של הרעלת מטמון DNS

- גניבת זהות – האקר יכול ליצור אתר זהה לאתר המקורי (לדוגמא "paypal.com"). כשהמשתמש מתחבר לאתר ע"י שימוש במטמון המורעל, הוא יכול להשאיר מידע אישי כמו שם משתמש, סומא, מס' טלפון, כתובת וכו' בידי ההאקר.
- הפצת קוד זדוני – אחת המטרות של ההאקר היא להפיץ קוד זדוני. דרך אחת להשגת המטרה היא לשחרר את הקוד באינטרנט ולחכות. דרך זו תתן בד"כ תוצאות מעטות ואקראיות. הרעלת המטמון והתחזות למס' אתרים פופולריים ייקדו את ההתקפה. ברגע שהמשתמש יוזם פגישה עם האתר של ההאקר, ניתן להוריד את הקוד הזדוני למחשב של המשתמש ללא ידיעתו.
- הפצת מידע שגוי – ניתן להשתמש במידע שגוי או פרסום שגוי כדי להפיץ מידע self serving על ארגון. מידע שגוי זה יכול להחשב כמקורי כאשר הוא מושג מאתר אמין כמו: www.nasdaq.com.
- התקפת Man-in-the-middle – בהתקפה זו המשתמש נכנס לאתר של האקר, אשר בתורו נכנס לאתר המקורי. המשתמש ממשיך בעבודתו ללא חשד, כאשר כל המידע שמועבר בין המשתמש לאתר המקורי עובר דרך ונקלט ע"י האתר של ההאקר.

זרימת פרוטוקול DNS:

החלק הבא ייתן הסבר קצר על הפרוטוקול תוך הדגשת הנושאים החשובים:
ב-"Name Resolution process" יש 2 סוגים של שאילתות שלקוח יכול לעשות לשרת DNS:
רקורסיבית ואיטרטיבית. כשדנים ב-"name resolution", שרת DNS אחד יכול להיות הלקוח של שרת DNS אחר.

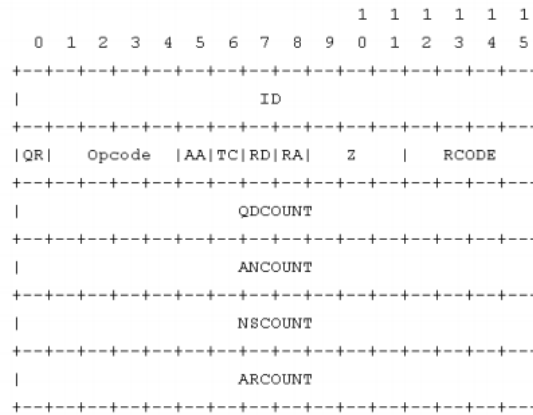
- שאילתות רקורסיביות – בשאילתה רקורסיבית, שרת ה-DNS הנשאל נדרש להגיב עם המידע המבוקש או עם הודעת שגיאה לפיה נתונים מהסוג המבוקש או שם ה-domain המבוקש לא קיימים. שם השרת לא יכול פשוט להפנות את הלקוח לשם שרת אחר. סוג כזה של שאילתה נעשה בד"כ ע"י לקוח לשרת ה-DNS המקומי שלו.
- שאילתות איטרטיביות – בשאילתה איטרטיבית, שרת ה-DNS הנשאל מחזיר את התשובה הטובה שיש לו באותו רגע ללקוח. סוג זה של שאילתה נעשה בדרך כלל ע"י שרת DNS לשרתי DNS אחרים לאחר שהוא קיבל שאילתה רקורסיבית מלקוח.



(איור 1.1: זרימת פרוטוקול DNS. דוגמא לסוג שאילתה איטרטיבית, בה כל השאלות מטופלות ע"י שרת DNS מקומי.)

כאשר משתמש מעוניין לגשת לאתר מסוים, למשל: google.com, הדפדפן שלו מבצע את התהליך הבא:

- המטמון המקומי של משתמש הקצה נבדק. אם התשובה נמצאה, האתר במטמון חוזר. אם לא, משתמש הקצה שולח שאילתה לשרת ה-DNS המקומי.
- שרת ה-DNS המקומי מקבל את השאלתה ובודק במטמון שלו אחר התשובה. אם תשובה נמצאה, היא מוחזרת. אם היא לא נמצאת, שרת ה-DNS המקומי שולח את השאלתה ל-ROOT DNS server.
- שרת ה-ROOT מחזיר תשובה חדשה עם ה-IP: ".com DNS server".
- שרת ה-DNS המקומי שולח שאילתה ל-".com DNS server" לכתובת ה-IP.
- ".com DNS server" מחזיר תשובה חדשה עם ה-IP "Authoritative DNS server" של האתר המבוקש.
- שרת ה-DNS המקומי שולח שאילתה ל-IP של האתר לשרת ה-DNS המוסמך.
- שרת ה-DNS המוסמך מוצא את כתובת ה-IP המבוקשת לשם האתר הנתון ושולח את התשובה לשרת ה-DNS המקומי.
- שרת ה-DNS המקומי מחזיר את כתובת ה-IP ומכניס את התוצאה למטמון שלו, במקרה שמישהו יחפש את אותו האתר בעתיד הקרוב.



(איור 1.2: פורמט חבילות DNS. מציג את הפורמט של חבילת DNS. השדות החשובים מוסברים בפרק זה.)

- ID (או Transaction ID). 16 ביטים. משמש לקשר בין חבילות שאילתה לתשובה.
- QR. ביט 1. דגל שאלה או תשובה.

QR	Description
0	Request
1	Response

- AA. ביט 1. תשובה סמכותית (Authoritative Answer) המציינת כי שרת ה-DNS המגיב הוא סמכות עבור שם הדומיין המבוקש.
- Rcode (Return code). 4 ביטים. הטבלה הבאה מתארת את ה-Rcodes הבסיסיים כפי שזוהו ב-RFC 1035.

Rcode	Description
0	No error.
1	Format error.
2	Server failure.
3	Name Error.
4	Not Implemented.
5	Refused.

- Total Questions. 16 ביטים. מס' הכניסות ברשימת הבקשות שחזרו.
- Total Answer RRs. 16 ביטים. מס' הכניסות ברשימת ה-response resource record, שחזרו.
- Total Authority RRs. 16 ביטים. מס' הכניסות ברשימת ה-authority resource record, שחזרו.
- Total Additional RRs. 16 ביטים. מס' הכניסות ברשימת ה-additional resource record, שחזרו.

- Questions. אורך משתנה. רשימה של אפס או יותר request record structures. כל שאילתה מכילה את שם השאילתה, סוג השאילתה ודרגת השאילה. דרגת השאילה היא בד"כ 1 עבור כתובת אינטרנט. סוגי השאילות הפופולריות ביותר מפורטות בטבלה 1.1 להלן:

Type	Value	Description
A	1	Host's IP address
NS	2	Host's or domain's name server(s)
CNAME	5	Host's canonical name, host identified by an alias domain name
MX	15	Host's or domain's mail exchanger
ANY	255	Request for all records

טבלה 1.1: סוגי שאילות DNS פופולריות.

- Answer RRs. אורך משתנה. רשימה של אפס או יותר response record structures.
- Authority RRs. אורך משתנה. רשימה של אפס או יותר authority record structures.
- Additional RRs. אורך משתנה. רשימה של אפס או יותר additional record structures.

סקירת העבודה

במאמר זה פיתחנו אלגוריתם חדש לזיהוי חריגות בחבילות DNS. האלגוריתם מזהה חבילות עם זמן הגעה חריג לעומת אחרים חבילות אחרות עם אותם המאפיינים. החבילות הללו נחשבות כחבילות "מהירות מידי" ומעוכבות לזמן מסויים. במקרה של התקפה אמיתית, בזמן העיכוב נשלחת תשובה חוקית משרת Authoritative DNS אמיתי והאלגוריתם מזהה את המקרה של תגובות מרובות עבור שאילתה אחת ולכן מסיק כי שרת ה-DNS מותקף. אלגוריתם ה-DFP זורק את 2 התשובות ומאפשר לשרת ה-DNS לטפל בשאילתה חדשה, כאילו התרחש אובדן חבילות רגיל.

האלגוריתם נבחן על תנועה אמיתית בשרת ה-DNS HUN. השתמשנו בסט של ניסויים כדי למצוא את הפרמטרים האופטימליים לאלגוריתם. האלגוריתם משתמש בפרמטרים אלו כדי לאתר את החבילות ה"מהירות מידי" הללו.

פרק 2: סימונים והגדרות

פרק זה מספק את המונחים והסימונים אשר נמצאים בשימוש נפוץ ברשתות מבוססות IP / TCP ומפרטי פרוטוקול DNS. מונחים אלו חיוניים להבנה מלאה של האלגוריתם המוצע בעבודה זו. מידע נוסף לגבי IP, TCP, ו-UDP ניתן למצוא ב-RFC 791 [4], RFC 793 [5], ו-RFC 768 [6].

2.1 הגדרות כלליות:

הגדרה 2.1 מגדירה את **5-Tuple** כרשומה בעלת 5 אלמנטים המייצגת קשר ברשת מבוססת IP/TCP. ה-5 Tuple מכיל מקור IP, יעד IP, יציאת מקור TCP, יציאת יעד TCP, ו-IP הפרוטוקול הבא (בד"כ TCP או UDP).

הגדרה 2.2 מגדירה **local DNS server** כשרת ברירת מחדל למשתמש. כל בקשת DNS מהמשתמש מועברת קודם לשרת ה-DNS המקומי, ורק אם השאילתה לא יכולה להתממש ע"י שרת ה-DNS המקומי היא מועברת לשרתי DNS אחרים.

הגדרה 2.3 מגדירה **Authoritative DNS server** כשרת DNS שנותן תשובות שהוגדרו ע"י מקור מקורי, בניגוד לתשובות שהתקבלו באמצעות שאילתת DNS רגילה ל-**Authoritative DNS server** אחר ואוסנו במטמון.

הגדרה 2.4 מגדירה **הרעלת מטמון DNS** או בקיצור **הרעלת מטמון**, כמצב שנוצר באופן זדוני או לא רצוני המספק מידע ל- **caching name server** שמקורו לא ממקורות authoritative Domain Name System (DNS). ברגע ששרת DNS קיבל נתונים לא אותנטיים כאלו ומכניס אותם למטמון כדי להעלות את הביצועים, הוא נחשב למורעל ומספק את המידע הלא-אותנטי למשתמשים של השרת.

2.2 Round Trip Time:

הגדרה 2.5 מגדירה **Round Trip Time (RTT)** כזמן שלוקח להעביר חבילה ממקור מסויים ליעד מסויים וחזרה.

ה-RTT תלוי בד"כ במס' פקטורים, וביניהם:

- המידע על קצב המעבר של חיבור האינטרנט של המקור.
- החומר המשמש להעברה (נחושת, סיבים אופטיים, אלחוטי או לוויני).
- המרחק בין המקור ליעד.
- מס' המכונות המחוברות בין המקור ליעד.
- עומס התנעה ברשת.
- מס' הבקשות האחרות המנוהלות ע"י nodes ביניים והשרת המרוחק.
- המהירות שאיתה מתפקדים ה-nodes הביניים והשרת המרוחק.

