Technologic Papers

Subverting BIND's SRTT Algorithm: Derandomizing NS Selection

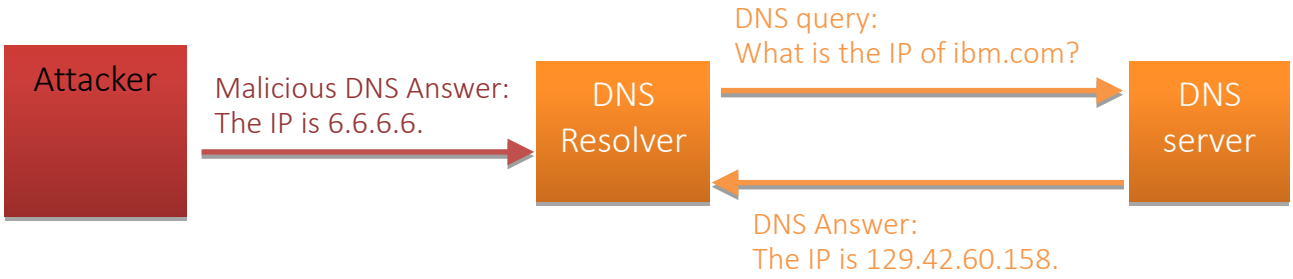
מאת רועי חי

הקדמה

במסמך זה אתאר מתקפה חדשה על שרת ה-DNS הנפוץ בעולם, BIND. המתקפה הוצגה בכנס USENIX WOOT '13. פרסום המתקפה נעשה בתיאום מלא עם ISC, הארגון שאחראי על BIND.

מבוא

DNS הוא פרוטוקול שעובד בדרך-כלל מעל UDP. אופן-הפעולה של התוקף ב- Off-path (Blind) DNS attack poisoning הוא לאלץ Name Server (NS) או DNS Resolver מסויים לשלוח בקשה, ולהחזיר תשובה זדונית כ-NS אליו הוא פונה, לפני שמגיעה התשובה המקורית. אנו מניחים שהתוקף אינו רואה את המידע. אחרת כללי המשחק משתנים, ופרוטוקול ה-DNS פשוט אינו מסוגל להתמודד עם מצב זה ללא DNSSEC.

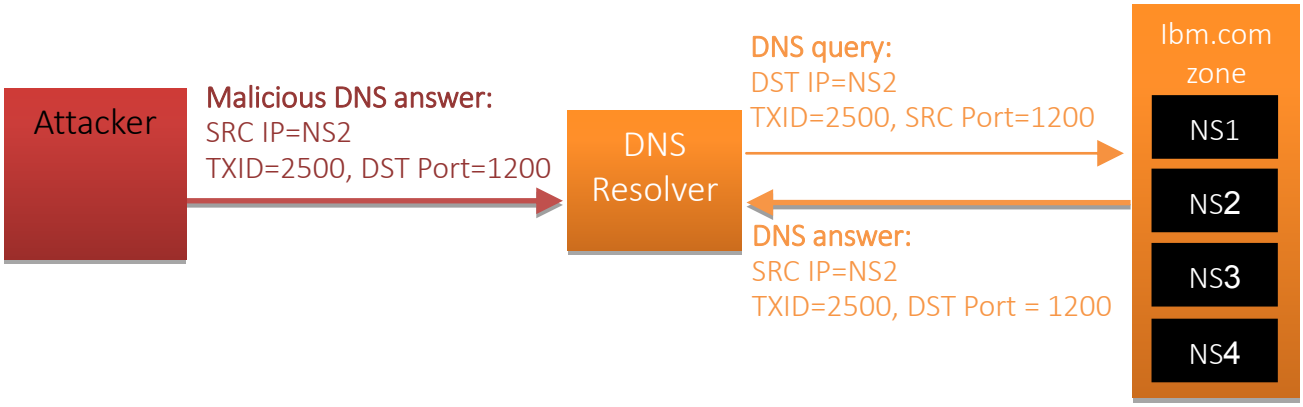


המצב המתואר למעלה הוא טריוויאלי מבחינת התוקף, כי הוא יכול לתזמן את המתקפה כך שתמיד או ברוב המקרים ינצח את השרת המקורי. לכן, כל בקשת DNS מכילה מזהה ייחודי, nonce, המורכב משלושה פרמטרים:

1. שדה ה-TXID ב-DNS header (16 ביטים רנדומליים).
2. UDP source port (16 ביטים רנדומליים בקירוב).
3. IP destination address.

