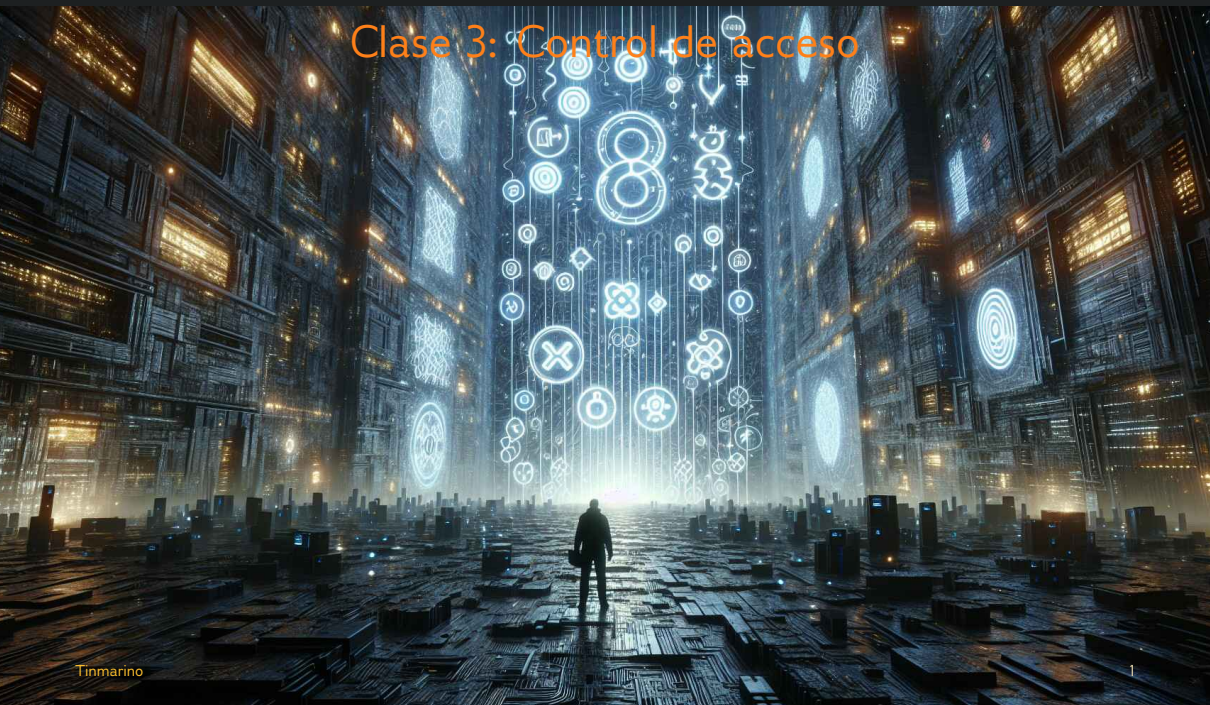
Pentest Web

Clase 3: Control de acceso



Clase 3: Control de acceso

N.	Clase	M1	M2	M3	M4
1	**Introducción**	Contexto	Ciberseguridad	HTTP	Hacktitud
2	**Reconocimiento**	Subfinder	Nmap	FFuF	BurpSuite
3	**Acceso**	Fundamentos	Criptografía	Tecnología	IDOR
4	**Incursión**	Clasificación	Divulgación	Cliente	Avanzados
5	**Lógica**	Negocio	Flujo	Aritmética	Diseño
6	**Inyección**	SQL	OS	Código	Parámetros
7	**informe**	Equipos	Objetivo	Metodología	Reporte
8	**Conclusión**	Resumen	Reflexiones	CVE	Futuro