ה-RTT יכול לנוע בין מס' מילי-שניות כשמחושב בין נק' קרובות (באותו LAN), למס' שניות בתנאים גרועים יותר בין נק' המופרדות ע"י מרחק גדול (כמו בחיבור לוויני).

כדי להעריך את איכות וזמן איחזור החיבור, משתמשי קצה מודדים כל הזמן את ה-RTT של החבילות שלהם. אך בגלל הכמות הגדולה של המשתנים ונעלמים, בעיקר בסביבת האינטרנט, בערכת RTT יכולה להשתנות באופן דרסטי.

2.2.1: TCP

לאחר שהמקור מעביר חבילת (TCP) Transmission Control Protocol ליעד שלו, הוא חייב לחכות זמן מסוים לאישור. אם האישור לא הגיע במהלך הזמן המצופה, משערים שהחבילה אבדה והיא נשלחת שוב. השאלה: "כמה זמן TCP צריך לחכות?" קשורה מאוד לזמן ה-RTT הצפוי של הרשת. דרך local area connection (LAN) יספיקו לא יותר מכמה מיקרו-שניות. דרך האינטרנט ידרשו מס' שניות. ואם התנועה עוברת דרך קישור לוויני סביב כדור"א, דרוש הרבה יותר זמן.

מאחר ו-TCP חייב להיות מסוגל לעבוד בכל סביבה, TCP מנטר את החילוף הרגיל של חבילות מידע ומפתח הערכה של כמה זמן הוא צריך לחכות לחבילה הבאה. תהליך זה נקרא "Round Trip Time estimation". להערכה מדוייקת יש השפעה מכרעת על הביצועים של הרשת: אם ה-RTT נמוך מידי, חבילות רבות יועברו מחדש ללא סיבה, ולכן יאטו את הרשת. אם ה-RTT גבוה מידי, החיבור יבזבז זמן יקר בציפיה שזמן ההמתנה יסתיים.

בשל העובדה שלכל רשת יש מאפיינים שונים, TCP חייבת להיות מסוגל להתאים את ערכת ה-RTT בהתאם. יתרה מכך, המאפיינים או הפריסה של הרשת יכולים להשתנות מהר מאוד ליותר טובים (מהירות גבוהה יותר) או לגרועים יותר (מהירות נמוכה יותר). לדוגמא, מעבר מחיבור ע"י אמצעי תקשורת לוייני לאמצעי המבוסס על סיב אופטי, הופך את הרשת למהירה יותר, ואילו שינוי מ-"Data over Carry" ל-"Pigeon" - "Data over Snail" מאט את הרשת. לכן, כדי שיהיה אפשר להסתגל לשינויים ברשת, הערכת ה-RTT החדשה של ה-TCP צריכה להיות מחושבת עבור כל חבילת TCP נכנסת.

הגדרה 2.6 מגדירה $EstimatedRTT$ כ-RTT מוערך של החבילה הבאה הנשלחת ע"י המקור. הנוסחה הבאה מראה כיצד ה-RTT המוערך החדש מוסק מה-RTT המוערך הישן והמדגם החדש של ה-RTT:

$$EstimatedRTT = (1 - \alpha) X EstimatedRTT' + \alpha X RTT$$

α ברירת המחדל ביישום TCP של היום שווה ל-0.125. משמעות α כזו היא שבחישובי $EstimatedRTT$ ניתן יותר משקל לחלק העתיק מאשר לחלק שהתקבל לאחרונה. זה נעשה במטרה להחליק כל פיק חיובי או שלילי. אך אפילו אם מתרחש שינוי מהיר במאפיינים של הרשת, TCP יתאים את ה- $EstimatedRTT$ במהירות.

במטרה לזהות חבילות אבודות, TCP לוקח "קצוות ביטחון" כסטייה של ה-RTT מה- $EstimatedRTT$. הגדרה 2.7 מגדירה $DevRTT$ כסטייה של ה-RTT מה- $EstimatedRTT$.

הנוסחה הבאה מראה כיצד ה-DevRTT החדש מחושב:

$$DecRTT = (1 - \beta) X DevRTT' + \beta X |RTT - EstimatedRTT|$$

β ברירת המחדל ביישום TCP של היום שווה ל-0.25. אז ה-Timeout מחושב כך:

$$TimeOut = EstimatedRTT + 4 X DevRTT$$

אם האישור עבור חבילה מגיע אחרי שה-Timeout חלף (או לא מגיע כלל), החבילה נחשבת כאבודה ונשלחת שוב ע"י המקור.

DNS 2.2.2:

DNS משתמש בעיקר ב-User Datagram Protocol (UDP) כפרוטוקול מנוף רביעי. לשאלות DNS יש בד"כ מטען קטן יחסית, שנכנס לתוך חבילת UDP אחת. שאילתה מלקוח מלווה ע"י חבילת UDP אחת מהשרת. מאחר ו-UDP לא מאבחן אובדן חבילות, שרת ה-DNS חייב להיות מסוגל לייצא שליחה חוזרת בעצמו או להמשיך לעבוד ללא החבילה שאבדה. כמובן במקרה של ה-DNS, החבילה שעבדה תישלח שוב. ה-RFC הבסיסי ממליץ כי מרווח השליחה החוזרת צריך להיות מבוסס על סטטיסטיקה קודמת, אם זה אפשרי. שליחה חוזרת אגרסיבית מידי יכולה בקלות להאט תגובות עבור הקהילה ברשתות גדולות. מרווח השליחה החוזרת המינימלי תלוי בכמה טוב הלקוח מחובר לשרתים הצפויים שלו. בד"כ זה אמור להיות 2-5 שניות.

פרק 3: עבודה הקשורה לנושא

3.1 סקירת ההתקפות:

DNS הוא אחד מהשירותים החשובים ביותר ברשתות פרטיות וגם ברחבי האינטרנט. קשה לדמיין איך ללא השירות שלו היינו יכולים לגלוש באינטרנט או להשתמש בכל יישומי האינטרנט. עם זאת, הפופולריות של ה-DNS והתלות שלנו בו, הם גם החסרונות שלו. כל קישור חדש לאתר או ליישום בד"כ מתחיל בשאלת DNS לכתובת IP של היעד. אם מסיבה כלשהי ה-IP שחוזר שגוי, המשתמש יכול להמצא בסופו של דבר בידי האקרים. הסביבה העיקרית לבעיה זו היא שפרוטוקול DNS חסר כל התייחסות או תכנון של אבטחה. לכן, במשך השנים שרתי ה-DNS היו יעד קבוע של האקרים. בינתיים, האקדמיה וחברון אבטחה עשו כל מאמץ למנוע את ההתקפות ע"י תיקון שרתי ה-DNS (ובכך שיפרו את התוכנה שמיישמת את הפרוטוקול). עם זאת, לא הוכנסו שינויים לפרוטוקול עצמו.

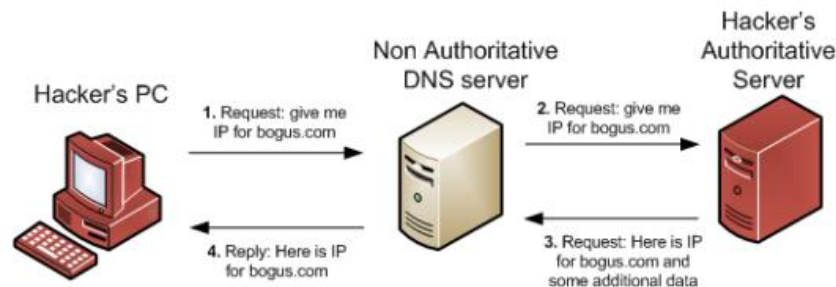
פרק זה יפרט על חלק מההתקפות הפופולריות ביותר במשך השנים ומה נעשה כדי למנוע אותן. למרות שחלק מההתקפות אלו לא אפשריות יותר, הן עדיין מעניינות היות והן מדגישות את העובדה ש-DNS לא תוכנן כפרוטוקול מאובטח ולכן יישום DNS היה בהתחלה פגום. הגבלות הבטחה פותחו במשך השנים כשההתקפות התגלו. מכאן ניתח להניח שבגרסת ה-BIND המעודכנת ביותר חלק מההתקפות עדיין לא התגלו. יש לבצע עבודה יסודית יותר כדי למנוע מהתקפות Zero-Day להצליח. מכאן, כפי שהוזכר, עבודה זו מציעה אלגוריתם לזיהוי ומניעת התקפות Zero-Day.

נתחיל בהצגת כמה מההתקפות ה-DNS הנפוצות כפי שפורסמו במשך השנים באינטרנט. כל ההתקפות ידועות אך כמה מהן עדיין מהוות איום אפילו על שרת ה-DNS המעודכן ביותר.

3.1.1 נתוני התקפה נוספים:

הרעיון מאחורי ההתקפה הזו הוא לשלוח שאילתה חוקית ולהפיק תשובה חוקית ובעקבותיה מידע נוסף. המידע הזה יתקבל למטמון של שרת ה-DNS לשימוש עתידי. 2 הדוגמאות הבאות להתקפות הן דיי ישנות ולא יעבדו על גרסאות ה-BIND החדשות, אך שווה להזכיר אותן כדי להבין את נסיונות הרעלת המטמון והנזק הפוטנציאלי שלהם. המשתמש יכול לבחון את שרת ה-DNS שלו עם האתר הבא [7] לגבי פגיעות זו.

(איור 3.1: נתוני התקפה נוספים. זרימת החבילות של 2 ההתקפות.)



מידע על התקפות שלא קשור:

ההאקר מייצר שאילתה לשרת ה-DNS עבור דומיין שלא קיים, מהדומיין הנשלט שלו. מאחר ול-DNS אין רשומות עבור הדומיין הזה, הוא ישלח שאילתה ל-Authority DNS Server של הדומיין. לאחר קבלת השאילתה, ההאקר שולח תשובה חוקית תוך הוספת נתונים. הנתונים הנוספים מכילים כתובת IP לדומיינים שלא נכללו בשאילתה המקורית. ה-IP של הדומיין שלא קיים ושל המוסף (שלא ביקשו אותו) מאוחסנים במטמון של שרת ה-DNS לשימוש עתידי.

התקפה זו לא עובדת יותר: אף דומיין לא נשמר במטמון חוץ מהדומיין המקורי בשאילתה.

מידע הקשור להתקפה:

האקר משתמש באותה טכניקה כמו ב"מידע על התקפות שלא קשור", אך הפעם המידע הוא פרטים נוספים הקשורים לדומיין הנשאל. בד"כ נתונים אלו כוללים MX, CNAME, ורשומות NS. רשומות אלו מצביעות על המידע שההאקר רוצה שייאוכסן במטמון. התקפה זו לא עובדת יותר: אף מידע חוץ מהמידע הנשאל לא נכנס למטמון.

