

EL PRINCIPIO DE INTERCAMBIO V2.0

Román Ramírez <rramirez@chasesun.es>
Septiembre de 2004

Introducción

Esta es una nueva revisión de un artículo ya publicado por Chase The Sun sobre el Principio de Intercambio, en la que hemos querido ampliar ciertos detalles técnicos que no se habían incluido en la versión original.

En la ciencia forense tradicional han existido diversos individuos que han provocado revoluciones, y que han traído avances sin los que muchos criminales nunca habrían sido capturados y condenados.

Personajes como Karl Landsteiner, Paul Uhlenhuth, o el mismísimo Sir Arthur Conan Doyle han aportado sus ideas innovadoras a la ciencia forense tradicional.

El caso que nos ocupa, Edmund Locard, provocó un giro en la metodología de investigación forense, elevando a imprescindibles una serie de evidencias forenses que antes, o se ignoraba siquiera su existencia o se desechaban por inútiles.

Con su famosa frase, "cada contacto deja un rastro", Locard creó uno de los principios más importantes de la ciencia forense, el Principio de Intercambio o Principio de Locard.

"Cada contacto deja un rastro"

Si mantenemos una mente abierta, y observamos atentamente aquellas acciones cotidianas que todos y cada uno de nosotros realizamos a lo largo de un día normal, veremos que en muchas de ellas podemos afirmar, sin dudar, que ha existido un intercambio.

Si paseamos por un campo lleno de rosales, nosotros dejamos nuestra huella en ese campo a través de nuestros zapatos, que marcan sus huellas en la tierra, a través de pequeñas hebras de nuestra ropa, que han podido quedar enganchadas en las ramas y espinas, e incluso, a través de hojas, ramas o flores rotas que accidental o intencionalmente hemos roto con nuestro paso.

Pero el campo, también ha dejado su huella sobre nosotros, y si analizamos la suela de nuestros zapatos observaremos tierra correspondiente a la zona, si investigamos el polen sobre nuestra ropa, detectaremos el tipo y la secuencia genética de las plantas que nos rodeaban, e incluso, puede ser que pequeños trozos de ramita u hoja, así como una rosa que hemos cortado para nuestra pareja, hayan viajado de regreso a casa con nosotros.

Este sencillo ejemplo sobre el Principio de Intercambio es una muestra evidente, pero podemos poner otros muy comunes dentro de la ciencia forense, como por ejemplo, las fracturas de cristales.

Cada vez que rompemos un cristal, digamos que mediante un puñetazo, dejamos una serie de evidencias que nos sitúan en la escena, pero sobre nosotros y nuestra ropa, automáticamente, han quedado restos microscópicos de ese mismo cristal que, con el análisis adecuado, demostrarán que fuimos nosotros quienes lo rompimos.

Restos de tejido bajo las uñas de una persona que ha sufrido una agresión, restos de la pintura de un coche en un caso de atropello, residuos minerales tras enterrar un cadáver, restos de pólvora en las manos al disparar un arma... hay múltiples puntos de intercambio que deben ser investigados entre una víctima y un sospechoso: imaginen por un momento todos los culpables que han quedado libres, y todos los inocentes que han sido encarcelados por ignorar la existencia de rastros definitivos, como son los que propone Locard.

La ciencia forense informática y el Principio de Locard

¿Qué ocurre con la ciencia forense informática? No es tan evidente que cada contacto deje un rastro, cuando hablamos de evidencias lógicas.

Cualquier experto forense o cualquier investigador con un mínimo de experiencia puede dar una lista de evidencias informáticas (generalmente en función a su volatilidad) de las que podemos obtener indicios para descubrir la secuencia de acontecimientos de, o los actores involucrados en, tal o cual hecho.

Pero este gran principio de la ciencia forense tradicional, no se está aplicando de forma sistemática en la investigación forense informática; muy pocos les apuntarán detalles sobre la interacción de un presunto atacante informático y los sistemas que este ataca, cuando de hecho hay diversas evidencias, que en muchos casos son evidencias finales que demostrarán una interacción única con un sistema informático.