מידת הרנדומליות של הפרמטר השלישי תלויה במספר שרתי ה-DNS האחראים על הדומיין המתושאל ובאלגוריתם הבחירה. למשל, אם אלגוריתם הבחירה הוא פשוט להגריל שרת באופן אחיד (לא המצב ב-BIND, כפי שנראה בפירוט בהמשך), ויש 8 שרתי DNS האחראים על הדומיין, אז נוספים 3 ביטים רנדומליים ל-nonce.

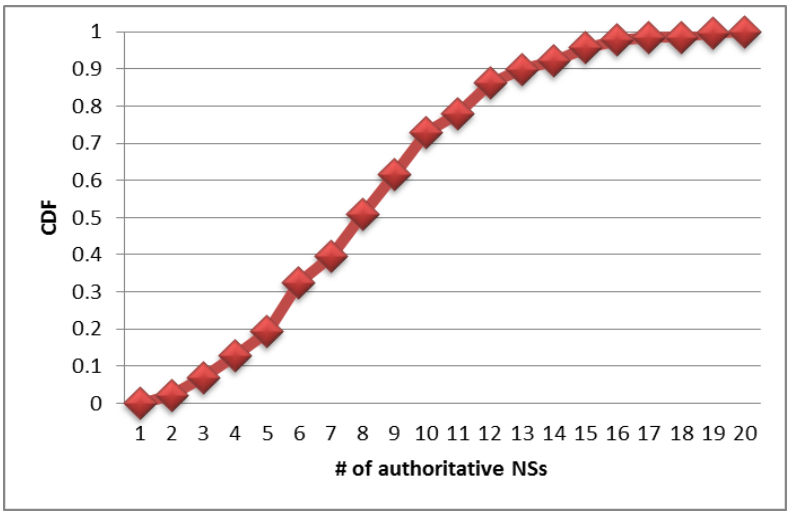
כדי להצליח בתקיפה התוקף חייב לנחש נכון את ה-nonce, ולכן מידת הרנדומליות שלו משפיעה באופן ישיר על זמן התקיפה. בפועל, תקיפת DNS (מוצלחת) נראית באופן הבא:



מה נשאר לחקור?

הנחת העבודה שלנו הייתה שמידת הרנדומליות של שדה ה-TXID וה-UDP Source Port במימושים נפוצים נחקרה רבות בעבר. לכן החלטנו להתמקד בשדה ה-IP destination address.

על מנת להסיק את הפוטנציאל של שדה זה (כלומר, באופן אופטימלי, כמה ביטים רנדומליים הוא יכול להוסיף ל-nonce), הורדנו את ה-root zone file (http://www.internic.net/domain/root.zone), ויצרנו את ה-CDF graph הבא:



גרף זה מלמד כי בערך למחצית מה-Top-Level Domains (TLDs) יש למעלה מ-8 שרתי DNS אחראיים ולכל ה-TLDs יש פחות מ-20 שרתי DNS אחראיים. עובדה זו מלמדת כי אפילו אם אלגוריתם הבחירה הוא רנדומלי לחלוטין (כלומר הוא מגריל את שרת ה-DNS באופן יוניפורמי), אז מספר הביטים הרנדומליים הנוספים הוא די נמוך.

לכן, אם נצליח לחזות את שרת ה-DNS הנבחר לא תתאפשר מתקפת DNS חדשה, אך כן נוכל לקצר מתקפות קיימות או חדשות אם תמצאנה. בנוסף לכך, אם התוקף הוא Man-in-the-Middle בין ה-Resolver לשרת DNS מסוים, והוא מסוגל לכפות תשאול מול אותו שרת, אז מתקפת Off-path DNS poisoning הופכת ל-On-path.

