PROJECT 2. ENTITY AUTHENTICATION

Abraham J. Reines

February 12, 2024

1 PVD File Study

This report is a detailed analysis of entries for Password Value Data (PVD) files for: Windows 7, Windows 2003, and Ubuntu/Linux systems. The uniqueness and security of the password hashes is the focus.

1.1 Windows 7 PVD Analysis | Non-Empty Fields

- Username: Morpheus
- User ID (RID): 1002
- LM Hash: aad3b435b51404eeaad3b435b51404ee (Placeholder)
- NT Hash: d5d630da69ebd8ab040d77be7bb046dd

1.2 Security Implications

LM is a placeholder, indicating there is no LM hash used. This is a good security measure. LM hashes are less secure. The NT Hash is the value actually used for password storage. This needs to be unique and the only password hash value present.

1.3 Windows 2003 PVD Analysis | Non-Empty Fields

- Username: Morpheus
- User ID (RID): 1010
- LM Hash: B902F044A44585DF93E28745B8BF4BA6
- NT Hash: D5D630DA69EBD8AB040D77BE7BB046DD

1.4 Security Implications

LM hash presence indicates compatibility with old systems and poses a security risk. NT hash being used is identical to the Windows 7 PVD suggesting the same password was used across the systems. This is a vulnerability which could lead to catastrophic security weaknesses.

1.5 Linux PVD Analysis | Non-Empty Fields

- Username: Morpheus
- Password Hash: SHA-512 with salt ykumbyCz
- UID/GID: 1001
- Home Directory: /home/morpheus
- Shell: /bin/sh

1.6 Security Implications

The SHA-512 hash usage with a unique salt ykumbyCz suggests a secure practice. This makes sure the password is secure and difficult to crack.

1.7 Conclusion for PVD File Study

PVD entries for Morpheus in these systems exemplify the necessity for unique hashing in security mechanisms. The reuse of passwords across systems is a notable security concern which should be addressed to maintain system integrity.

Explanation of Wordlists

Created a custom wordlist for this assignment, concatenating several wordlists derived from data breaches, available on GitHub. Also employed a variational algorithm of the known passwords cracked using hybrid attack strategies. Catching onto the trend, developing the code to produce the permutations for common substitutions and capitilizations was straight forward. Systematic permutations of the passwords: "DENTURES," "DEPENDS," "MODELT," "OLDFOGY," "KANDW," "CORNBREAD," "SUSPENDERS," and "WHITE_RABB1T." This was instrumental, especially for Linux SHA-512 crackings.

Sources of Wordlists:

1. `kkrypt0nn/wordlists`
2. `gmelodie/awesome-wordlists`
3. `berzerk0/Probable-Wordlists`
4. `assetnote/wordlists`

Commands for Concatenation:

1. `cat password_variations.txt wordlist1.txt wordlist2.txt > comprehensive_wordlist.txt`
2. `sort -u comprehensive_wordlist.txt -o comprehensive_wordlist.txt`

Python Code:

```python
import itertools

class PVG:
    """
    Generates variations on a list of passwords.
    Author: Abraham Reines
    Date: 2024-02-05
    """

    def __init__(self, passwords, output_file="/Volumes/StorageAJR/School/CS559/Project2/password_varations.txt"):
        """Initialize the generator."""
        self.passwords = passwords
        self.output_file = output_file

    def generate_variations(self):
        """
        Generates and writes the password variations.
        """
        with open(self.output_file, ’w’) as file:
            for password in self.passwords:
                # Generate all combinations of upper and lower case for each character
                variations = [’’.join(var) for var in itertools.product(*([letter.lower(), letter.upper()] for letter in password))]
                file.write(’\n’.join(variations) + ’\n’)
        print(f"Password variations generated successfully: {self.output_file}")

if __name__ == "__main__":
    passwords = ["MSKITTY666", "DENTURES", "DEPENDS", "MODELT", "OLDFOGY", "KANDW",
                 "CORNBREAD", "SUSPENDERS", "WHITE_RABB1T"]
    pvg = PVG(passwords)
    pvg.generate_variations()
```

2 Off-line dictionary attack: crack Windows 7 password authentication

This section documents the process and outcomes of a dictionary attack to crack Windows 7 password authentication for a set of user accounts.

2.1 Methodology

To conduct the offline dictionary attack, the following steps were taken:

1. Preparation of a comprehensive dictionary of potential passwords. This included a custom 'comprehensive_wordlist.txt' file, including all wordlist provided by Hashcat. See the 'Explanation of Wordlists' section above.
2. Utilization of password cracking software to match dictionary entries with the extracted hashes. We utilized hybrid attacks, combining brute force methodology with the wordlist.