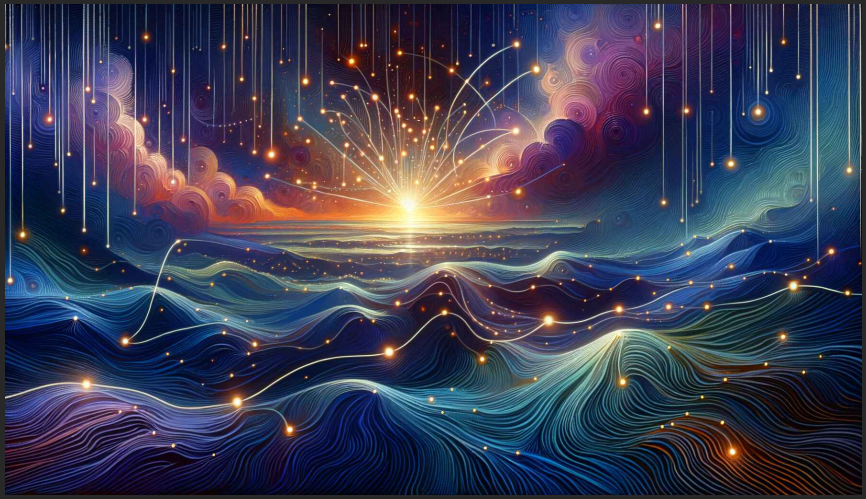
Clase 3: Control de acceso

N.	Clase	M1	M2	M3	M4
1	**Introducción**	Contexto	Ciberseguridad	HTTP	Hacktitud
2	**Reconocimiento**	Subfinder	Nmap	FFuF	BurpSuite
3	**Acceso**	Fundamentos	Criptografía	Tecnología	IDOR
4	**Incursión**	Clasificación	Divulgación	Cliente	Avanzados
5	**Lógica**	Negocio	Flujo	Aritmética	Diseño
6	**Inyección**	SQL	OS	Código	Parámetros
7	**informe**	Equipos	Objetivo	Metodología	Reporte
8	**Conclusión**	Resumen	Reflexiones	CVE	Futuro

Clase 3: Control de acceso

M	Nombre	Descripción
1	Fundamentos	Autorisación y autenticación
2	Criptografía	Como encryptar datos
3	Tecnología	Tecnologias comunes del control de acceso
4	IDOR	Referencia directa de objeto insegura

Módulo 1: Fundamentos



Módulo 1: Fundamentos

S	Nombre	Descripción
1	Autenticación	¿Quien es este usuario?
2	Autorisación	¿A que recurso puede acceder?
3	Métodos	¿Como se controla el acceso?
4	Fallos	¿Como se puede eludir el control?

Módulo 1: Fundamentos

Según el OWASP Top 10 el control de acceso es importante (y vulnerable).

- A07:2021-Fallas en la identificación y autenticación
- A01:2021-Pérdida de control de acceso
- A02:2021-Fallas criptográficas

Sesión 1: Autenticación

(Módulo 1: Fundamentos)

¿Que es la autenticación?

La autenticación es el proceso de verificar la identidad de un usuario o de sistema.

La autenticacíon garantiza que solo los usuarios autorizados accedan a los recursos.

Categorías de autenticación

Los sistemas de TI utilizan diversos métodos de autenticación, que se clasifican en tres categorías principales.

N.	Nombre	Descripción	Ejemplo
1	**Conocimiento**	Información que solo el usuario conoce	Contraseña
2	**Posesión**	Elementos que el usuario tiene	2FA
3	**Inherencia**	Características biométricas del usuario	Huellas dactilares

Ejemplos de authenticación

Conocimiento	Posesión	Inherencia
Contraseña	Tarjeta de identificación	Huella dactilar
PIN	Token de seguridad	Patrón facial
Respuesta	Aplicación de autenticación	Reconocimiento de voz

Desafíos de la autenticación

- Contraseñas débiles
- Phishing
- Gestión de credenciales
- Divulgación de información
- Bypass
- Intercepción

Mejores prácticas

- Controlar el acceso a la autenticación (CAPTCHA).
- Implementar autenticación multifactor.
- No revelar nada en caso de autenticación fallada.
- Usar contraseñas fuertes y únicas.
- Capacitar a los usuarios sobre seguridad.

Autenticación basada en conocimiento

La autenticación basada en conocimiento es común y fácil de atacar. Se basa en información personal estática que puede ser obtenida, adivinada o forzada.

Los atacantes explotan debilidades a través de ingeniería social y filtraciones de datos.