3.1.2 ניחוש Transaction ID:

המכשול האמיתי היחיד העומד בין ההאקר לבין הרעלת מטמון מוצלחת של שרת ה-DNS הוא שדה ה-Transaction ID בפרוטוקול ה-DNS. לכן, ההאקרים מחפשים נק' חולשה ביישום הפרוטוקול שייאפשר להם לנחש טוב את ה-Transaction ID ובכך להפריע לתנועה. הסעיף הבא עוסק בשיטות שבהן משתמש ההאקר כדי להתגבר על המכשול הזה ואיך שרתי ה-DNS יכולים להיות מוגנים מפני זה.



(איור 3.2: ניחוש Transaction ID. זרימת החבילות של ההתקפה.)

תוספת של אחד:

שרתי ה-BIND הראשוניים טיפלו מעט מאוד בבעיות ההבטחה. כדי להמנע מאותן חזרות ה-Transaction ID באותו הזמן ברשת, השרת השתמש בשיטת "Increment by One". כל שאילתה חדשה הונפקה עם ה-TransactionID הישן +1. במקרה כזה, ניחוש ה-Transaction ID הוא קל מאוד. חולשה זו טופלה וגרסאות ה-BIND החדשות מנפיקות Transaction ID רנדומלי לכל שאילתה חדשה.

הנפקת מספר Transaction ID ב-BIND 9:

כפי שצוין לעיל, היה קל לנחש את מס' ה-Transaction ID בגרסאות BIND קודמות. עם זאת, בגרסה החדשה (BIND 9) מס' ה-Transaction ID נבחר באופן אקראי, או ליתר דיוק הוא: Pseudo – PRNG Random Generated Number. האלגוריתם שמפיק את ה-PRNGs בכל אחת מהגרסאות אל BIND פתוח לציבור וקל להשיג וללמוד אותו. כפי שמוצג בגרסאות BIND 9 [8]: 9.2.0-9.2.8, 9.3.0-9.3.4, 9.4.0-9.4.1, 9.0.0-9.0.1, 9.1.0-9.1.3, PRNG הוא חלש והמספר האקראי הבא יכול להיות מוסק מהקודם. בעיה ספציפית זו תוקנה בגרסת BIND מס' 9.5.0.

התקפת פרדוקס יום ההולדת:

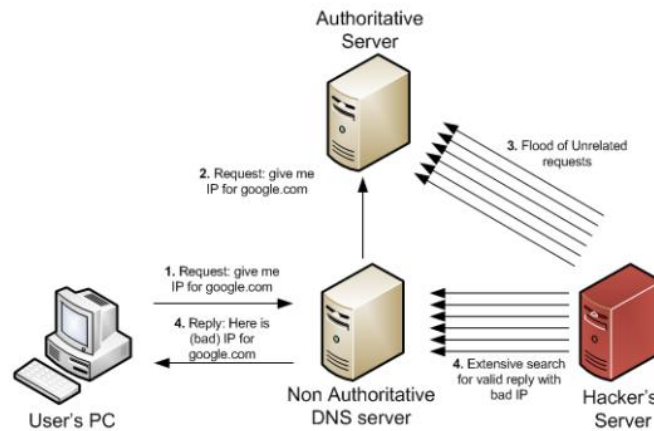
על מנת לבצע את ההתקפה הזו ההאקר שולח בהתחלה במקביל מס' חבילות גדול לשרת ה-DNS, המבקשות את אותו שם דומיין. שרת ה-DNS מייצר את אותו מס' הבקשות ושולח אותן ל-Authority Server. ההאקר מייצר את אותה כמות של תשובות DNS מזויפות עם מס' Transaction ID רנדומלי. פרדוקס יום ההולדת מכתוב שכמה מאות חבילות יספיקו להבטיח 50% הצלחה כדי להתאים בין Transaction ID של לפחות שאילתה אחת ותשובה מזויפת אחת. וכך ניתן להצליח להרעיל את המטמון של שרת ה-DNS. התקפה זו תוארה במלואה ב[9] וב[10].

התקפת יום ההולדת מבטיחה סיכויי הצלחה גבוהים עם מס' מנות דרושות נמוך. בזיוף חבילות רגיל ההאקר שולח N תשובות לשאילתה אחת, ולכן ההסתברות להצלחה היא $\frac{N}{T}$ כש-N הוא מס' החבילות שנשלחו ו-T הוא סך מס' החבילות האפשרי (במקרה של ה-DNS: $T = 2^{16} - 1 = 65535$). בהתקפת פרדוקס יום ההולדת ההאקר רק צריך להתאים את אחת השאילתות לאחת מהתשובות, ההסתברות להצלחה יכולה להיות מחושבת ע"י הנוסחה הבאה:

$$P(success) = 1 - 1 \left(1 - \frac{1}{T}\right) \left(1 - \frac{2}{T}\right) \dots \left(1 - \frac{N-1}{T}\right) = 1 - \frac{T!}{T^N(T-N)!}$$

הכוח של "התקפת פרדוקס יום ההולדת" על "התקפת זיוף רגיל של חבילות" הוא שהיא דורשת יחסית מס' קטן של חבילות כדי ליצור התקפה מוצלחת. רק 300 חבילות יבטיחו 50% הצלחה בעוד ש-750 חבילות יבטיחו 99% הצלחה. בזיוף חבילות רגיל התקפה עם 750 חבילות תבטיח רק $\frac{750}{65535} = 1.14\%$ הצלחה.

התקפת פרדוקס יום ההולדת מראה שאפילו Transaction ID שמונפק באופן אקראי המשמש את גרסאות ה-BIND האחרונות, פגיע להתקפות.

3.1.3 התקפות הצפה:

(איור 3.3: התקפת הצפה)

על מנת להצליח בהתקפה, ההאקר זקוק לזמן רב ככל שאפשר. הזמן שהוא יכול לנצל הוא כאשר שרת ה-DNS מייצר את הבקשה ונגמר כאשר התשובה מתקבלת משרת ה-Authority DNS. בד"כ זה לא מספיק זמן כדי להריץ חיפוש נרחב אחר ה-Transaction ID החוקי על החבילות. כדי להרוויח עוד זמן, ההאקר יכול לנסות להציף את שרת ה-Authority DNS בכדי להאט אותו ואולי להפיל אותו. התקפת ההצפה דומה להתקפת ניחוש ה-Transaction ID והיא מוצגת באיור 3.3.

3.2 פתרונות:

יש מס' פתרונות המציעים כיצד להתמודד ולמנוע התקפות הרעלת מטמון ונסינות להתקפות [11], [12], [13], [16]. בפרק זה אנו מציגים חלק מהם:

3.2.1 גרסאות BIND:

BIND היא תוכנת ה-DNS הכי נפוצה ברחבי האינטרנט [3], ולכן היא יעד קבוע להתקפות האקרים. גרסאות חדשות ועידכון גרסאות משוחררים באופן קבוע עם עדכונים ותיקונים חדשים של באגים ובעיות אבטחה. לכן, אחת הדרכים החשובות ביותר שמישהו יכול לשפר את האבטחה של השרת ה-DNS המקומי שלך הוא להריץ את הגרסה האחרונה ביותר של BIND, מאחר וכל הגרסאות הקודמות לאחרונה כנראה רגישות לפחות לכמה התקפות ידועות [8]. עם זאת, התקפות חדשות כנראה עוד לא תוקנו ע"י הגרסה החדשה ולכן יכולים עדיין להזיק למע' שלך. דרך אחת לגלות על התקפות חדשות ודרכים להתגונן מפניהן היא לקרוא את הדיון: comp.protocols.dns.bind באופן קבוע [12].

3.2.2 תצורה וניהול DNS

אחת הטעויות הנפוצות ביותר היא תצורה לא נכונה של ה-DNS. כאשר התצורה שלו שגויה, אפילו שרת ה-DNS המעודכן ביותר פגיע להרבה התקפות ידועות (לא בהכרח הרעלת מטמון) כפי שתואר ב[13], [14] ו-[15]. לדוגמא, אחת התצורות החשובות ביותר הוא מגבלת פקודות Zone Transfers לכתובות IP מוכרות ומהימנות בלבד.

כאשר אתה בטוח שה-DNS המקומי הותקף, יש מס' צעדים שניתן לעשות כדי למנוע נזק נוסף:

- ודא שאתה לא נגוע בתוכנות ריגול. תוכנות ריגול רבות משנות את תצורת ה-DNS.
- נסה לברר את כתובת ה-IP של שרת ה-DNS הזדוני.
- הכנס את כתובת ה-IP הזדונית ל-IPs Black List.
- נקה את המטמון של שרת ה-DNS ע"י איפוס השרת או ע"י כל פקודות ניקוי מטמון אחרות.

שיטות טובות אחרות כשעוסקים בתצורה הן אבטחה פיסית של שרתי ה-DNS, הסתרת הגרסה של ה-BIND שפועל [17], הסרת כל השירותים המיותרים הרצים על שרת ה-DNS, הפיכת יציאת המקור UDP לרנדומלית [18] וניטור באופן קבוע של שרתי ה-DNS וקבצי ה-log של ה-DNS.

3.2.3 שינויי שאלות:

ההנחה המרכזית בפתרונות הבאים היא כי פרוטוקול DNS לא ישתנה בזמן הקרוב, ולכן חלק מהפתרון לשלמות הפעולות חייב להיות במגבלות המפרטים הנוכחיים. הרעיון הבסיסי הוא למצוא שדות שניתנים לשינוי בדרך כלשהי, ושינוי זה יגדיל את מס' השאילתות לכזו כמות שלהאקר לא יהיה סיכוי למצוא ולשלוח את השינוי הנכון של התשובה לפני שהאמיתית תגיע מה-Authentication Server.

- [19] Bit 0x20: הרעיון הוא להשתמש בסעיף השאלות כדי להוסיף ביטים ראנדומליים לשאילתה. שרת ה-DNS יצטרך להתאים את הביטים הללו בכל תשובה לגיטימית שהוא מייצר. לרוב שרתי ה-DNS לא משנה אם השאלה מוצגת בתיבה העליונה או התחתונה, ולכן שילוב של התיבות יכול לספק את הביטים הרנדומליים ההכרחיים לשאילתה. ההבדל בין "A" (0X41) ל-"a" (0X61) או "Z" (0X5a) ל-"z" (0X7a) הוא הביט 0X20 (ומכאן שם האלגוריתם). לדוגמא:

• WWW.IETF.ORG	000	0000	000
• WwW.iEtF.oRg	010	1010	101
• wWw.IeTf.OrG	101	0101	010
• www.ietf.org	111	1111	111

- מס' הביטים שמיוצר בשיטה זו הוא פונקציה של אורך הדומיין: לדומיינים ארוכים יהיו הרבה ביטים רנדומליים ודומיינים קצרים ישארו רק עם מס' ביטים קטן וישאירו את האתרים שלהם פגועים להתקפות ה"ל.
- קידומת אקראית [20]: שיטה זו מציעה להשתמש ב-Wildcard Domain Names כדי להגדיל את האנטרופיה. לדוגמא: אם משתמש רוצה לפענח את www.example.com, שרת ה-DNS ייצר קידומת אקראית לשאלתה וישלח: ra1bc3twqj.www.example.com. שרת ה-Authoritative DNS יחזיר את אותו שם הדומיין עם כתובת ה-IP של www.example.com. שיטה זו המשתמשת בקידומת של אורך 10 תייצר בערך $\log_2 36^{10} \approx 52$ ביטים.

