Strong payload obfuscation and encryption

Ariel Waissbein

Core Security Technologies
Humboldt 1967 2
Cda. de Buenos Aires, Argentina
Ph: (5411) 5556-2673
www.coresecurity.com

Introduction

ABOUT

- Networked computing devices are exploitable.
- If the attacker can access the computer through a wire, wirelessly or physically, he might exploit it.

Introduction

NEXT

- Understanding the attacker, yes.
- Once the attacker has compromised the network, the *Security Officer* in charge must know
  - What has happened?
  - Could the attack have been avoided?
  - Is there any legal evidence?
  - What were the attacker’s plans?

The Anatomy of an Attack

The Beginning of an Attack

THE EXPLOIT

- The attacker exploits a bug in a vulnerable system.
- This also means that the exploit evaded the computer's protection.
- We must admit that this will happen.
  - 0-days
  - Unpatched systems

Caveat: all the following attacks require that the attacker has the ability to run his code in the hacked system.

The Second Step of an Attack

THE PAYLOAD

- Targeting step:
  - Search for a specific user/system, email account, etc.
  - Or grow a well-sized network of attacked systems.

- Execution step(s):
  - Steal credentials, credit cards, source code or contacts.
  - DDoS a list of IP addresses.
  - Delete all databases; etcetera.
  - Spying

Detection

TUNING UP

- The hacked network can have some *detection programs* that ***log*** *security-related events*.

- Paradigm:
  - some attacks can be stopped, those we stop.
  - Others are detected with some innocent behavior, this we log.

- The Security Officer (SO) inspects these logs periodically, looking for suspicious information.

After detection, Investigation

WHAT'S DONE

- Logs and all the forensic information allow the SO to answer:
  - What has the attacker done?
  - Could it be prevented in the future?
  - Did he leave any evidence? Backdoors?
  - What was his aim?

- Let's say the attacker is caught in the act:
  - Same questions as above.

Let’s study attacks!

Attack 1: will do anything

A FIRST PAYLOAD EXAMPLE

- The payload is a system call (syscall) proxying server.
- Through a syscall-proxying client the attacker sends syscalls to the target, which are executed there, and the result is returned.
- It is a lightweight payload *injected* in a running process.
- No data is written to disk!
- Communications
  - Go through standard channels
  - encrypted with AES.

Defense

DETECTING ATTACK 1

- Only a combination of reverse engineering and a log-everything approach will work.
  - A word on host-IDSs that monitor syscalls

- The SO must read *IDS logs* and locate the syscall-proxying server, the encrypted commands and the returned answers.

- *Reverse engineering* will help to
  - understand that the payload is a syscall-proxying server and
  - recover the AES key.

Attack 2: with one goal in mind

A 2nd EXAMPLE

- The attacker
  - Generates an AES key *K*,
  - Computes `c:=SHA-1(K)`
  - Encrypts his code as *`encryptedCode:=AES(K,code)`*

- The payload is embedded in a process
  - It listens in a fixed port for all incomming packets

```
For every packet x it computes SHA-1(x)
If SHA-1(x)=c then
    execute AES^-1(x,encryptedCode)
```

Attack 2': with *n* goals in mind

A 2nd EXAMPLE (revisited)

- In fact, there are many keys $k_1, \ldots k_{15}$
- Each encrypting a different functionality.
- The hashes $c_i := \text{SHA-1}(k_i)$ are stored in a hash table together with many nonces.
- The encrypted functions are stored one after the other in memory, and its startpoint and endpoint is computed from the keys.

After detection, what?

- Now the SO has to reverse engineer the obfuscated payload, and find out what it does.
- Look for (candidate) keys, i.e., every string that entered the network.
- Brute force the encrypted functions.
- But he doesn't know n! In fact, it could be made worse...

Attack techniques

Before things get too complicated,
let's check two useful tricks

Trick 1: Cryptography, use cryptography

The attacker can use cryptography

- An authentication/key generation to authenticate agent with the attacker and ensure confidentiality.
- An encryption scheme with backwards secrecy to prevent the SO from obtaining old messages.
- MACs to ensure the integrity of communication.

Implementing Cryptography in Agents

THREE VARIANTS

- V1: The attacker must do some pre-processing:
  - Compile it using shellforge ([P. Biondi]) or Gera's magic makefile ([Gera]).
  - Send the source code for a C implementation of this crypto functions.
- V2: Using Virtual machines for remote execution
  - Install an agent in the target machine that runs Mosquito
  - Send the source code for a Lisp1 implementation of this crypto functions

Implementing Cryptography in Agents

THREE VARIANTS

- V3: Save the trouble, but spend some more bandwith:
  - In a first stage send a compiled **userland exec**. This is an agent that receives any executable (in the targeted machine) and runs it.
  - Next compile in **his** machine all the crypto functions, and send the .exe to the port where userland exec listens.

