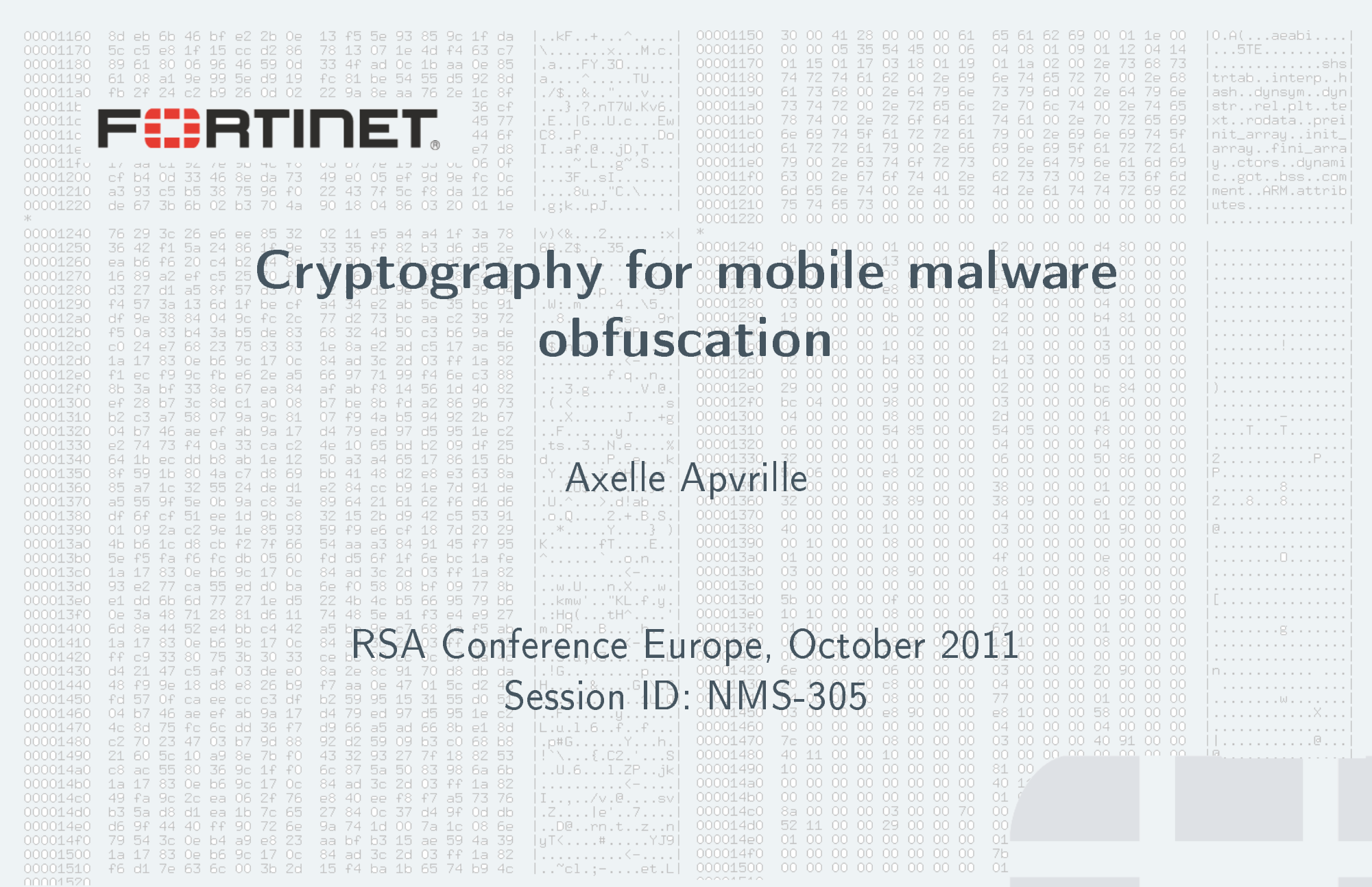


Cryptography for mobile malware obfuscation

Axelle Apvrille

RSA Conference Europe, October 2011

Session ID: NMS-305

Summary

Session objectives

Get familiar with real life mobile malware

Discuss (very) recent malware
Wrong ideas:

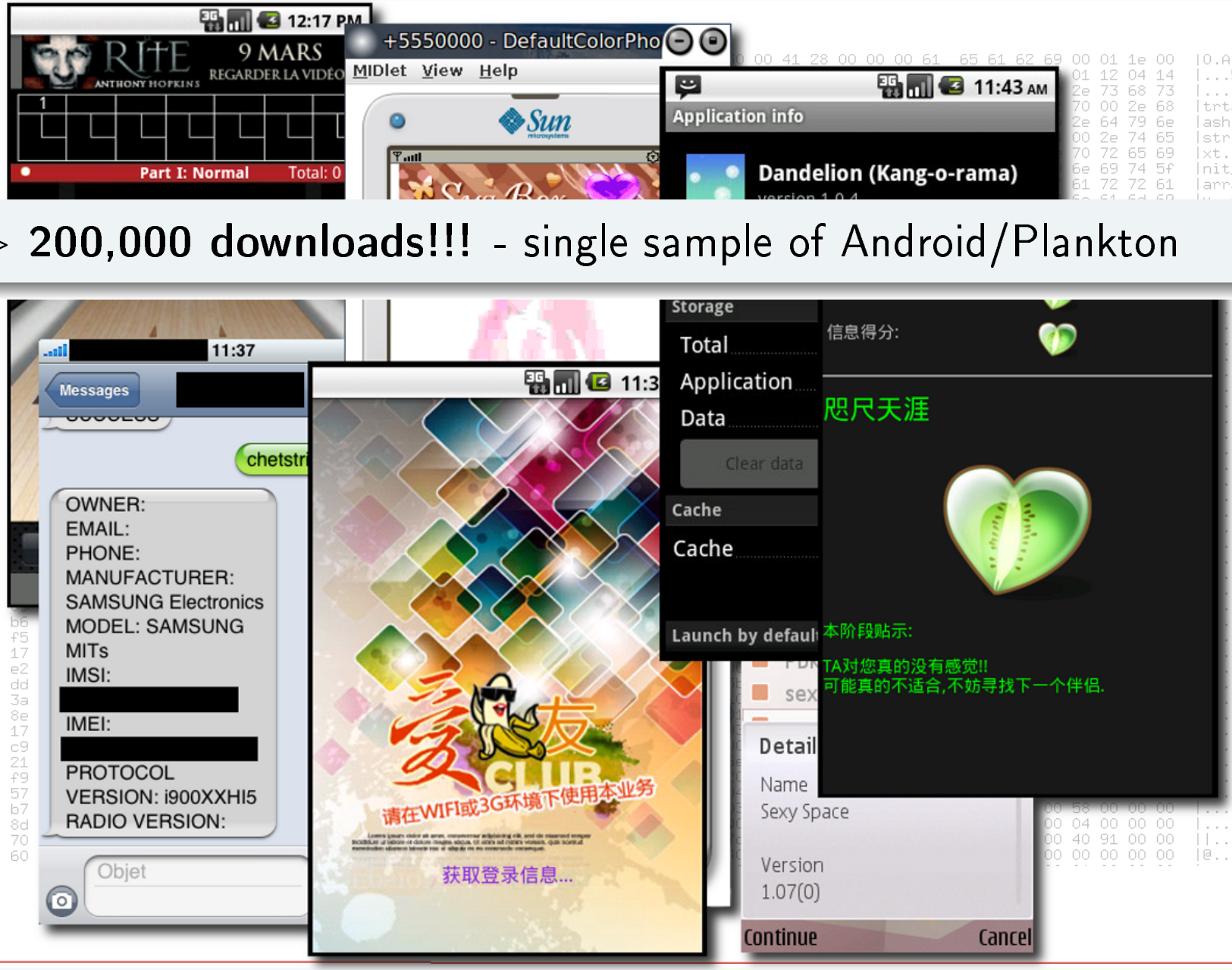
- "This never happens, I need not be concerned"
- "They do not use exploits, no interesting stuff in there"

How To See What's Hidden!