3.2.4 אבטחת DNS:

מאחר ופרוטוקול DNS לא כולל אבטחה, פותחו (DNSSEC) Domain Name System Security Extensions כפי שתואר ב-RFC 3833 [16], [22]. DNSSEC תוכנן למניעת הרעלת מטמון ע"י חתימה דיגיטלית של כל התשובות שלו, כך שניתן בקלות לאמת את נכונות ושלמות המידע. DNSSEC הוא פרוטוקול חדש ורק לאחרונה כמה מהחלקים העיקריים שלו הוגדרו באופן רשמי. פריסת הפרוטוקול ברשתות בקנה מידה גדול היא משימה מאתגרת. הסיבה לכך היא שנדרשת רמה מינימלית של פריסה כדי שיוזר יפיק תועלת מהשירות החדש. קשה למצוא את המשתמשים הראשונים שלמשך תקופה מסויימת יהיו מוכנים לשלם את המחיר של שימוש בפרוטוקול חדש.

3.2.5 התרומה שלנו:

בעבודה זו אנו מציעים אלגוריתם שמתמודד עם הסוג חזק יותר של האקרים מזה שתואר בפתרונות הקודמים (פרט ל-DNSSEC). ההאקר שלנו יכול לבחון (אך לא לשנות) כל חבילה שנשלחה או התקבלה ע"י שרת ה-DNS המקומי. הפתרון שלנו יכול להיות מיושם כקופסה שחורה ישר אחרי שרת ה-DNS, ולכן אין דרישות לשינויים לא בפרוטוקול DNS ולא בקוד השרת של BIND.

פרק 4: אלגוריתם Delay Fast Packets

כפי שהוזכר קודם לכן, פרוטוקול DNS משתמש בד"כ ב-User Datagram Protocol (UDP) כפרוטוקול מדרגה רביעית לתקשורת הנתונים שלו. אם מסיבה כלשהי, השאלתה או התשובה לא מגיעים ליעד שלהם, שרת ה-DNS שולח בקשה חוזרת. במקרה כזה, שרת ה-DNS צריך לדעת איך להתמודד עם מצב שנובע מעיכוב חבילות היות וזה יכול להיות מתורגם בטעות כאיבוד חבילות. הדרך שבה ה-DNS פועל היא הדרך הכי פשוטה: קבלת התשובה החוקית הראשונה (תשובה חוקית היא תשובה מ-Authoritative Server עם Transaction ID נכון) והתעלמות מכל שאר התשובות. במילים אחרות, רק התשובה הראשונה תועבר למשתמש ותישמר במטמון לשימוש עתידי. זהו חיסרון באבטחה של ה-DNS ודרך להאקרים להתקפת ה-DNS והרעלת המטמון.

פרק זה מציג את "Delay Fast Packets algorithm" או בקיצור אלגוריתם DFP המאתר ומונע נסיונות להתקפת הרעלת מטמון. הרעיון של אלגוריתם DFP הוא הוא העניין הבא: בכדי ש"ניסיון התקפה" יהפוך ל"התקפה מוצלחת" הוא חייב לנצח את התשובה האמיתית (מ-Authoritative Server) במרוץ חזרה לשרת ה-DNS המקומי. כדי להצליח בכך, ההאקר מנסה לשלוח תשובה בהקדם האפשרי, בד"כ הרבה יותר מהר משזה לוקח ל-Authoritative Server לייצר תשובה. אלגוריתם ה-DFP שלנו מזהה את הנק' הזו ולכן מאבחן ומונע את ההתקפות.

4.1 הצגת אלגוריתם DFP:

הרעיון המרכזי של האלגוריתם הוא להעריך את ה-RTT בין שרת ה-DNS לכל אחד מה-Authoritative Servers שהוא פוגש אי פעם ומעכב את התשובות שמגיעות מהר מידי בהתאם להערכה (לכן הוא נקרא אלגוריתם DFP – Delay Fast Packets). יתרה מכך, אלגוריתם DFP מעריך את ה-RTT לכל סוג שירות שה-Authoritative Server יכול לספק (MX, A, AAAA, CNAME, PTR וכו'). לחלק משירותים אלו הערכת TRR גבוהה יותר מלאחרים בגלל חיפוש Data Base נרחב יותר בצד ה-Authoritative. לכל Authoritative Server וסוג שירות, ה-Estimated RTT מנבא את הזמן הממוצע הנדרש לתשובה הבאה להגיע. אם מסיבה כלשהי, התשובה מגיעה מוקדם מידי לפי אלגוריתם DFP, שרת ה-DNS מחכה במשך זמן מסויים. אם תשובה חוקית נוצפת מגיעה בחלון הזמן הזה, 2 התשובות נזרקות ושאלתה חדשה נוצרת (כאילו התרחש איבוד חבילות DNS רגיל). אם התוקף מתעקש ושולח תשובה לכל שאלתה, המשתמש יחוה התקפת Denial of Service היות ואלגוריתם DFP לא יעביר אף אחת מהתשובות חזרה למשתמש. התקפה זו נקראת (Denial of Service) DoS ונושא זה לא נסקר בעבודה זו.

בסעיף הבא (4.2) אנו מציגים pseudocode פשוט המדגים את הרעיון של אלגוריתם DFP. כדי להעביר את הרעיון מבלי לסבך את ה-pseudocode, אלגוריתם ה-DFP הבא לא עוסק בהתקפה מרובת חבילות. אם ההאקר מחליט לשלוח תשובות "חוקיות" רבות עבור שאלתה אחת, אלגוריתם ה-DFP הבא יזרוק רק את ה-2 הראשונות ויכשל בזריקת התשובות הבאות של ההאקר. עם זאת, הכללת אלגוריתם DFP היא פשוטה: לכל חבילה מהירה, כל תשובה לאותה השאלתה חייבת להשמר לתקופה מסויימת, ואז כל התשובות חייבות להזרק. כמובן כדי למנוע הצפה בזיכרון של שרתי ה-DNS, רק המידע המינימלי הרלוונטי חייב להשמר, כמו 5-tuple והזמן עד שהתשובות כבר לא תקפות (ה-DNS timeout הרגיל).

סוגיה לא מטופלת נוספת באלגוריתם DFP היא שחבילות "מהירות מידי" משפיעות על הערכת RTT. במקרה שבו אין התקפות, חבילות "מהירות מידי" אלו יכולות לגרום לשינוי בטופולוגיה של הרשת ולכן חייבות להחשב בהערכת ה-RTT. אך במקרה של התקפות אפשריות, זה יהיה רעיון רע להכליל אותן בהערכות כי הן יכולות להיות ניסיון של ההאקר להוריד את הערכת ה-RTT כדי ליצור התקפה מוצלחת בעתיד הקרוב. לכן, חבילות "מהירות מידי" חייבות לא להשפיע על ה-RTT עד שנהיה בטוחים שהן

תשובות אמיתיות וחוקיות ולא ניסיונות להתקפה. זה יהיה ידוע רק כשיעבור פרק זמן מסויים ולא יגיעו תשובות נוספות במהלך פרק הזמן הזה.

4.2 Pseudocode:

Algorithm 1 DFP - Delay Fast Packets

Main Method

```

1: PacketDictionary.Init()
2: StatsDictionary.Init()
3: loop
4:   NewPacket  $\leftarrow$  SniffDNSPacket()
5:   ID  $\leftarrow$  NewPacket.TransactionID + NewPacket.Type
6:   5Tuple  $\leftarrow$  NewPacket.5Tuple
7:   IsQuery  $\leftarrow$  NewPacket.IsQuery
8:   if IsQuery then
9:     if [5Tuple, ID] already in PacketDictionary then
10:      PacketDictionary[5Tuple, ID] = NewPacket
11:      print DUP REQUEST FOUND: REMOVING PACKET AFTER TIMEOUT
12:     else
13:      PacketDictionary.Insert([5Tuple, ID],NewPacket)
14:     end if
15:   else
16:     RequestPacket  $\leftarrow$  PacketDictionary.Get([5Tuple, ID])
17:     if [5Tuple, ID] is delayed then
18:       Drop all packets with ID [5Tuple, ID]
19:       SEND: WARNING, 'TOO FAST' PACKET RECEIVED, POSSIBLE ATTACK
20:     else
21:       if RequestPacket not exists then
22:         SEND: WARNING, UNMATCHED REPLY FOUND, POSSIBLE ATTACK
23:       end if
24:       RTT  $\leftarrow$  NewPacket.TimeOfArrival – RequestPacket.TimeOfArrival
25:       DelayTime  $\leftarrow$  StatsDictionary[[NewPacket.SourceIP,
NewPacket.ReplyType].AddSample(RTT,NewPacket.SourceIP,
NewPacket.ReplyType])
26:       DelayPacket(DelayTime)
27:       PacketDictionary.Remove([5Tuple, ID])
28:     end if
29:   end if
30: end loop

```

AddSample(RTT, IP, Type)

```

1: if First Sample for this IP and TYPE then
2:   EstimatedRTT  $\leftarrow$  RTT
3:   DevRTT  $\leftarrow$  0
4: end if
5: EstimatedRTT  $\leftarrow$   $(1 - \alpha) \times$  EstimatedRTT +  $\alpha \times$  RTT
6: DevRTT  $\leftarrow$   $(1 - \beta) \times$  DevRTT +  $\beta \times |RTT - EstimatedRTT|$ 
7: if RTT < EstimatedRTT - DevRTT  $\times$  FactorWindow then
8:   return (EstimatedRTT + DevRTT  $\times$  FactorWindow) - RTT
9: else
10:  return 0
11: end if

```

4.2.1 תיעוד אלגוריתם DFP:

השיטה העיקרית (מפורטת לפי שיטות קוד):

(1,2): אלגוריתם DFP משתמש ב-2 PacketDictionary:hashtables כדי להתאים בין השאליות היוצאות לתשובות הנכנסות, ו-StatsDictionary לאיחסון הסטטיסטיקות לכל שרת Authoritative DNS.

(3): לולאה ראשית. רצה ללא הגבלת זמן כל עוד התוכנית פועלת.

(4): קבלת חבילה חדשה מהתקן הרשת.

(5,6,7): חילוץ הפרמטרים מהחבילה החדשה, קבלת ה-5-Tuple, ה-Transaction ID, Packet Type וה-IsQuery flag.

(8): ניהול חבילות השאלתה.

(9): בדיקה אם יש חבילה עם אותו 5-Tuple ו-ID ב-PacketDictionary hashtable.

(10,11): החלפת הבקשה הישנה עם הבקשה החדשה ב-PacketDictionary hashtable. הערה: מצב זה יכול לקרות רק אם אף תשובה לא התקבלה והשאלתה החדשה היא בעלת אותו 5-Tuple ו-ID כמו הבקשה הישנה.

(12,13): הכנסת השאלתה החדשה ל-PacketDictionary hashtable.