<u>Note:</u> In the future, when we need to execute *any* functionality, we shall use one of these techniques.

Trick 2: Obfuscation with Secure Triggers

Program Obfuscation

- Informally, a program in binary or source code form is said *obfuscated* if it cannot be analyzed.

- Historically:
  – There are many *ad hoc* methods for obfuscating code.
  – Theoretical results imply that obfuscation is not possible in a general setting.

- We have a practical and theoretically secure obfuscation method for the attack scenario.

Code obfuscation (2)

- Let `k` be an AES key and `c:=SHA-1(k)` its hash.
- Let `P` be data (e.g., computer code) and let `e:=AES(k,P)` be its encryption with `k`.
- Consider the program

```
INPUT x
if SHA-1(x)=c then execute AES^-1(x,e);
```

- Assume you are analyzing this code without further info. What does it do?

More triggering criteria

- Simple trigger: the input mathces a given bitstring
- Subset trigger: the input is a bitstring where certain bits contain a prespecified value (e.g., the input $(x_1,\ldots,x_n)$ in $\{0,1\}^n$ verifies $x_{11}=1$, $x_{25}=1$, $x_{72}=0$, ...).
- Multiple-strings trigger: the input bitstring contains a set of presepcified sub-strings
- Fuzzy combinations, operations, ...

Payload obfuscation and cryptography in the benefit of the attacker

Private information stealing

- Is the analog of Private Information Retrieval.
  - Search: The attacker is searches for a directory/file or email account, and he knows its name.
  - Privacy: He doesn't want anyone to analyse the code and guess what it is looking for.
- Next, the attacker wants to mail him this file, encrypted.
- A simple trigger can be used to this end!

Setup

```
K:=filename;
hk:=Hash(K);
C:=E(K,mailingProgram)
```

Code

```
Scan the hard drive;
For every file found do:
       If Hash(file)=hk then
              execute D(file, C);
```

Private information stealing: analysis

- This program makes little noise (in terms of generating security logs).
- An *a priori* analysis will render that the code is searching for *something*, and after it is found it will execute *some* functionality.
- To learn the *some*, first we must discover the *something*.
- Finding *something*:
  - Bootstrap the code and wait for the *something* to be found
  - Attempt to guess what it is looking for.

More searching

- Actually, using the bit-string trigger the attacker can also look for
  - Specially-formed packets or files (protocols)
  - Use combinations (e.g., an email from X to Y)
- Similar analyses apply to this variants.
  - Only that the brute force search can be made more difficult!

Time bombs

USING TIME-RELEASED CRYPTO [Rivest-Shamir-Wagner 96]

- A TRC encryption scheme
  - allows to set a “time counter” and then encrypt a secret, so that it can be decrypted after the counter reaches zero.
  - It relies on un-parallelizable number theory computations (i.e., g^(2^22.

- The attacker develops a worm that
  a) spreads
  b) “starts the counter” in order to decryt the secret (an executable functionality).
  c) When the secret is decrypted, it executes the functionality and broadcasts the key to other bots

Time bombs: analysis

- The computations will require little prorcessor time and memory.
- There is always the sit-and-wait approach.
- There is no brute-forcing (too big key space)
- Breaking the crypto scheme

Anonymization: analysis

- If the SO captures the bots/agents
  - Stop messages from being delivered (in both directions)
  - he can witness what they encrypt before it is done
  - Then try to catch the attacker when he connects to the forum. Although this approach has some problems.
- Else, there's little he can do!

Anonymization of the attacker

- The attacker can be anonymized when he sends and receives messages.
- Say we are in the setting of Example 2 (many triggers)
- He communicates with his bots through public forums, there's a preset list.
  - Messages from bots are posted encrypted.
  - "Orders" for the bots are posted in the forum as links: http://wormIP/key , and indexers take care of the rest!

Coercion attacks

- The attacker can make any “good-willed” entity surrender their private key.
- He simply makes a worm that
  - Spreads as much as possible
  - Enrypts the hard drive using the underlying public key
  - Then prints the message: Call target entity and ask them for the key.
- That’s it!

In closing...

- Reminder: know your enemy
- Studying how harmful can attacks be.
- About logging: what and where?
- Reverse engineering
- We need better detection mechanisms

So long and many thanks.

Any questions?