Seguridad del Diseño del Software - 2019

- Melanie Mariam Cruz Morgado
- Pedro Giménez Aldeguer
- Álvaro Garcerán Olivares

Contenido

PRÁCTICA

FASES

1. Registro.

Tenemos el servidor y si lo tenemos el cliente 1 sabe la dirección y le dice que se quiere registrar con este usuario y esta contraseña. Entonces el servidor dice okey, ya te he registrado, cuando lo meta a la base de datos que ya está creada, es decir, añade otro registro. Errores: ya existe, fallo al registrar…

2. Log-in.

Un usuario ya registrado le manda el usuario y (el hash de) la contraseña (que se puede usar para cifrar). Si te logueas correctamente el servidor te da un token para las siguientes peticiones (num aleatorio que solo puede conocer el servidor – servidor ssl lo hace solo) o un canal seguro que se puede usar para comunicarnos.

Una vez hecho esto ya podemos hacer las funcionalidades…

3. Añadir usuario, contraseña y url al servidor.

(min) Para esto lo que haremos es enviarle al servidor algo que quieres añadir y le mandas el usr, pass y url teniendo el token. Si está registrado, así añadirá en tu base de datos personal que tienes guardada en el servidor y se devuelve si se ha conseguido o se ha producido error.

4. Listar mis usuarios, contraseñas y url.

(min) El cliente pide un listado con unas características enviando el token también asegurando quién eres, etc. y el servidor hará una búsqueda en la base de datos sobre estas características y devolverá dicha lista en un Json, xml o lo que sea, o un error. Al recibirlo en el cliente, se mostrará de forma gráfica.

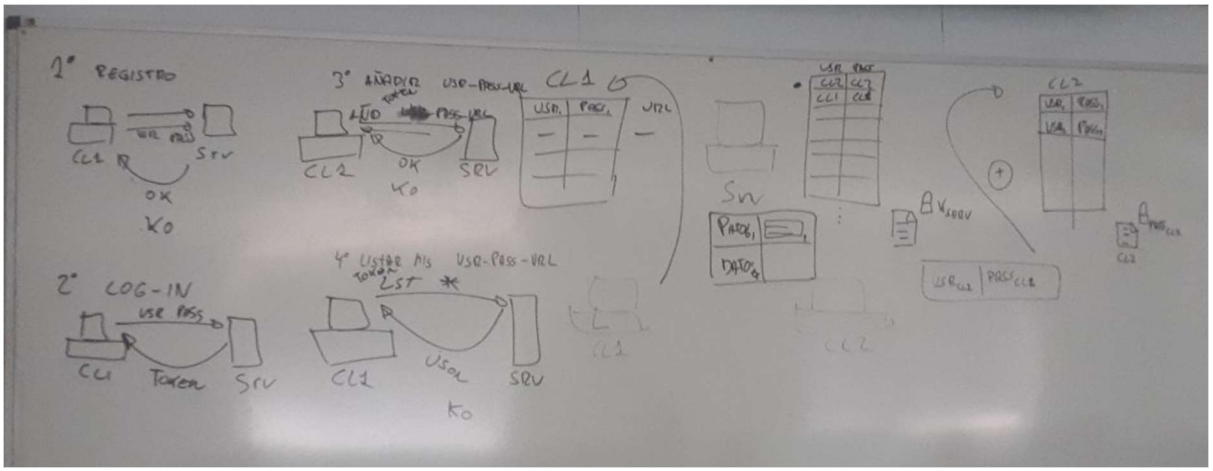
Si tienes conocimiento cero, para poder acceder a más de un 6 de nota, el servidor no sabrá absolutamente nada, así que en la consulta se devolverá todo y que el cliente haga ese filtro.

¿Cuándo se codifica?

(min) El archivo de base de datos del registro de los usuarios tiene que estar cifrado y también las base de datos personales de los diferentes clientes. Cuando la app esté abierta se descifra, se guarda lo que se tenga que guardar y se vuelve a cifrar para guardarse.

(max) El servidor al no poder descifrar la base de datos del cliente, le devuelve todo al querer añadir elemento, el cliente lo descifra, lo añade, lo vuelve a cifrar y se lo vuelve a pasar al servidor.

Varias contraseñas: Las de los usuarios para la autenticación y después las que se están guardando.



PRIMERA ENTREGA (0.5p) →

- Exponer en clase
    - Decir los objetivos que tenemos (qué vamos a hacer).
    - Cómo vamos a implementar esas partes.
    - Decir el estado actual de la práctica.

TEORÍA

Tema 1

19.02.2019

Un bit no se pierde, como mucho se altera. Te llega un patrón de bits y sabes que un bit se ha cambiado, pero no se puede saber cual, se debe de pedir entero de nuevo.

Lo de autosincronizante es muy interesante, pero no tiene aplicación práctica útil. Mucho mejor los síncronos.

LFSRs viene de la época del HW porque eran mucho más fáciles de implementar en estos. La semilla es el estado inicial y el registro tiene capacidad de 2^n-1 porque la secuencia de todo 0 no tiene gracia xD, no sé explicarlo, no tiene sentido que se desplace. PROBLEMA: predecibles, porque se puede encontrar la semilla con n bits de secuencia (con un algoritmo). NO SE PUEDE USAR TAL CUAL EN CRIPTOGRAFÍA, se ha usado, pero no está bien. El GSM del móvil utiliza un LFSRs. La mayoría de criptografos intentan combinarlos para aumentar la seguridad, porque el resultado sería no lineal. El snow3d es el único que conoce que no está roto.