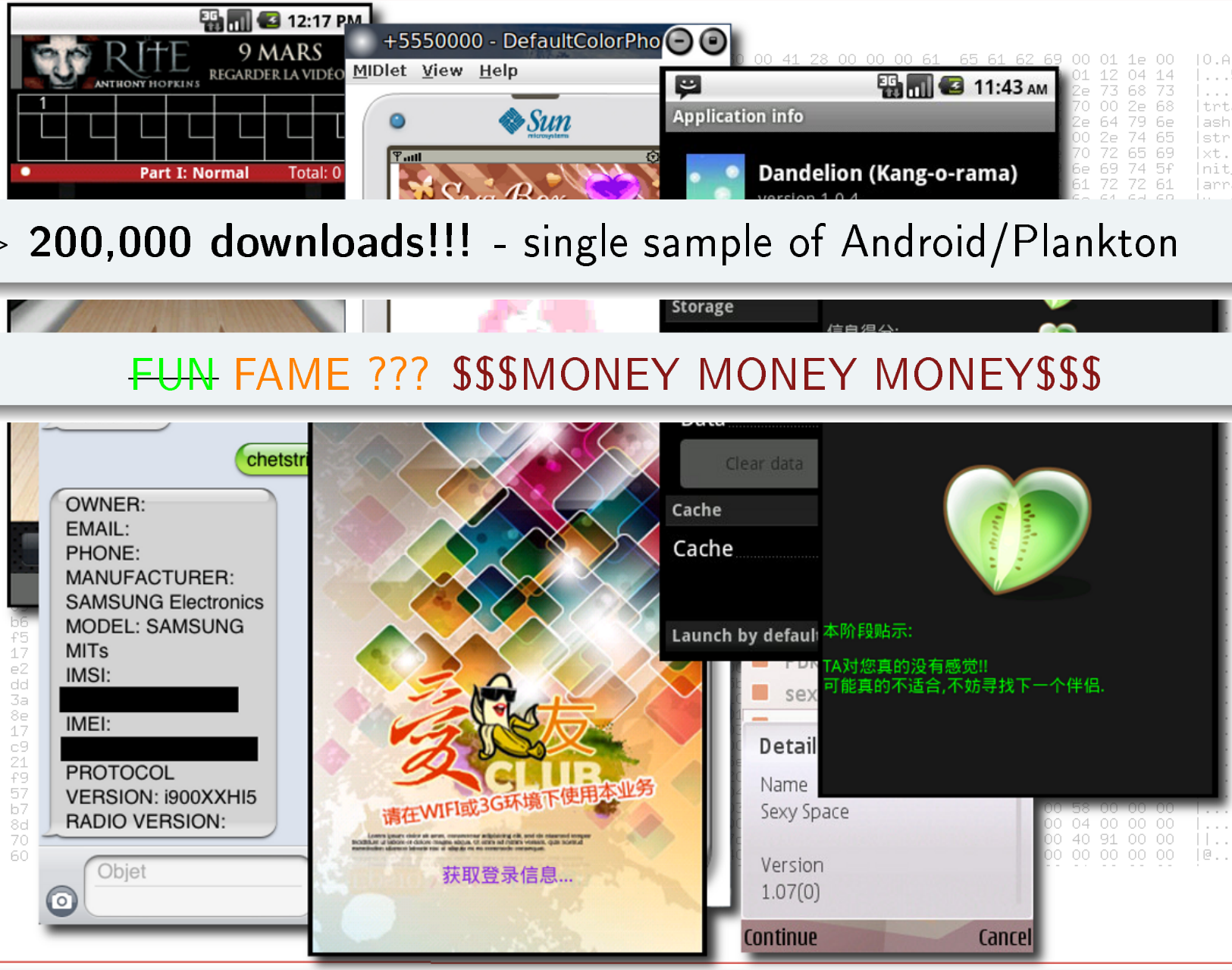
- Spot encryption routines in assembly listings
- Spot the key
- Decrypt!

Step by step examples with real malicious samples!

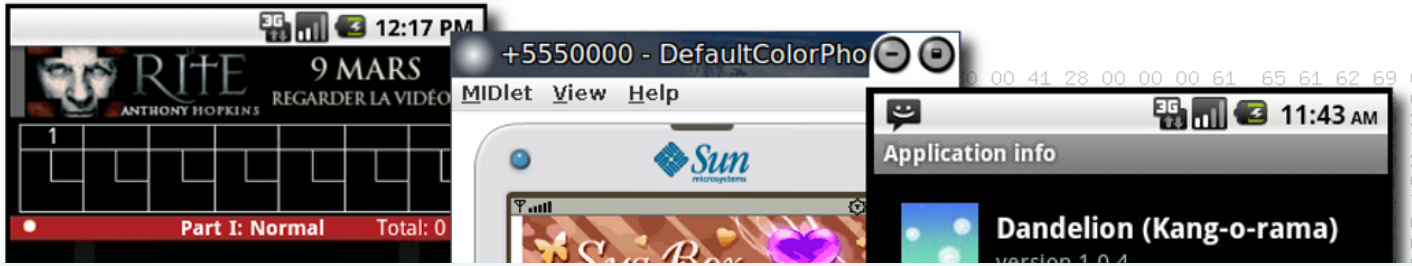
Mobile malware



Mobile malware



Mobile malware



> **200,000 downloads!!!** - single sample of Android/Plankton

~~FUN~~ FAME ??? $$$MONEY MONEY MONEY$$$

Basic malware: very successful

Advanced: exploits, polymorphic code (=code mutates), botnets, crypto...

Crypto in mobile malware

Motivations

- Obfuscation
- Hide maliciousness
- Harden reverse engineering
- Harden detection
- Keep control of their own malicious network

Crypto in mobile malware

Motivations

- Obfuscation
- Hide maliciousness
- Harden reverse engineering
- Harden detection
- Keep control of their own malicious network

Algorithms

Encryption algorithms only: Base64 $\rightarrow$ **AES**
Hash functions, signatures: not used (yet)

What's encrypted?

Encrypted payload	Motivation
SMS short codes and bodies	Analysis difficulty?

What's encrypted?

Encrypted payload	Motivation
SMS short codes and bodies	Analysis difficulty?
URLs of remote servers or C&C to contact	Hide maliciousness

What's encrypted?

Encrypted payload	Motivation
SMS short codes and bodies	Analysis difficulty?
URLs of remote servers or C&C to contact	Hide maliciousness
Communication with remote servers	Keep control + Harden reversing

What's encrypted?

Encrypted payload	Motivation
SMS short codes and bodies	Analysis difficulty?
URLs of remote servers or C&C to contact	Hide maliciousness
Communication with remote servers	Keep control + Harden reversing
Variable names, keywords, filename	Harden reversing

What's encrypted?

Encrypted payload	Motivation
SMS short codes and bodies	Analysis difficulty?
URLs of remote servers or C&C to contact	Hide maliciousness
Communication with remote servers	Keep control + Harden reversing
Variable names, keywords, filename	Harden reversing
Exploits or nested executables	Hide maliciousness + Harden reversing & detection

What's encrypted?

Encrypted payload	Motivation
SMS short codes and bodies	Analysis difficulty?
URLs of remote servers or C&C to contact	Hide maliciousness
Communication with remote servers	Keep control + Harden reversing
Variable names, keywords, filename	Harden reversing
Exploits or nested executables	Hide maliciousness + Harden reversing & detection
Binaries: polymorphic exec.	Harden reversing & detection

Summary

Simple obfuscation in Java/SmsBoxer.N!tr 1/2

Decompiled source code

```
this.jdField_b_String = a(b("L1RodW1icy5kYg=="));
```

b() does Base64 decoding, and a() reads a resource.

Simple obfuscation in Java/SmsBoxer.N!tr 1/2

Decompiled source code

```
this.jdField_b_String = a(b("L1RodW1icy5kYg=="));
```

b() does Base64 decoding, and a() reads a resource.

Decode the string

```
$ echo "L1RodW1icy5kYg==" | openssl base64 -d
/Thumbs.db
```

Simple obfuscation in Java/SmsBoxer.N!tr 1/2

Decompiled source code

```
this.jdField_b_String = a(b("L1RodW1icy5kYg=="));
```

b() does Base64 decoding, and a() reads a resource.

