



**DEPARTAMENTO
DE COMPUTACION**

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico

LACHIWA - HONEYTOKENS

Seguridad de la Información

Integrante	LU	Correo electrónico
Cielo Serena Roccella	122/21	cielorocella@gmail.com
Florencia Rosenzuaig	118/21	f.rosenzuaig@gmail.com
Al Kamber	x/x	al.kamber@mail.polimi.it



Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2160 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (54 11) 4576-3359

<http://www.fcen.uba.ar>

1. Introducción

Los honeytokens son datos o recursos especialmente diseñados que se implantan en bases de datos y sistemas con el propósito de detectar accesos no autorizados. Estos elementos, al ser manipulados por un atacante, envían una alerta al propietario del recurso. Dicha alerta incluye información valiosa obtenida del atacante, como su dirección IP y detalles del recurso comprometido, lo cual es esencial para rastrear el origen del ataque y tomar las medidas necesarias para mitigar el riesgo.

Estos recursos trampa son una herramienta crucial en la seguridad informática, ya que permiten la detección temprana de actividades sospechosas que podrían comprometer la integridad del sistema. Al identificar rápidamente un uso indebido, los propietarios de los recursos pueden implementar contramedidas efectivas para proteger tanto el sistema como la seguridad de los datos.

Los honeytokens emplean un mecanismo conocido como 'call home', mediante el cual notifican de un acceso indebido a un servidor central. Este mecanismo varía según el tipo de token creado, pudiendo incluir inyecciones de código o el uso de macros, adaptándose así a diferentes formatos y escenarios de uso.

En este trabajo práctico, hemos desarrollado una aplicación de línea de comandos llamada Lachiwa, la cual tiene la capacidad de generar honeytokens para diferentes tipos de archivos. Además, hemos implementado un servidor central al cual los honeytokens se conectan para reportar accesos no autorizados, desde donde se envía la correspondiente alerta por correo electrónico al propietario del recurso. Este enfoque no solo fortalece la seguridad del sistema, sino que también proporciona una herramienta efectiva para la vigilancia y protección de los datos.

2. Funcionamiento

La herramienta Lachiwa nos permite generar tokens que pueden ser **url**, **qr**, **pdf**, **ini** o **exe**.

Cuando se crea el token, se especifica a qué mail notificar junto con un mensaje. Además, se puede especificar a qué dirección redirigir luego de comunicarse con el servidor. Esta información es guardada en una URL que está insertada en el token.

Para contener la información que necesitamos enviar en la URL, generamos un mensaje el cual contiene nuestros datos (mail, nota y dirección de redireccionamiento) separados por la una secuencia de caracteres ('@@@') la cual permiten al server separar los datos requeridos. Una vez creado este mensaje, lo encriptamos con la clave publica del servidor, de forma que los datos no sean visibles por el atacante al activar el honeypot.

```
enc_data = encrypt(mail + @@@ + note + @@@ + redirect, public_key)
```

```
url = https://<servidor>/enc_data
```

Cuando alguien activa el honeypot, se abre esta URL. Entonces el servidor desencripta el pedido con la clave privada, y manda un mail con el mensaje correspondiente. De ser especificada una dirección de redireccionamiento, se va a redirigir allí.

```
dec_data = decrypt(enc_data, private_key)
```

```
mail, note, redirect = dec_data.split(@@@)
```

3. La herramienta: lachiwa.py

La aplicación lachiwa.py responde por linea de comando y debe ser llamada con las siguiente información: el tipo de archivo que debe ser el honeypot, un mail a donde se requiere que llegue la alerta y una nota que se enviara en el mail de alerta. Esta nota contiene la información que el usuario desee para poder identificar el honeypot. Por ejemplo, si una empresa quiere generar honeypots para cada computadora y detectar cual fue atacada, la nota debería incluir información de a quien pertenece el dispositivo.

```
python lachiwa.py {pdf,url,qr,exe, ini} mail nota
```

Como opcional, a los parámetros anteriores se les puede agregar -name seguido del nombre deseado para asignar al honeypot, y -redirect seguido de una URL la cual se desea que redirija el honeypot al ser abierto. Si no se definen estos parámetros el nombre default es token y no redirige a ningún otra URL.

```
python lachiwa.py [-name NAME] [-redirect REDIRECT] {pdf,url,qr,exe,ini} mail note
```

Luego, lachiwa.py se encarga llamar al generador de token correspondiente y le envía los datos necesarios para crear el token.