(15): ניהול חבילות התשובה.

(16): קבלת חבילת השאלתה המתאימה עם ה-5-Tuple וה-ID הנכונים.

(17): בדיקה שלאף תשובה מעוכבת אין את אותו ID כמו של התשובה הנכונה.

(18,19): אם נמצאו הרבה ID, זרוק את כולם ושלח אזהרה.

(21,22): אם לא נמצאה שאלתה מתאימה (יש לנו תשובה ללא שאלתה), נשלח אזהרה היות ומצב זה לא יכול להתממש אף פעם אלא אם כן יש התקפה על שרת ה-DNS.

(23): ניהול המצב הרגיל שבו נמצאו שאלתה ותשובה.

(24): חישוב ה-RTT בין שרתי ה-DNS ע"י מדידת ההבדלים בין זמן ההגעה של השאילתה ושל התשובה.

(25): הרצת AddSample method על המדגם החדש וה-RTT.

(26): מניעה מחבילת התשובה להכנס לליבת שרת ה-DNS, לפרק הזמן שנקבע ע"י AddSample method.

(27): הסרת הבקשה מה-PacketDictionary Hashtable.

AddSample:

(1): בדיקה אם זה המדגם הראשון עבור ה-IP וה-Type.

(2,3,4): איתחול הפרמטרים הסטטיסטיים.

(5): חישוב ה-EstimatedRTT עם ה- α הנתונה.

(6): חישוב ה-DevRTT עם ה- β הנתונה.

(7): בדיקה האם החבילה מהירה מידי בהתאם לחלון הנקבע ע"י הפרמטרים EstimatedRTT, DevRTT, FactorWindow.

(8): אם זוהי חבילה מהירה, להחזיר את מרווח הזמן שבו היא חייבת להיות מעוכבת.

(9,10,11): אחרת, להחזיר 0.

4.3 מפרטי אלגוריתם DFP:

אלגוריתם DFP משתמש בנוסחה הבאה כדי לאתר חבילות "מהירות מידי":

$$RTT < EstimatedRTT - devRTT \times FactorWindow$$

כל תשובת DNS המגיעה מוקדם מידי עפ"י הנוסחה נחשבת לחשודה ומעוכבת, כדי לתת מספיק זמן לתשובה אחרת עם אותו TID להגיע. המשתנים בנוסחה נשלטים ע"י 3 פרמטרים: α , β ו-FactorWindow. הביצועים של אלגוריתם DFP, במונחי מהירות, דיוק הזיהוי והקצאת זיכרון תלויים בטיב התצורה שלהם. בפרק זה אנו מתארים את השיקולים והניסויים שהובילו אותנו לבחור בערכים של 3 פרמטרים אלו.

4.3.1 חלון:

הגדרה 4.1 מגדירה **חלון** כנק' זמן כלשהי המתחילה לאחר שליחת בקשת ה-DNS. כל תשובה המגיעה לפני שהחלון מתחיל נחשבת לחשודה. לדוגמא: ל-www.xkcd.com שרת Authoritative DNS עם סוג A, החלון יכול להתחיל ב-3400 ms, ואילו לסוג MX יכול להתחיל ב-3800.

הגדרה 4.2 מגדירה **נקודת התחלת חלון** כזמן שעבר מאז הבקשה לתחילת החלון.

הגדרה 4.3 מגדירה **הזעקת שווא** כחבילה שמגיעה מה-AuthoritativeServer לפני נק' התחלת החלון ואפילו אם מספיק זמן עובר, לא מגיעה תשובה נוספת עם אותו TID. כל אחת מהזעקות שווא אלו גורמת לשרת ה-DNS לשמור את החבילה בזיכרון לזמן קצר ולשחרר אותה רק אחרי שזה בטוח.

החלון הוא בעל אופי דינמי מאוד, נק' ההתחלה שלו משתנה כל הזמן וזזה על ציר הזמן. זאת בשל האופי הדינמי של רשת האינטרנט והשינויים התמידיים של ה-RTT של השאלות המגיעות.

נק' התחלת החלון מתייבה אילו חבילות נחשבות "מהירות מידי" ולכן צריך לעכב אותן, ואילו חבילות בגבול הזמן הנורמלי ולכן יכולות לעבור ללא עיכוב. כדי להתאים את הפרמטרים המגדירים את נק' התחלת החלון, יש להתחשב ב-2 נקודות מרכזיות:

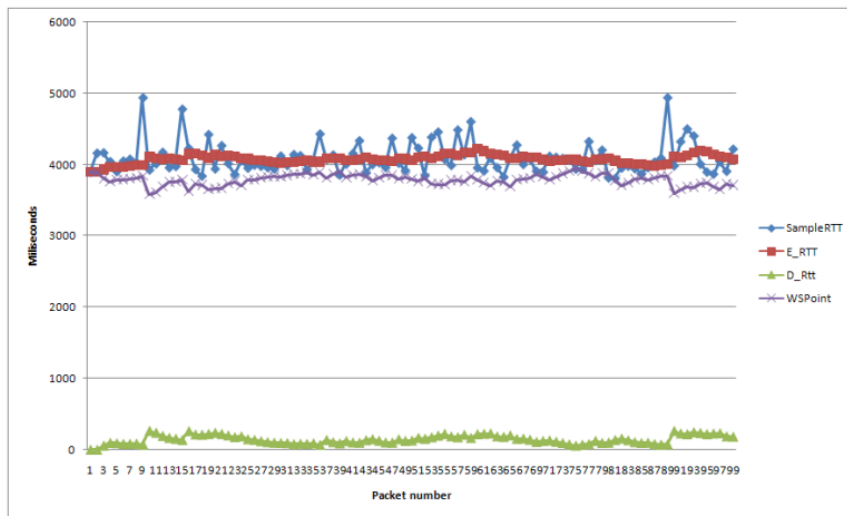
- נק' התחלה מוקדמת מאפשרת ליותר חבילות לעבור ללא עיכוב. מצד אחד, זה עוזר לשרת ה-DNS לעבוד מהר יותר ללא עיכוב של יותר מידי חבילות חשודות (עד שזה בטוח להעביר אותן הלאה). אך מצד שני, התקפות מתוכננות היטב ינסו לפגוע בדיוק מעל נק' ההתחלה ובכך לעבור את הפילטר ללא הפעלת ההזעקה.
- נק' התחלה מאוחרת מעכבת יותר חבילות היות והיא מחשיבה אותן ל"מהירות מידי". מצד אחד, זה הופך את העבודה של ההאקר להרבה יותר קשה מכיוון שהוא חייב להתחרות על מרווח זמן הרבה יותר קטן מה שהופך לניסיון הרעלת המטמון שלו להרבה קשה יותר. אך מצד שני, נק' התחלה מאוחרת מאלצת את שרת ה-DNS לעכב הרבה חבילות ולהחשיב אותן כאיום פוטנציאלי, זה גורם לתשובות איטיות יותר של שרת ה-DNS למשתמשים וצריכת זיכרון גדולה יותר בשרת ה-DNS.

בפרקים הבאים (4.3.2 ו-4.3.3) אנו מתייחסים ל- α , β ו-FactorWindow. אנו מבצעים סט ניסויים כדי להדגים את ההשפעה של כל אחד מהפרמטרים על נק' התחלת החלון ומכאן על ה-tradeoff בין מס' הזעקות השווא לסיכוי לאתר ולמנוע התקפה. יש לשים לב כי על מנת להדגיש את התוצאות בצורה ברורה, 2 הסטים של הניסויים נעשו על מערכי נתונים שונים (בעוד שהמסקנות זהות על כל המע' נתונים הזמינות לרשותינו, ניסינו למצוא גרפים ברורים ככל שניתן).

4.3.2 שיקולי α ו- β :

הם β ו- α הם 2 פרמטרים המשפיעים על EstimatedRTT ועל DevRTT באלגוריתם DFP. הם קובעים את המשקל של מדגם ה-RTT החדש לעומת ההיסטוריה, ובכך משפיעים על נק' התחלת החלון. כדי למצוא איך α ו- β משפיעים על המס' של ה"חבילות המהירות" המזוהה ע"י אלגוריתם DFP. ביצענו כמה ניסויים על תנועה אמיתית ללא ניסיונות התקפה. בכל ניסוי נקבע מס' ההזעקות ה-*false positives*. האיורים הבאים 4.1, 4.2, 4.3 מראים את תוצאות הניסויים שהורצו בדיוק על אותה התנועה, עם ערכי α ו- β שונים בכל פעם.

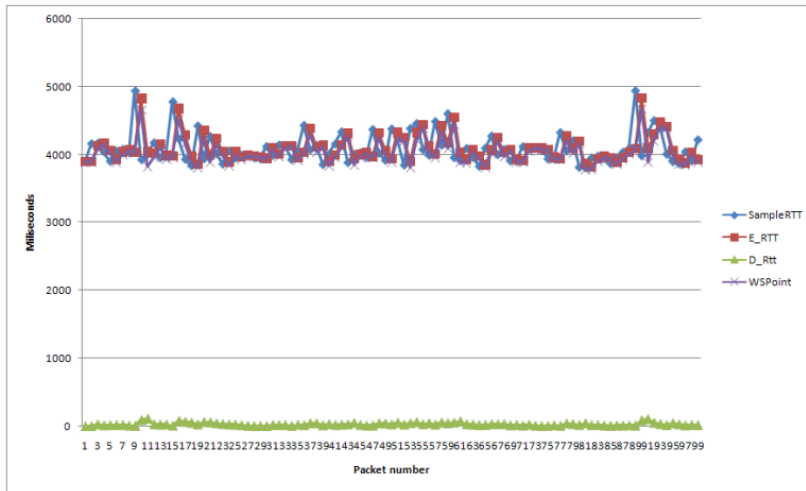
הערה: הפרמטר FactorWindow הוגדר כ-2 בכל אחד מהניסויים הבאים.



(איור 4.1: עוסק במצב של ערכים נמוכים של α ו- β , $\alpha=0.125$, $\beta=0.25$, $\#FastPackets=2$)

ערכי α ו- β נמוכים כמו בהערכת TCP RTT מחליקים את פונקציית ה-RTT Estimated מאחר וניתן יותר משקל להיסטוריה של המדגמים מאשר למדגם החדש ביותר בהשוואה לערכי α ו- β גבוהים יותר בניסויים הבאים. בכל אחד מהניסויים ניתן למדגם החדש משקל רב יותר והוא מנבא טוב יותר את ה-RTT העתידי. Estimated RTT מיוצג ע"י עקומה אדומה בגרף. עם זאת, לכל פיק ב-RTT (עקומה בצבע כחול כהה) ה-DevRTT (עקומה ירוקה) עולה, כפי שמצופה ממשוואת נק' התחלת החלון. (הערה: לדוגמא הפיק ב-RTT והעליה ב-DevRTT בחבילה מס' 10). כתוצאה מכך, נק' התחלת החלון יורדת. נק' התחלת החלון מוצגת בצבע סגול. המצב מאפשר לחבילות רעות פוטנציאליות חלון הזדמנויות רחב יותר לתקוף את שרת ה-DNS.