Algunos ejemplos de intercambio en evidencias informáticas

Protocolo SSH

Pienso que el ejemplo más obvio lo podemos encontrar en el funcionamiento del *protocolo y las claves de SSH*.

El protocolo SSH funciona de una manera que implica un intercambio de claves públicas al inicio de una sesión, de forma que muchos clientes de SSH almacenan las claves públicas de los servidores conocidos; de esta manera pueden acelerar ese intercambio en las siguientes conexiones, y además, pueden asegurarse de que no ha cambiado la clave (evitando así ataques de Hombre-en-medio).

Si recordamos lo básico de la criptografía de clave pública, un par clave privada y clave pública están relacionados, y podemos asegurar con cierta fiabilidad que no puede existir una clave pública que corresponda a una clave privada que no sea la asociada.

En el caso del cliente gratuito para Microsoft Windows, Putty, podemos encontrar esas claves públicas dentro del registro, en:

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

Donde podemos encontrar un ejemplo de clave (inventada):

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

¿No es esta la culminación del Principio de Intercambio? ¿No queda demostrado que ha existido un contacto y un intercambio entre un servidor www.ejemploprueba.com y la máquina que almacena las claves de los sistemas con los que ha interactuado?

Concretando, tenemos una evidencia indiscutible de que al menos el ordenador o el sistema operativo de nuestro sospechoso ha estado en contacto con el servidor remoto que consideramos víctima o escena del crimen (todo esto al margen de los archivos históricos que podamos encontrar en el sistema víctima o escena).

De forma adicional, Putty guarda un listado de las sesiones más comunes, dentro de la clave de registro:

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

En algunos casos, puede ser que el usuario del programa Putty, decida eliminar de la lista de sus sesiones el servidor comprometido; aún en este caso, podríamos obtener la clave pública que es definitiva.

En el caso de Openssh en entornos Unix, podemos encontrar el archivo `known_hosts`, dentro del directorio `.ssh`:

- \$HOME/.ssh/known_hosts

En este fichero se van almacenando las claves públicas de los servidores con los que ha entrado en contacto nuestro usuario, veamos un ejemplo (inventado):

```
www.ejemploprueba.com,192.168.1.1 ssh-dss
AAAAB3NzAC1kc3MAAACBALoFeOdt47tqujliR/c6tCmJ1D+zgiuLDAYaAILkHm
lQOaNK+kpbvWodocPcWmyWKzGcAcdMVOMvZ4/MJEKW8ScPxibOhF4bITmO6eyJRPR
+P8AWf0BiA3s0sddtOfpGSh6Y5UtcifouXJ
bYVLZhH4WAeEpjzvt/mq6x5AiUKI/LAAAAFQDKDyg6mOghliGQNbIb0S17LKjlyQAAAI
AfVdZP+YUTwpTxyVOr283XiECmMCPmQY
MzpgZMbNNz4MYrd0IDQLer4RQL3kzhuGhIAs6PSJNfwPD3HefNL00iVBxlef4MCmnjJTS
AtKldsYNLtc373mN5uJv+OrYfEJQIUd
LbWc/pksGLYUAau5BLhzV5G6PFRmACSQAntRuOrQAAAIBHpVgp0bmQr5aABnJnqq9U
OTWFtBN9QG486WylMlvNR1MFN/jxMDEqjG
k4/fy2fSk0Eq5flit6jP8W3zyDJygnzJD1/AQHFAOQVVVSq0Sx6rUouEhmaTIAEwXiljTOI
m35UZw5RX7Xh9uciBvLZg1rw5g5Yh
QrqWMqWT7ZRuu4pA==
```

De la misma forma que en Putty, en el cliente de Openssh tradicional podemos localizar las claves públicas de los servidores con los que hemos establecido contacto.

Y repetimos, esta prueba es **definitiva**, **verificable** e **irrefutable**, por lo antes comentado sobre las bases de la clave pública y privada.