El flag -h o --help muestra un mensaje de ayuda que enumera todos los argumentos posibles que se pueden pasar a la aplicación, útil para los nuevos usuarios o para aquellos que necesitan un recordatorio rápido de las opciones disponibles.

Al ejecutar el comando `python lachiwa.py -h` se obtendrá la siguiente descripción de los parámetros:

```
usage: lachiwa.py [-h] [-name NAME] [-redirect REDIRECT] {pdf,url,qr,exe,ini}
mail note
```

[illegible]

```
positional arguments:
  {pdf,url,qrcode,ini}  Specify the type of honey token to generate. Choices:
                        'pdf', 'url', 'qrcode', 'exe', 'ini'.
  mail                  The email address where the notification will be sent
                        when the token is accessed.
  note                  A note to include in the notification email, i.e. a
                        description of the type of token triggered

options:
  -h, --help            show this help message and exit
  -name NAME            Optional: specify the file name for the token. Default
                        is 'token'.
  -redirect REDIRECT    Optional: specify a URL to redirect to when the token
                        is accessed. Default is an empty string, which
                        redirects to a dummy page.
```

Figura 1: Mensaje de ayuda que se obtiene al activar la opción -h

4. El servidor: server.py

Nuestro servidor va a levantarse en la IP **0.0.0.0** con el puerto **8080**. El servidor va a recibir pedidos del tipo:

```
url = https://<servidor>/enc_data
```

o equivalentemente:

```
url = https://<servidor>/acá/puede/ir/cualquier/cosa/enc_data
```

Donde la URL puede contener o no información de un token. El servidor va a intentar descriptarla usando la clave privada:

```
dec_data = decrypt(enc_data, private_key)
mail, note, redirect = dec_data.split('@@@')
```

Si se logra descryptar `enc_data`, entonces conseguimos `mail`, `nota` y `redirect`. En ese caso, el servidor manda una notificación al mail obtenido con la nota obtenida. Además, informa desde qué IP fue accedida la URL.

Si se especificó dirección de redireccionamiento (`redirect != null`), el servidor redirige al atacante a ella. En caso contrario, por motivos de test, el servidor muestra `mail`, `nota` e `IP`.

Para enviar el mail, utilizamos la librería `smtplib` y la plataforma Mailtrap, que nos otorga un servidor SMTP, usuario, contraseña y un inbox para testear el código.

Si `enc_data` no se logra desencriptar, el servidor muestra un mensaje de error.

Algorithm 1 Definición del método GET, el cual responde a los pedidos del servidor

```
1: function GET(path, IP_address)
2:   enc_data ← la última parte de path
3:   if ¿Puedo desencriptar enc_data con la clave privada? then
4:     mail + '@@@" + note + '@@@" + redirect ← decrypt(enc_data, private_key)
5:     if redirect != null then
6:       Me redirijo a redirect
7:     else
8:       print_on_web_page(You are connecting from '{ip}')
9:       print_on_web_page(We'll send a mail to '{mail}')
10:      print_on_web_page(saying '{note}')
11:    end if
12:    send_mail(mail, note, 'Alguien accedió a tu Honey Token desde la IP {IP_address}')
13:  else
14:    return Error: token inválido
15:  end if
16: end function
```

4.1. Sobre la dirección del servidor

Para testear el servidor y la aplicación, el servidor corre en la IP 0.0.0.0. Lo hicimos de esta forma para que sea sencillo testear lachiwa desde la misma computadora que corre el servidor. Para acceder desde un dispositivo en la red local, basta con acceder con la IP privada de la computadora que corre el servidor (es decir, en vez de acceder a `http://0.0.0.0:8080`, acceder a `http://<ip-privada>:8080`).

Intentamos hacer que el servidor sea accesible desde redes externas (es decir, desde dispositivos fuera de la red local), pero no pudimos implementarlo porque por distintos motivos ninguno de nosotros tenía acceso a la configuración de su respectivo router.