Decode the string

```
$ echo "L1RodW1icy5kYg==" | openssl base64 -d
/Thumbs.db
```

Read Thumbs.db

The file contains base64-encoded data:

```
UO1TTn ...  HJlZmYuLi4NCg==
```

Simple obfuscation in mobile malware 2/2

Decode Thumbs.db

```
SMSNum-1: 3353
SMSText-1: xesss 3689
SMSNum-2: 3353
SMSText-2: xesss 3689
SMSNum-3: 7132
SMSText-3: xesss 3689
end of preff...
```

Simple obfuscation in mobile malware 2/2

Decode Thumbs.db

```
SMSNum-1: 3353
SMSText-1: xesss 3689
SMSNum-2: 3353
SMSText-2: xesss 3689
SMSNum-3: 7132
SMSText-3: xesss 3689
end of preff...
```

Base64 obfuscates:

1. the filename
2. the payload (SMS numbers and text)

Summary

Crypto apprentice no. 1: Android/PJapps

- Discovered in 2011, affects Android phones
- Remotely controls the phone: send SMS, add bookmark, visit URL, install app

Code builds this URL:

```
StringBuilder localStringBuilder1 =
    new StringBuilder("http://");
String str1 = Base64.encode(
    "alfo3gsa3nfdsrfo3isd21d8a8fccosm", 1);
...
```

This is **not** base64 + it's **decoding**:

alfo3gsa3nfdsrfo3isd21d8a8fccosm

l o g a n d r o i d 1 8 8 c o m

http://log.android188.com

Crypto apprentice no. 2: Java/Konov.S!tr

Using a hand-made and obscure algorithm

```
public String encryptSFrom = "R$...THE KEY";
String str = "";
char [] paramString = param.toCharArray();
int i = (encryptSFrom.toCharArray()).length - 1;
char [] enc = encryptSFrom.toCharArray();
int j = paramString.length - 1;
for (int l = 0; l <= j; ++l)  {
  int k = -1;
  for (int i1 = 0; i1 <= i; ++i1)
    if (enc[i1] == paramString[l])  {
      k = i1; break;
    }
  if (k != -1)  {
    if (k == 0) k = i;  else k -= 1;
    paramString[l] = enc[k];
  }
  str = str + paramString[l];
}
return str;
```

Java/Konov.S!tr: Decrypting the ciphertext

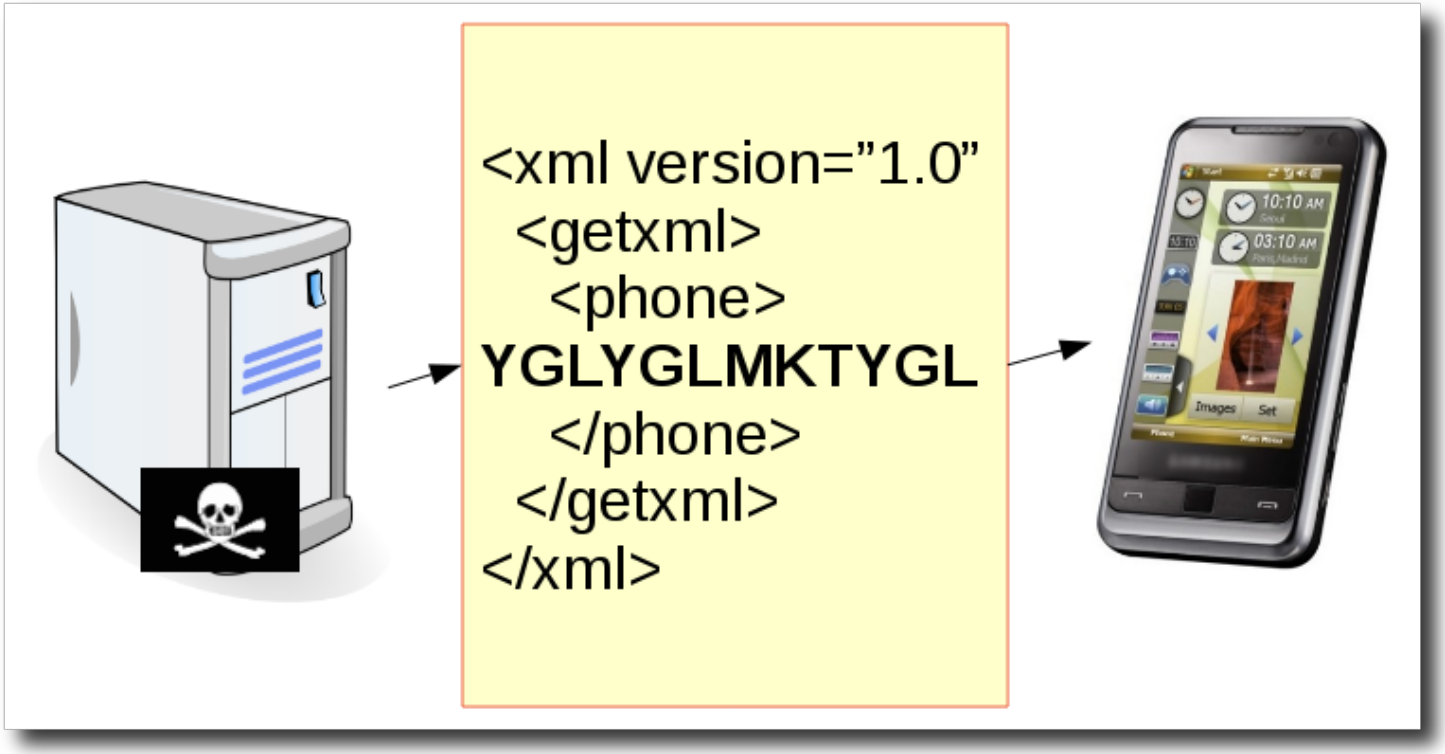
- Encrypts a file named /numbers.cfg (=ciphertext)
- No need to *understand* the algorithm, just to decrypt the ciphertext!
- Write a basic Java class, copy / paste the algorithm, call it on the ciphertext:

```
String str = decodeCes(getText("/numbers.cfg"));
System.out.println("Decoding string: "+ str);
```

Result: SMS numbers, body and corresponding price

```
7122::suksa1837::241.55py6.
7132::suksa1837::141.66py6.
8355::suksa1837::86.00py6.
```

Crypto apprentice no. 3: WinCE/Sejweek - 2009



Simple cryptographic substitution

```
Parameters::codeTable->AddShifrRow(S"YGL", S"1");
Parameters::codeTable->AddShifrRow(S"HKR", S"2");
Parameters::codeTable->AddShifrRow(S"DPO", S"3");
Parameters::codeTable->AddShifrRow(S"WHR", S"4");
Parameters::codeTable->AddShifrRow(S"MKT", S"5");
...
```

Crypto apprentice no. 4: SymbOS/ShadowSrv - 2010

Looks like a video downloading application, but sends SMS messages...

```
LDR R1, Y0RIoij[cR?dijWbbRH[]_ijhoR?dij...
```

Simple cryptographic translation (0x0A)

```
c:\System\Install\Registry\Install.reg
```

`Install.reg` is encrypted too (XOR).

Summary

XOR encryption in mobile malware

XOR encryption is basic, is it really used?

Yes.

- Java/Espaw.D!tr, Java/Swapi.AF, Java/Konov.K...
- WinCE/Pmcryptic (2008)
- SymbOS/Yxes (2009)
- SymbOS/Shurufa (2009)
- SymbOS/ShadowSrv (2010)
- SymbOS/Zhaomiao (2010)
- Android/DrdDream (2011)
- ...

Also found on **Pushdo** (PC malware)
Historical: red phone USA / Russia.

Why XOR ?!

Reasons

1. Perfect algo in theory if key truly random + as long as cipher
2. Efficient: 1 instruction
3. Easy to code

Malicious implementations

- Often 1-byte key
- Simple to break: frequency analysis

Spotting an XOR-encryption routine in Symbian

- a function with a buffer and a key as parameter

```
LDR R1, R4 ; R4 is a counter
LDR R0, R6 ; R6 is the buffer
```

- load one byte of the buffer: LDRB

```
BL _ZNK6TDesC83AtCEi ; TDesC8::AtC(int)
```

- apply the XOR key: EOR

```
EOR R3, R8, R3 ; R3 = R3 XOR R8 (key)
```

- increment a counter ADD

```
ADD R4, R4, #1 ; increment counter
```

- loop until all buffer has been processed: CMP, BL

```
CMP R4, R7 ; R7 is the maximum value
BLT loop   ; loop if not finished
```

Finding the value of the XOR key

Method 1. Close to the calling function

```
04 00 A0 E1 MOV     R0, R4              ; string
6E 10 A0 E3 MOV     R1, #0x6E ; 'n'     ; key
3F FA FF EB BL      analyst_xorencrypt  ; arg1 = string
                                        ; arg2 = key
```

Method 2. Break it!