RC4 era secreto de RSA. Es muy rápido. Es un cifrado en grupo. Funciona a nivel de byte, que es muy eficiente en ese sentido. ¿Qué pasa con los algoritmos? Los algoritmos no se pueden patentar, porque sino se puede saber, ya que una patente es pública. Que, por cierto, es muy difícil patentar un algoritmo. Curiosamente, lo que ha permanecido en secreto es el nombre, por eso en vez de RC4 en algunos sitios ponía ARC FOUR, que se lee tal cual, pero no se podía usar RC4. Los intercambios aleatorios no producen un estado aleatorio, por eso entre algunas variables del algoritmo, hay mucha relación. Nadie a conseguido romper el RC4 en sí, pero sí que han podido reducir mucho la búsqueda, por eso ya se empieza a sospechar que no es el más adecuado. La NASA dice que puede romper un algoritmo, pero no dice cual, se sospecha que es este. AES es mucho más lento que RC4, está acelerado por hw. Este permitía hacer mucha cosa con hw muy barato.

Salsa20 (sí, el creador tiene tendencia al baile y es muy joven). Utiliza dos bases para realizar las operaciones para impedir que se resuelvan las incógnitas mediante sistema de ecuaciones. En criptografía se utiliza mucho usar dos bases matemáticas simultáneas.

A5 está totalmente roto. Si vamos a hacer algo, no lo hagamos desde el 2G.

HC128 no se usa casi nah.

Spritz es 10 veces más lento que RC4 y es muy distinto y funciona como un algoritmo de esponja. Ha perdido punch, ya que nadie le hace caso en la comunidad científica.

¿Qué cifrador en flujo elegirías en la actualidad, atendiendo a la seguridad y el rendimiento? Salsa20 o ChaCha, claramente. Porque no hay ataques conocidos capaces de romperlos y por el buen rendimiento.

Tema 2

26.02.2019

Diferencia entre cifrador en bloque y uno en flujo: El cifrador en flujo es variable en el tiempo.

IDEA no está roto, pero no ha tenido éxito. MD es como resumir el mensaje. RC6 participó en el concurso de AES y se lo cargaron.

Scheneier es un criptógrafo muy influyente, tiene parecido con C. Norris y tiene mucho ego. La B de bcrypt es de Blowfish y se ha usado para algunas criptomonedas porque garantiza bastante. Usa RAM y ralentiza, pero ha cogido fama.

En el mundo del cifrado en bloque existe Feistel y la gran mayoría de cifradores en bloque están basados en esta estructura, menos AES. 2^n es pragmático, no se puede implementar. Si la función de permutación es segura, el cifrador es seguro. Es fácil hacer un cifrador en bloque que no se pueda descifrar. Por eso, los cifradores en bloque usan esta estructura. Cada ronda tiene una subclave diferente. La subclave es una modificación de la clave y la generación de esta subclave depende de cada cifrador.

Si algo no está claro el porqué se utiliza, la gente no lo utiliza.

DES surge de un concurso de la NSA, pero no lo eligen tal cual lo crea Feistel, pero cuando lo estandarizan por DES lo cambian Tuchman y Meyer y reducen la clave de 128 a 56 que es menos de la mitad. LA NSA cambia las cajas de Lucifer, pero nunca dice por qué. La gente dice que el diseño estaba manipulado por la NSA para que solo ellos pudiesen piratearlas. Al contrario de lo que la gente pensaba, en realidad es que la NSA en realidad conocía ataques que nadie más conocía y lo convirtió en algo más seguro. Pero eso no quiere decir que lo otro no lo hubiesen hecho en alguna otra ocasión.

La única parte que no se puede medir con una ecuación interlineal es la tabla.

Cuando no se tiene el mensaje de origen, hay que ser capaz de identificar este también.

Los cifradores no son lineales. El criptoanálisis es muy complejo y no es fácil de hacer desde un punto de vista matemático.

Ataque de criptoanálisis dinámico: Cuando un algoritmo de cifrador en bloque tiene en la CPU en proceso de cifrado y comparte la caché con otro proceso que está en la misma máquina. El otro puede ir viendo a qué posición de la tabla está accediendo y es capaz de ir hacia atrás en el algoritmo. Es una forma de hackear el cifrador mediante cajas.

Ataque de criptoanálisis lineal: No da versión exacta del cifrador porque los cifradores no son lineales. Este no requiere que el atacante elija los mensajes, pero sí que los conozca.

No se puede hacer un sistema que tenga una puerta secundaria que abra solo para los gobiernos, ya que la abres también para los terroristas, etc. La idea de que las claves las tenga el gobierno surge de la época de DES.

Los surfaces están todos cifrados. En la aduana a veces te pueden pedir la clave y sino la das pueden pensar que llevas algo en plan malo. China es bastante hardcore en estos aspectos. A veces es mejor no llevarse el pc de viaje.

La criptografía se usa como una arma militar en todo el mundo.

AES tiene una base matemática muy fuerte, usando matrices para todo.

La gran mayoría de los pcs que tenemos hoy en día soportan los cifradores en hardware.

El cuerpo de Galois usa una aritmética especial sobre polinomios, teniendo en cuenta que, por ejemplo: 1101 → x^3+x^2+1.

