מבוא לרשתות – עבודה 1

1. חולשות DNS
   1. מתקפת NXDomain זו מתקפה שבמסגרתה תוקף מנצל את מנגנון הDNS על מנת להציף Authoritative DNS Server, ולגרום לDOS אצלו.  
        
      הדרך שבה מממשים את המתקפה היא כזו -   
      תוקף מנסה לרזלב כמות גדולה מאוד של דומיינים **שלא קיימים**.   
      הוא יכול לעשות זאת בצורה מבוזרת (ממספר רב של מכשירים שונים במקביל - בוטים) כדי להגביר את נפח התעבורה ובכך את עוצמת המתקפה.  
        
      המתקפה מנצלת מימוש לקוי של שרתי הDNS - מנגנון היעילות של הcaching יודע עקרונית להתמודד רק עם דומיינים שכן קיימים, אבל לא עם כאלו שלא.   
      לכן עבור המתקפה הזו, שבמסגרתה נשלחות המון בקשות DNS לדומיינים **שונים ושאינם קיימים**, יוצא מצב בו שרת הDNS מוצף ועמוס בבקשות דמה, ולכן יגרמו עיכובים בטיפול בבקשות לגיטימיות, עד כדי DOS לחלוטין של אותו שרת DNS.
   2. ישנם מספר דרכים למדוד את יעילות/הצלחת המתקפה:
      1. מדידת Packet Factor Amplification:

למדוד לאורך מספר ניסויים את פקטור ההגברה של הפקטות, כלומר בהינתן פקטה אחת שנשלחה (כתוצאה מבקשת DNS), כמה פקטות בפועל נשלחו כתוצאה מפרוטוקול ואופן פעולת שרתי הDNS במהלך בקשות DNS נוספות שהם ביצעו, ולחשב את היחס הזה (קרי – Packet Factor Amplification).

* + 1. מדידת נפחי התעבורה וביחס למשאבי השרת -   
       לאסוף מידע על נפחי התעבורה (כמות בקשות, גדלי פקטות, זמנים של פיקים, התפלגות סוגי הבקשות השונות וכד') במהלך תפקוד רגיל של שרת הDNS ובמהלך הרצת המתקפה, ולהשוות את הנתונים.  
       - נצפה לראות **עלייה** גדולה מאוד בכל המטריקות והפרמטרים של נפחי התעבורה השונים (ושונות **קטנה** יותר בסוגי הבקשות השונות, עקב הספמה של בקשות A).  
       - נרצה להשוות גם את "נפחי העומס" על השרת לפני ובמהלך המתקפה; מבחינת עומס על רוחב הפס ועל משאבי השרת (על הCPU והRAM למשל).
    2. מדידת משך המתנה בתור הבקשות –  
       לאסוף מידע על משך זמני ההמתנה של בקשות בתוך שרת הDNS, החל מרגע שבקשה נכנסת לתור הבקשות של השרת, ועד שתשובה נשלחת עבור אותה בקשה.  
       - נצפה לראות **עלייה** בדיליי/משך זמני ההמתנה האלו במהלך הרצת המתקפה, עקב התמלאות של התור בבקשות לא לגיטימיות.
    3. מדידת משך ההמתנה מצד הלקוח –  
       בדומה למדידה לעיל, רק מצד הלקוח. כלומר לאסוף מידע על זמני קבלת התשובה (RTT), מרגע שליחת הבקשה ועד רגע קבלת התשובה.  
       - במהלך ההתקפה נצפה לחוות **עלייה** מסיבית בזמני ההמתנה.
    4. מדידת כמות הלקוחות הגולשים באתרים הנמצאים תחת שרת הDNS –

אם המידע הזה פומבי, ניתן להתבונן בכמות המשתמשים הגולשים באתרים שנמצאים תחת שרת הDNS המותקף (ב"תחת" הכוונה באתרים שאותו שרת DNS המותקף הוא הAuthoritative שלהם, כלומר אם ננסה לרזלב אותם נקבל לבסוף תשובה ממנו).  
- נצפה לראות **ירידה** חדה בכמות הזו במהלך המתקפה.

* 1. מתקפת NSNXDomain היא מתקפה שבמסגרתה מנצלים את אופן פעולת מנגנון הDNS, על מנת להציף שרת DNS כלשהו בכמות מסיבית של בקשות, הדבר היכלול לגרום לו לDOS.  
       