5. Criptografía

Dado que la información conteniendo al mail del usuario y la nota con detalles sobre el token se envía al servidor cuando se interactúa con el token, debemos asegurarnos de que la misma este protegida. Para ello utilizamos criptografía asimétrica RSA para crear una clave pública y una clave privada para el server. De esta forma, la herramienta lachiwa contiene la clave pública del servidor con la cual se encriptarán los mensajes que se usan de extensión en la URL, y el servidor almacena su clave privada que usa para desencriptar estos mensajes. En nuestro proyecto guardamos ambas claves en el mismo directorio para facilitar el testeo, pero en un caso más realista, la aplicación no va a ser distribuida con la clave privada, solo con la clave pública.

La información enviada en la URL puede ser vista por el atacante, por esto la ventaja de utilizar criptografía asimétrica es que nos aseguramos de que la información solo pueda ser desencriptada por el server. De esta forma, la única forma de acceder a dicha información es teniendo la clave privada del server, lo cual implicaría primero haber atacado al server. Por otro lado, al no guardar información de los usuarios en el server, si este es atacado, el atacante no tendría acceso directo a todos los datos sino que debería conocer también a las URLs de cada honeypot generado.

6. Sanitización de entrada

Para evitar errores en el servidor, se chequea que la url provista para redireccionamiento sea una url válida. Además, dado que la información encriptada se encuentra separada por "@@@", no está permitido que mail, nota y redirect contengan "@@@".

Finalmente, por motivos de test si un token válido es recibido y no se especificó dirección de redireccionamiento, entonces en la página web se imprime la información recibida descriptada. A esta página la llamamos *dummy page*.

**You are connecting from 1.2.3.4
We'll send a mail to user@mail.com
saying "Potential attack registered from Honey Token access"**

Figura 2: Ejemplo de dummy page

Dado que tanto el mail como la nota son dados como entrada por el creador del token, y la dummy page está escrita en HTML, entonces cabe la posibilidad de que el usuario ingrese como mail o nota:

```
<script>alert("Hola manola");</script>
```

Resultando en que la página ejecute el código provisto (como muestra la Figura 3).

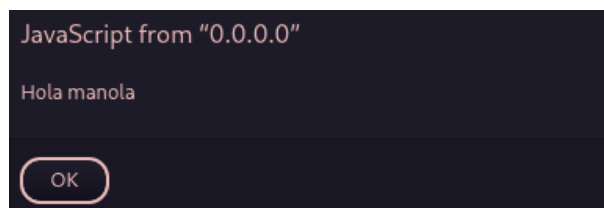


Figura 3: Resultado de ingresar el script de ejemplo sin sanitizar.

Para evitar que el usuario logre ejecutar código arbitrario, usamos la librería `bleach` y un regex para sanitizar la entrada. La librería `bleach` reemplaza caracteres especiales, de forma tal que la entrada pierde su posible sintaxis HTML. Luego, el regex por si acaso elimina substrings del tipo `<script>*</script>`. De esta forma, aunque se ingrese un script HTML como entrada, no tendrá efecto alguno.

7. Tokens

Finalmente, vamos a ver el funcionamiento de cada token, pero antes de explicar cada uno es importante ver la función que todos tienen en común: `url_creator`. Esta función se encarga de tomar los datos provistos por el usuario y crear el url que luego será inyectado en el token. El funcionamiento es el siguiente:

Algorithm 2 Crea URL que contiene la información encriptada.

```
1: function URL_CREATOR(mail, note, name, redirect)
2:   message ← mail + '@@' + note + '@@' + redirect
3:   enc_data ← encrypt(message, public_key)
4:   if name == null then
5:     return http://0.0.0.0:8080/ + enc_data
6:   else
7:     return http://0.0.0.0:8080/ + name + / + enc_data
8:   end if
9: end function
```

7.1. URL

Para crear el url token simplemente llamamos a `url_creator` y le devolvemos el mismo al usuario.

Por ejemplo, al correr el siguiente comando:

Algorithm 3 Crear un URL Token.

```
1: function GENERATE_URL_TOKEN(mail, note, name, redirect)
2:   url ← create_url(mail, note, name, redirect)
3: end function
```

```
python lachiwa.py url user@mail.com "Potential attack registered from URL access" -name
"site_urlredirect "https://youtube.com"
```

El programa devuelve:

URL token generated: http://0.0.0.0:8080/site_url/...