No es muy difícil extender AES a 512bits ya que solo es hacer más rondas.

Se diferencia de Feistel porque no divide el mensaje, por ejemplo, pero sí que usa el tema de subclaves y tal.

La tabla de sustitución de la ronda no es aleatoria sino que se corresponde con la función de inversión y es la más no lineal de ese tamaño. Está demostrado matemáticamente que además garantiza que es la mejor posible.

La suma es una suma de polinomios, no es una suma normal.

El intercambio de las filas tiene una modificación diferente para cada fila. La primera se deja tal cual.

Recuerda: El producto de una matriz por un vector, da un vector y el mix de columnas está diseñado para ello.

¡¡ Nadie ha conseguido atacar AES !!

No es plan de que nos memoricemos las tablas y tal (aunque se ve tentado a ello), sino que la idea es aprendernos cómo funciona.

Han sido capaces sacar la clave privada pasando la clave pública todo el rato y a partir del sonido que emitía después de varias semanas escuchando. I'm flipping…

ECB: El modo directo se puede usar cuando todos los tamaños del mensaje sean menores al bloque para que sea algo seguro. Pero no se suele usar, porque no es muy seguro.

OFB: La clave es la que es secreta.

CTR: Se incluye el contador inicial que suele ser aleatorio, porque sino no se puede descifrar. Puede parecer inseguro, pero en realidad lo es porque la clave es privada. Es el único que garantiza que va a recorrer todos los estados posibles sin repetición, consiguiendo el periodo máximo posible. No hay descifrado, sino que se usa el mismo cifrado. Permite acceder de forma secuencial al segmento cifrado.

XTS-AES: Doble cifrador de AES que es como hacer un CTR. En muchos protocolos no tiene sentido porque a veces no tiene sentido repetir la autentificación. Se usa cuando no tienes autentificación y esta es la única manera. De esta forma se puede acceder al disco sector a sector pero con una misma clave.

- ¿Qué cifrador en bloque elegirías en la actualidad atendiendo a la seguridad y el rendimiento? AES. ¿Con qué modo de operación? Modo contador.
- ¿Hay situaciones en las que sería preferible un cifrador en bloque a uno en flujo o viceversa? Un cifrador en bloque produce latencia y es más lento que el cifrador en flujo, porque como va bit a bit no hay que esperar y lo podemos ir cifrando a tiempo real. Pero en la mayoría de los casos son intercambiables.
  AES en cifrado rápido y Salsa también son bastante rápidos y se parecen mucho.

5/3/2019

El hash tiene una entrada variable y una salida fija, generalmente manejable. NO DECIR "ENCRIPTAR CON UN HASH", los hashes no tienen clave y, por lo tanto, no se puedes encriptar.

Por alguna razón la gente se aferra a algoritmos ya rotos solo por ser más rápidos, aunque saben que no son seguros.

Siempre se pone el tamaño del mensaje y así se evitan errores. Tiene la siguiente propiedad: Para cambios muy pequeños en el mensaje, el resumen cambia mucho. Se usa el hash en firmas digitales porque se firma el resumen, que es más eficiente. De esta forma, siempre se tarda lo mismo en firmarlo.

El resumen se va juntando como si fuese una secuencia de resúmenes y esa secuencia se puede usar como cifrador en flujo aunque no se usa porque los cifradores en flujo son más rápidos.

Los hashes son muy útiles, seguramente vamos a tener la necesidad de usarlo a lo largo de nuestra vida laboral.

Tiene que comprimir en sentido figurativo, porque no se puede descomprimir, es decir, pierdes datos. Tiene que ser rápido de computar. Tiene que ser fácil calcular la función hash, pero difícil computacionalmente a partir de un resumen obtener un mensaje que genere ese resumen, por eso se le llama una vía. Porque si alguien es capaz de esto puede obtener una forma de romper la función hash. (Cuando habla de fácil y difícil se refiere a rápido y lento) Objetivo: obtener a partir del resumen un mensaje que te produce un resumen.

La colisión débiles tienen que ver con la paradoja del cumpleaños. Ya que es más difícil la colisión fuerte que la débil. (Relacionar con la explicación de dicha paradoja).

El ataque por fuerza bruta también existe y lo que se prueba es todos los resúmenes, pero en realidad estamos comprobando todas las entradas. Por preimagen es lo más difícil y hay muy pocas. Cuando tienes un hash y lo quieres romper usas la “resistencia a colisiones”.

Las funciones hash son una perspectiva diferente de un cifrador en bloque.

¿Cómo convertir un cifrador en bloque en una función hash?
Tengo una entrada y una salida del mismo tamaño. La clave (del mismo tamaño) en vez de estar en la función del cifrador en bloque, lo introduce como otra entrada.

MD5 (MD = digestión de mensajes) sigue funcionando por la costumbre de la gente y porque es muy rápido, pero se ha roto incluso por preimagen. (La costumbre es más difícil que romper que MD5 xD)

RIPEMD-160 no está roto, pero es muy pequeño. Lo crea quien rompe MD5.

Consejito del profe: NO USAR MD5 NI RIPEMD-160.

No hay ninguna explicación matemática que asegure que algo es más seguro si se pasa por dos cifradores que si se pasa por el mismo con una clave distinta. A diferencia de lo que mucho piensan. (No chicos, la criptografía no funciona como vosotros os pensáis).