Autenticación basada en posesión

Este método es más resistente a amenazas comunes como phishing. Los tokens físicos son más difíciles de replicar. Sin embargo, la distribución y gestión de estos dispositivos puede ser un desafío.

También son vulnerables a ataques físicos, como el robo o la clonación de objetos, y ataques criptográficos en el algoritmo utilizado.

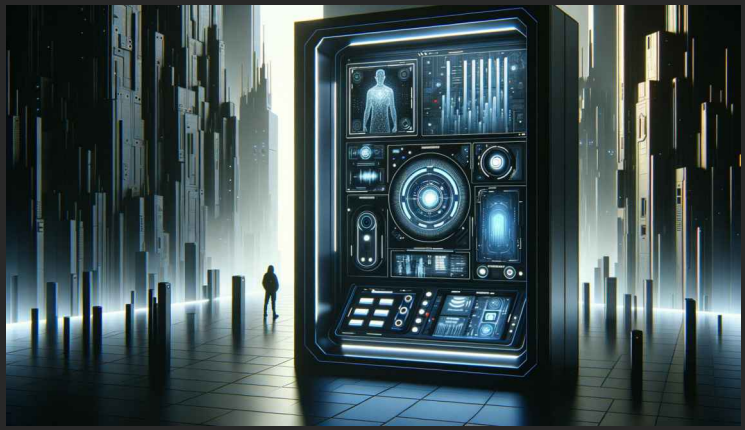
Autenticación basada en inherencia

Proporciona conveniencia al no requerir contraseñas complejas. Los usuarios utilizan datos biométricos para acceder. Sin embargo, debe abordar preocupaciones sobre privacidad y seguridad de datos.

Si hay una violación de datos, las características biométricas no se pueden cambiar, lo que representa un riesgo significativo.

Sesión 2: Autorización

(Módulo 1: Fundamentos)



¿Qué es la autorización?

La autorización es el proceso de otorgar permisos a un usuario autenticado para acceder a recursos específicos.

La autorización garantiza que los usuarios solo puedan acceder a los recursos y realizar acciones para los cuales tienen permisos.

Grandes categorías de autorización

Los sistemas de TI utilizan diversos métodos de autorización, que se clasifican en tres categorías principales.

N.	Nombre	Descripción
1	**Basada en Roles**	Líder de equipo de Pentest accede a herramientas.
2	**Basada en Atributos**	Personal de TI accede a sistemas internos.
3	**Basada en Políticas**	Acceso permitido solo durante horas laborales.

Ejemplos de autorización

Basada en Roles	Basada en Atributos	Basada en Políticas
Administrador	Usuario con privilegios especiales	Acceso restringido por tiempo
Usuario	Acceso según ubicación	Acceso según tipo de dispositivo
Invitado	Acceso temporal	Acceso basado en condiciones

Desafíos de la autorización

- Configuración incorrecta de permisos
- Escalación de privilegios
- Complejidad en la gestión de roles
- Falta de auditoría de accesos
- Falta de políticas de datos
- Dependencia de sistemas heredados
- Dificultades en la integración de sistemas

Mejores prácticas

- Diseñar según el concepto de **confianza cero**.
- Implementar el principio de **menor privilegio**.
- Utilizar **autenticación multifactor** para acceso a recursos críticos.
- **Documentar** y mantener políticas de acceso claras.
- **Capacitar** a los usuarios sobre la gestión de accesos.
- **Monitorear y auditar** regularmente los permisos y los accesos.

Autorización basada en roles

La autorización basada en roles asigna permisos a los usuarios según su rol en la organización.

Esto simplifica la gestión de accesos, pero puede ser vulnerable a la escalación de privilegios si no se gestiona adecuadamente.

Autorización basada en atributos

Este método permite una mayor flexibilidad al otorgar permisos basados en atributos específicos del usuario, como su ubicación o tipo de dispositivo.

Sin embargo, puede ser más complejo de implementar y gestionar.

Autorización basada en políticas

La autorización basada en políticas utiliza reglas definidas para controlar el acceso a recursos.

Esto permite una gestión más granular, pero requiere un mantenimiento constante para asegurar que las políticas sean efectivas y actualizadas.