A notification containing "Potential attack registered from URL access" will arrive to "user@mail.com" when the URL is opened, and then it will redirect to "https://youtube.com"

You can specify where to redirect with -r [valid url](#).

7.2. QR

La idea detrás del token de código QR es muy simple, primero llamamos a nuestra función url-creator, que genera nuestra URL incluyendo los datos encriptados para que el servidor los descripte. Finalmente, codificamos dicha URL como un código QR con la librería `qrcode`. Este código QR se guarda en el directorio desde el que se ejecuta el programa, y puede colocarse donde el usuario desee.

Algorithm 4 Crear un QR Token.

```
1: function GENERATE_QR_TOKEN(mail, note, name, redirect)
2:   url ← create_url(mail, note, name, redirect)
3:   img ← crearQR(url)
4:   img.save('{name}.png')
5: end function
```

En nuestra implementación, al ejecutar el siguiente código:

```
python lachiwa.py qr "user@mail.com" "Potential attack registered from QR code scan"
-name "site_qr" -redirect "https://youtube.com"
```

El programa devuelve:

QR token generated: [site_qr.png](#)

A notification containing "Potential attack registered from QR code scan" will arrive to "user@mail.com" when the QR is opened, and then it will redirect to "https://youtube.com".

Y se generará un código QR como el que se muestra en la figura 4.



Figura 4: Ejemplo de código QR generado

Este código QR ofrece una gran versatilidad al usuario que lo ha creado y puede utilizarse en numerosos contextos. Por ejemplo, puede colocarse en los puntos de entrada a zonas seguras, como salas de servidores, centros de datos o espacios de oficina restringidos. El personal autorizado escanea el código QR para acceder. El escaneo activa una alerta al equipo de seguridad, registrando la entrada y proporcionando supervisión en tiempo real del acceso a zonas sensibles.

Otro caso de uso del código QR podría ser colocarlo en los puestos designados para informar de incidentes o cerca de equipos de emergencia como extintores y DEA. En caso de incidente, el escaneo del código QR puede alertar inmediatamente al equipo de seguridad, facilitando la ubicación y naturaleza del incidente para una respuesta más rápida.

Estos ejemplos demuestran lo versátil que puede ser el código QR y la ventaja adicional que supone que sea un soporte físico y no un sistema de detección que sólo existe en el software.

7.3. PDF

Para crear el Token PDF, primero creamos un archivo pdf en blanco con la librería `fpdf`. Luego, utilizamos `pypdf` para inyectar código en javascript que nos permite abrir la url obtenida con `url_creator`. Luego, una vez que el Token PDF es abierto con *Adobe Acrobat*, se va a notificar que el archivo intenta abrir el enlace insertado.

Lamentablemente, abrir o no el enlace va a ser decisión del usuario, ya que en las últimas versiones de *Adobe Acrobat* ya no hay forma de hacerlo automáticamente.

Algorithm 5 Crear un PDF Token.

```
1: function GENERATE_PDF_TOKEN(mail, note, name, redirect)
2:   pdf ← archivo pdf en blanco
3:   url ← create_url(mail, note, name, redirect)
4:   pdf.agregar_javascript("app.launchURL('{url}', true);")
5:   pdf.save("{name}.pdf")
6: end function
```

Al ejecutar el siguiente código:

```
python lachiwa.py pdf "user@mail.com" "Potential attack registered from opening a PDF file"
-name "password list" -redirect "https://youtube.com"
```

El programa devuelve:

PDF token generated: `password list.pdf`

A notification containing "`Potential attack registered from opening a PDF file`" will arrive to "`user@mail.com`" when the token is opened with Adobe Acrobat, and then it will redirect to "`https://youtube.com`".

Y se generará un archivo pdf con el nombre especificado.

Dichos archivos PDF pueden crearse y colocarse en numerosos lugares con nombres que podrían interesar a posibles atacantes, atrayéndolos así y activando la alerta al servidor una vez que se accede a ellos. Por ejemplo, un PDF de este tipo podría colocarse en depósitos de documentos confidenciales con un nombre atractivo como "Lista de contraseñas".

