CAI 2. CONSULTA SOBRE LA INTEGRIDAD DE LAS TRANSMISIONES DE UNA ENTIDAD BANCARIA CON APP

Escuela Técnica Superior de
Ingeniería Informática

Índice

Alcance y Objetivos de la Consultoría

La importancia de la seguridad para las empresas bancarias y sus usuarios es algo primordial para mantener la confianza en el sector, y se debe llevar más allá de los sistemas finales, cliente y servidor, tan importante es la seguridad en ellos como la forma en la que se comunican.

En esta comunicación se debe salvaguardar la integridad y la confidencialidad de la información para evitar que un tercero intercepte los mensajes y pueda acceder al contenido de los mismos, pudiendo suplantar o perjudicar nuestra identidad. Por ello, esta consultoría tiene como objetivo conseguir la integridad mencionada y proporcionar mecanismos y estrategias para evitar su vulneración.

Estudio Inicial

Como primera instancia de esta auditoría, debemos tratar la información que nos ha proporcionado la empresa.

En primer lugar, se hace uso de una aplicación móvil, donde se hace uso de códigos de autenticación de mensajes, MAC, con clave secreta de32 bits para el cifrado de mensajes.

En segundo lugar, para el intercambio de claves secretas, se entrega un dispositivo de almacenamiento al cliente con la clave de un año de validez.

A pesar de las medidas usadas, el Equipo de Gobierno de la Seguridad de la Información duda de la robustez de los protocolos, por tanto se procederá a hacer un análisis más exhaustivo de los métodos de generación de MAC, así como del tamaño de las claves.

Recursos para la Auditoría

Recursos Humanos

El equipo de seguridad para esta asignatura se compone de los siguientes miembros:

- Barragán Candel, Marina - estudiante de Ing. Informática – Tecnología Informática, en la mención de Tecnologías de la información
- Calcedo Vázquez, Ignacio - estudiante de Ing. Informática – Tecnología Informática, en la mención de Sistemas de Información
- Polo Domínguez, Jorge - estudiante de Ing. Informática – Ingeniería de Computadores
- Sala Mascort, Jaime Emilio - estudiante de Ing. Informática – Tecnología Informática, en la mención de Computación

Recursos Tecnológicos

El equipo dispone de los siguientes dispositivos para realizar la auditoría:

- Ordenador 1, las características de este ordenador son:
  - Intel® Core™ i5-3570K
  - GTX 660 Ti OC 2GB GDDR5
  - RAM 8GB DDR3
  - SSD 500GB Samsung 860 EVO
- HP Pavilion x360,
  - Intel Core i5 8250U
  - Nvidia GeForce 940MX

Escuela Técnica Superior de Ingeniería Informática

- 12 GB DDR4
- HDD 1TB

- ASUSPRO P2520LA,
  - Intel Core i3-4005U
  - RAM 4GB DDR3L SODIMM
  - HDD 500GB
- HUAWEI MateBook D 14 AMD,
  - AMD Ryzen 5 3500U
  - Radeon™ Vega 8 Graphics
  - 8 GB DDR4
  - 512 GB PCIe SSD

Actividades de la Consultoría

Las distintas actividades que se realizaran en esta auditoría serán las siguientes:

Análisis de Generación de MAC y Tamaño de Claves Secreta

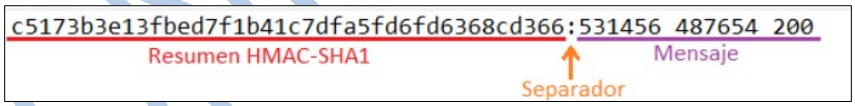
A continuación, se procederá a demostrar la robustez de los medios de generación MAC, así como también se probará la seguridad de las claves secretas.

Para este estudio, se hará uso de una herramienta de libre acceso llamada Hashcat, la cual mediante comandos en un terminal del sistema, es capaz de descifrar contraseñas dependiendo de los parámetros y ficheros que le indiquemos automáticamente.