Sesión 3: Métodos

(Módulo 1: Fundamentos)

Mecanismos de control

Mecanismo	Descripción
Carga firmada	Token de acceso que contiene información firmada. Ex: JWT
Sesión	Identificador de sesión que indexa datos en el backend. Ex: GUID, Base64, ID, Número
Intermediario	Protocolo que permite delegar la autenticación. Ex: OAuth
Claves API	Identificador único para autenticar solicitudes a una API. Ex: API KEY: OpenAI, CTFD, Shodan
Autenticación básica	Envío de usuario y contraseña. Ex: Basic Auth Base64, URL Authority

Ejemplo de roles

Rol	Descripción
Administrador	Gestor de permisos y configuraciones del sistema. Ex: Asigna roles y controla accesos.
Usuario	Accede a recursos según permisos asignados. Ex: Puede ser un empleado o cliente.
Auditor	Revisa y verifica el cumplimiento de políticas de acceso. Ex: Realiza auditorías de seguridad.
Desarrollador	Crea y mantiene aplicaciones. Ex: Implementa autenticación y autorización en el código.
Operador de Seguridad	Monitorea y responde a incidentes de seguridad. Ex: Gestiona alertas y responde a brechas de seguridad.

Implementación de roles

```php
<?php session_start();  // Leer PHPSESSION (cookie) y establecer $_SESSION (PHP)
// Simular un usuario autenticado con un rol (admin, user o auditor)
$_SESSION['user'] = [ 'username' => 'john_doe', 'role' => 'admin' ];

// Verificar el rol del usuario
function hasRole($role) {
    return isset($_SESSION['user']) && $_SESSION['user']['role'] === $role; }

// Clausula: Control de acceso a una página específica
if (!hasRole('admin')) {
    die("Acceso denegado. No tienes permisos suficientes."); }

// Contenido accesible solo para administradores
echo "Bienvenido, " . $_SESSION['user']['username'] . "! Tienes acceso a la sección
↪  de administración."; ?>
```

Transporte

1. Cookie de sessión
2. Cabecera «Authorization: Basic Ym9zY236Ym9zY28=»
3. Cabecera «Authorization: Bearer eyJhbGciOiJIU...»
4. URL autoridad (http://username:password@dev.myapp.com/api/users)
5. URL parametro (?access_token=mytoken123)

Controlar la autenticación

¿Como controlar el acceso a la interfase de autenticación?

Controlar la autenticación

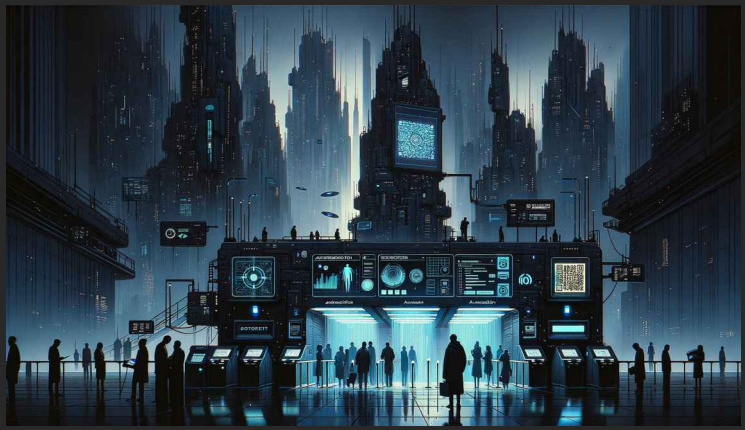
¿Como controlar el acceso a la interfase de autenticación?

CAPTCHA: **C**ompletely **A**utomated **P**ublic **T**uring test to **T**ell **C**omputers and **H**umans **A**part

Cada *endpoint* tiene que tener o **control de acceso** o **CAPTCHA**.

