

THE EXCHANGE PRINCIPLE V2.0

Román Ramírez <rramirez@chasesun.es>
September, 2004

Introduction

This is a revision about an article already published by Chase The Sun, about The Exchange Principle, where we want to improve some technical concepts that we can't find in the original.

In the traditional forensic science there existed some individuals that made revolutions and carried improvements that jailed criminals that it will be impossible to jail without them.

People like Karl Landsteiner, Paul Uhlenhuth, or Sir Arthur Conan Doyle had improved the traditional forensic science with original ideas.

This one, Edmund Locard, turned the forensic methodologies, making relevant evidences that no one known about or that were rejected seeming useless.

With the famous phrase, "every contact, leaves a trace", Locard built one of the most important principles of forensic science, The Exchange Principle or Locard's Principle.

"Every contact leaves a trace"

If we have an open mind and watch every daily action we perform, we will see that many times we can say and exchange happened.

If we walk through a field filled with rosebushes, we will leave our trace as we wear shoes that will make a deep sign in the land, or through small pieces of branches in our clothes, even leaves or broken flowers that we hit passing through.

But the field has left its trace too, if we analyze our shoes we will find grass or soil that will match what we can find in this area, if we investigate pollen over our clothes, we will detect the kind and the genetic sequence of the flowers and plant that we crossed, we can even analyze these small pieces of leaves or branches or the rose we took for our couple.

This example about the exchange principle is quite simple, but we have many others that commonly are used in forensic investigation, for example crystal fractures.

Whenever we break a glass or crystal, let's say with a punch, we leave a lot of evidences that can place us into the crime scene, but over us and our clothes we carry tiny rest of this glass or crystal we broke which with the right analysis will demonstrate that we are guilty without a doubt.

Rests of clothing under the nails, rests of car-paint in a run, mineral rests after burying a corpse, powder in our hands when shooting a gun... there exist many exchange points that must be investigated between an victim and a suspect: imagine for a moment how many guilty criminals have been freed, and how many innocents were jailed because no one knew anything about these kind of tracks and evidences.

Computer forensic science and Locard's Principle

What about computer forensics? It's not as evident that every contact leaves a trace.

Every expert in forensics or any qualified investigator, with a minimum of experience, will give you a detailed list of computer evidences (usually depending of the volatility of every evidence) that can be used to discover the specific sequence of events or the list of the players involved in this or that case.

But this important principle of traditional forensic science it is not used systematically in computer forensics; not much people will show us details about the exchange between a suspect of computer crime and the systems that he attacks and we will probe that there exist a lot of evidence, that many times are final, that will demonstrate the unique relationship with a specific computer system.

Some examples of exchange in computer evidences

SSH Protocol

One of the evident examples can be found in the way SSH protocol and keys work.

SSH protocol depends on public keys exchange in the beginning of every session, so SSH protocol clients use to store known-remote public keys into a "cache"; in this way they can bypass the key exchange in the next session so they will improve the speed of the connection, and they can check the key is the same, so no one will be able to perform a Man-in-the-Middle attack.

If we remember public cryptography basics, we will see that we have a pair of keys, public and private, that have an unique relationship, so we can say without error that there are no public keys that will match another private key that is different from its twin (and vice versa).

For example, in the case of the freeware ssh client for Microsoft Windows, Putty, we can find these public keys into the registry:

- [HKEY_CURRENT_USER\Software\SimonTatham\PuTTY\SshHostKeys]

Where we can show an example with an invented key:

```
rsa2@22:www.testexample.com
0x23,0xef1a930c701358cca64b07b03e3a6c1ade29a71bba776bfe555c131fe84a7423
69fe7a63c3a2ce8bc1cb39396ea61c685769337b3e471403b5e8cc24c69110ff5d89434
4024fdd08f3852c74da0002de1d96319f38261397ebee35a86faf441e4a15ac7277a9be
46a86a56046a7584bc96176c13af99e5b6ddbcb0668daf62d5
```

This is the culmination of the exchange principle. Don't we have demonstrated that there was a contact and exchange between www.testexample.com and the host that stores this key?