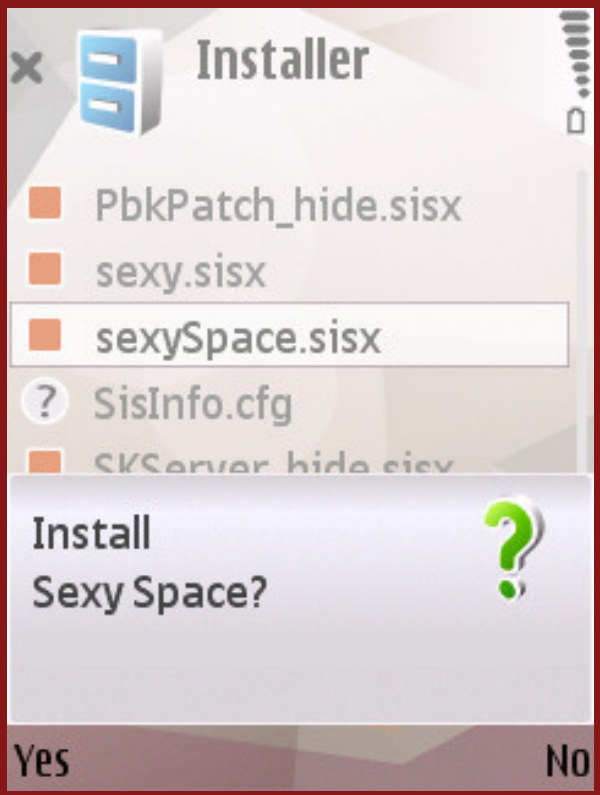
Use XORSearch tool (Didier Stevens)

```
$ wine XORSearch.exe -s Srv.cfg http
Found XOR 57 position 0000: http://[CENSORED]banw.com/api/
```

Taken from SymbOS/Shurufa.A!tr

Hands On Symbian Yxes Worm

Hands On Symbian Yxes Worm



What is it?

SymbOS/Yxes!worm is a **worm** for mobile phones. It sends SMS and connects to Internet. Discovered in 2009.

Why is it important?

1. High bills for victims
2. Said to have affected *"hundreds of thousands"* devices in China [source: Daniel Hoffman, CTO of Smobile]
3. First malware for Symbian OS 9
4. Advanced: hidden connections to Internet and SMS sending...

Hands On Yxes: spot the decryption routine

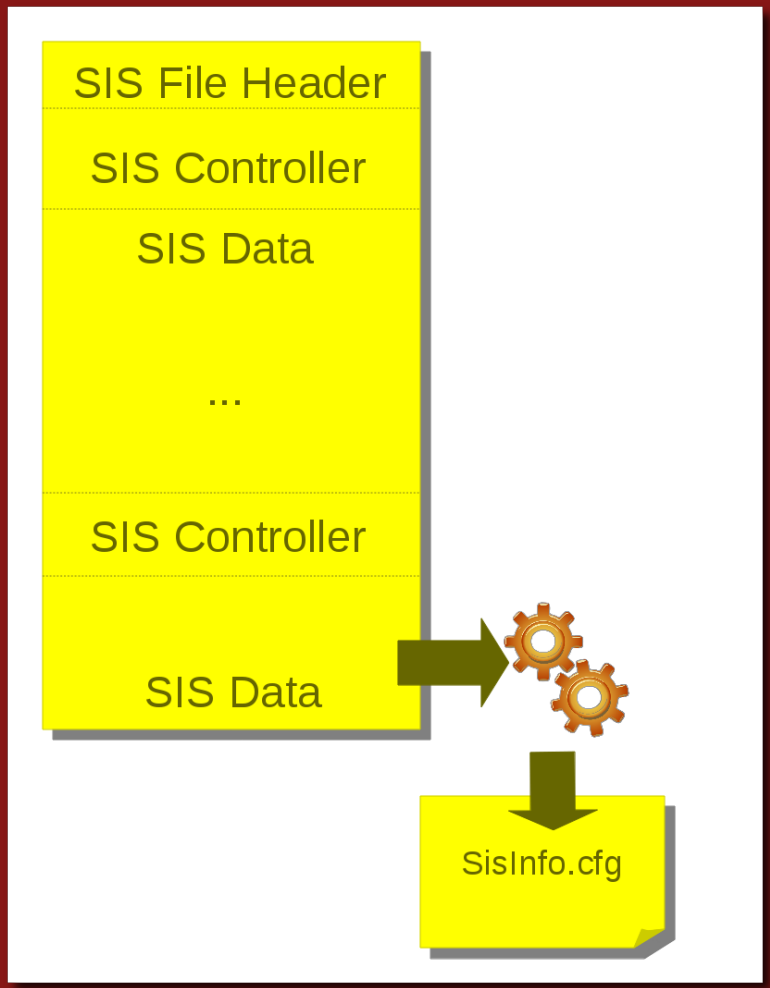
Where is the config file SisInfo.cfg?

Contents of Symbian package:

- Resource file [20026CA5].rsc: No
- Main malicious executable: AcsServer.exe No
- Installer: 0x20026CA6.exe Try here

Downloaded from a URL? which URL?

Hands On Yxes: Identify the encryption key



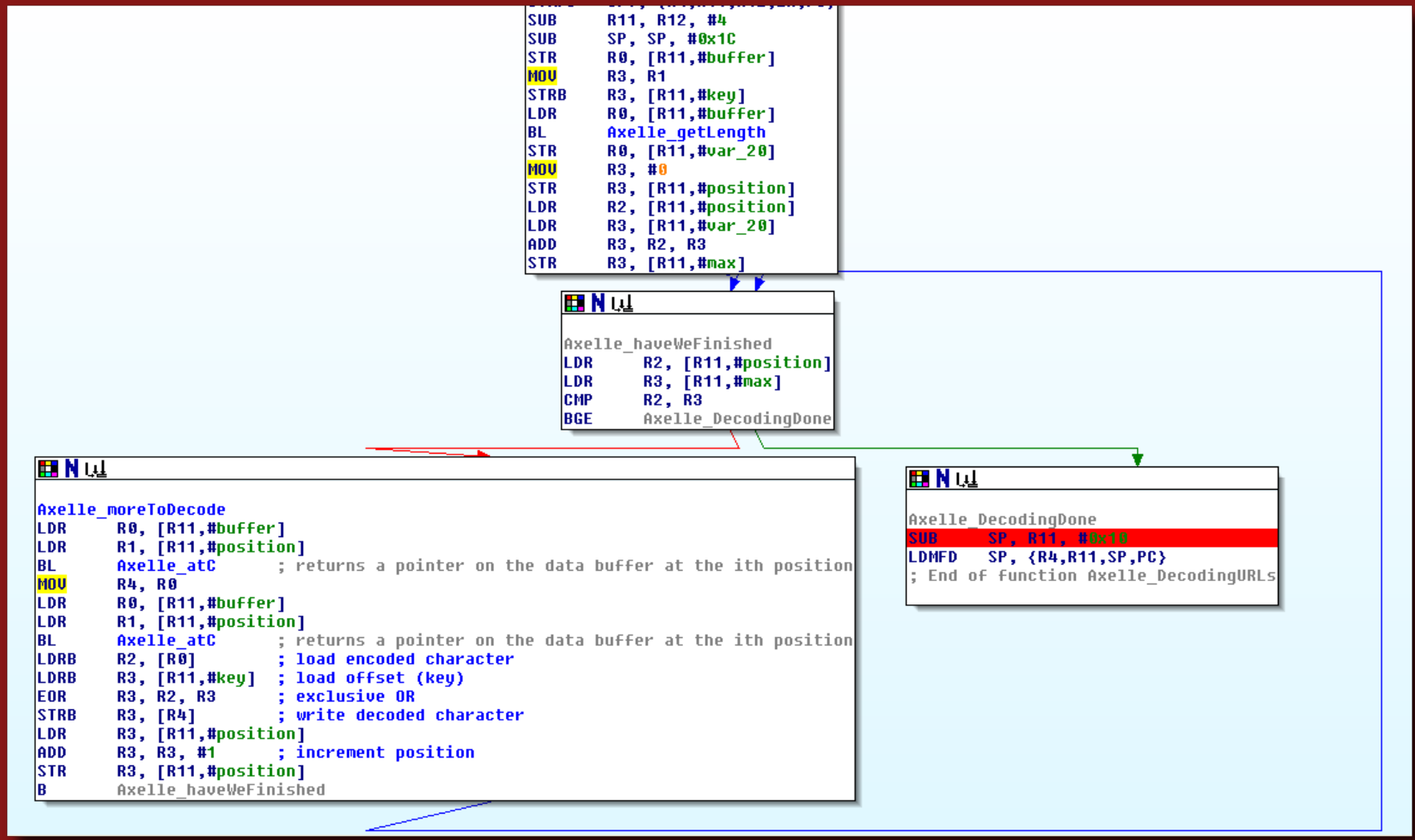
Reversing of the installer

Read section in Symbian package:
huh? why?