Sesión 4: Fallos

(Módulo 1: Fundamentos)



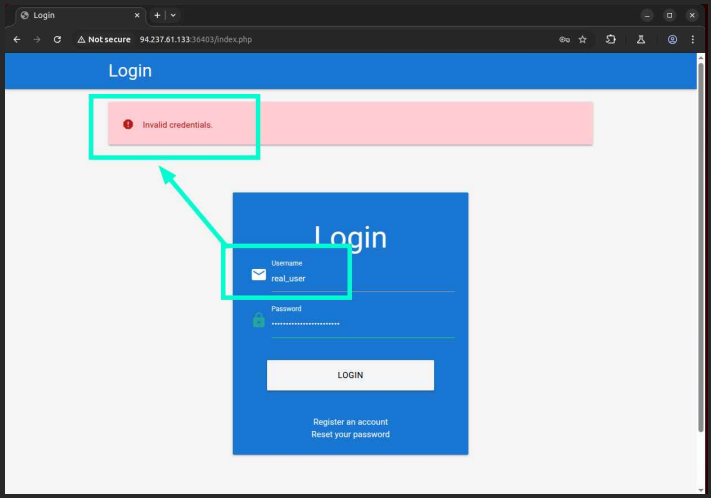
Enumeración de usuario

Descripción: La enumeración de usuario es un proceso mediante el cual un atacante intenta identificar nombres de usuario válidos en un sistema.

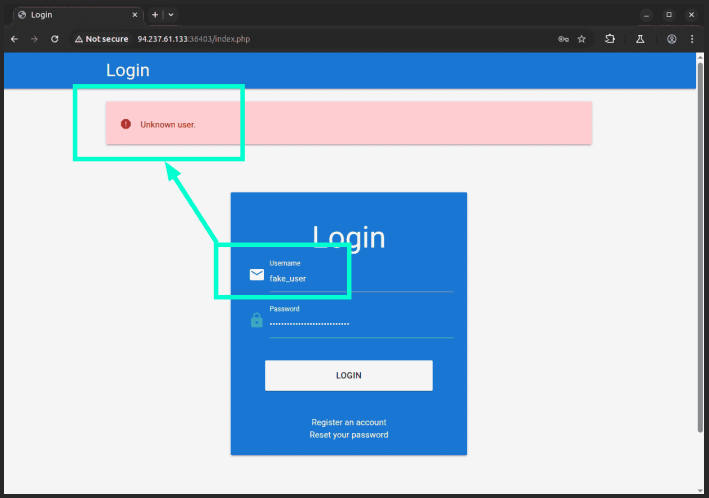
Problematica: Este tipo de ataque se vuelve especialmente problemático cuando los nombres de usuario son predecibles o fácilmente accesibles.

1. En Chile, el nombre de usuario suele ser el RUT.
2. Non repudiacion
3. Facilita ataques de password spray
4. Facilita bloqueo masivo
5. Facilita divulgación masiva