At least, we have absolute evidence that the suspect's computer system has been in contact with the remote server that is the victim or the scene of the crime (not taking care of logs that we will check of course).

Additionally, Putty stores a list of the most common sessions into the registry:

- [HKEY_CURRENT_USER\Software\SimonTatham\PuTTY\Sessions]

Many times we will find users that will try to remove the list of common sessions, but we will have the public keys list, that is absolute.

In Openssh, in Unix environments, we can find the "hosts" file inside the ".ssh/" directory:

- \$HOME/.ssh/known_hosts

In this file our openssh client will store every public key from remote servers, look at this invented example:

```
www.testexample.com,192.168.1.1 ssh-dss
AAAAB3NzAC1kc3MAAACBALoFeOdt47tqujliR/c6tCmJ1D+zgiuLDayaAILkHm
IQOaNK+kpbvWodocPcWmyWKzGcAcdMVOMvZ4/MJEKW8ScPxibOhF4bITmO6eyJRPR
+P8AWf0BiA3s0sddtOfpGSh6Y5UtcifouXJ
bYVLZhH4WAeEpjzvt/mq6x5AiUKI/LAAAAFQDKDyg6mOghliGQNblb0S17LKjlyQAAAI
AfVdZP+YUTwpTxyVOr283XiECmMCPMqY
MzpgZMbNNz4MYrd0IDQLEr4RQL3kzhuGhIAs6PSJNfwPD3HefNL00iVBxleF4MCmnjJTS
AtKlDsYNLtc373mN5uJv+OrYfEJQIUd
LbWc/pksGLYUAau5BLhzV5G6PFRmACSQAntRuOrQAAAIBHpVgp0bmQr5aABnJnqq9U
OTWFtBN9QG486WyiMlvNR1MFN/jxMDEqjG
k4/fy2fSk0Eq5flit6jP8W3zyDJygnzJD1/AQHFAOQVVVSq0Sx6rUouEhmaTIAEwXlIjTOI
m35UZw5RX7Xh9uciBvLZg1rw5g5Yh
QrqWMqWT7ZRuu4pA==
```

In the same way as Putty, we can find public keys in the ssh traditional client application.

And we must repeat, that this evidence is *absolute*, *verifiable* and *irrefutable*, because what we commented about the basics of public and private key crypto.

Web and cookies

How is it generated the value of a cookie in a web application? Many times a MD5 or SHA1 value is calculated with information taken from the remote user, as the remote address (REMOTE_ADDR) date, timestamp, and kind of browser...

Cookies can be stored into a local browser's database into the user's computer, and a session index will be stores into the remote server database, so the application can compare data from the cookie and the session authorization list.

This another evidence of the exchange principle between systems.

In deep, we can take the browser cache as a pool of evidences about the contact and exchange between an user and the server. Every html page, images and documents we could find into the cache can be originated from the server we are investigating.

What happens if we can confirm the contact and relationship between the remote user and the server comparing signs or fingerprints from the server's resources and the user's cache

Even better, take on mind a web application that needs a high level of security, for example, a stock options application.

It would be a great idea to insert some invisible images into the web pages, so the browser will store them into the cache but the user will not see a trace about. If the organization detects a security breach or incident, it will be necessary to demonstrate that this user is guilty, and we can check the invisible images (that will be unique and with limited access to the people that were in contact with these restricted access pages) and comparing MD5 and SHA1 fingerprints we will demonstrate that this particular user ABSOLUTELY was in contact with the restricted areas of the server.

HTTPS protocol

In a different way from SSH, the HTTPS protocol will not give us a final evidence about the contact between the suspect and the server.

In the moment that we try to connect to an SSL server (HTTPS), we can have two different situations:

- a) We have the Certification Authority (CA) certificate that signs the server certificate installed in our browser.
- b) We don't have this CA certificate.