Gracias a la tabla que nos ha proporcionado la entidad bancaria donde se nos muestran mensajes y MAC generados por distintos, vamos a proceder con el primer mensaje

Primer Mensaje, 531456 487654 200

Creamos un documento de texto (.txt) y escribimos el mac y el mensaje del primer cliente, en el siguiente formato:



Es necesario que sea así, para que la herramienta detecte cada parámetro, a raíz del algoritmo de codificación que creemos que usa el resumen. A continuación, abrimos un terminal del sistema y escribimos el siguiente comando:

```
hashcat.exe -m 150 -a 3 [DIRECTORIO DEL ARCHIVO]  ?b?b?b?b
```

Siendo -m 150 el algoritmo de generación, HMAC-SHA1, -a 3 el tipo de ataque, fuerza bruta, y por último ?b?b?b?b, siendo esto el patrón de la clave privada.

Tras definir todos los parámetros, lanzamos la herramienta para ver si es posible averiguar las claves en poco tiempo. Obtenemos lo siguiente como resultado:

```
Host memory required for this attack: 204 MB

c5173b3e13fbed7f1b41c7dfa5fd6fd6368cd366:531456 487654 200: a6A

Session..........: hashcat
Status...........: Cracked
Hash.Name........: HMAC-SHA1 (key = $pass)
Hash.Target......: c5173b3e13fbed7f1b41c7dfa5fd6fd6368cd366:531456 487654 200
Time.Started.....: Mon Nov 02 02:02:09 2020 (8 secs)
Time.Estimated...: Mon Nov 02 02:02:17 2020 (0 secs)
Guess.Mask.......: ?b?b?b?b [4]
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 83095.2 kH/s (11.03ms) @ Accel:16 Loops:128 Thr:64 Vec:1
Recovered........: 1/1 (100.00%) Digests
Progress.........: 692060160/4294967296 (16.11%)
Rejected.........: 0/692060160 (0.00%)
Restore.Point....: 2695168/16777216 (16.06%)
Restore.Sub.#1...: Salt:0 Amplifier:128-256 Iteration:0-128
Candidates.#1....: $HEX[cd61394d] -> $HEX[ffff3f3d]
Hardware.Mon.#1..: Util:65536% Core:1200MHz Mem:1200MHz Bus:16

Started: Mon Nov 02 02:02:07 2020
Stopped: Mon Nov 02 02:02:18 2020
```

Como podemos observar, hemos conseguido averiguar la clave en muy poco tiempo, tan solo 8 segundos. El estado de la clave es Cracked y su valor es " a6A".