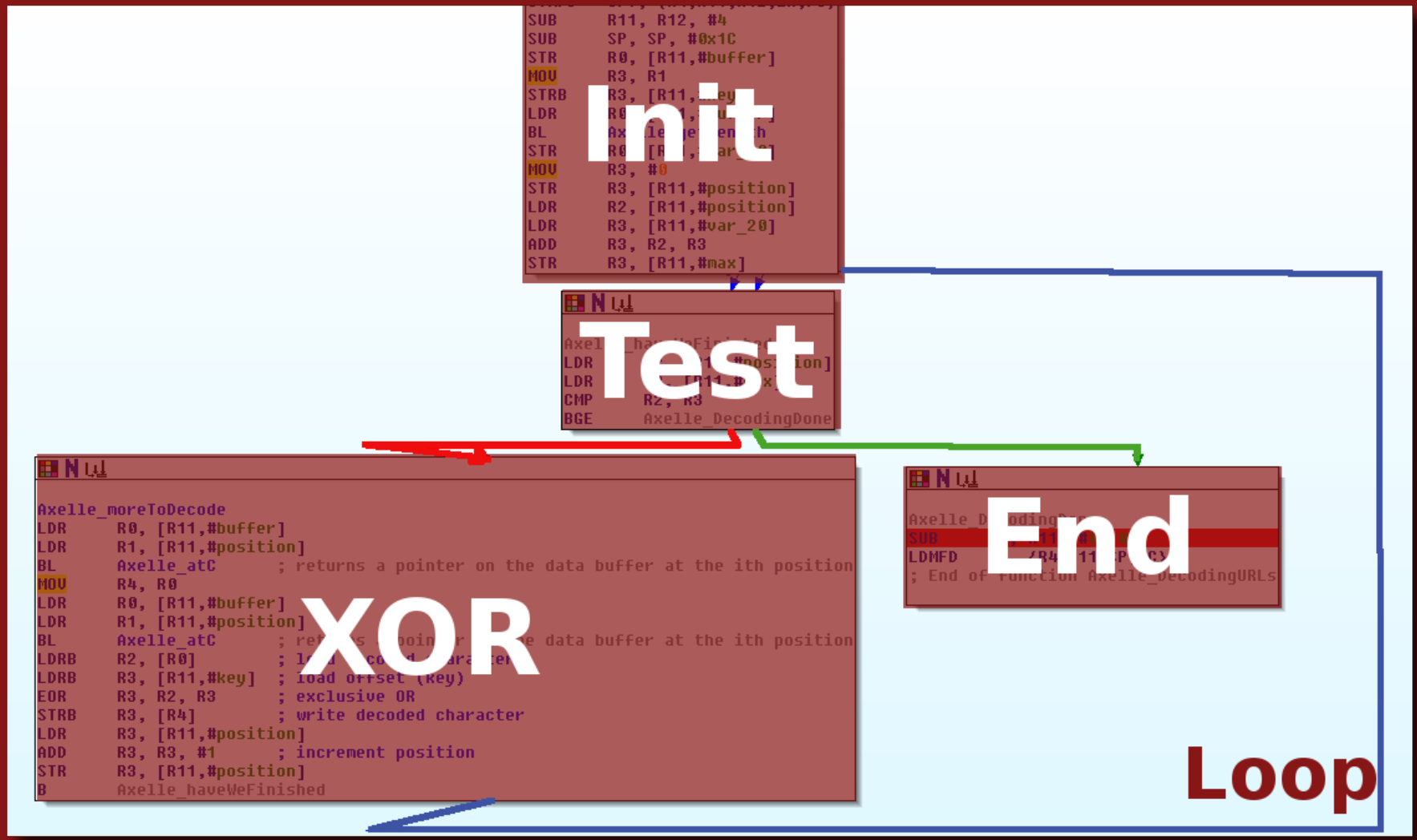
```
SUB R0, R11, #0xBC
MOV R1, #0xBF
BL Yxes_func
```