Browsers like Mozilla, Netscape or Internet Explorer have special databases to store known certificates, existing a specific storage for CA certificates and keys.

If we try to connect to a server which certificate is signed by an unknown CA ("b" case), our browser will show an alert and give us the opportunity to store this new key or the opportunity to cancel the session because of the risk.

What happens with those secure web servers where people generate their own certificates?

Many are self-signed, or signed by unknown CAs built on top of OpenSSL or OpenCA.

When the remote client tries to connect to a SSL server with these characteristics, and decides to go ahead with the connection and stores this new CA certificate, we can say the exchange principle is in the way to be applied. If we investigate the certification databases in user's browsers, we will find evidences of the exchange in the CA certificate.

Compiled unique binaries

This kind of example could be parallel to a ballistic investigation, trying to determine from which gun a bullet was shot in a scene.

We know MD5, SHA1 and other hashing algorithms that will give a unique and different result for every different input, and we know we can use them for integrity checking in files.

But what we usually don't take on mind is that we can use these fingerprints against our suspect, that could leave tools inside of our systems (for example rootkits) developed and compiled by himself in his machine.

Look at Trojan Horses or Worms that are distributed in binary format. Imagine they have been compiled using the freeware compiler "gcc", could it be possible that every binary created in a system has a specific fingerprint about this system?

To make a quick check about this issue, we have created this "hello.c":

```
#include <stdio.h>

int    main(void){
        printf("Hello world\n");
        return 0;
}
```

The classic "Hello world", which we will compile in two different systems.

In the first, a Redhat, we got:

```
-rwxrwxr-x 1 user  user  11361 may 18 21:56 hello
MD5 hello: 21ca7fa63d21cc616d3ccbdefe8fc111
SHA1 hello: a09085289250308efd4e1d07b9008c175d30ae8c
```

In our second system, a Debian, we got:

```
-rwxrwxr-x 1 user  user  11256 2004-05-18 21:56 hello
MD5 de hello: ada22274e5fc96ffe7cac22a735c548b
SHA1 de hello: fb19f2431a6de2fbba47acc614618e69af99364a
```

It is obvious that there exist many factors that can make different binaries, among them libraries, different linking ways, optimization changes... but the fact is that in two different systems we have got different binaries with different signatures.

Anyway we cannot have a final thesis about the origin of a specific binary file, but we can get evidence enough to select or discard a suspect.

Think about "rootkits" installed in our system, that many times are not only binary files and including source code files; look at the code comments.

If we can grant access to the suspect's machine, won't it be possible to compile from this source code to check the binaries signatures?

Proprietary application's files

What happens with applications like Microsoft Word, Excel, Powerpoint or like Adobe Acrobat, Distiller or Sun's OpenOffice? These applications use to have binary (and proprietary) file formats and many times they insert "tags" or "signatures" that can identify in a unique (an obvious) way the creator of a document.

Many times we can even extract IP address information from this kind of documents, or going beyond, exact information about the application license number that was used to write it.

Even applications like Windows Media, or the ATTRACK music format, can carry information about the creator of a specific document using fingerprints and signatures (we know about a case when an attacker inserted a joke sound file into the Windows start sequence of his victim).

And what about image file formats like Adobe Photoshop? And Watermarks?

Conclusions

It was NOT our idea to make a deep inventory of every computer evidence that can be used to probe the exchange principle; actually, we wanted to show to the community that traditional computer forensics has many things to offer and we should take care of this kind of resources than not many people use to check.

The Exchange Principle is fully applicable in digital evidences, what we have to do is to, systematically, decide where and how we can find these irrefutable evidences that will let us to catch "the person who hacked this or that system".

We do not comment about printers (that leave signatures in the way they print), paper classes, and other evidences that will improve the investigation, as they are related to traditional forensic science, but don't forget your "trashing" abilities as very valuable resources can be found at the bottom of a basket. And don't forget: "every contact leaves a trace".