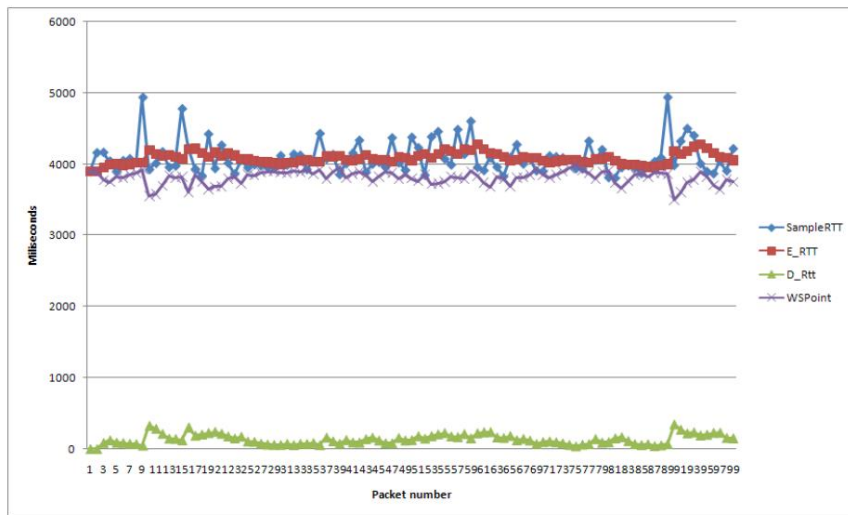
אנו רואים שרק 2 חבילות נחשבו כ"מהירות מידי" בתצורה זו: חבילה מס' 40 וחבילה מס' 79. רואים בגרף שזמן RTT של חבילות אלה חרג מנק' ההתחלה.



איור 4.2: עוסק במצב של ערכים גבוהים של α ו- β . $\alpha=0.875$, $\beta=0.75$, $\text{FastPackets} = 23$

ערכים גבוהים של α ו- β נותנים את רוב המשקל למדגם החדש ביותר (בהשוואה לניסויים אחרים). ולכן סטיית ה-RTT (עקומה ירוקה) קטנה מאוד. נק' התחלת החלון מאוחרת מאוד (עקומה סגולה) מה שמקשה על הצלחת הנסיונות להתקפת ה-DNS. מצב זה יוצר גם הרבה הזעקות שווא, כי כל תנודה ב-RTT כנראה תשים את הדגימה החדשה לפני נק' התחלת החלון.

אנו רואים שבתצורה זו כחמישית מכלל החבילות נחשבו "מהירות מידי".



איור 4.3: עוסק במצב של ערכים בינוניים של α ו- β . $\alpha=0.2$, $\beta=0.4$, $\text{FastPackets} = 5$

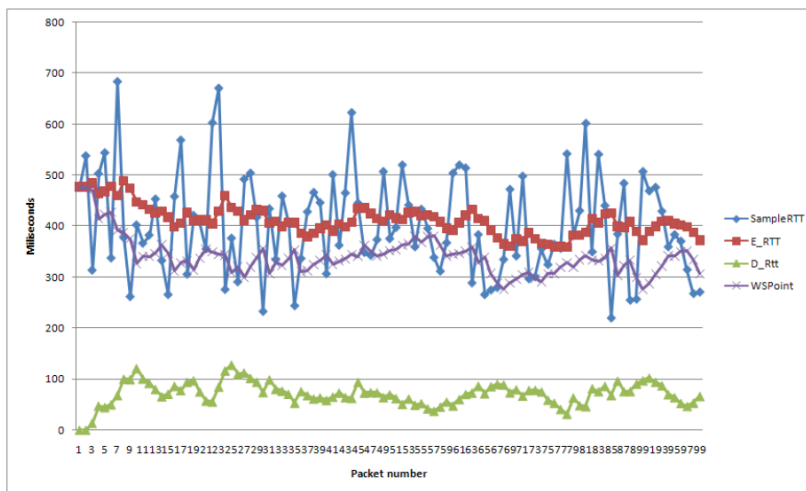
ערכי α ו- β הן הממוצע של תצורות "גבוהים" ו"נמוכים". נק' ההתחלה שנוצרה במקרה זה מאוחרת יותר מתצורות ה"נמוכים" וסטיית ה-RTT (עקומה ירוקה) גבוהה יותר מתצורות ה-"גבוהים".

אנו רואים שבתצורה זו 5 מהחבילות נחשבו ל"מהירות מידי".

הניסויים שלנו מראים כי רוב הזמן הסטייה של RTT יחסית נמוכה, ולכן נקודת ההתחלה היא דיי גבוהה. השינוי בסטייה מתרחש כאשר מגיעה חבילה איטית מאוד. במצב זה נק' התחלת החלון נמוכה יותר לתקופת זמן קצרה, ולהתקפות אפשריות יש יותר סיכויים להצליח. למרות זאת, הניסויים שלנו מראים כי מצב החבילה האיטית מתרחש לעיתים רחוקות ואנו חושבים כי חשוב יותר למנוע הזעקות שזו שכיחות להיווצר ע"י חבילות חוקיות "מהירות מידי". שיקול זה הביא אותנו להחלטה שהתצורה הנבחרת ביישום שלנו של אלגוריתם DFP הוא גרסת ה"נמוכים". עם זאת, אם שרת ה-DNS המקומי יכול להרשות לעצמו לשמור על צריכת זיכרון גדולה יותר שנוצרת ע"י הזעקות שזו, התצורה הטובה יותר היא זו שמונעת יותר התקפות ולכן במקרה זה עדיף לבחור ב"בינוניים" או אפילו (אם זיכרון לא מהווה בעיה) תצורת ה"גבוהים".

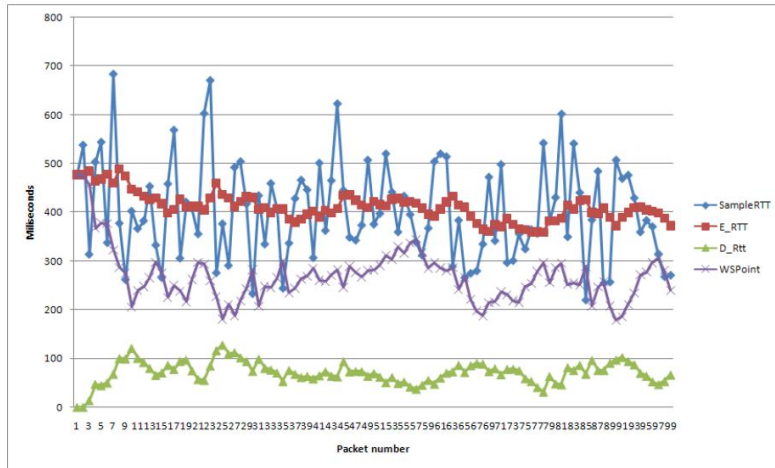
4.3.3 שיקולי FactorWindow

לאחר הגדרת הפרמטרים α ו- β , תצורת הפרמטר FactorWindow צריכה להקבע. מטרת הפרמטר היא להוריד את נק' ההתחלה הנוצרת ע"י α ו- β , ע"י גורם קבוע. כפי שצוין קודם, ה-tradeoff בין מס' הזעקות השווא והסיכוי להתקפה מוצלחת מכתוב איזה ערך ייבחר. הערה: הפרמטרים α ו- β מוגדרים ל-0.125 ו-0.25 בהתאמה בכל אחד מהניסויים הבאים.



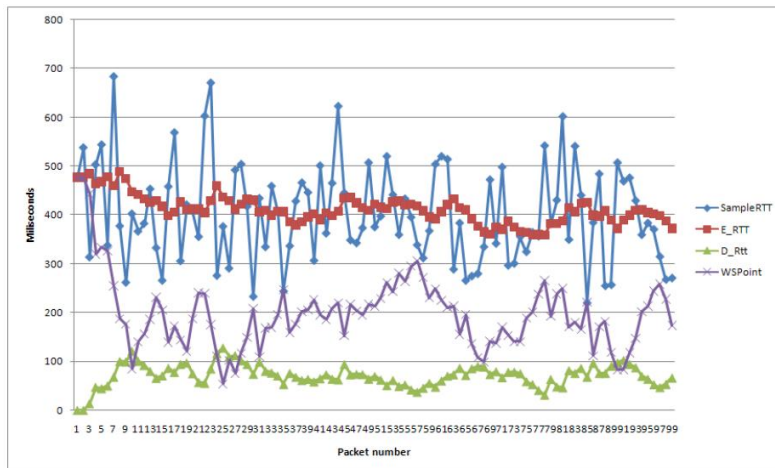
איור 4.4: עוסק במצב שבו FactorWindow=1. #FastPackets=28

FactorWindow = 1 אומר שנק' התחלת החלון מותאמת רק ע"י α ו- β ולכן נק' ההתחלה שנוצרת גבוהה והסיכוי שחבילה תגיע לפני נק' ההתחלה הוא גבוה. במקרה זה הרבה חבילות צריכות להיות מעוככות. אנו רואים שבתצורה זו כשליש מהחבילות נחשבות ל"מהירות מידי".



איור 4.5: עוסק במצב שבו $FactorWindow=2$, $\#FastPackets=9$

$FactorWindow = 2$ אומר שנק' ההתחלה היא 2 סטיות משוערות מה-RTT המשוער. ע"י שימוש ב- Chebyshev inequality, ההסתברות שחבילה תעבור את נק' ההתחלה היא פחות מרבע. אך בפועל, הקשר חזק יותר. הניסויים שלנו מראים שרק כ- $\frac{1}{10}$ מהחבילות נחשבות "מהירות מידי".
אנו רואים שבתצורה זו 9 מהחבילות נחשבו ל"מהירות מידי".



איור 4.6: עוסק במצב שבו $FactorWindow=3$, $\#FastPackets=1$

$FactorWindow = 3$ אומר שנק' ההתחלה היא 3 סטיות משוערות מה-RTT המשוער. ע"י שימוש ב- Chebyshev inequality, ההסתברות שחבילה תעבור את נק' ההתחלה היא פחות מ- $\frac{1}{9}$. אך בפועל, כמעט

אף חבילה לא עוברת את נק' ההתחלה. היא כ"כ נמוכה שהתוקף יכול להסתכן בקלות, אפילו בלי לדעת שרץ שם אלגוריתם DFP לזיהוי ומניעה.

אנו רואים שבתצורה זו רק אחת מהחבילות נחשבת ל"מהירה מידי".

השיקול העיקרי לבחירת תצורת הפרמטר FactorWindow היא שוב, ה-tradeoff בין מס' ה- false positives לסיכוי להתקפה מוצלחת. בשימוש התוצאות של הניסויים, הערך של FactorWindow אמור להיות בין 2 ל-3. אנו החלטנו שאלגוריתם DFP ירוץ עם $FactorWindow = 2$.