Web y cookies

¿Cómo genera una aplicación web los valores de las cookies que un navegador utilizará? En muchos casos, se calculan un valor de MD5 o SHA1 sobre datos del usuario remoto, como por ejemplo su dirección IP, fecha y hora, tipo de navegador etc.

Las cookies pueden quedar almacenadas en la base de datos del navegador del usuario, y un índice de sesión quedará almacenado en el servidor, para que la aplicación pueda comparar los datos de la cookie del usuario con las sesiones autorizadas.

¿No es esta otra evidencia del principio de intercambio entre sistemas?

Yendo más lejos aún, podemos considerar la caché de los navegadores como puntos de contacto entre usuario y sistema remoto. Las páginas web, imágenes y documentos que podemos encontrar en la caché tienen su origen en las páginas web que el usuario ha visitado.

¿Qué ocurre si podemos confirmar el contacto de un usuario con un servidor web verificando que las firmas de los ficheros de su caché son iguales a las de ese servidor?

Mejor aún, imaginemos un aplicativo web que necesita un alto nivel de seguridad, por ejemplo una página de compra y venta de valores a través de Internet.

Sería una gran idea insertar imágenes invisibles dentro de las páginas web, que el navegador podría almacenar en su caché pero que el usuario no vería dentro de la página. Si la empresa detectara algún tipo de incidente, y necesitara demostrar que un usuario ha sido el responsable, recurriría a esas imágenes (que serían únicas, irrepetibles e inaccesibles excepto para aquellos que hubieran accedido al web) y comparando sus firmas MD5 y SHA1, demostraría que el usuario SÍ ha estado en contacto con su servidor.

Protocolo de web seguro: HTTPS

A diferencia de SSH, el protocolo HTTPS no nos llevará a tener una evidencia concluyente del contacto del presunto atacante con nuestro servicio de web seguro.

En el momento en que intentamos conectar a un servidor de web seguro, se pueden dar dos casos principales dentro de la conexión:

- a) Que tengamos la clave de la autoridad de certificación que firma el certificado del servidor de web seguro, dentro de nuestro navegador.
- b) Que no tengamos esa clave.

Navegadores como Mozilla, Netscape o Internet Explorer, cuentan con bases de datos que almacenan los certificados conocidos, existiendo un apartado especial para las claves de firma de las autoridades de certificación.

En caso de intentar conectar con un servidor cuyo certificado para HTTPS no esté firmado por una autoridad conocida (el caso "b"), nuestro navegador nos dará la opción a almacenar esa nueva clave de una autoridad de certificación desconocida o la de cancelar la conexión por el riesgo implícito de confiar en una autoridad cuestionable.

¿Qué ocurre con todos esos servidores de web seguro en los que mucha gente ha insertado sus propios certificados? Muchos de esos certificados son auto-firmados, o firmados por autoridades de certificación particulares, que han sido creadas con herramientas como OpenSSL u OpenCA.

Cuando el cliente remoto se intente conectar a un servidor SSL de estas características, y decida almacenar la clave de esa autoridad de certificación desconocida, podemos concluir que se ha cumplido el principio de intercambio. Si investigamos las bases de datos de certificados de los navegadores, podremos encontrar evidencias del intercambio.

Archivos binarios únicos compilados

Este tipo de ejemplo sería paralelo a mediante investigación balística, determinar que un arma del crimen corresponde con unas balas localizadas en una escena.

Conocemos el MD5, SHA1 y otros algoritmos de "hashings" que nos dan un resultado único y distinto para cada entrada de datos diferente, y sabemos que podemos utilizarlos para verificar la integridad de ficheros de todo tipo.

Pero lo que pocas veces nos planteamos es utilizar este tipo de firmas contra nuestro presunto atacante, que ha podido dejar dentro del sistema operativo atacado, herramientas desarrolladas y compiladas por él, dentro de su propia máquina.

Imaginemos un Caballo de Troya o un Gusano (Word) que se distribuyan en formato binario. Imaginemos también, que han sido compilados mediante el compilador gratuito "gcc", ¿puede ser posible que cada binario creado en un sistema tenga una firma determinada de ese sistema?