7.4. INI

La idea de un token de archivo INI, a menudo llamado `token desktop.ini`, es aprovechar el comportamiento de los sistemas operativos Windows para detectar accesos no autorizados a un directorio.

El archivo desktop.ini es un archivo de sistema utilizado por Windows para personalizar la apariencia de un directorio. Al colocar un archivo desktop.ini especialmente diseñado en una carpeta, se puede crear un honeypot que active una alerta cuando se acceda a la carpeta.

Algorithm 6 Crea un INI Token.

```
1: function GENERATE_INI_TOKEN(mail, note, name, redirect)
2:   url ← create_url(mail, note, name, redirect)
3:   desktop_ini_content ← f"\"[.ShellClassInfo] IconResource=url,0\""
4:   ini_file.write(desktop_ini_content)
5: end function
```

En nuestra implementación, al ejecutar el siguiente código:

```
python lachiwa.py ini "user@mail" "Potential attack registered from ini file accessed"
-name "initoken"
```

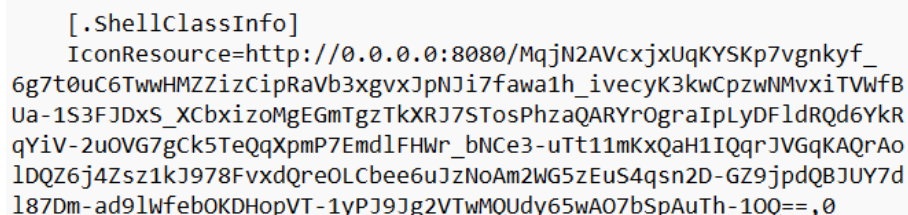
El programa devuelve:

INI token generated: `path to directory\initoken\desktop.ini`

A notification containing 'Potential attack registered from ini file accessed' will arrive to 'user@mail.com' when the folder containing the token is opened, and then it will redirect to a dummy page. You can specify where to redirect with `-r <valid url>`.

y se va a crear un archivo desktop.ini en la carpeta initoken del directorio desde el que se está llamando a lachiwa.py.

Este archivo desktop.ini contiene, por ejemplo, el código que se muestra en la siguiente imagen:



```
[.ShellClassInfo]
IconResource=http://0.0.0.0:8080/MqjN2AVcxjxUqKYSkp7vgknyf_
6g7t0uC6TwwHMZZizCipRaVb3xgvxJpNji7fawa1h_ivecyK3kwCpzwNMvxiTVWfB
Ua-1S3FJDxS_XCbxiZoMgEGmTgzTkXRJ7STosPhzaQARYrOgraIplYDFldRQd6YkR
qYiV-2uOVG7gCk5TeQqXpmp7EmdlFHW_r_bNce3-uTt11mKxQaH1IQqrJVGqKAQrAo
1DQZ6j4Zsz1kJ978FvxdQreOLCbee6uJzNoAm2WG5zEuS4qsn2D-GZ9jpdQBjUY7d
187Dm-ad9lwfebOKDHopVT-1yP9Jg2VTwMQUdy65wA07bSpAuTh-10Q==,0
```

Figura 5: El código contenido en el archivo desktop.ini creado

Cuando un usuario acceda a la carpeta que contiene este archivo desktop.ini, Windows intentará recuperar el icono de la URL especificada, activando así la llamada a la URL y llamando al servidor, que descifrará el contenido de la URL y enviará la alerta correspondiente al correo electrónico especificado, con la nota que se haya establecido.

La incrustación de un honeypot en un archivo desktop.ini proporciona varias ventajas de seguridad.

En primer lugar, dado que se trata de un archivo de sistema bastante discreto, es menos probable que los actores maliciosos se percaten de su presencia. Esta colocación sigilosa aumenta las posibilidades de detectar accesos no autorizados sin alertar al intruso.

También es rentable y tiene un impacto mínimo en el rendimiento del sistema, por lo que es una medida eficaz para detectar accesos no autorizados.

7.5. EXE

Nuevamente, creamos la url correspondiente y creamos un script de python que abra la misma url. Luego, guardamos el script en un archivo y usando *PyInstaller* creamos un ejecutable a partir del mismo script. Para evitar crear archivos de más, trabajamos en una carpeta temporal y una vez creado el ejecutable lo movemos hacia nuestra carpeta y eliminamos la temporal.