Podemos usar SHA-2 (no roto y que no salió de concurso, sino que lo sacaron y fin) y se aconseja el 512.

La función esponja tiene un estado interno que se divide en dos mitades: una para absorber (recaudar información) y la otra para estrujar la secuencia cifrante (creo, porque Pedro no me ha dejado oír). Más lento que un cifrador en flujo.

En Tangle (enredo) hay un problema porque el valor que suma W y K, y usaron eso para romperlo.

Sin la sal un atacante puede precomputar de antemano. Con la sal dificultas el problema y a las GPU les da igual la sal, siguen yendo rápido, porque no utiliza la RAM.

Lo correcto actualmente para el almacenamiento de contraseñas es usar PBKDF + sal, que es específico para esto. Va lento y usa mucha RAM, que es lo que necesitamos para esto.

No está roto el PBKDF2, pero no tiene las ventajas de los PBKDF más nuevos. A partir de este empezaron a motivarse con la creación de algoritmos que utilizan RAM a tope. El primero fue BCRYPT y han encontrado una forma en la que cabe en una GPU, así que… mal. Luego surge SCRYPT (La S es de SALSA) define un modelo de funciones que cumplen una serie de propiedades que va a ser difícil que se conviertan en otras que no utilicen RAM. En este dejan elegir tiempo y RAM dependiendo para lo que queremos usarlo, hasta hace poco era lo mejor, porque dependiendo de los parámetros elegimos el objetivo con el que se quiere usar.

Si yo hago cálculos según los datos de entradas y alguien mide la caché, me pueden romper por el “time in attack”.

SSL no es un protocolo de Web, es uno TCP. El problema viene que al principio de la web cada navegador tenía y todo tenía que ser específico al navegador, pero poco a poco se ha ido generalizando. Dos protocolos: De conexión y de sesión. Las conexiones son las peticiones pequeñitas que se le hacen al servidor.

En un navegador normal no se tiene un certificado de clave de cliente, por eso en SSL hay autentificación del servidor al cliente, pero no a la inversa. Este está diseñado para que se pongan ambos de acuerdo en algo que soporten los dos.

HMAC es un hash con clave. Coje los datos y pone el HMAC que está parametrizado con una clave y un atacante no puede descifrarlo sin saber dicha clave.

El HTTP1 estaba enfocado a conexiones múltiples.

Con HTTPS no se puede ocultar dónde accedes porque hay un log de IPs y muchos tenemos IPs dinámicas y también hay un Log de a quién le asignaron una IP. Pero también tenemos las DNS.

La mejor forma de hacer un ataque sin que te pillen es usar VPN. La mejor forma es ser anónimo: irte a otro sitio, cambiarte la MAC, no salir en las cámaras, etc…

SSH surge como versión segura de TELNET. Es como una navaja de doble filo, nos abre un abanico de oporunidades: Servicios que no soportan cifrados se lo puedo poner, etc. No entrarnos en redes, pero hay que tener claro para qué sirve y sus funcionalidades.

Soliniti está basado en Go.

Pregunta: ¿Qué aplicaciones se benefician de protocolos o servicios como SSL/TLS, HTTPS o SSH?
Opciones:

1. Solo aquellas que tengan elementos que requieran seguridad/privacidad.
2. Todas, tengan arquitectura C/S o no, porque escribir en disco y volver a leer es como hacer C/S contigo mismo.

Respuesta: Lo normal es pensar el 1, pero es el 2, que garantiza confidencialidad.

~ En Matrix se ve un ataque por SSH real, vamos a ver esa peli, plis.
~ Al profe le molesta la gente.
~ Nos ha contado una vaina y luego ha dicho: os estoy mintiendo un poco.

12/03/2019

Melanie: Hola, hoy solo atiendo yo… mis compis están a las suya. Álvaro de profe para Pedro, pero… ya le tocará explicármelo a mí. Me acabo de dar cuenta de que encima se le parece al profe de teoría.

Parciales: 26/03, 16/04 y 21/05

La distribución de claves es un protocolo sofisticado y tiene una parte de confianza.

Con clave pública, cada usuario tiene un par de claves y usan la clave pública del destino. De esta forma, solo hacen falta el número de claves como nodos tenga. De otra forma esto sería exponencial. La firma digital no existe en criptografía simétrica, por mucho que varios hayan intentado imitarlo. En Clave Pública a partir de una clave no puedo obtener la otra, es eso lo que permite tener clave pública y privada, y aunque conozcan la pública no pueden descubrir la privada. Lo lógico sería cifrar con la pública y descifrar con la privada, pero los algoritmos que consiguen la firma digital lo hacen a la inversa. Antes de cifrar con clave pública, se guarda una copia ya descifrada antes de pasarla.

La paradoja de la firma digital es que no cifra, todos la pueden descifrar y lo que se firma es el hash (resumen) del mensaje, pero todos pueden ver dicho mensaje. Esto no da confidencialidad, sino autenticidad. Si quisiéramos tenerlo cifrado, además tendríamos que pasarlo por un cifrado aparte.

Criptografía simétrica es usar la misma clave para cifrar y para descifrar. En algún caso son un poco distintas, pero con mucha relación, por ejemplo, que sean la inversa o algo así.

Go no va a dejar cifrar (buscar para completarlo)

Alguien rompió la clave de google de 512 bytes y le envió un correo al que la hizo firmado por él mismo y la cambió jajajaja Por no hacerle caso, ale! se lo tenían que demostrar y pasó.