Mensaje de error



Mensaje de error

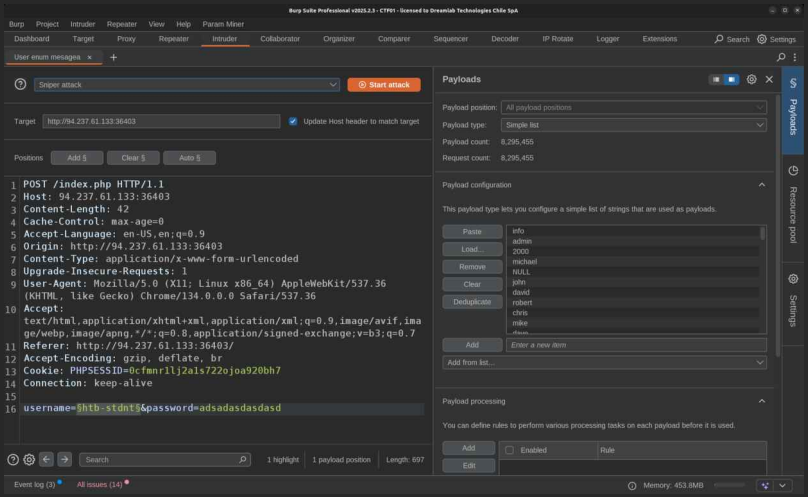


Explotación FFuF

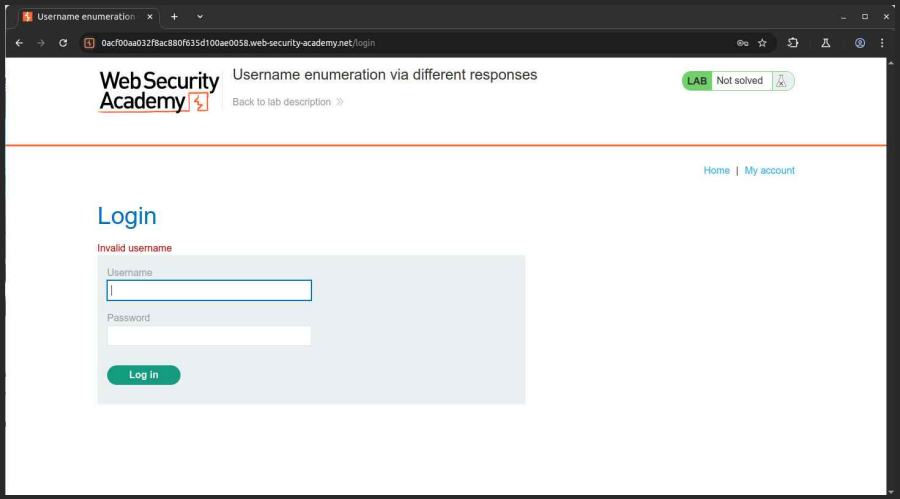
Ver clase sobre FFuF.

```
ffuf \
  -w xato-net-10-million-usernames.txt \
  -u http://94.237.61.133:36403/index.php \
  -X POST \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "username=FUZZ&password=azazzaza" \
  -fr "Unknown user." \
  -t 1000
```