Algorithm 7 Crear un EXE Token.

```
1: function GENERATE_EXE_TOKEN(mail, note, name, redirect)
2:   url ← create_url(mail, note, name, redirect)
3:   script ← "Abrir {url}"
4:   script.save("file.py", /tmp/lachiwa)
5:   exe ← Genero ejecutable de file.py
6:   Muevo exe de /tmp/lachiwa al directorio actual
7:   Borro /tmp/lachiwa
8: end function
```

En nuestra implementación, al ejecutar el siguiente código:

```
python lachiwa.py exe "user@mail" "Potential attack registered from exe file accessed"
-name "exetoken"
```

El programa devuelve:

EXE token generated: `path to directory\exetoken\`

A notification containing 'Potential attack registered from exe file accessed' will arrive to 'user@mail.com' when the token is executed, and then it will redirect to a dummy page. You can specify where to redirect with `-r <valid url>`.

7.6. Documentos Office

Si bien no realizamos la implementación de documentos excel o word de Microsoft Office como honeytokens debido a dificultades para testear los mismos, investigamos como se implementaría el mecanismo callhome. Los documentos Microsoft Office tienen la posibilidad de usar macros en VBA (Visual Basic for Applications) con los cuales se puede ejecutar código cuando se realiza una determinada acción, como puede ser abrir el archivo. Luego, el código a ejecutar sería un request con la URL creada específicamente para el honeytoken.

8. Cómo probarlo

Para levantar el servidor de lachiwa en Docker, se debe ejecutar el siguiente comando en /lachiwa:

```
docker build -t lachiwa .
docker run -p 8080:8080 lachiwa:latest
```

Luego, desde otra terminal, se puede testear la aplicación en si:

```
docker exec -it <container-id> python lachiwa.py {pdf,url,qr,exe} mail note [-h] [-name NAME] [-redirect REDIRECT]
```

De ser necesario, es posible copiar los archivos dentro del contenedor en la PC del host (Por ejemplo, para abrir el token pdf con Adobe Acrobat)

```
docker cp <container-id>:lachiwa/file /home/user/
```

9. Trabajo a futuro

Para distribuir la herramienta, se podría modificar para que la información del servidor y de la herramienta lachiwa estén separadas (por ejemplo, en repositorios distintos), para simplificar su uso a futuros usuarios. Además es importante almacenar la clave privada del servidor en un directorio seguro, de forma que un atacante no la pueda acceder fácilmente.

Finalmente, nos gustaría modificar el servidor para que sea accesible desde fuera de la red local. Para esto se debería modificar la IP de la URL para que se dirija específicamente a la dirección donde este planeado levantar el servidor.

10. Conclusión

En este trabajo, desarrollamos una herramienta para generar honeytokens, junto con un servidor el cual se encarga de emitir las alertas cuando se activa un token. Nuestra implementación crea una URL para cada token, que incluye datos del usuario encriptados. Esta URL realiza una solicitud a nuestro servidor cuando se accede, lo que desencadena el envío de una alerta.

La generación de un nuevo honeytoken implica encontrar una manera efectiva de abrir dicha URL. Esto puede lograrse mediante la inserción de un script que abra la URL automáticamente o redirigiendo al usuario hacia la misma. En este trabajo, logramos implementar honeytokens funcionales para **URL**, **QR**, **PDF**, **EXE** e **INI**.

El uso de varios tipos de honeytokens ha demostrado que es posible crear trampas efectivas en una variedad de formatos de archivos y medios, lo que aumenta la probabilidad de detectar accesos no autorizados en una variedad de contextos y aplicaciones.

El uso de URLs encriptadas garantiza que los datos que se transmiten entre los honeytokens y el servidor central permanezcan confidenciales y protegidos contra la interferencia. Una parte importante de la integridad y seguridad del sistema de detección es la capacidad del servidor para descryptar estas URLs y procesar la información correctamente.

El diseño del sistema permite la fácil expansión y adaptación a nuevas formas de honeytokens o métodos de ataque emergentes. Esto garantiza que el sistema pueda evolucionar y seguir siendo relevante en un panorama de amenazas en constante cambio.