Key = 0xBF
Decryption routine = Yxes_func

Hands On Yxes: decryption routine



Hands On Yxes: decryption routine



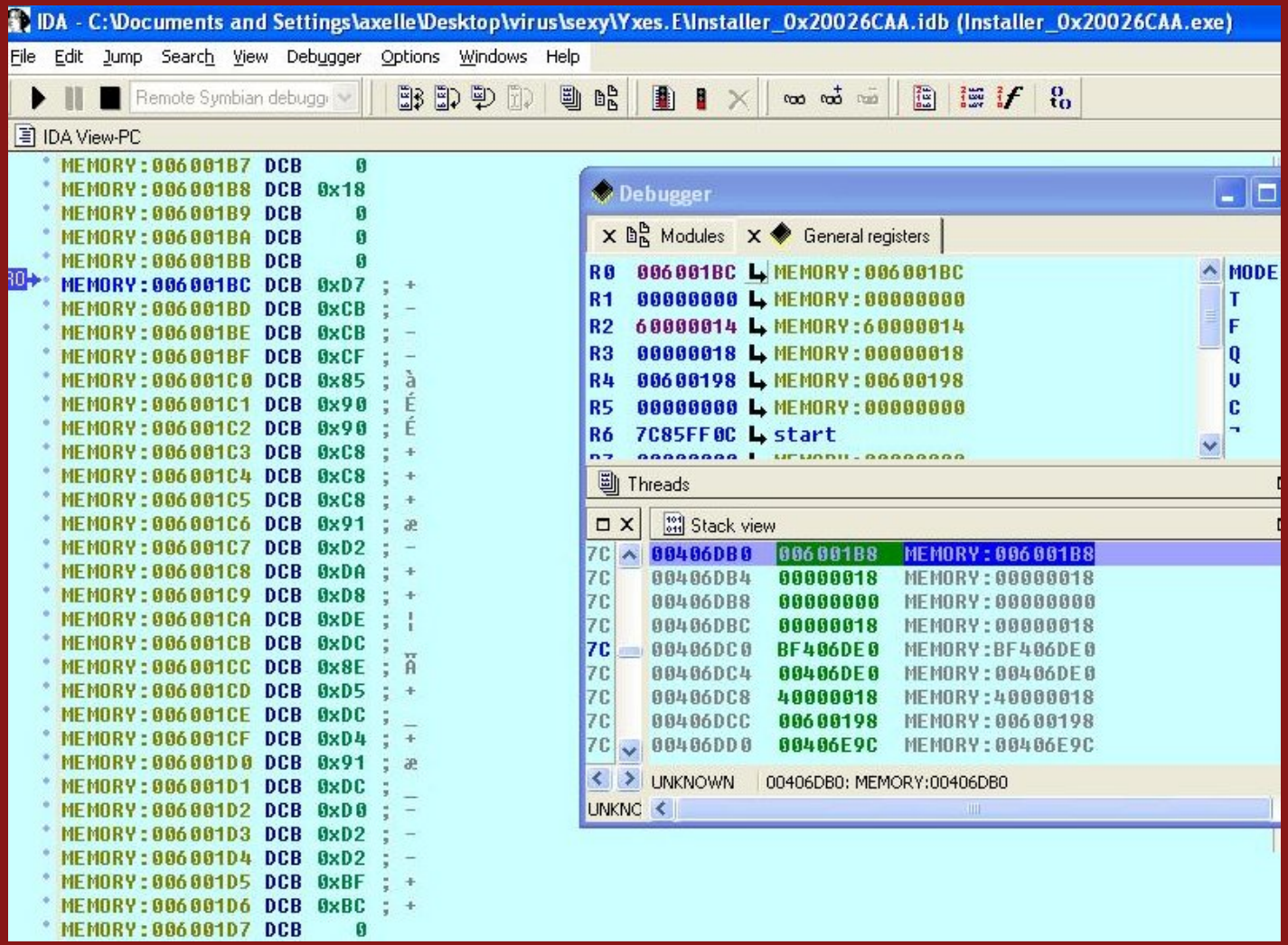
Decrypt it! Method 1

Read the configuration file on the phone

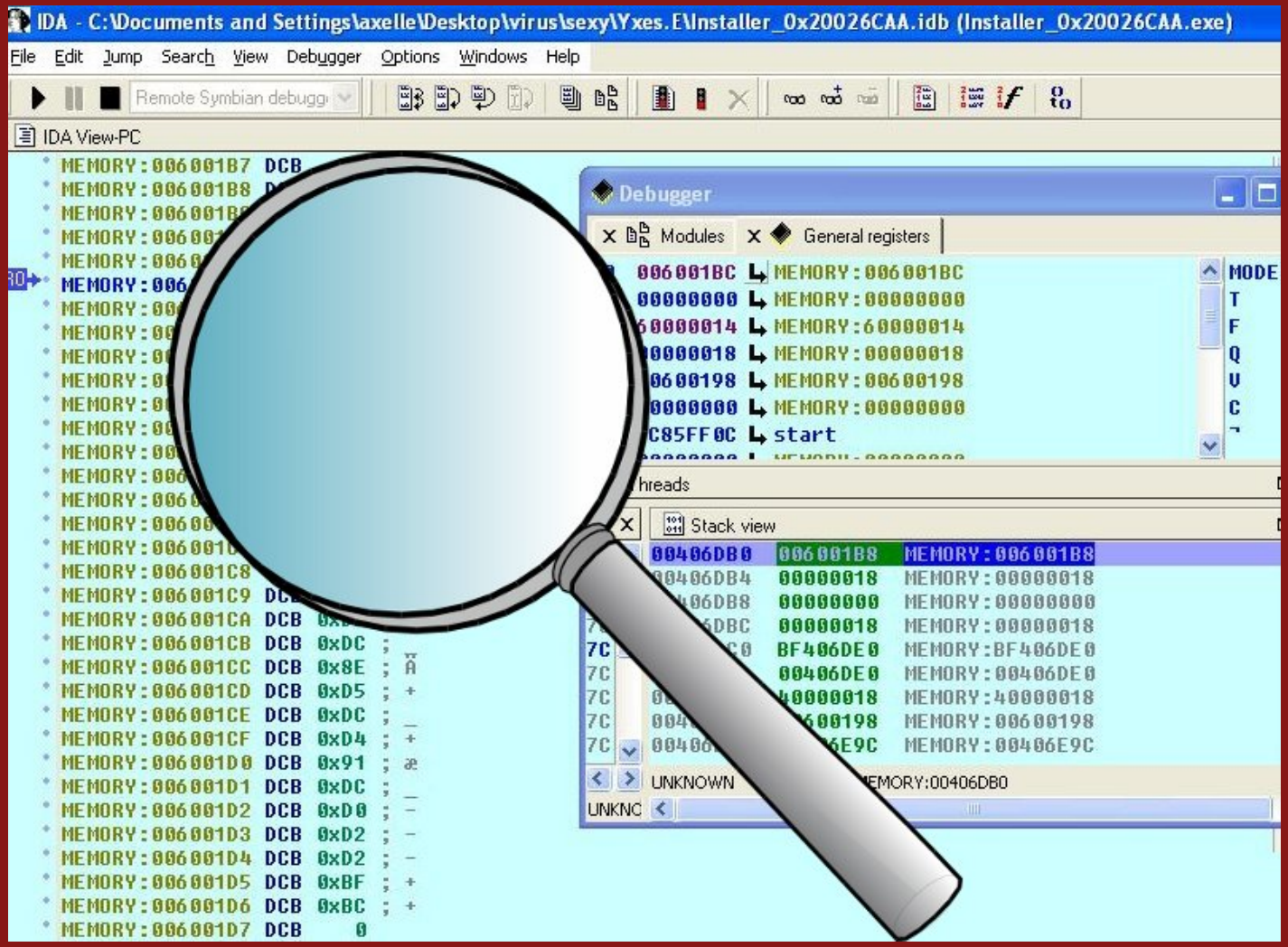
Easy but beware: ensure you do not propagate the virus

```
$ hexdump -C SisInfo.cfg
2f 2f 77  |........http://w|
63 6f 6d  |ww.megac1jck.com|
77 2e 6d  |....http://www.m|
00 00 68  |akt000b.com....h|
69 61 66  |ttp://www.mediaf|
74 70 3a  |ir8.com....http:|
30 61 64  |//www.megaup10ad|
2f 2f 77  |.com....http://w|
6d        |ww.mozi11a.com|
```

Decrypt URLs from a debug session - Method 2



Decrypt URLs from a debug session - Method 2



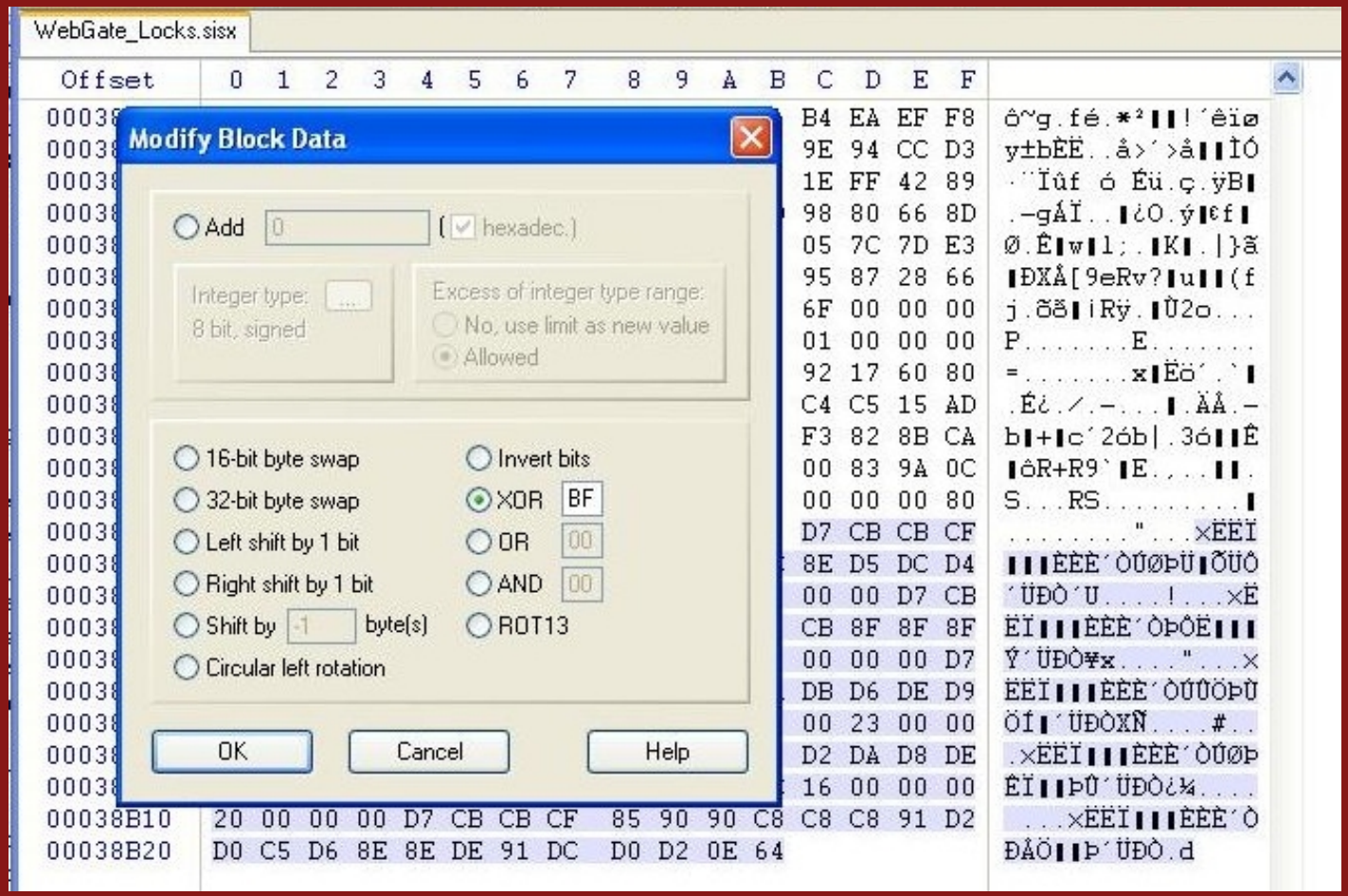
Decrypt URLs from a debug session - Method 2

```
MEMORY:006001BC DCB 0xD7 ; +
MEMORY:006001BD DCB 0xCB ; -
MEMORY:006001BE DCB 0xCB ; -
MEMORY:006001BF DCB 0xCF ; -
MEMORY:006001C0 DCB 0x85 ; à
MEMORY:006001C1 DCB 0x90 ; É
MEMORY:006001C2 DCB 0x90 ; É
MEMORY:006001C3 DCB 0xC8 ; +
MEMORY:006001C4 DCB 0xC8 ; +
```

Decrypt URLs from a debug session - Method 2

```
MEMORY:006001BC DCB 0x68 ; h
MEMORY:006001BD DCB 0x74 ; t
MEMORY:006001BE DCB 0x74 ; t
MEMORY:006001BF DCB 0x70 ; p
MEMORY:006001C0 DCB 0x3A ; :
MEMORY:006001C1 DCB 0x2F ; /
MEMORY:006001C2 DCB 0x2F ; /
MEMORY:006001C3 DCB 0x77 ; w
MEMORY:006001C4 DCB 0x77 ; w
```

Apply the XOR key to the ciphertext - Method 3



Apply the XOR key to the ciphertext - Method 3

XOR encryption in Java code

Java/SmsBoxer.F!tr

- the package contains an encrypted file a.zip and Java classes
- we decompile the classes
- one of the classes shows it loads the a.zip resource and then:

```
new a(...).field ^ 0x78 ^ 0x78;
```

where the constructor of **a** does:

```
this.a = (paramInt ^ 0x78);
```

XOR key

value xor 0x78 xor 0x78 xor 0x78 = value xor 0x78

Decrypting the resource of Java/SmsBoxer.F!tr

Quick Perl script to decrypt a.zip

```
$ cat a.zip |
 perl -ne 'print pack "C*", map {$_^0x78} unpack "C*", $_'
06159395 smswap 473151350 vsxwap 473159395
  smswap 473147122 gywap 473159395 smswap
  473159395 smswap 473
```