Para hacer una verificación rápida, hemos creado el fichero "hola.c", con el siguiente contenido:

```
#include <stdio.h>

int main(void){
    printf("Hola mundo\n");
    return 0;
}
```

El clásico "Hola mundo", que vamos a compilar con la misma versión de "gcc" en dos sistema diferentes.

En nuestro primer sistema, un Redhat, obtenemos el siguiente fichero:

```
-rwxrwxr-x 1 usuario usuario 11361 may 18 21:56 hola
MD5 de hola: 21ca7fa63d21cc616d3ccbdefe8fc111
SHA1 de hola: a09085289250308efd4e1d07b9008c175d30ae8c
```

En el segundo sistema, un Debian, obtenemos el siguiente fichero:

```
-rwxrwxr-x 1 usuario usuario 11256 2004-05-18 21:56 hola
MD5 de hola: ada22274e5fc96ffe7cac22a735c548b
SHA1 de hola: fb19f2431a6de2fbb47acc614618e69af99364a
```

Es evidente que hay diversos factores que pueden llevar a obtener binarios diferentes, entre ellos librerías diferentes, diversas formas de enlazado de los objetos, cambios en las configuraciones de optimización... pero el hecho es que en dos sistema diferentes hemos obtenido diferentes binarios, con diferentes firmas MD5 y SHA1.

Puede que no podamos elaborar una tesis concluyente sobre la procedencia de un archivo binario, pero como mínimo obtendremos indicios que nos permitirán descartar o localizar sospechosos.

Imaginemos que nuestro presunto atacante ha dejado un "rootkit" instalado en el sistema que estamos analizando, y ha dejado, no solo los ficheros binarios sino que ha dejado también el código fuente.

Si pudiéramos acceder al sistema de nuestro presunto atacante y compilar de nuevo esos binarios en ese sistema, ¿no obtendríamos binarios exactamente iguales a los que ha instalado en nuestra máquina?

Archivos de aplicaciones propietarias

¿Qué ocurre con aplicaciones como Microsoft Word, Excel, Powerpoint, o con aplicaciones como Acrobat Distiller, o con OpenOffice? Estas aplicaciones suelen utilizar formatos binarios para los documentos que emiten, y en muchas ocasiones introducen dentro de estos documentos firmas que identifican de una forma bastante obvia al creador, e incluso el sistema donde se creó el archivo.

En muchos casos incluso podemos extraer direcciones IP de este tipo de documentos, o más lejos todavía, datos del número de registro exacto que ha generado un documento determinado.

Incluso aplicaciones como Windows Media, o el formato de música Attrack, pueden registrar mediante ciertas firmas detalles sobre el equipo donde se crearon ficheros de sonido (en un caso encontramos un presunto atacante simpático, que había insertado un sonido creado por el mismo en el arranque del sistema de su víctima).

¿Y en ficheros de imagen de aplicaciones como Adobe Photoshop o JASC Paintshop Pro?

Conclusiones

No hemos querido hacer un inventario exhaustivo de evidencias informáticas y sus interacciones con los atacantes; más bien nuestra idea era conseguir que los investigadores forenses le presten más atención al intercambio de información entre los servidores y los atacantes.

El Principio de Intercambio tiene plena aplicación en las evidencias digitales, simplemente debemos determinar como y donde vamos a encontrar esas evidencias que nos hagan llegar a concluir que la persona que insertó un binario y la que hemos detenido son la misma.

No hemos hablado de las firmas en la impresión (cabezales dañados, impurezas en las tintas, tipos de papel...) que pueden llevar a concluir en la caza de la impresora (y su dueño) que dio salida a un documento determinado. La ciencia forense tradicional ya investiga de forma exhaustiva todos los detalles en sus unidades de investigación documental, siendo capaces de rastrear el origen de un documento de forma clara.

Y recordemos: *"Cada contacto deja un rastro"*.