El tema del tamaño de una contraseña es un dilema que lleva ya tiempo.
Pasar de **n** a **p** y **q** es lo que se busca que sea complicado, ya que se hace buscando los valores de estos y no probando todos los valores de **n**. Lo que dicen de que la criptografía cuántica va a romper este sistema es que va a conseguir la descomposición de **n** en **p** y **q** que son los factores primos.

Hay gente que se piensa que ha roto algo y alguien le demuestra que no.

Cual es el número mínimo que se deben poner en un sudoku para que se pueda resolver sin ambigüedades. Descubrieron que era 17 con algoritmos de cómputo, tras un estudio bastante tocho. ¿Para qué sirve esto? Para

Dicen mucho: RSA va a caer pronto. Pero… jeje Nunca acaba de caer.

Imaginar que estoy usando RSA para enviar claves de DES de 56 bits. ¿Qué tiene que hacer un atacante? Hay veces que se sabe qué es lo que se está pasando, pero no qué contiene. Esto facilita el ataque, ya que, como está cifrado con la clave pública, te la puedes saltar probando todos los casos de DES y cifrarlos con esta y compararlos. Además, no hace falta saber 100% siempre qué es el mensaje, sino que a veces basta con que sea muy probable que sea eso y al probarlo, tachaaan… lo consiguen.

Había un momento en el que si quería cifrar cualquier cosa había que pagarle a RSA. Ahora ya no porque ya perdió la patente, se le acabó el tiempo.

Es fácil saber que un número es primo. Los número primos no son primos 100%, sino que son números aleatorios que pasa una serie de tests y demostrar que no este primo es romper RSA. Es relativamente fácil calcular o identificar un número primo, pero lo realmente difícil es pasar de n a p por q, con número grandes, claro. No hay prueba matemática de que sea seguro, esto es muy difícil y solo hay un algoritmo que está demostrado matemáticamente, pero RSA no ha podido romperse hasta ahora, así que se considera uno de los más seguros aunque no se usa tanto porque hay otro mucho más rápido.

Si hacemos un hackeo, no usemos los certificados que nos da la generalitat… xD Parece lógico, pero no te creas…

A veces para pasar la clave privada se opta porque sea en mano porque es lo más seguro.

En el algoritmo diffie-hellman las claves públicas y privadas dan la misma clave compartida (k) aunque cada uno calcula la suya, pero da lo mismo (mirar diapositivas pág. 17) y si queremos que esto cambie hay que cambiar las claves públicas y privadas. Esto es lo que usa whatsApp.

Hasta ahora solo hemos visto la parte idílica de la clave pública. Pero la vida no es tan bonita.

El ataque MITM es que M consigue hacerse pasar por B y A le pasa su clave pública y este se la pasa a B haciéndose pasar por A. De esta forma puede estar de forma pasiva y simplemente funciona de intermediario y A y B se piensan que están hablando entre ellos, pero esto no siempre es así, depende de la intención.

La autoridad certificadora es la tercera parte de confianza exige que nos fiemos. Esto consigue identificar que estás recibiendo un ataque MITM porque al recibir la clave pública te das cuenta que no está firmada con la autoridad certificadora y te das cuenta que algo pasa. Así que de esta forma se evita suplantación de la clave pública. A veces lo que puede pasar es que ha caducado o cualquier otra cosa, pero ya sabes que algo pasa y lo compruebas. No evita que lo hagan, sino que identificas que lo han hecho.

T. ElCamal es uno de los creadores de SSL y trabajaba en Mozilla. Su algoritmo es muy parecido a diffie-hellman pero añade algún detallito más. Ambos soportan las curvas elípticas. Esto no tiene nada que ver con las elipses. Es una aritmética matemática basada en una serie de expresiones en el plano que muchos matemáticos han intentado pelearse con esto y lo han dejado porque es muy tocho.

Al mensaje se le pueden añadir metadatos (no obligatorio). De todo esto se calcula el hash: HASH (MENS/METAD) y lo firmamos con la clave privada: Epk[HASH (MENS/METAD)] y todo todito es la firma, es decir el HASH + Epk[HASH (MENS/METAD)]. Como podemos ver, el mensaje no es privado, pero como ya hemos explicado antes, el objetivo de la firma digital no es ocultarlo sino autentificarlo.

Hay como varios niveles de ataque. No entra mucho en los ataques a la firma digital, así que solo hay que mirarlo por encima. No es muy importante, solo pensar que es un algoritmo de clave pública y la firma se hace sobre el resumen y no sobre los datos reales.

Con el DNI electrónico en España hubo problemas porque el HASH estaba roto.

¿Si la clave pública sirve para firmar por qué se usa el HASH? Por el tamaño. (Explicación: Muchos se piensan que es por la integridad, pero no, porque la firma digital no necesita el hash para la integridad, ya la garantiza persé, pero es por el tiempo, ya que se hace en un único paso y siempre se tarda lo mismo independientemente del tamaño. Sí que es verdad que da integridad pero no es por eso, sino por el rendimiento).

¿Por qué no usamos criptografía de clave pública para propósito general y qué alternativas tenemos a nuestra disposición? Porque la criptografía asimétrica es tan lenta como Pedro. Alternativa: criptografía simétrica. No me he enterado de lo otro.