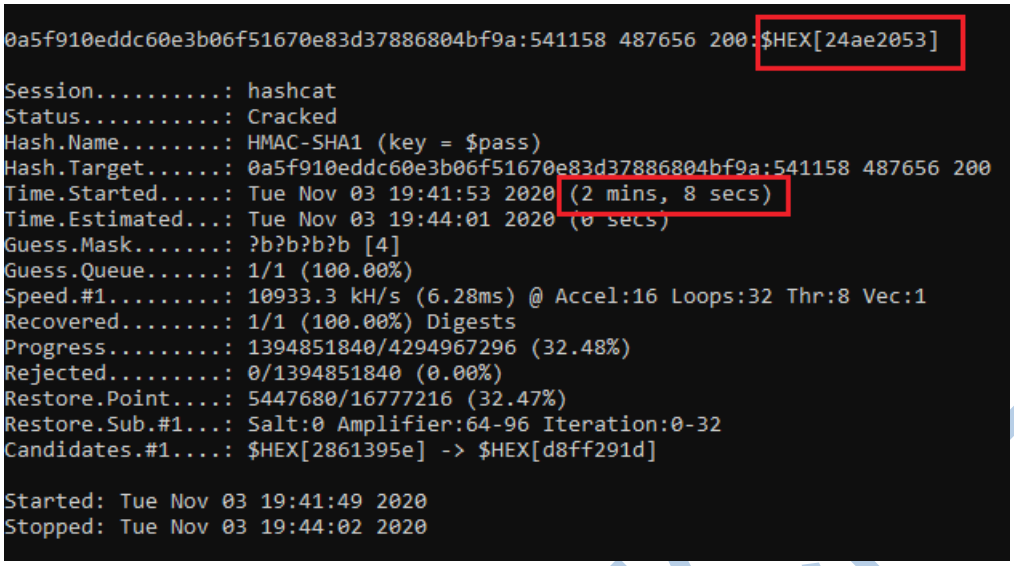
Segundo Mensaje, 541157 487655 200

Realizando el mismo procedimiento, se obtienen los siguientes resultados:

```
Host memory required for this attack: 204 MB

158413dd62eada5273a72f9fa35f4e19ddb864b8:541157 487655 200:$HEX[21ae2d41]

Session..........: hashcat
Status...........: Cracked
Hash.Name........: HMAC-SHA1 (key = $pass)
Hash.Target......: 158413dd62eada5273a72f9fa35f4e19ddb864b8:541157 487655 200
Time.Started.....: Mon Nov 02 02:29:29 2020 (7 secs)
Time.Estimated...: Mon Nov 02 02:29:36 2020 (0 secs)
Guess.Mask.......: ?b?b?b?b [4]
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 83287.9 kH/s (11.02ms) @ Accel:16 Loops:128 Thr:64 Vec:1
Recovered........: 1/1 (100.00%) Digests
Progress.........: 623902720/4294967296 (14.53%)
Rejected.........: 0/623902720 (0.00%)
Restore.Point....: 2433024/16777216 (14.50%)
Restore.Sub.#1...: Salt:0 Amplifier:0-128 Iteration:0-128
Candidates.#1....: $HEX[7361392e] -> $HEX[d2ff3f37]
Hardware.Mon.#1..: Util:65536% Core: 200MHz Mem:1200MHz Bus:16

Started: Mon Nov 02 02:29:27 2020
Stopped: Mon Nov 02 02:29:37 2020
```

Observamos que de nuevo, se ha podido descifrar la clave de otro cliente. En este caso ha tardado 7 segundos, la clave está Cracked y su valor se obtiene en hexadecimal, con un valor de $21ae2d41. Al convertirla de hexadecimal a ASCII, tenemos "!®-A" como clave.

Volvemos a repetir el mismo procedimiento, obteniendo esta vez os siguientes resultados:



Como las otras dos claves, hemos podido obtener el valor de la tercera. Se ha realizado la prueba en una máquina menos potente, para así poder comprobar si existe un cambio sustancial, pero no es el caso en términos de seguridad, ya que la clave se ha obtenido tras 2 minutos y 8 segundos de ejecución, pasando al estado de Cracked y su valor en hexadecimal es de $24ae2053. Mediante conversión conseguimos "$® S" como clave.

Valoración de la Generación de MAC y Tamaño de Claves Secreta

Tras las pruebas realizadas, las tres claves suministradas por la entidad bancaria para asegurar la integridad de las transmisiones, son débiles y fácilmente descifrables. Su tamaño no es el indicado para la seguridad de transacciones bancarias y por supuesto, el tiempo necesario para averiguarlas es muy preocupante.

Dicho esto, vamos a mostrar una serie de cálculos y pruebas para que la entidad bancaria pueda elegir de manera correcta, la longitud de clave suficiente para proteger las transacciones durante un año completo, además de cambiar de algoritmo de cifrado.

Para estos cálculos, vamos a tener en cuenta que el atacante posea una alta potencia de computación para realizar operaciones, asumiendo que tiene en su poder una tarjeta gráfica acorde para tal cometido.

Hemos supuesto que el atacante posee la gráfica más potente del mercado actualmente, GeForce RTX 3090 | Nvidia, cuya potencia de cómputo asciende a los 705.5 MH/s(M Hashes/segundos) para hashes con algoritmo HmacSHA512 según un benchmark no oficial, sin embargo, podemos asegurar que son certeros ya que muestra una mejora del 35% respecto a la generación anterior, coincidiendo con el 40% anunciado por la compañía propietaria en el caso óptimo.

A continuación aparece el tiempo necesario para obtener las claves de los clientes, con distinto tamaño, mediante desencriptado por fuerza bruta, es decir, probando todas las combinaciones posibles.