2.2 Tools and Commands

A combination of advanced password recovery tools was employed to orchestrate the dictionary attack. The specific software and command sequences executed are detailed below:

- **Tool:** *Hashcat* – a robust password recovery utility renowned for its speed and versatility.
- **Command(s):**
  - Using the wordlist:

    ```
    hashcat -m 1000 -a 0 2024Spring-PVDwindows7.txt comprehensive_wordlist.txt
    ```

  - To initiate the attack with lowercase alphabetic character assumptions:

    ```
    hashcat -m 1000 -a 3 -i --increment-min=4 --increment-max=10
    -1 ?l 2024Spring-PVDwindows7.txt ?1?1?1?1?1?1?1?1?1?1
    -o cracked_passwords_windows7.txt
    ```

  - To incorporate numeric characters into the attack vector:

    ```
    hashcat -m 1000 -a 3 -i --increment-min=4 --increment-max=10
    -1 ?d 2024Spring-PVDwindows7.txt ?1?1?1?1?1?1?1?1?1?1
    -o cracked_passwords_windows7.txt
    ```

- **Tool:** *John the Ripper* – an open-source password security auditing and password recovery tool used for detecting weak passwords.
- **Command(s):**
  - To initialize a brute force attack using john.

    ```
    john --incremental 2024Spring-PVDwindows7.txt
    ```

2.3 Results

The following table lists the cracked passwords for each user account:

Table 1: Cracked passwords for Windows 7 user accounts along with crack time and attempt count.

User Account	Cracked Password	Time to Crack	Attempt Count
Morpheus	`dentures`	< 10 mins	1
Neo	`cornbread`	< 10 mins	1
Trinity	`KandW`	> 12 hrs	2

Table 1 continued from previous page

User Account	Cracked Password	Time to Crack	Attempt Count
Cypher	`Depends`	> 1 hr	2
Smith	`modelT`	> 1 hr	2
Cobb	`oldfogy`	< 20 mins	1
Arthur	`suspenders`	< 10 mins	1

2.4 Computing Environment

The password cracking process was executed within a controlled and secure virtualized setting, leveraging the following specifications:

- **Hardware:**
  - Processor Cores: 4
  - Memory: 4008 MB
  - Network Interface: Virtualized within the hypervisor
  - Hard Disk: 2.6 MB used for the operation snapshots (dynamic allocation)
- **Software:**
  - Operating System: Kali Linux, version 6.5.0-kali3-arm64
  - Hypervisor: Identified as a virtual machine within a hypervisor environment.
  - Penetration Testing Tools: Hashcat 6.2.6, John the Ripper 1.9.0

This setup was deliberately chosen for its balance of performance and isolation, providing a secure sandbox for password recovery simulations.

2.5 Discussion

We cracked the passwords required. The cracking process for this PVD file was moderately difficult.

2.6 Conclusion

The dictionary attack was successful in identifying the passwords for all user accounts in the PVD file. The details of the computing environment and the tools used were instrumental in the process.

3 Off-line dictionary attack: crack Windows 2003 password authentication

This section is a review of the results of a off-line dictionary attack for cracking the Windows 2003 authentication for specified user account passwords.

3.1 Cracked Passwords | User Accounts

The passwords for user accounts have been successfully cracked and are presented:

Table 2: Cracked passwords for user accounts.

Account Name	Cracked Password
Administrator	mskitty666
Morpheus	dentures
Neo	cornbread
Trinity	KandW
Cypher	Depends
Smith	modelT
Cobb	oldfogy
Arthur	suspenders

3.2 Cracked Passwords | Service Accounts

The passwords for service accounts have been successfully cracked and are presented:

Table 3: Cracked passwords for service accounts.

<table>
<tr><th>Service Account Name</th><th>Cracked Password</th></tr>
<tr><td>ASPNET</td><td>><0,K)ND5912K8</td></tr>
<tr><td>IUSR_JAMES-BFCF5F2D5</td><td>;3PZ#PA(A{|W62</td></tr>
<tr><td>IWAM_JAMES-BFCF5F2D5</td><td>\5e9K)K}MGNKW&</td></tr>
<tr><td>System32</td><td>WHITE_RABB1T</td></tr>
<tr><td>SYS_32</td><td>WHITE_RABB1T</td></tr>
</table>

3.3 Methodology | Tools and Commands

The tools and commands used to perform the dictionary attacks are as follows:

- **Tool:** *Hashcat* – a robust password recovery utility renowned for its speed and versatility.
- **Command(s):**
  - Using the wordlist:

```
hashcat -m 1000 -a 0 2024Spring-PVDwindows2k3.txt comprehensive_wordlist.txt
```

- **Tool:** *John the Ripper* – an open-source password security auditing and password recovery tool used for detecting weak passwords.
- **Command(s):**
  - To initialize a brute force attack using john.

```
john --incremental 2024Spring-PVDwindows2k3.txt
```

- To use a customized comprehensive_wordlist.txt file.

```
john 2024Spring-PVDwindows2k3.txt comprehensive_wordlist.txt
```

3.4 Time Taken

The time taken to find the passwords for the accounts is as follows:

- Administrator: < 5 mins
- Morpheus: < 5 mins
- Neo: < 5 mins
- Trinity: < 5 mins
- Cypher: < 5 mins
- Smith: < 5 mins
- Cobb: < 5 mins
- Arthur: < 5 mins
- Service accounts: > 12 hrs (some brute-force was necessary)

3.5 Computing Environment

The details of the computing environment used to crack the passwords are:

- **Hardware:**
  - Processor Cores: 4
  - Memory: 4008 MB
  - Network Interface: Virtualized within the hypervisor
  - Hard Disk: 2.6 MB used for the operation snapshots (dynamic allocation)
- **Software:**
  - Operating System: Kali Linux, version 6.5.0-kali3-arm64
  - Hypervisor: Identified as a virtual machine within a hypervisor environment.
  - Penetration Testing Tools: Hashcat 6.2.6, John the Ripper 1.9.0

3.6 Conclusion

This section successfully details the cracking of passwords of user and service accounts in a Windows 2003 system using hybrid attacks. The specific passwords cracked, the tools and commands utilized, the time required for each account, and the computing environment setup comply with the assignments requiements. The cracking process for this PVD file was not difficult.