אלגוריתם בחירת שרת ה-DNS ב-BIND.

כאמור מבחינת Security, הדבר הכי טוב שה-Resolver יכול לעשות הוא להגריל באופן אחיד את השרת המתושאל. אולם, מבחינת ביצועים הדבר אינו אופטימלי: המוטיבציה של ה-Resolver היא לבחור את השרת אשר יענה לו הכי מהר, כלומר עם ה-Round-Trip Time (RTT) הנמוך ביותר. מכיוון שה-RTT משתנה לעתים תכופות (תלוי בעומסי הרשת, עומס השרת וכד'), BIND ממצע את ערכי ה-RTT. הממוצע נקרא Smoothed Round-Trip Time (SRTT).

באופן מפורט יותר, ערך ה-SRTT לכל שרת DNS הוא במיקרו-שניות, ונקבע באופן הבא:

1. **אתחול (Initialize):** הערך ההתחלתי מוגרל בין 1 ל-32.
2. **עדכון (Update):** כאשר מתקבלת תשובה חדשה משרת DNS מסויים, אז ל-BIND יש מידע אודות ה-RTT שלו. לכן האחרון נלקח ומשתקלל עם ה-SRTT הישן. כאשר משקל 0.3 ניתן ל-RTT ומשקל 0.7 ניתן ל-SRTT הישן.
3. **דעיכה (Decay):** על-מנת למנוע הרעבה, כאשר יש מספר שרתי DNS אופציונאליים, אז ערך ה-SRTT של שרתי ה-DNS הבלתי-מתושאלים יורד ב-2% לאחר כל שאילתה.
4. **שגיאה (Error):** כאשר שרת ה-DNS המתושאל מחזיר שגיאה או שלא עונה, ערך ה-SRTT שלו נענש ב-200ms (עם גבול עליון של שנייה).

כל ערכי ה-SRTT נשמרים ע"י BIND ב-cache. הכניסות ל-cache הן כתובות ה-IP של שרתי ה-DNS.

מה נעשה בעבר ומה הוא החידוש שלנו?

באופן כללי אם התוקף יכול להשפיע על ערכי ה-SRTT של שרתים שרירותיים הוא יכול לחזות לאיזה שרת BIND יפנה, כלומר הוא מבצע דרנדומזיציה (derandomization) לאלגוריתם הבחירה :

1. אם התוקף מצליח להוריד את ערך ה-SRTT של שרת מסויים לערך נמוך מערכי ה-SRTT של שאר השרתים האחראים על דומיין כלשהו, BIND יפנה אליו בבקשה הבאה.
2. אם התוקף מצליח להעלות את ערך ה-SRTT של כל השרתים האחראים על דומיין מסויים, למעט שרת בודד, כך שערכי ה-SRTT יהיו גבוהים מזה של האחרון, BIND יפנה לאותו שרת בבקשה הבאה.

בעבר הוצעו שתי מתקפות עיקריות על מנגנון בחירת ה-NS:

1. המתקפה הראשונה [Petr 2009], מנצלת את פעולת ה-Update באלגוריתם ה-SRTT. התוקף פשוט מתחזה לשרת אשר הוא רוצה להוריד את ה-SRTT שלו, ושולח תשובות מהירות בשמו.
2. המתקפה השניה [Herzberg & Shulman, 2012], מנצלת את פעולת ה-Error באלגוריתם ה-SRTT. התוקף מעלה את ה-SRTT של כל שרת אחראי למעט אחד. הוא מנצל IP fragmentation כדי לייצר תשובות שגויות.

נשים לב ששתי המתקפות הן הסתברותיות: התוקף חייב לנחש ערכים מסויימים. לעומת זאת, המתקפה שלנו היא דטרמיניסטית, כלומר היא תמיד עובדת. בנוסף לכך במתקפה שלנו השרת אשר אנו משפיעים על ערך ה-SRTT שלו בכלל לא מודע לתקיפה.

המתקפה

במאמר זה אתאר וריאציה פשוטה של המתקפה שלנו. במסמך המלא ניתן למצוא וריאציות נוספות.