Tamaño en bits	NºCombinaciones	Tiempo Aproximado (s)	Tiempo aproximado
48	$2{,}8 \times 10^{14}$	398972	4,61 Días
64	$1{,}8 \times 10^{19}$	26147050423	828,5 Años
72	$4{,}7 \times 10^{21}$	$6{,}69 \times 10^{12}$	212108,8 Años

En conclusión, a partir de claves cuya longitud superen los 64 bits, es decir 8 Bytes, el tiempo para descifrarlas será superior a un año natural, tiempo en el cual se produce el cambio de clave por parte de la entidad bancaria y el cliente. Recomendaríamos el uso de una clave de seguridad de 64 bits, 8 bytes, dado que su tiempo de vida en este caso corresponde a más de ochocientos años, acompañada de un algoritmo de cifrado más fuerte que el usado, que es HmacSHA1, pasando a usar HmacSHA512.

Despliegue de un Verificador de Integridad

Se ha desarrollado un verificador de integridad en los sistemas cliente y servidor de forma que se pueda realizar de forma práctica en la transmisión de los mensajes.

Robustez del sistema desarrollado

Con respecto a las medidas que se han tomado en la implementación, el equipo de desarollo ha implementado dos medidas principales contra ataques.

Mejora de la Generación de MACs

Se ha implementado un algoritmo de generación de MAC basados en algoritmo de cifrado SHA-512, de forma que el tiempo necesario para obtener la clave por fuerza bruta resulte muy costoso en tiempo y recursos. Haciendo que ataques como man-in-the-middle resulten ineficaces al estar capturando un tráfico con alta seguridad.

Uso de Timestamps

Se han hecho uso de timestamps en los mensajes, de forma que en caso de encontrarse ante un ataque de replay, en el que se envían varias peticiones idénticas una vez capturada una petición correcta o se retrasa el envío de otra petición idéntica para confundir al sistema.

Un timestamp es una marca temporal que se genera cuando se crea el mensaje. Mediante esta medida, se asegura que el mensaje contiene una capa de identificación adicional.

Informe Final

A través de las actividades realizadas anteriormente, hemos llegado a las siguientes conclusiones.

En primer lugar, se debe asegurar una transmisión segura de las claves privadas entre cliente y entidad bancaria, haciendo partícipe al menor número de implicados para evitar problemas dentro de la misma entidad, ya que no podemos asegurar las intenciones de un tercero. Por tanto el sistema de entrega de un dispositivo físico debería sustituirse por una transmisión unilateral por parte del cliente con el servidor, asegurándose de que cumplen una serie de condiciones para la transmisión de la contraseña, implementando también medidas para asegurar la fortaleza de la misma.

Un método para esto sería Diffie-Hellman, que se trata de un protocolo para el establecimiento de claves en un medio inseguro, donde se genera una clave compartida mediante dos números públicos y uno secreto para cada parte de la comunicación. Cada parte hace una serie de operaciones con los dos números públicos y su número secreto, y se intercambian los resultados de forma pública, Posteriormente aplicarán una fórmula sobre la información y ambos llegarán al mismo resultado.

La entidad bancaria debería almacenar la clave cifrada en un servidor distinto a donde se encuentre el nombre de usuario asociado, haciendo que este no se un punto único de fallo,

En segundo lugar, ya que el cálculo de una clave HMAC-SHA-512 de más de 8 bytes es de 800 años, no tenemos ningún problema temporal, sin embargo, por seguridad se recomienda

un cambio anual de la contraseña, haciendo que el cálculo de la misma sea mucho más difícil poniendo un límite temporal.

Anexo I: Especificaciones técnicas de los recursos

En este anexo, se procederá a exponer los distintos benchmarks realizados con los recursos tecnológicos disponibles, para distintas operaciones de encriptación.