4 Off-line Dictionary Attack: Linux Password Authentication

4.1 Introduction

This section reports the outcome of the off-line dictionary attack to ascertain passwords for Linux user accounts as specified. A significant milestone was achieved by identifying the password for the root account, which has been highlighted.

4.2 Cracked Passwords

The passwords for the Linux user accounts were successfully retrieved and are tabulated separately to maintain clarity and prevent any amalgamation with other tasks within this project.

Table 4: Cracked passwords for Linux user accounts.

Account Name	**Cracked Password**
Morpheus	`dentures`
Neo	`cornbread`
Trinity	`KandW`
Cypher	`Depends`
Smith	`modelT`
Cobb	`oldfogy`
Arthur	`suspenders`

4.3 Methodology | Tools and Commands

The tools and commands used to perform the dictionary attacks are as follows:

- **Tool:** *Hashcat* – a robust password recovery utility renowned for its speed and versatility.
- **Command(s):**
  - Using the wordlist:

```
hashcat -m 1800 -a 0 2024Spring-PVDlinux.txt comprehensive_wordlist.txt
```

- **Tool:** *John the Ripper* – an open-source password security auditing and password recovery tool used for detecting weak passwords.

- **Command(s):**
  - To initialize a brute force attack using john.

```
john --incremental 2024Spring-PVDlinux.txt
```

- To use a customized comprehensive_wordlist.txt file.

```
john 2024Spring-PVDlinux.txt comprehensive_wordlist.txt
```

4.4 Timeframe

The duration taken to retrieve the passwords is documented, and the computing environment details are provided hereunder:

- Administrator: < 5 mins
- Morpheus: < 5 mins
- Neo: < 5 mins
- Trinity: < 5 mins
- Cypher: < 5 mins
- Smith: < 5 mins
- Cobb: < 5 mins
- Arthur: < 5 mins

4.5 Computing Environment

The details of the computing environment used to crack the passwords are:

- **Hardware:**
  - Processor Cores: 4
  - Memory: 4008 MB
  - Network Interface: Virtualized within the hypervisor
  - Hard Disk: 2.6 MB used for the operation snapshots (dynamic allocation)
- **Software:**
  - Operating System: Kali Linux, version 6.5.0-kali3-arm64
  - Hypervisor: Identified as a virtual machine within a hypervisor environment.
  - Penetration Testing Tools: Hashcat 6.2.6, John the Ripper 1.9.0

4.6 Free Lunch Achievement

Unfortunately, the 8th password for 'root' was not cracked. Very interested to learn how this can be done!

4.7 Conclusion

In summary, this section encapsulates the successful application of an off-line dictionary attack to crack passwords for Linux user accounts. The process, tools, and commands used, time taken, and computing environment specifics are comprehensively documented herein. The cracking process for this PVD file would be highly difficult without application of comprehensive_wordlist.txt.

5 Comparison of Three PVD Cracking

5.1 Hashing Algorithms and Their Efficacy

In Windows 7, the NTLM hash function is utilized. This is an improvement over the LM hash function used in older Windows versions like Windows 2003. The LM hash, case insensitive and split into two parts, is less secure. Ubuntu uses the SHA-512 hashing algorithm, which proved to be more secure, with unique salts, improving security during hash collision attacks.

5.2 The Role of Salt in Password Security

When we include a unique/random data string to a password before hashing occurs, we call this salt. Windows 7 & 2003 do not use salt. Instead, NTLM hashes were easily cracked with hybrid attacks. Ubuntu/Linux's use of SHA-512 hashes proved to be troublesome when utilizing pre-computed rainbow table attacks. Noteably, SHA-512 also ensures identical passwords generate different hashes.

5.3 Iteration Count as a Deterrent to Brute-Force Attacks

When we apply the hashing function a number of times during password hashing, we call this iteration count. Windows 7 & 2003 systems do not employ iteration count in hashing. Ubuntu uses iterated hashing of 5000, which proved difficult to crack and required copious computation power.

5.4 Conclusion

Ubuntu/Linux PVD is the most difficult to crack. This is due to the use of a modern hashing algorithm, the addition of salt, and a high iteration count. The Windows systems proved to be less secure due to the lack of these security measures. This analysis underscores the critical importance of employing advanced hashing strategies, including the use of salt and high iteration counts, to bolster system security.

Academic Integrity Pledge

"This work complies with the JMU honor code. I did not give or receive unauthorized help on this assignment."