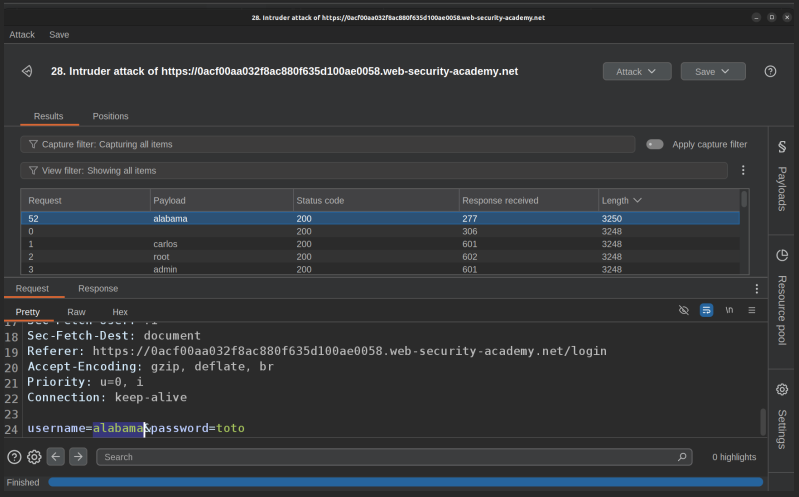
Explotación BurpSuite



Mensaje de error



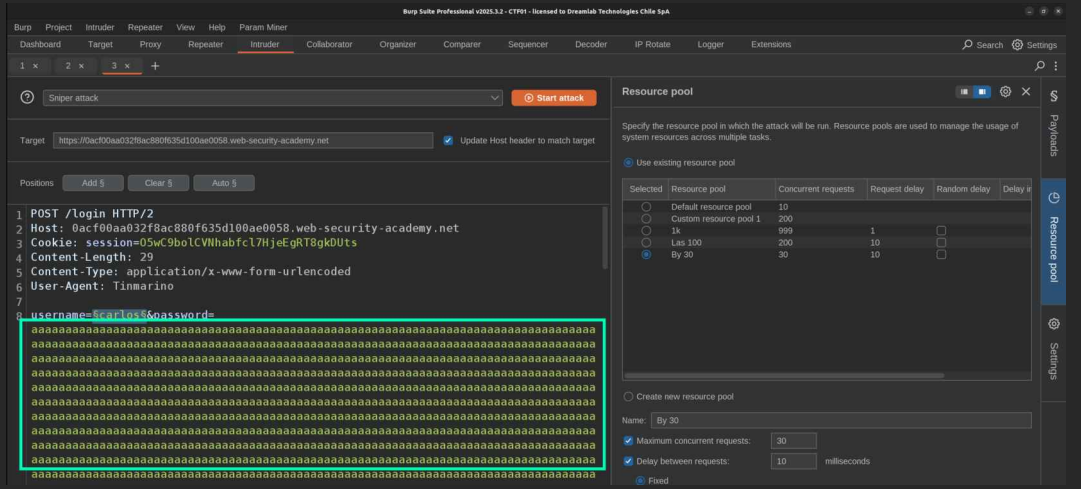
Mensaje de error



Ataque temporal

```php
<?php
$username = $_POST['username']; $password = $_POST['password'];

if (!array_key_exists($username, $users)) {
    echo "Invalid username."; exit 1; }

if (!password_verify(hash($password),
 ↪  $users[$username].password_hash)) {
    echo 'Invalid password'; exit 2; }
?>
```

Ataque temporal



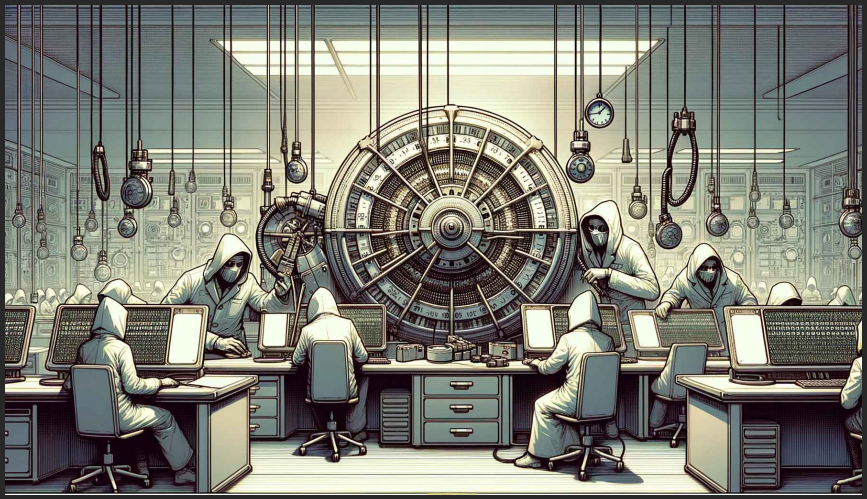
Ataque temporal

```php
<?php
$username = $_POST['username']; $password = $_POST['password'];
 ↪  $isblocked = False;

# Multiple checks
if (!array_key_exists($username, $users)) { $isBlocked = True }
if (!password_verify(hash($password),
 ↪  $users[$username].password_hash)) { $isBlocked = True }

# Single exit
if (isBlocked){echo 'Invalid username or password'; exit 1; }
?>
```

Otros fallos de control de acceso

1. Ausencia de control
2. Bypass del control
3. Bloqueo masivo de usuario.
4. Brute forcing del OTP del reinicio de clave.
5. Falta de entropia en los secretos
6. Falta de validación de roles
7. Ausencia de políticas de acceso claras

Módulo 2 Criptografía



Módulo 2: Criptografía

S	Nombre	Descripción
1	Conceptos	Definición y aplicaciones
2	Simétrica	Cifrar y descifrar con la misma clave
3	Asimétrica	Cifrar con clave privada, descifrar con clave pública
4	Claves	Gestión de claves: generación, distribución, almacenamiento

Sesión 1: Conceptos

(Módulo 2: Criptografía)

¿Que es la criptografía?

La criptografía (o criptología) es la práctica y estudio de técnicas para la **comunicación segura** en presencia de comportamientos **adversariales**.

Su objetivo principal es **proteger la información** mediante el cifrado, lo que permite garantizar la confidencialidad, integridad y autenticidad de los datos.

Los 5 pilares de la criptografía

N.	Nombre	Descripción
1	**Confidencialidad**	Asegura que solo las partes autorizadas puedan acceder a la información.
2	**Integridad**	Garantiza que los datos no sean alterados.
3	**Disponibilidad**	Asegura el acceso contanste al canal.
4	**Autenticidad**	Verifica la identidad de los involucrados.
5	**Responsabilidad**	Asegura la no repudiacón de los involucrados.