     המתקפה נשענת על העובדה שכל אחד יכול להרים שרת DNS משלו (ודי בקלות), ובכך שבמידה ושרת DNS שולח תשובת NS (תשובת DNS שבה מוכלים שמות שרתי DNS הAuthoritativeים של דומיין המבוקש כלשהו), שרת הDNS הרקורסיבי (של הספקית נניח) מרזלב את כולם במכה כדי למקסם יעילות (כדי שאם במידה ואחד מהם לא מגיב/איטי, אז יהיו בstandby מוכנים עוד IPים שניתן לפנות אליהם כדי לנסות לרזלב את הדומיין). אלו הפרימיטיביים של המתקפה.  
       
     למתקפה יש כמה ווריאציות, כולן דומות ברעיונם ובשיטתם. השוני הוא בגדול במי השרת יעד לתקיפה (הTLD, הROOT או הSLD), ובשוני מעט בשיטה (בין אם שימוש בTCP או UDP, בDOS או DDOS וכד').

נתבונן ונעמיק לצורך העניין במתקפה על שרת הDNS של הTLD:

קנפוגים מקדימים:

1. התוקף קונה שרת DNS, ודומיין נניח בשם dnsdomain.com עבורו (כלומר מחווט את הדומיין לשרת שלו).
2. מקנפג את השרת בצורה כזו שעבור כל בקשה שתתקבל – תוחזר תמיד רשומת NS ענקית, המכילה 135 NSים **שונים ולא קיימים** (135 זו הכמות המקסימלית האפשרית ברשומתNS אחת).

נניח כולם מהצורה ns-i.fake-i.target, כאשר הtarget זה הTLD שאותו אנחנו רוצים לתקוף.

מהלך המתקפה:

1. התוקף מנסה לרזלב את הדומיין fake.dnsdomain.com (אין באמת חיוב לfake, זה יכול להיות כל דבר שלא באמת קיים).
2. השרת DNS של התוקף מקבל את הבקשה, ומחזיר תשובת/רשומת NS, המכילה 135 NSים מהצורה ns-i.fake-i.target (כאשר target זה TLD כלשהו, כמו com או net למשל), כאשר כל הדומיינים האלו לא באמת קיימים.
3. רשומות הNS האלו מגיעות לrecursive DNS server (של הספקית ברוב המקרים), וכעת שרת זה יבצע 135 רזלובים (אחד לכל NS שהופיע בתשובה שחזרה) במכה.
4. כל בקשות הDNS (רזלובים) הרבות אלו יגיעו לשרתי הDNS של הTLD (לtarget)

כלומר בכך השגנו, שע"י בקשת DNS אחד מקורית שלנו, נעשו בפועל כ-135 בקשות DNS (ו"במכה").  
ואם נתבונן באמפליפיקציה ברמת כמות הפקטות שנשלחו, השגנו "מקדם הגברה" של 1620, כלומר על כל פקטה אחת ששלחנו, נשלחו לשרת הtarget 1620 פקטות.   
החישוב הוא בגדול כזה – 135 בקשות DNS, כפול 2 עבור אחת לipv4 ואחת לipv6, כפול 2 עבור בקשה ועבור תשובה, כפול 6 עבור פקטור שנובע משימוש בTCP במקום UDP עקב כך שהבקשות גדולות מדי, חלקי 2 עקב הפקטות של התוקף עצמו שכביכול מכבידות עליו ו"מורידות" את האמפליפיקציה.

* הסיבה שמתקפה זו יותר רצינית ומסוכנת ממתקפת NXDomain סטנדרטית טמון בפקטור האמפליפיקציה העצום.   
    
  אם במתקפת NXDomain רגילה עשינו abuse למנגנון הcaching שלא יודע להתמודד באמת עם בקשות לדומיינים שלא קיימים, בפקטור הגברה של בגדול 1 - היינו צריכים כמות מאסיבית של בוטים כדי באמת להשפיע ממש ולגרום לDOS/עיכובים רציניים (כמו Mirai Attack ב2016).  
    
    
  לעומת זאת, במתקפת NXNSDomain אנחנו גם עושים abuse למנגנון הcaching, אך מנצלים גם את מנגנון הDNS עצמו, ואיך שrecursive resolver עובד כאשר הוא מקבל תשובת NS.  
  בכך אנו משיגים פקטור אמפליפיקציה עצום, שבעצם משיג את אותה מטרת NXDomain Attack, שזה DOS לשרת DNS, אך בהרבה פחות משאבים.   
    