XOR encryption in Android/DrdDream.A!tr - 2011

- In the Android Market (removed)
- Root the phone (Rage against the cage)
- Leak private info, install without consent

Ciphertext and key are hard-coded

```
arrayOfByte[0] = 94;
arrayOfByte[1] = 42;
arrayOfByte[2] = 93;
...
KEYVALUE = "6^)(9-p35...";
```

XOR encryption

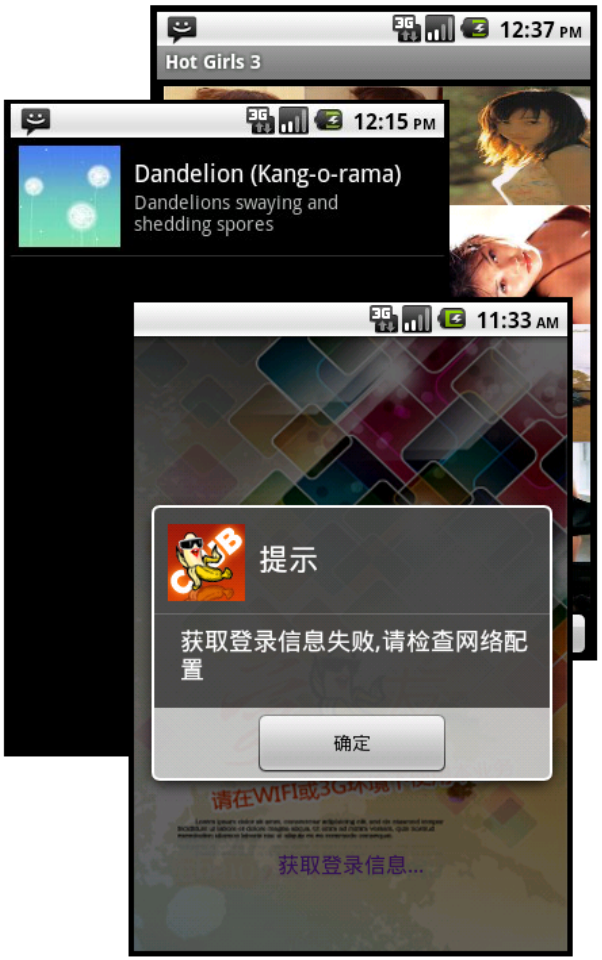
```
int l = arrayOfByte[j];
int i1 = KEYVALUE[i];
int i2 = (byte)(l ^ i1);
```

Decrypted value

http://[CENSORED]45.17:8080/GMServer/GMServlet

Summary

Modern algorithms in malware



- Android/Geinimi (Jan 2011): DES
- Android/Hongtoutou (Feb 2011) : DES
- Android/DrdLight (June 2011): DES
- Android/DroidKungFu (June 2011): AES
- ...

Locating the encryption code: search for KeySpec

In Android/DroidKungFu.A!tr

```
SecretKeySpec localSecretKeySpec =
    new SecretKeySpec(arrayOfByte, "AES");
```

In Android/DrdLight.A!tr (smali code)

```
new-instance v1, Ljavax/crypto/spec/DESKeySpec;
const-string v2, "DDH#X%LT"
invoke-virtual {v2}, Ljava/lang/String;->getBytes()[B
move-result-object v2
invoke-direct {v1, v2}, Ljavax/crypto/spec/DESKeySpec;
    -><init>([B)V
```

Beware

Several advertisement kits use encryption!

Identify the key

- Search for hard-coded constants

In Android/Geinimi.A!tr

```
b = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8 };
```

In Android/Hongtoutou.A!tr

```
const-string v0, "48734154"
```

- Search for the code that creates the KeySpec

In Android/DroidKungFu.A!tr

```
private static byte[] defPassword = { 70, 117, 99, 107,
    95, 115, 69, 120, 121, 45, 97, 76, 108, 33, 80, 119 };
..
byte[] arrayOfByte = defPassword;
SecretKeySpec localSecretKeySpec =
   new SecretKeySpec(arrayOfByte, "AES");
```

- Search in assets or resources for unusual data

Decrypt it!

It's not difficult!

Write a standalone program, and copy/paste the malware's encryption code!

MyDecrypt.java for Android/Hongtoutou.A!tr

```
public class MyDecrypt {
    private byte [] desKey;
    public String kk;
    public static String CIPHER = "39...";
    public MyDecrypt() { this.kk = "48734154";   }
    public static String decrypt(String paramString1,
       String paramString2) throws Exception {
byte[] arrayOfByte1 = convertHexString(paramString1);
Cipher localCipher = Cipher.getInstance(
        "DES/CBC/PKCS5Padding");
byte[] arrayOfByte2 = paramString2.getBytes("UTF-8");
     ...
}
```

Android/Hongtoutou.A!tr decrypted

The result

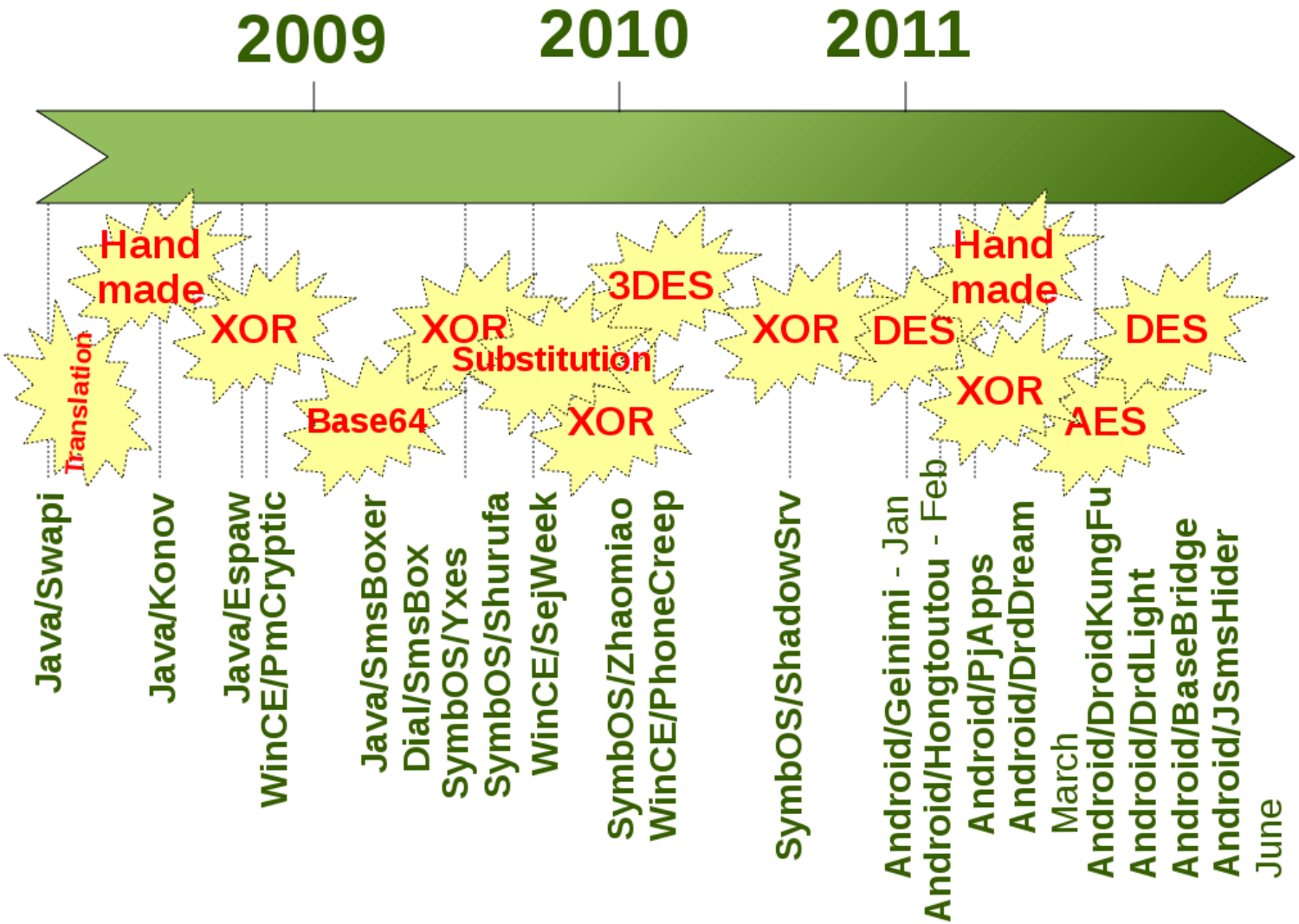
```
Result=B#1#963a_w1|http://[CENSORED]2.105/g/g.ashx?w=963a_w1
|1|http://[CENSORED]2.105/add/pk.aspx$B#1#961a_w1|
http://[CENSORED]2.105/g/g.ashx?w=961a_w1|1|
http://[CENSORED]2.105/add/pk.aspx$B#1#964a_w1|
http://[CENSORED]2.105/g/g.ashx?w=964a_w1|1|
http://[CENSORED]2.105/add/pk.aspx$B#1#978a_w1|
http://[CENSORED]2.105/g/g.ashx?w=978a_w1|1
```

What is this?