¿Qué papel tienen las funciones hash en los protocolos de firma digital? El rendimiento. (Explicado anteriormente)

¿Qué diferencia hay entre un certificado y un par de claves pública/privada? En realidad un certificado es un fichero con un formato estándar y tiene el par de claves pública/privada(esta última cifrada por AES con una clave pin, para que si roban el fichero no esté expuesta) además está firmado por una firma de autenticidad certificadora. Entonces la diferencia es que ese par de claves es solo parte del certificado, pero tiene más cosas.

Muchas apps están haciendo la función de certificado sin este. En los casos que no haga falta exportarlos e importarlos y tal.

Tema 3

26/03/2019

Hoy a Melanie le da palo y me toca a mi apuntar las cosas. En realidad es mejor así porque las notas de Melanie son una mierda. (Notas de anotaciones, aunque las otras tampoco son la hostia)

Firewall

Hoy en día la mayoría de ataques son a las aplicaciones. Esto provoca que el firewall no sea 100% eficaz, debido que un ataque en la red interna o un ataque con un formato de mensaje válido no sería detectado por el firewall.

Además, si los mensajes están cifrados no se pueden filtrar mediante el firewall.

Firewall se está quedando en la mierda. =' (

Melanie le dice a Álvaro: Yo escribo muchiiiisimo más rápido nene.
Álvaro le dice a Melanie: Calidad ante cantidad, nena.

Libro: Las 7 cualidades de un no sé qué seguro A los del firewall les ponemos a explicar la seguridad de un desarrollo y ¿qué es lo que pasa? que la solución que se les ocurre es crear otro firewall.

Concepto: WAF (Firewall de la Aplicación Web)

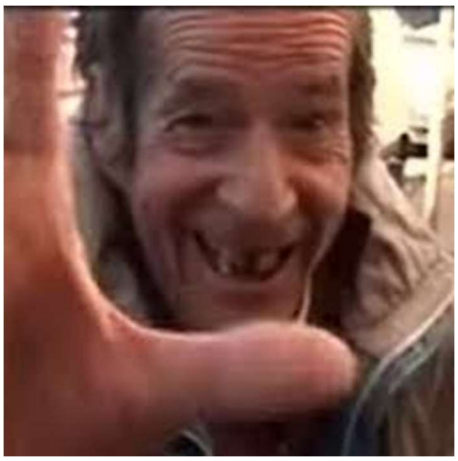
El firewall no aliza la salida y no puede evitar que se filtre información.

Me cansao, así que esto se queda aquí. Pedro está sad porque ha perdido su wordpress. Eso le pasa por no haberse guardado una copia de local. Si me hubiesen hecho caso cuando les dije que primero se desarrolla en local y luego se sube al hosting….

Moxie es un nombre muy ridículo para ser su nombre de verdad, el profe cree que no lo es.

Álvaro: Voy a seguir los ideales de Melanie y solo voy a apuntar lo que no es importante.

No es fácil robar tarjetas de crédito. Se tiene que separar en 3 sectores: robar, vender y usar.



A robar carteras que no hay pasta para comer

No hay que ir en moto. El profesor tiene un problema con las motos porque es bajito y las motos pesan mucho, pero tiene primos motoristas.

Enano en moto

Profe: Estáis muy poco puestos… (Menos Melanie que va todo el día puesta de pastillas de “hierro”) Referencia musical: pastillas de freno a toda pastilla…(8)

Imagen de stock de Melanie

Ha dicho algo de Stuxnet (hecho por la USA), ataque que se utilizó para ralentizar el desarrollo nuclear de Irán alterando la maquinaria para que el proceso de refinamiento del uranio falle de manera aleatoria.

Iraní random

MySpace está dep. El creador supo vender en el mejor momento. La compró Justin Timberlake (el pavo de NSYNC).

Creador de Myspace

El profesor no quiere fichar en las aulas de teoría porque el ordenador es una patata y para fichar a tiempo tendría que llegar 10 min antes.

Ordenador de clase

Antes los virus eran para sacarse la polla. Había uno llamado T64 (porque ocupaba 64 bits). El que conseguía meterse, teniéndolo más pequeño gana.

Se mete en jardines.

Anonymous se lo atribuyen de forma un poco aleatoria y por eso no se tiene claro qué es lo que atribuye eso.

Hace falta un tipo de defensor distinto que explique cómo funciona el software.

En toda la jerga de ciberseguridad, se les llama controles de seguridad a todos los mecanismos que utilizo para mejorar la seguridad.

Proactivo es que intervienen antes para evitar el ataque. Reactivo actúa luego para arreglar lo ocurrido por culpa del ataque.

ROI → Retorno de inversión

Melanie: Me arrepiento de no haber atendido hoy, pido perdón a diosito y aviso que a Álvaro no va a volver a tomar apuntes con fotos, porque le mato. PD: Esa parte va a ser borrada y quien avisa no es traidor@...

¿A que aunque hagamos copias de seguridad no solemos probar a reinstaurar la copia de seguridad? Pues eso, que la mayoría de las veces la copia de seguridad no sirve para nada porque la usamos mal.

Resiliencia: Soy capaz de doblarme y volver a mi forma normal o habitual.

¡¡Hay resumen en esta parte!! GG

2/04/2019

Tema 3c

9/4/2019

Melanie: Como se nota que ya he vuelto, después de dos semanas sin tomar yo apuntes… No me puedo fiar de estos… Tranqui apuntes, ya está aquí mamá.