4.4 חולשות:

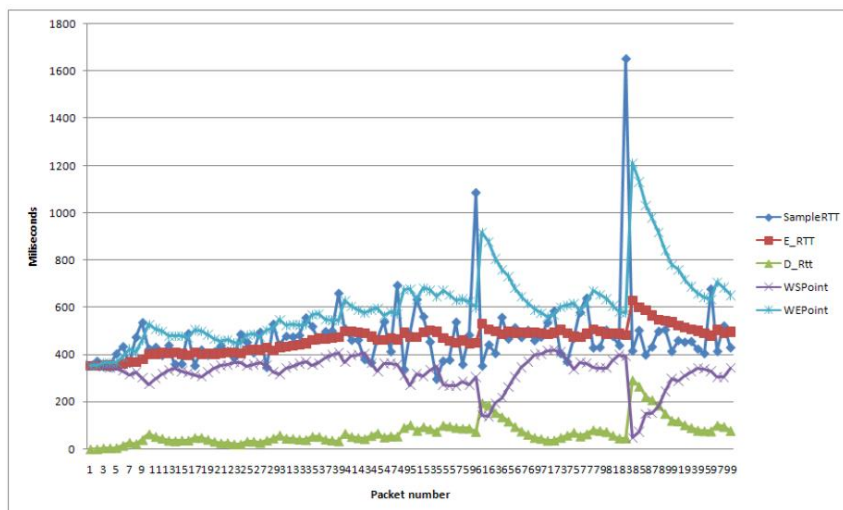
הגדרה 4.4 מגדירה נקודת סיום חלון כנק' מעל ה- $Estimated\ RTT$. כל תשובה המגיעה אחרי נק' הסיום תחשב כחבילה "איטית מידי".

ההנחה העיקרית של אלגוריתם DFP היא שהסטייה מה- $EstimatedRTT$ קרובה לאפס. כפי שיודגם בפרק "סימולציות", הנחה זו הוכחה כנכונה בניסויים רבים שבוצעו על תנועה אמיתית. אך במקרים מסויימים, הסטייה עולה לפרק זמן קצר. ברגעים אלו, נפתחת הזדמנות להתקפות פוטנציאליות מכיוון שהרבה חבילות "מהירות מידי" נופלות לנק' התחלת החלון של קשר:

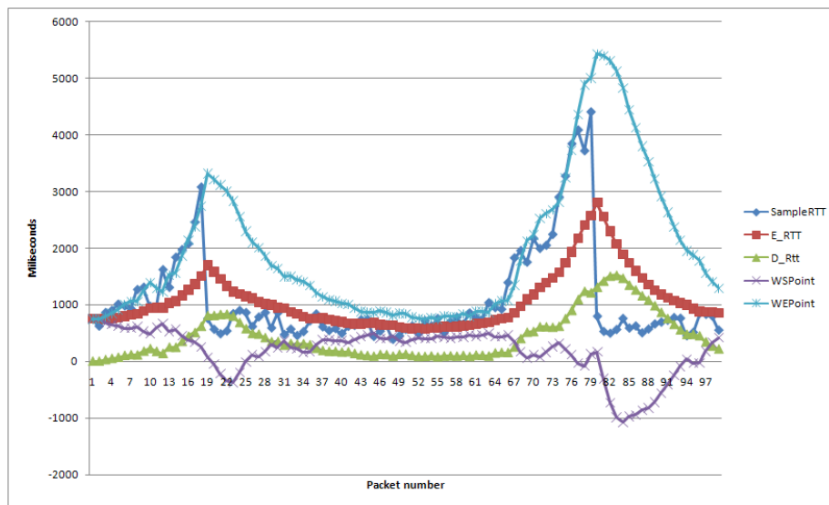
$$(EstimatedRTT - DevRTT) \times FactorWindow$$

מצב זה מתרחש לאחר שמתקבלת חבילה מאוד איטית. החבילה ה"איטית מידי" יוצרת עליה זמנית של הסטייה ומורידה את נק' ההתחלה, כפי שרואים בתמונות 4.7 ו-4.8. נק' ההחלה חוזרת לערכים נורמליים לאחר כ-3-5 חבילות, כשההשפעה של החבילה ה"איטית מידי" ההיסטורית נחלשת (ההשפעה תלוי בתצורת α ו- β שמשמשת את אלגוריתם DFP, כפי שהוסבר בסעיף הקודם). הירידה הזמנית של נק' ההתחלה יוצרת הזדמנות להאקר לתקוף את שרת ה-DNS.

דוגמא 1:



(איור 4.7: נק' התחלה נמוכה שנוצרה ע"י פיקים גבוהים.)



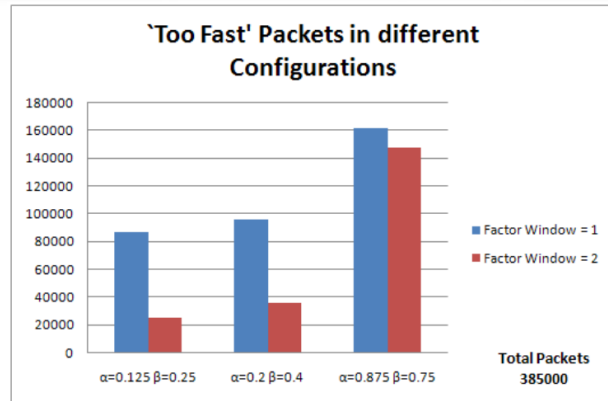
(איור 4.8: נק' התחלה שלילית.)

הדרך למנוע חולשה זו היא להסיר חבילות "איטיות מידי" מהחישוב של הסטייה. ובכך למנוע את הירידה הזמנית של נק' ההתחלה. עם זאת, אלגוריתם DFP חייב לקחת בחשבון את האפשרות של שינוי מהיר במאפייני או טופולוגיית הרשת. לכן, יש להתחשב בחבילות ה"איטיות מידי" בחישוב ה-*Estimated RTT*. אחד מהרעיונות להבדיל בין חבילה איטית אחת לרצף חבילות כאלה הוא למנוע ממס' החבילות הראשונות ה"איטיות מידי" מלהשפיע על ההערכות, ולאחר מס' מסויים של חבילות "איטיות מידי" ברצף – להכליל אותן בחישובים, מה שגורם ל-*RTT* החדשים לבנות הערכות חדשות ונכונות.

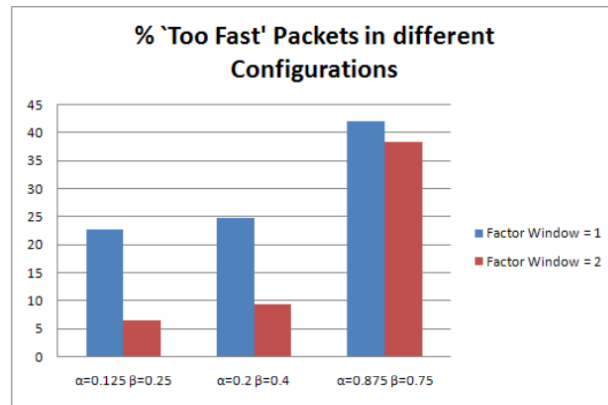
4.5 צריכת זיכרון:

השיקול העיקרי בבחירת התצורה של אלגוריתמי DFP הוא למנוע התקפות, לשם כך נק' ההתחלה צריכה להיות צמודה ככל האפשר ל-*Estimated RTT*. עם זאת מחשבה זו תיצור גם הרבה הזעקות שווא (כפי שהוסבר לעיל), הזעקות שווא אלו הן חבילות ששרת ה-DNS שומר בזיכרון עד שזה בטוח שהן תשובות חוקיות והגיעו מוקדם מידי. אז הן משוחררות מהזיכרון ונשלחות למשתמש (ולמטמון). בפרק זה אנו מציגים כיצד ערכי נק' התחלה שונים משפיעים על צריכת הזיכרון.

באיור 4.9 אנו רואים כמה חבילות נחשבות ל"מהירות מידי" בכל אחת מהתצורות שהוצגו מוקדם יותר בעבודה זו. באיור 4.10 אותו המידע מוצג באחוזים. איורים אלו יעזרו לנו להעריך מהי כמות הזיכרון שיש להקצות לאלגוריתם DFP בכל אחת מהתצורות.

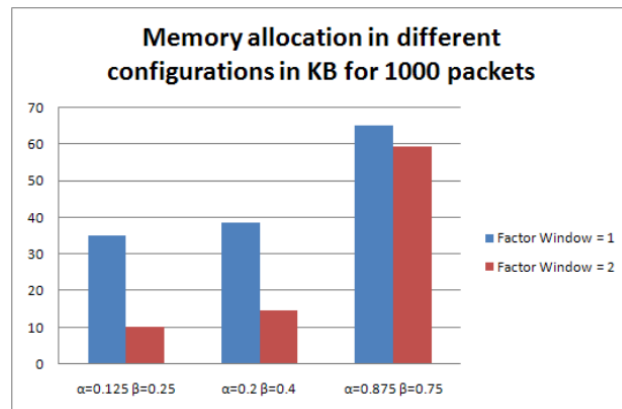


(איור 4.9: חבילות מהירות בתצורות שונות.)



(איור 4.10: אחוז החבילות המהירות בתצורות שונות.)

הניסויים שלנו מראים כי תשובה ממוצעת היא באורך 155 ביטים, אלגוריתם DFP חייב להקצות אותם בזיכרון לכל חבילה שמעוכבת, בד"כ לכמה מאות מילי-שניות, עד שהתשובה משוחררת או נזרקת. הזיכרון שאלגוריתם DFP זורש תלוי בכמה עסוק שרת ה-DNS המקומי. איור 4.11 מעריך כמה KB אלגוריתם DFP צורך, בהנחה שהוא מטפל באלף חבילות בכל זמן נתון:



(איור 4.11: הערכת הזיכרון ב-KB ל-1000 חבילות.)

חלק II: סימולציות

פרק 5: תוצאות נסיוניות:

5.1 תוצאות של אלגוריתם DFP:

האלגוריתם המוצג יושם ובדק על תנועה אמיתית שנאספה מהשרת DNS של HUJ. התנועה הוקלטה ונשמרה בקבצי pcap, ששימשו מאוחר יותר לבדיקה וניתוח של תצורות שונות. התנועה סוננה כדי שתכיל רק תשובות DNS עם Authoritative flag, מה שהבטיח שהתשובות יבואו מ-Authoritative server. ראה איור 5.1.