במתקפה זו אנו מנצלים שרתי DNS סגורים, כלומר כאלה שלא עונים על שאילתות על דומיינים שהם אינם אחראיים עליהם. אלו הם **רוב** שרתי ה-DNS בעולם, כאשר התוקף רק צריך להכיר את כתובת ה-IP שלהם. אנחנו נראה כיצד התוקף יכול לנצל אותם על מנת להוריד את ערך ה-SRTT של שרת DNS שרירותי לערך שרירותי כרצונו, וכפי שצויין לעיל, כך הוא יוכל לדעת בוודאות ש-BIND יפנה אליו בשאילתה הבאה.

נניח כי הדומיין הנתקף הוא ibm.com, ויש שני שרתי DNS האחראים עליו: ns1.ibm.com ו-ns2.ibm.com. כמו כן נניח ששני שרתים אלו הם כבר ב-SRTT cache של ה-DNS Resolver הנתקף, וה-SRTT של ns1.ibm.com גבוה יותר. נראה כיצד ניתן להוריד את הערך של ns2.ibm.com, כדי שבשאילתה הבאה ה-DNS Resolver יפנה בוודאות ל-ns2.ibm.com.

התוקף מצויד בארסנל הבא:

1. רשימה של שרתי DNS סגורים.
2. שרת DNS עליו הוא שולט, לדוגמא ns.malicious.foo.
3. דומיין אשר השרת הנשלט אחראי עליו, למשל malicious.foo.

המתקפה מתחילה בתשאול ה-DNS Resolver על דומיין אשר השרת הנשלט אחראי עליו, למשל 666.malicious.foo. בסופו של יום ה-Resolver יפנה לשרת ה-DNS עליו התוקף שולט: ns.malicious.foo. אשר יחזיר Delegation, כלומר לא יענה על השאילתה בעצמו אלא יפנה את ה-Resolver לשרתים אחרים. הוא מחזיר את ה-Delegation הבא:

1. רשימה גדולה מאוד של שרתי DNS סגורים אשר לא ב-SRTT cache של ה-Resolver.
2. ns2.ibm.com.

כאשר ה-Resolver יקבל את ה-delegation הוא יכניס את הרשימה של שרתי ה-DNS הסגורים ל- SRTT cache שלו עם ערך נמוך מאוד, בין 1 ל-32, לפי פעולת האתחול באלגוריתם ה-SRTT. לכן הוא יפנה אליהם לפני שהוא יפנה ל-ns2.ibm.com. בשלב הבא הוא יפנה אליהם באופן סדרתי, והם יחזירו לו Query refused (כי הם סגורים).

בכל תשאול של שרת DNS סגור, תתבצע פעולת הדעיכה של האלגוריתם, כלומר ה-SRTT של ns2.ibm.com מוכפל בפקטור 0.98 (יורד ב-2%) לאחר 30 שניות ה-DNS Resolver יבצע timeout לבקשה. לכן בסוף התהליך ערך ה-SRTT של ns2.ibm.com יהיה 0.98^n מערכו המקורי, כאשר n הם מספר שרתי ה-DNS הסגורים שתושאלו עד ה-timeout. לפיכך הצלחנו להוריד את ערך ה-SRTT של ns2.ibm.com באופן דטרמיניסטי לערך שרירותי!

כיצד ניתן לתקן את הפגיעות?

נשים לב שהמתקפה מנצלת את העובדה ששרת DNS זדוני יכול להשפיע על ערך ה-SRTT של שרת DNS שרירותי. שורש הבעיה הוא שה-SRTT cache הוא גלובאלי. ניתן לתקן את הבעיה ע"י חלוקה של ה- SRTT cache, למשל לפי ה-zone המתושאל.

קישורים לקריאה נוספת

- המאמר המקורי מ-USENIX WOOT '13: http://bit.ly/19Gy734
- המצגת מ-USENIX WOOT '13: http://bit.ly/15AdyZ4
- הדיווח של ISC על הפגיעות: http://bit.ly/1beaBfJ