Se prioriza en función del riesgo que supone.

- Un acceso anónimo es mucho más arriesgado que el autentificado.
- El acceso administrativo es más arriesgado que el nivel usuario, porque no tiene tantas limitaciones.
- La vulnerabilidad que se puede explotar en red tiene más riesgo que en local, porque esta última requiere otro paso más.

- UDP más arriesgado que TCP, porque esta última está orientado a conexión y el UDP ni siquiera requiere al atacante mantener una conexión.

Ejemplo de ataque con formato de fichero: ponerle más número de tamaño al paquete del que tiene y así lea más de lo que hay y así no lee el archivo correcto.

Los ataques de protocolo pasan en servicios integrados, embebidos, etc.

Si alguno de los verbos HTTP a los que se puede atacar no se necesitan, mejor se quitan y te ahorras el tener esa vulnerabilidad.

Lo mejor es no tener un sendmail, pero si se tiene, lo mejor es limitarlo al máximo, ya que los atacantes no paran de buscar abiertos para enviar SPAM.

En linux una parte se compila con el kernel, este puede gestionar la memoria virtual y por eso el SO corre por toda la memoria. La otra parte está en el espacio de usuario.

El código extenso tiene más posibilidad de ataque porque es más difícil de encontrar ahí un fallo que en uno más corto.

El que algo sea open source no significa que alguien lo mire, porque hay códigos extensísimos y no hay quien lo lea.

El control de acceso implica una incomodidad del usuario. El control de acceso fuerte implica una “molestia” al usuario y por la rapidez se acaba usando el acceso débil, pero es más vulnerable. La biometría no es más segura que la contraseña, porque esta es lo único voluntario que hay para la autentificación. Las contraseñas NUNCA van a morir, por mucho que la prensa lo diga.

<<Hay que verse la peli “gataca” y la serie “killing eve”.>>

No se puede tener privacidad sin seguridad, pero sí a la inversa; porque si no se tiene seguridad, aunque se tenga privacidad, cualquier atacante se puede llevar dicha información y deja de ser privada.

Hay que tener mucho cuidado con la información, porque a veces no se sabe para qué se va a usar dicha información en el futuro. Hay expertos que son capaces de personalizar a partir de datos privados y despersonalizados.

Lo de comportamientos y control está especificado con información de EEUU.

Si hay mucha renovación de personal es imposible mantener la seguridad, porque no hay nadie que sepa el funcionamiento correcto del sistema, entre otras.

Hoy en día hacer una rrss mundial es un percal, porque cumplir los controles de todos los países es una locura, ya que incluso se pueden contradecir.

Una amenaza es una vulnerabilidad en potencia y la vulnerabilidad es una amenaza concreta.zzz

Nunca hay que pensar que el sistema es seguro. Nunca existe el riesgo cero. Cada funcionalidad que se ponga, aumenta el riesgo, pero a medida que se van poniendo, tengo que saber a qué riesgos me estoy exponiendo.

SSL a cada cierto tiempo se le encuentra una vulnerabilidad diferente.

Si pones cifrado adicional, añades una capa adicional. A veces, tener logs es una segunda capa de seguridad. Depende de la situación, serán útiles unas capas u otras, ya que lo que hay que pensar es qué ocurre si se me cae una capa y entonces buscar una solución a esto, para añadir capas “de seguridad”.

Si puedes llamar a GCC, puedes hacer lo que quieras.

¿Cuál es el nivel de acceso mínimo que requiere la aplicación para realizar sus funciones? La aplicación tiene que estar lo más limitada posible, para protegerse de ataques. Esto a veces es dinámico, puede que una parte de la aplicación necesita elevar privilegios, pero si luego no hace falta, hay que bajarlo de nuevo. No hay que dejar una aplicación al máximo nivel que se vaya a necesitar en dicha aplicación. Error común: dejar la clave en RAM. Correcto: cuando ya no se necesite dicha clave, sobreescribirla, para si alguien accede a la RAM, no esté dicha clave tontamente. Todo esto no significa que no vaya a tener problemas, sino que estoy intentando que estos sean los menores posibles.

Hay que hacer que el usuario se lea el manual y si no es capaz de configurarlo, que no pueda acceder a él.

Si lo pensamos fríamente volveremos al ensamblador para programar HTML y me ha encantado la metáfora que ha hecho de reinventar la scootter con el patinete eléctrico y el walkie talkie con los audios de whatsApp.

El código en ejecución no es el mismo que se ejecuta.

Hay mecanismos de compilación impiden que se ensamble. Cuando se quiere evitar piratería, se ponen cosas anti-tampering (esto no es tontería, son gente que sabe muchísimo) y los depuradores llaman a interrupciones a tu programa y el depurador puede ver el estado actual y puedes hacer cosas que pueda realentizar esto, que no es del todo seguro, pero algo es.

En el análisis de código cada uno tiene su panacea.

GoSec deja encontrar vulnerabilidades en código en Go.

Ventajas del análisis en código binario: Hay vulnerabilidades o ataques posibles que dependen de cómo ha optimizado el compilador y no en el código fuente. Además, se puede ver el resultado final, es decir, lo que se está ejecutando en el procesador.

Las opciones de optimización del compilador a medida que suben empiezan a quitar seguridad (buscarlo si queremos para informarnos bien).

La dependencia del lenguaje implica que si no tenemos analizador para dicho lenguaje, nos obligará a emplear otro.