  בעצם בשורה התחתונה צריך כפי 1620 יותר משאבים, כדי שבמתקפת NXDomain רגילה נשיג אותו אפקט כמו במתקפת NSNXDomain, וזו הסיבה בעצם למה מתקפה זו רצינית ומסוכנת יותר.
* אופציות לפתרון של הבעיה (mitigations):

1. MaxFetch(k) – הRecursive DNS Server/Resolver לא ירזלב במכה את כל הNSים, אלא לכל היותר k במקביל, ובכך למזער את פקטור האמפליפיקציה.
   * הtradeoff כאן הוא כמובן בין יעילות לבטיחות. אנחנו יודעים מה קורה כשכולם מתרזלבים במכה, וכאשר נרזלב k לכל היותר, קיים סיכוי שאותם k שרוזלבו לא מגיבים/איטיים ובכך נגרום לפגיעה ביעילות ובזמן תגובה במהלך פעילות תקינה של המערכת.
2. MaxBreadth - bound – להגביל את כמות הNSים שיכולים להופיע בתשובת/רשומת NS (ממחקר שעשו הרוב לא מכילים יותר מ6), ובכך למזער גם כן את פקטור האמפליפיקציה.
   * הtradeoff כאן הוא גם בין יעילות לבטיחות, וגם השינוי טומן בתוכו שינויים הכרחיים במנגנון ובפרוטוקול הDNS.   
     אם הגבלנו את המספר נניח ל6 כפי שהם הציעו, במידה ולשרת מסויים יש 10 NSים מתאימים, איך נדע את מי מהם להחזיר? מה אם נחזיר 6 רנדומליים ובמקרה כולם לא מגיבים כרגע או שסתם הם פחות מהירים משאר השרתים? יש כאן שאלות פתוחות שלא ענו עליהן, שכאמור – פתרונם מצריך עדכון של פרוטוקול הDNS עבור המקרים האלו, ובכללי מחשבה נוספת כדי למקסם יעילות.
3. זיהוי תשובות NX NS – הRecursive DNS Server/Resolver יצטרך לזהות תשובות NS לא קיימות, ובכך לזהות את המתקפה ובעצם למנוע אותה ע"י אי רזלוב של אותם NSים לא קיימים מראש.
   * הtradeoff כאן הוא פחות חד משמעי, ותלוי בעיקר ביכולת הזיהוי של NX NS. כלומר אם הזיהוי מאוד איכותי וטוב – זה ניצחון חד משמעי, אך במידה והוא שוגה (בין אם חושב שתשובות NS לגיטימית היא NX ומפיל אותה, או בין אם הוא לא מזהה כל מתקפה ברגע שהיא קורית) – נוכל לגרום לDOS אצל לקוחות לגיטימיים, ובעצם לא נשיג הרבה, או שלא נשיג מספיק. שוב, תלוי בעיקר ביכולת הדיוק והאיכות של אותו מנגנון זיהוי/מודל AI וכו'.

1. שימוש בDNSSEC – כיוון שDNS הוא פרוטוקול עתיק וכולו עובר גלוי בתווך (ועוד כמה סיבות אבטחה מעניינות), יצא מנגנון חדש המתלבש על גבי המנגנון המוכר בשם DNSSEC.  
   מנגנון זה פותר את עניין הNSים הלא קיימים, בכך שקיימת רשומה בשם NSEC, שבמידה וניסינו לרזלב דומיין שלא קיים, לא נקבל תשובה ריקה (NX) אלא נקבל תשובה המכילה את הIP הכי "קרוב" (אלפביתית) למה שביקשנו. ובכך אנחנו בעצם נשארים תחת מסגרת הcaching התקינה, כי אכן חוזרת תשובה, כביכול כמו במצב תקין בו רזבלנו דומיין קיים ואמיתי.
   * הtradeoff כאן הוא בעצם השימוש עצמו בDNSSEC, שהוא אומנם מאובטח יותר (גם למתקפה שלנו וגם לסיבות אבטחה נוספות רבות) אך עקב כך גם מגיע overhead גדול מאוד, שבסופו של דבר מעלה את זמני ההמתנה לקבלת תשובה (יוצר דיליי גדול יותר, ולקוחות עצבניים יותר).
2. שינוי מנגנון/פרוטוקול הDNS: לרדת רק מטה בהיררכיית ה – DNS אחד הפרימיטיביים עליהם מתבססת המתקפה היא העובדה ששרת הDNS הזדוני, שהוא Authoritative DNS Server, כלומר נמצא הכי תחתון בהיררכיה, יכול לגרום לRecursive DNS Server/Resolver, שוב לפנות לשרתי DNS גבוהים יותר בהיררכיה (לRoot או TLD).
   * הtradeoff כאן הוא בעיקר בשינוי הפרוטוקול DNS, הם לא הסבירו לעומק ובפרטים איך הם מציעים לשנות את הפרוטוקול (אין RFC מלא שהוציאו), ויכול להיות שהדבר חולשתי לא פחות (מאותה פרצה או מפרצות אחרות שיצוצו עקב השינוי), או פשוט קשה למימוש, מה שיגרום לחברות גדולות אולי לזנוח את הכיוון/לקחת זמן עד שהוא יכנס לפועל in the wild.