- Ordenador 1, mediante "john --test" en Manjaro KDE 20.1.1

```
Benchmarking: HMAC-MD5 [password is key, MD5 128/128 AVX 4x3]... (4xOMP) DONE
Many salts:  98770K c/s real, 24754K c/s virtual
Only one salt:       24870K c/s real, 6248K c/s virtual

Benchmarking: HMAC-SHA1 [password is key, SHA1 128/128 AVX 4x]... (4xOMP) DONE
Many salts:  40100K c/s real, 10686K c/s virtual
Only one salt:       13565K c/s real, 3909K c/s virtual

Benchmarking: HMAC-SHA256 [password is key, SHA256 128/128 AVX 4x]... (4xOMP) DONE
Many salts:  20078K c/s real, 5057K c/s virtual
Only one salt:       8720K c/s real, 2180K c/s virtual

Benchmarking: HMAC-SHA512 [password is key, SHA512 128/128 AVX 2x]... (4xOMP) DONE
Many salts:  7476K c/s real, 1907K c/s virtual
Only one salt:       3766K c/s real, 943863 c/s virtual
```

- HP Pavilion x360, mediante "hashcat64.exe -b" en Windows 10

```
Hashmode: 0 - MD5
Speed.#1.........:  2192.4 MH/s (88.56ms) @ Accel:64 Loops:1024 Thr:1024 Vec:8

Hashmode: 100 - SHA1
Speed.#1.........:   772.1 MH/s (64.86ms) @ Accel:32 Loops:512 Thr:1024 Vec:1

Hashmode: 1400 - SHA2-256
Speed.#1.........:   275.4 MH/s (91.03ms) @ Accel:8 Loops:1024 Thr:1024 Vec:1

Hashmode: 1700 - SHA2-512
Speed.#1.........: 77053.0 kH/s (81.30ms) @ Accel:2 Loops:1024 Thr:1024 Vec:1
```

- ASUSPRO P2520LA, mediante "hashcat64.exe -b" en Windows 10

```
Hashmode: 0 - MD5
Speed.#1.........:  1736 MH/s (37.45ms) @ Accel:256 Loops:1024 Thr:64 Vec:1

Hashmode: 100 - SHA1
Speed.#1.........:  1042.0 MH/s (49.05ms) @ Accel:128 Loops:1024 Thr:64 Vec:1

Hashmode: 1400 - SHA2-256
Speed.#1.........:   382.0 MH/s (63.42ms) @ Accel:512 Loops:128 Thr:64 Vec:1

Hashmode: 1700 - SHA2-512
Speed.#1.........: 75964.2 kH/s (71.28ms) @ Accel:16 Loops:1024 Thr:64 Vec:1
```

- HUAWEI MateBook D 14 AMD, mediante "hashcat64.exe -b" en Windows 10

```
Hashmode: 0 - MD5
Speed.#1.........:  2672.5 MH/s (48.82ms) @ Accel:256 Loops:1024 Thr:64 Vec:1

Hashmode: 100 - SHA1
Speed.#1.........:  1030.4 MH/s (63.54ms) @ Accel:128 Loops:1024 Thr:64 Vec:1

Hashmode: 1400 - SHA2-256
Speed.#1.........:   425.0 MH/s (77.50ms) @ Accel:512 Loops:128 Thr:64 Vec:1

Hashmode: 1700 - SHA2-512
Speed.#1.........: 95363.9 kH/s (86.34ms) @ Accel:16 Loops:1024 Thr:64 Vec:1
```

Desafortunadamente, no disponemos de los benchmarks de los ordenadores portátiles, sin embargo, se puede asegurar que la mayor potencia de cálculo la dispone el Huawei MateBook y la menor el Asuspro, donde se han realizado los cálculos de las hmac proporcionadas, dándonos un rango de tiempo muy bajo para descifrar las contraseñas.

Anexo II: Pruebas del Verificador Desplegado

En primer lugar se mostrará una prueba exitosa del envío de un mensaje entre cliente y servidor.