Tema 4

16/04/2019

El virus no se puede esperar de forma remota, porque no está esperando órdenes. Los virus se insertan en un ejecutable que se ejecutará a la hora de ejecutar el programa.

Si ocupas toda la RAM vuelcas el PC, porque incluso hoy en día, los ordenadores funcionan realmente mal cuando no tienen espacio en disco.

Los DLL también se pueden infectar si somos algo espabilados.

Hay mucho código ejecutable en cualquier tipo de archivo, por lo que se pueden meter virus en todos estos.

El número de personas que usan ordenadores aumenta más rápido que el número de personas que saben utilizarlo.

Un virus de arranque lo que hace es cambiar el salto de la secuencia que se ejecuta al arrancar, pero no suele tocar el MBR. Esto tiene un riesgo bastante elevado porque arranca antes que el SO y puede cambiar unos drivers que no permitirán que el SO lo elimine e incluso, que ni lo identifique.

El gusano de morris era multiplataforma, bastante avanzado para la época. No se mandaba el código ejecutable. Fue infectando todo lo que pillaba y solo podía infectar máquinas que tuviesen un compilador instalado, que por aquel entonces eran todas. Unix era un servidor y tenías que compilar el kernel seguro 100%.

El virus no se puede controlar de forma remota de forma normal, pero hay excepción a esto si se hace de forma que se conecte a una web, etc. etc.

Con un rootkit son el malware más caro, el que más mola. En un principio no puede ser visto por el SO. Puede manipular el SO para hacer lo que quiera, por ejemplo puede incriminarte.

Muchos ataques son combinaciones de varias formas.

Muchos SO han usado para compilarse el compilador de C.

La mayoría de gente no se da cuenta de que tienes programas trabajando en segundo plano.

IRC de lo primercito que había antes de la web al comenzar internet. Sigue existiendo, solo que no lo usamos. En un protocolo textual.

ULTIMITO DÍA
14/05/2019

No todas las empresas funcionan económicamente, muchas empresas juegan con la expectativa de que alguien les compre porque van a tener muchas ganancias, pero empresas como twitter, no han ganado dinero nunca.

ARP convierte IPs a MACs es un protocolo de Ethernet. Nadie se molestó en ponerle seguridad a ARP, ni a DSS.

Cada vez es más difícil hacer ataques basados en Ethernet porque este no se puede reproducir en un switch ya que es más inteligente porque es capaz de enviar el tráfico a un puerto específico y cada vez se usa más el switch.

B si solo ve la funcionalidad de A y este está mal configurado y B es inteligente. B puede ver a C usando A, ya que C tiene confianza en A. Dibujo: B → A → C

Las centralitas de los coches llevan siendo electrónicas mucho mucho tiempo.

El inalámbrico es menos seguro que el cableado aunque sea más cómodo, bueno, en algunos casos. Porque permite una serie de ataques que no tendríamos de otra forma y es más inhibible.

El bluetooth llega mucho más de lo que esperas. (Anécdota del taller y la casa de su amiga xD

Piconet es que hay el único master y Scatternet es que un nodo puede tener más de un master.

Cuanto más velocidad, menos penetración tiene. (Cada cual que entienda lo que quiera, pero se refiere al wifi)

Hay aplicaciones que van catalogando y mapeando en todo momento.

Cada vez se tienen más routers en casa en modo managed. Las cámaras externas se pueden conectar y funcionar como red punto a punto. El modo monitor puede guardar información y si se quiere guardar en paralelo de varias redes se necesitaría una antena para cada red (negativo)

Que vaya mal el wifi en un encuentro puede ser porque se quede sin ips o que no tenga el ancho de banda suficiente.

El sol inhibible, porque manda su resplandor a todas partes.

En Ethernet todos emiten a la vez y los otros detectan la colisión, lo vuelven a mandar y así se satura. Pasa lo mismo en wifi.

Los técnicos de la UA no se quieren molestar en ponerle seguridad a los servicios.

Se puede saber el fabricante por la MAC.

Todo el mundo puede esnifar el tráfico aunque no esté en la wifi. Es muy fácil asumir qué IPs se están usando en un DHCP. El filtrado de MACs solo funciona para capar el acceso a alguien de tu entorno y tal.

La seguridad basada en SSID es bastante tonta. SSID es el nombre de la red.

WEP no es seguro realmente, por lo que es una vulnerabilidad. El problema no es usa CRC, daría igual que usase otro y es porque se va repitiendo y si vas haciendo muchos ataques, al final lo sacas porque se van repitiendo, o sea, reutiliza la secuencia cifrante y esto es un error. En un ataque de WEP lo que hay es que generar mucho tráfico. Si se mandan paquetes desde fuera se puede saber si es un ping o es otro tipo. Si la red no tiene mucho tráfico, no funciona.

Hay aplicaciones que tienen listas de claves cifradas anteriormente y con esa lista se puede probar muy rápido y romper.

WPA es una mejora de WEP. Lo hace parecido, peor mejor.

Muchas veces el problema no es del protocolo, sino de que las personas suelen poner contraseñas tontas y muy fáciles de sacar.

No tiene sentido usar WPA, teniendo WPA2. Están trabajando en esto y pronto saldrá WPA3.

Es muy útil dividir en diferentes redes virtuales, en plan una para invitados y tal. Antes era muy difícil, pero ahora es muy fácil e incluso vienen algunas configuradas.