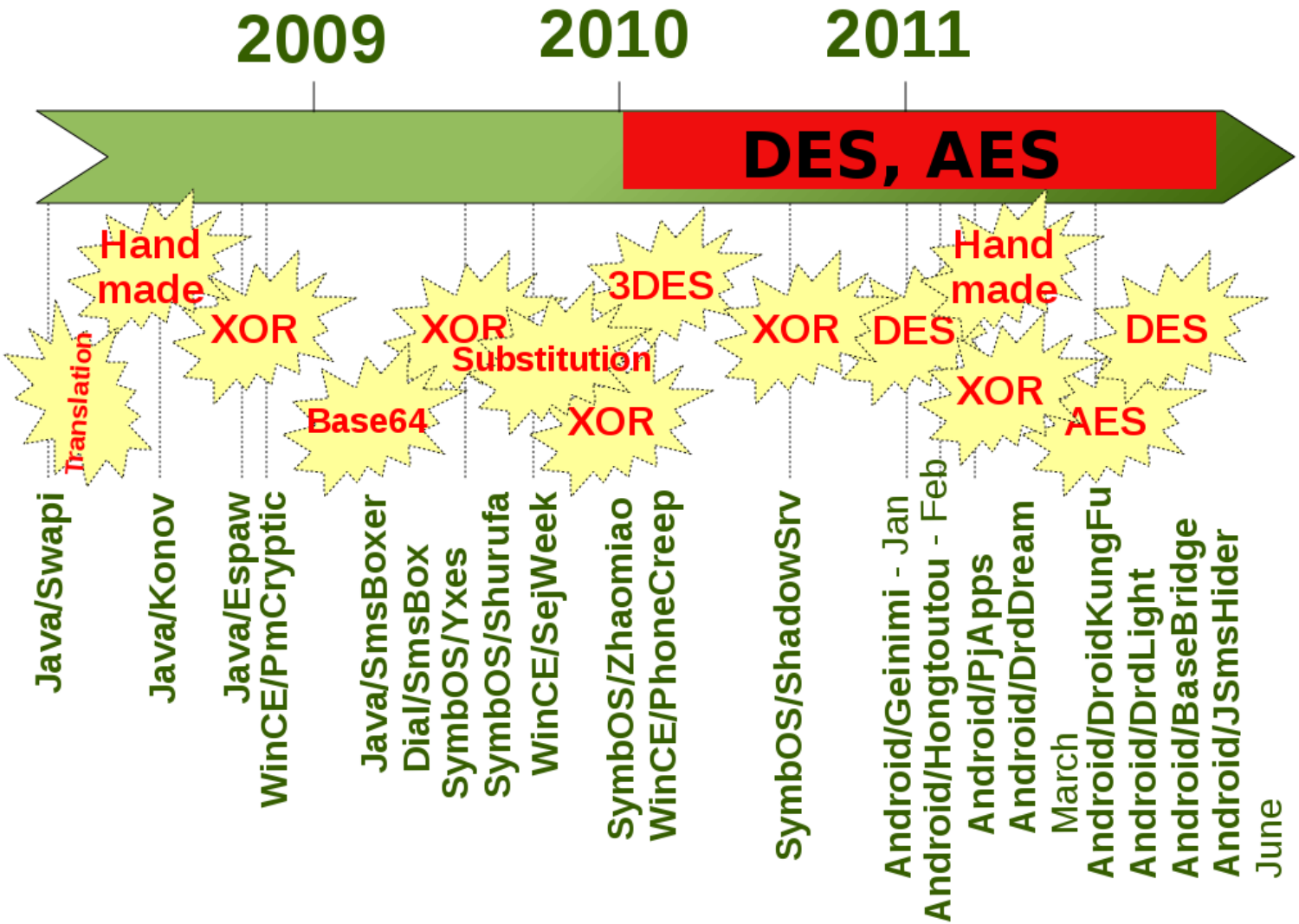
Result=Parameters | URL | Params ...

Summary

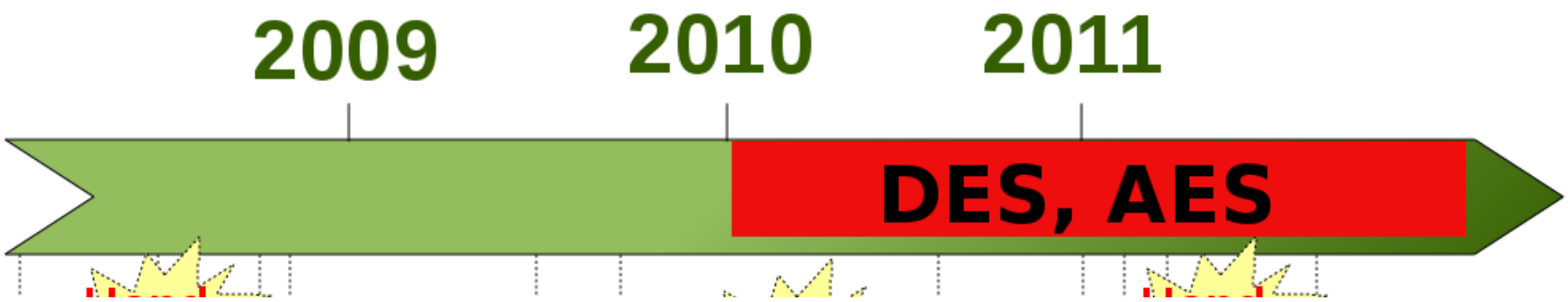
Noticeable Trends in Crypto



Noticeable Trends in Crypto



Noticeable Trends in Crypto



Explanations

- DES, AES in Android API
- Botnets

How efficient is it?

Efficiency against detection

Poor: AV signatures usually not based on encrypted data.
Only efficient against *basic* hash/checksum-based signatures.

Efficiency against analysis

Algorithm	Time to reverse
base64	☺
XOR	☹
AES, DES	☹
Custom	☹

- Hard-coded keys and ciphertext quite easy to spot
- Assembly more difficult to reverse

Apply

You are a ... mobile phone user

Do not trust your mobile phone (yet)

You are a Security Researcher, Architect, Cryptographer

- Mobile malware **IS** an issue.
- Mobile malware use crypto, exploits etc. Spread the word.

You are a (nice) developer

- Secret? Don't put it in the code ;)
- Have a look at Key Agreement schemes

You study malware

- Get your hands on a few mobile malware samples
- Spot the algo, the key, write your own decrypt code

References

- A. Apvrille, Symbian Worm Yxes: Towards Mobile Botnets?, in Proceedings of the 19th EICAR Annual Conference, pp. 31-54, Paris, France, May 8-11, 2010
- Description of Android/PJApps.A!tr
- Description of Java/Konov.S!tr
- Description of Java/SmsBoxer.F!tr
- Description of Java/SmsBoxer.N!tr
- Description of SymbOS/ShadowSrv.A!tr
- Description of SymbOS/Shurufa.A!tr.dldr
- Description of Android/DrdDream.A!tr
- Description of Android/Hongtoutou.A!tr
- X. Jiang, Security Alert: New Sophisticated Android Malware DroidKungFu Found in Alternative Chinese App Markets, June 4, 2011
- A. Apvrille, Android/DroidKungFu uses AES encryption, June 9, 2011
- A. Lelli, A Smart Worm for a Smartphone – WinCE.PmCryptic.A, June 29, 2009
- T. Strazzere, T. Wyatt, Geinimi Trojan Teardown, January 6, 2011
- D. Maslennikov, Trojan-SMS.WinCE.Sejweek, December 17, 2009

Thank You !

Follow us on http://blog.fortinet.com
or twitter: **@FortiGuardLabs**

Axelle Apvrille

aka *Crypto Girl*
/mobile malware reverse engineering/
aapvrille@fortinet.com

Slides edited with LOBSTER