Por parte del cliente se ha enviado un mensaje de forma exitosa al servidor y el servidor ha recibido el siguiente registro en el log.

```
nov. 11, 2020 3:23:42 A. M. codigo.CreacionLog creaLogIntegridad
INFO: Información de la Integridad de los mensajes intercambiados
nov. 11, 2020 3:23:42 A. M. codigo.CreacionLog creaLogIntegridad
INFO: 531456 487654 200 2020-11-11 03:23:42.274
nov. 11, 2020 3:23:42 A. M. codigo.CreacionLog creaLogIntegridad
INFO: Mensajes integros: 1      Mensajes totales: 1     Ratio:100%
```

A continuación tras intentar enviar un mensaje con una clave incorrecta, seha obtenido la siguiente información en el registro.

```
nov. 11, 2020 4:07:33 A. M. codigo.CreacionLog creaLogIntegridad
INFO: Información de la Integridad de los mensajes intercambiados
nov. 11, 2020 4:07:33 A. M. codigo.CreacionLog creaLogIntegridad
INFO: 531456 4444 4 2020-11-11 04:07:33.773
nov. 11, 2020 4:07:33 A. M. codigo.CreacionLog creaLogIntegridad
INFO: Mensajes integros: 2      Mensajes totales: 3     Ratio:66.0%

nov. 11, 2020 4:08:56 A. M. codigo.CreacionLog creaLogIntegridad
INFO: Información de la Integridad de los mensajes intercambiados
nov. 11, 2020 4:08:56 A. M. codigo.CreacionLog creaLogIntegridad
INFO: 531456 487654 200
nov. 11, 2020 4:08:56 A. M. codigo.CreacionLog creaLogIntegridad
INFO: Mensajes integros: 2      Mensajes totales: 4     Ratio:50.0%
```

Tras realizar un pseudo ataque de replay, duplicando el envío de la información, hemos comprobado que el servidor desestima el mensaje, sin embargo esto nos limita a que podemos hacer hasta mil transacciones iguales por segundo, pero no lo consideramos un problema para el usuario medio.

Anexo III: Manual de Uso

Para ejecutar de manera correcta el sistema cliente servidor, será necesario seguir los siguientes pasos:

1. Ejecutar el IntegrityVerifierServer. El servidor empezará a funcionar, creará la carpeta para almacenar el archivo log y se quedará esperando comunicaciones con el cliente. No hace falta volver a ejecutarlo, puesto que se quedará activado de manera pasiva, a la espera. Solo volver a ejecutar en caso de reiniciar el sistema.

2. A continuación, ejecutamos el IntegrityVerifierClient, con el que podremos empezar a comunicarnos con el servidor. A diferencia de este, en cuanto la comunicación termina, se cierra y para volver a enviar un mensaje, hará falta ejecutarlo de nuevo.

3. Tras esto, le aparecerá una ventana al cliente. En ella deberá proporcionar un mensaje con el formato "Cuenta Origen Cuenta destino Cantidad". Las cuentas corresponden con las que proporcionó la entidad bancaria. Al darle a OK o al botón Enter, aparecerá otra ventana que nos pedirá la clave de cliente. En el servidor se han suministrado una clave nueva para cada cuenta, atendiendo al tamaño óptimo para ser seguras, el cual ya hemos analizado. Ambas informaciones servirán para el cálculo del resumen Mac correspondiente. Si quisiéramos cambiar el algoritmo de cifrado, es necesario cambiarlo en VerificadorMac.

4. Todo lo que hemos introducido será leido por el servidor, que empezará a verificar si la Mac es correcta, y no se ha producido modificaciones (Ataque Man-In-The-Middle) o repeticiones de transmisiones (Ataque Replay). Si se diese el caso de que el Mac asignado a un mensaje ha variado, o se detecta que ha llegado una copia de algún mensaje con el mismo instante de tiempo de creación, el servidor nos mandará una aviso como clientes y nos aparecerá una ventana que nos informará, de que el mensaje no ha llegado de manera íntegra al servidor. En el caso de que no haya ningún problema en la comunicación, el aviso será simplemente otra ventana que nos indicará de que el mensaje ha llegado de forma íntegra.

5. En la carpeta raíz del sistema (Windows), se nos debe haber creado una carpeta llamada LogDirectory, que contendrá el archivo log. En su interior encontraremos las entradas correspondientes a los mensajes intercambiados, así como un ratio de los mensajes íntegros respecto a los totales, de toda la comunicación efectuada.