19283	10464.04437	92	128.143.2.125	132.65.16.8	DNS	standard query A planet3.cs.huji.ac.il
19284	10464.04455	146	132.65.16.8	128.143.2.125	DNS	standard query response A 132.65.240.104
19285	10467.35188	39	132.65.16.8	132.65.16.8	DNS	standard query PTR 68-80.65.132.in-addr.arpa
19286	10467.35224	137	132.65.16.7	132.65.16.8	DNS	standard query response PTR chopin.cs.huji.ac.il
19287	10467.49790	92	132.170.240.15	132.65.16.8	DNS	standard query A planet1.cs.huji.ac.il
19288	10467.49818	146	132.65.16.8	132.170.240.15	DNS	standard query response A 132.65.240.100
19289	10467.51251	92	132.170.240.15	132.65.16.8	DNS	standard query A planet3.cs.huji.ac.il
19290	10467.51270	146	132.65.16.8	132.170.240.15	DNS	standard query response A 132.65.240.102
19291	10468.26090	92	132.65.16.8	132.65.16.7	DNS	standard query TXT operator.passwd.ns.cs.huji.ac.il
19292	10468.26129	143	132.65.16.7	132.65.16.8	DNS	standard query response, No such name
19293	10468.26160	92	132.65.16.8	132.65.16.7	DNS	standard query TXT operator.passwd.ns.cs.huji.ac.il
19294	10468.26203	143	132.65.16.7	132.65.16.8	DNS	standard query response, No such name
19295	10468.26232	92	132.65.16.8	132.65.16.7	DNS	standard query TXT operator.grps.group.ns.cs.huji.ac.il
19296	10468.26270	133	132.65.16.7	132.65.16.8	DNS	standard query response TXT
19297	10468.26825	98	132.65.16.8	132.65.16.7	DNS	standard query TXT operator.grps.group.ns.cs.huji.ac.il
19298	10468.26866	151	132.65.16.7	132.65.16.8	DNS	standard query response, No such name
19299	10469.93170	92	168.95.192.73	132.65.16.8	DNS	standard query A planet2.cs.huji.ac.il
19300	10469.93198	146	132.65.16.8	168.95.192.73	DNS	standard query response A 132.65.240.101
19301	10470.14531	86	96.17.73.207	132.65.16.8	DNS	standard query PTR 58.191.65.132.in-addr.arpa
19302	10470.14551	134	132.65.16.8	96.17.73.207	DNS	standard query response, No such name
19303	10470.38304	92	168.95.192.73	132.65.16.8	DNS	standard query AAAA shuidig.cs.huji.ac.il

(איור 5.1: קובץ pcap סטנדרטי.)

לאחר תהליך הסינון, נמדד הזמן בין התשובה הנכנסת לשאילתה היוצאת, וכל אחד ממדגמי ה-RTTs נרשם בקובץ המתאים, בהתאם לכתובת ה-IP של ה-DNS וסוג השאילתה של ה-DNS. ראה איור 5.2.

	194.90.221.195	194.90.1.5	192.118.82.168	192.115.106.11
1	3892	4987	6301	9551
2	4159	4608	5595	3877
3	4164	4454	5469	4139
4	4041	5271	5468	4401
5	3900	4652	5447	3791
6	4049	4661	5992	25372
7	4074	5088	7380	5038
8	4026	4824	5459	4092
9	4933	4835	5292	4069
10	3923	4657	10967	4148
11	4010	4387	5238	3177
12	4171	4618	5505	4420
13	3955	4825	8541	3881
14	3973	4322	5692	4004
15	4773	5072	6563	3861
16	4225	4878	5583	4121
17	3930	4898	5779	5143
18	3837	4861	5228	4319
19	4419	4408	5359	4109
20	3938	4495	6136	25411
21	4264	4549	5611	4286
22	4013	4666	5406	22837
23	3861	4735	5456	3684
24	4053	4766	5334	3775
25	3949	4440	5432	4185
26	3992	4818	5919	3890
27	3975	4406	12211	4310
28	3956	4842	6065	4253
29	3937	4265	5418	2936
30	4118	7594	5510	3928
31	3986	5022	5297	3834
32	4139	4440	5896	4097
33	4124	4558	5379	451081
34	3925	4954	5508	3044

(איור 5.2: מדגם RTT (במילי-שניות) מתנועה המסודרת עפ"י כתובת IP וסוג השאילתה.)

לכל אחד מהמדגמים, האלגוריתם מחשב את ה-EstimatedRTT, ה-DevRTT ועם FactorWindow נתון הוא מסיק אילו חבילות נחשבות ל"מהירות מידי". דוגמאות קצרות עם חבילות "מהירות מידי" ניתן לראות באיור 5.3.

194.90.221.195					192.115.106.11					128.139.6.1				
RTT	E RTT	D RTT	FW=1	FW=2	RTT	E RTT	D RTT	FW=1	FW=2	RTT	E RTT	D RTT	FW=1	FW=2
4159	4125.625	8.34375			3877	4586.25	177.3125	fast	fast	358	374.625	4.15625	fast	fast
4164	4159.203	7.457031			4139	4194.906	146.9609			352	354.8281	3.824219		
4041	4055.775	9.286621	fast		4401	4375.238	116.6611			461	447.7285	6.186035		
3900	3919.472	11.83295	fast		3791	3864.03	105.7533			371	380.5911	7.037292	fast	
4049	4032.809	12.92246			25372	22683.5	751.439			350	353.8239	6.23394		
4074	4068.851	10.97907			5038	7243.688	1115.001	fast		348	348.728	4.857451		
4026	4031.356	9.573397			4092	4485.961	934.7412			345	345.466	3.759588		
4933	4820.295	35.35641			4069	4121.12	714.0859			402	394.9332	4.586379		
3923	4035.162	54.55776	fast	fast	4148	4144.64	536.4044			433	428.2417	4.62937		
4010	4013.145	41.70463			3177	3297.955	432.5421			363	371.1552	5.510829	fast	
4171	4151.268	36.21143			4420	4279.744	359.4705			489	474.2694	7.815772		
3955	3979.534	33.29195			3881	3930.843	282.0636			442	446.0337	6.870248		
3973	3973.817	25.17314			4004	3994.855	213.8339			473	469.6292	5.995383		
4773	4673.102	43.85433			3861	3877.732	164.5584			535	526.8287	6.539375		
4225	4281.013	46.89394	fast		4121	4090.591	131.0209			301	329.2286	11.96168	fast	fast
3930	3973.877	46.1396			5143	5011.449	131.1535			423	411.2786	11.90161		
3837	3854.11	38.8821			4319	4405.556	120.0041			446	441.6598	10.01126		
4419	4348.389	46.8144			4109	4146.07	99.27047			435	435.8325	7.716561		
3938	3989.299	47.93545	fast		25411	22752.88	738.9819			377	384.3541	7.625936		
4013	4040.083	40.1727			4286	6594.36	1131.327	fast	fast	477	465.4193	8.614637		

(איור 5.3: הערכות ומסקנות של חבילות מהירות עם 2 אפשרויות FactorWindow).

5.2 איתור התקפות:

כדי לבחון את אלגוריתם DFP, שתלנו מס' עותקים של חבילות תשובה אקראיות עם זמן הגעה אקראי בתנועה שנבחנה, זמני ההגעה של התשובות המזויפות היו קצרים יותר משל התשובות האמיתיות. אליגוריתם DFP עם התצורה הנ"ל היה מסוגל לסווג את כל ההתקפות כחבילות "מהירות מידי" ולכן עיכב אותם עד שהתשובה האמיתית הגיעה.

אנו מאמינים כי לא היו נסיונות אמיתיים לתקוף את שרת ה-DNS המקומי HUII כשהמדגמים נתפסו, היות ולא נמצאה כפילות של חבילות חוץ מהחבילות המזויפות שאנו שתלנו.

פרק 6: מסקנות:

6.1 עבודה עתידית

עבודה זו מתמקדת בפיתוח אלגוריתם חדש לזיהוי ומניעת התקפות על שרת ה-DNS. החוזק של האלגוריתם הוא בכך שהוא לא מסתמך על העובדות הקיימות על ההתקפות עצמן (לדוגמה: מחרוזות חתימה, כתובת IP ידועה וכו'). במקום זה, אלגוריתם DFP מסתמך רק עם התקשורת הבסיסית ותהליך זיהוי החבילות בפרוטוקול של ה-DNS: ה-5-tuple, TransactionID וסוג השאילתה. אנו מאמינים כי עבודה זו, מיושמת כמערכת משולבת על כל שרת DNS, תוריד משמעותית את הסיכוי להתקפה מוצלחת של ההתקפות הידועות וכמו כן גם ההתקפות zero-day שלא ידועות.

עדיין מחקר נוסף על אופטימיזציות אפשריות של האלגוריתם מומלץ במטרה לשפר את הנק' החלשות שלו. בפרק זה אנו מציגים כיוונים אפשריים למחקר עתידי.

- תצורה דינמית – באלגוריתם DFP, שלושת הפרמטרים המשפיעים על רוחב החלון מוגדרים רק פעם אחת, לפני שהאלגוריתם מתחיל. במחקר עתידי כדאי למצוא דרך להתאים את הפרמטרים בזמן הריצה של האלגוריתם, מה שיהפוך את התצורה להיות תלויה במאפייני הרשת וערכי ה-RTT הצפויים. ייתכן שהתצורה הדינמית תוכל לפתור את ההתרחבות הפתאומית של החלון הודות לבעיות הנדירות והמקומיות עם ה-RTT של חבילה בודדת.
- חבילות איטיות מידי – באלגוריתם DFP אין יחס מיוחד לטיפול בחבילות שמגיעות מאוחר מהגבול העליון של החלון. חבילות אלו יכולות להחשב כ"איטיות מידי" וצריכות להיות מטופלות בדרך שתמנע התרחבות משמעותית של החלון. מצד אחד, חבילות איטיות אלו יכולות להצביע על שינוי ברשת ולכן אי אפשר להתעלם מהן בהערכת ה-RTT. מצד שני, חבילות איטיות אלו הן בד"כ רק שגיאה זמנית בשרת ה-Authoritative DNS או אחד הקישורים ברשת. יש למצוא דרך שתוכל להבחין בין 2 מקרים אלו ותטפל בכל אחד מהן בהתאם.

ביבליוגרפיה

1. www.faqs.org/rfcs/rfc1034.html
2. www.faqs.org/rfcs/rfc1035.html
3. <http://www.isc.org/software/bind>
4. www.faqs.org/rfcs/rfc791.html
5. www.faqs.org/rfcs/rfc793.html
6. www.faqs.org/rfcs/rfc768.html
7. <http://ketil.froyn.name/poison.html>
8. BIND 9 DNS Cache Poisoning, Amit Klein, March-June 2007
9. <http://www.rnp.br/cais/alertas/2002/cais-ALR-19112002a.html>
10. <https://www.kb.cert.org/vuls/id/457875>
11. DNS and BIND, 4th Edition, by By Paul Albitz and Cricket Liu, Chapter 11, Security
12. <http://groups.google.com/group/comp.protocols.dns.bind/topics?pli=1>
13. <http://tldp.org/LDP/lame/LAME/linux-admin-made-easy/domain-nameserver.html>
14. DNS Cache Poisoning: Definition and Prevention, Tom Olzak, March 2006

15. DNS Spoofing (Malicious Cache Poisoning), Doug Sax
16. <http://www.dnssec.net/>
17. <http://slaptijack.com/software/how-to-hide-your-bind-version/>
18. <http://linuxgazette.net/153/moen.html>
19. Use of Bit 0x20 in DNS Labels to Improve Transaction Identity, P. Vixie and D. Dagon, March 2008
20. Solving the DNS Cache Poisoning Problem Without Changing the Protocol, Roberto Perdisci, Manos Antonakakis, and Wenke Lee, May 2008
21. <http://www.topbits.com/securing-dns-servers.html>
22. <http://www.ietf.org/rfc/rfc3833.txt>