Disciplinas de la criptografía

- Matemáticas
- **Informática**
- Física
- Eléctricidad
- Electronica

Aplicaciones prácticas de la criptografía

- Comercio electrónico
- Tarjetas de pago basadas en chip
- Monedas digitales
- Contraseñas de computadora
- Comunicaciones militares
- **HTTPS**
- **Control de acceso**

Tipos de criptografía

Tipo	Descripción
Simétrica	Usa la misma clave para cifrar y descifrar datos.
Asimétrica	Utiliza un par de claves (pública y privada).
Unidireccional	Transforma datos en un irreversible.

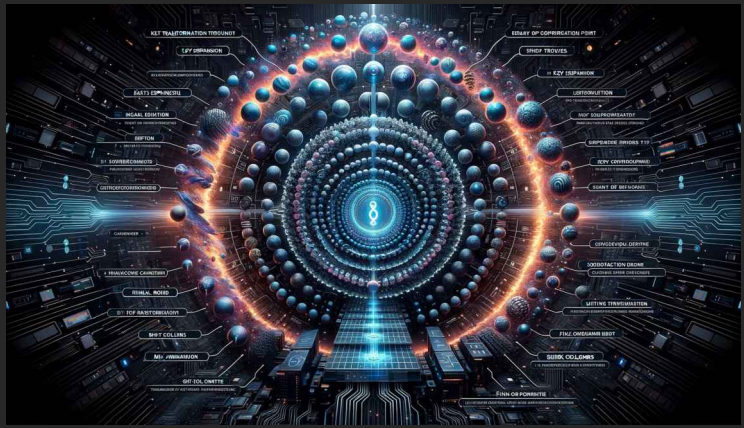
Conceptos recordar

La criptografía es la sciencia de encryptar datos.

La criptografía asymetrica tiene **dos claves**. Hay que esconder la clave pública.

Sesión 2: Simétrica

(Módulo 2: Criptografía)



Definición

La criptografía simétrica es un método de cifrado en el que se utiliza **la misma clave para cifrar y descifrar la información**. Esto significa que tanto el emisor como el receptor deben conocer y mantener en secreto la clave compartida.

Ventajas: Rápido y eficiente para grandes volúmenes de datos. **Desventajas**: La distribución segura de la clave es un desafío.

ÆS	Serpent	DES	3DES	Blowfish	TwoFish
IDEA	Tea	XTEA	RC4	ChaCha20	Camelia

Índice temático

v • t • e	**Block ciphers (security summary)**
Common algorithms	**AES** • Blowfish • DES (internal mechanics, Triple DES) • Serpent • SM4 • Twofish
Less common algorithms	ARIA • Camellia • CAST-128 • GOST • IDEA • LEA • RC5 • RC6 • SEED • Skipjack • TEA • XTEA
Other algorithms	3-Way • Adiantum • Akelarre • Anubis • Ascon • BaseKing • BassOmatic • BATON • BEAR and LION • CAST-256 • Chiasmus • CIKS-1 • CIPHERUNICORN-A • CIPHERUNICORN-E • CLEFIA • CMEA • Cobra • COCONUT98 • Crab • Cryptomeria/C2 • CRYPTON • CS-Cipher • DEAL • DES-X • DFC • E2 • FEAL • FEA-M • FROG • G-DES • Grand Cru • Hasty Pudding cipher • Hierocrypt • ICE • IDEA NXT • Intel Cascade Cipher • Iraqi • Kalyna • KASUMI • KeeLoq • KHAZAD • Khufu and Khafre • KN-Cipher • Kuznyechik • Ladder-DES • LOKI (97, 89/91) • Lucifer • M6 • M8 • MacGuffin • Madryga • MAGENTA • MARS • Mercy • MESH • MISTY1 • MMB • MULTI2 • MultiSwap • New Data Seal • NewDES • Nimbus • NOEKEON • NUSH • PRESENT • Prince • Q • QARMA • RC2 • REDOC • Red Pike • S-1 • SAFER • SAVILLE • SC2000 • SHACAL • SHARK • Simon • Speck • Spectr-H64 • Square • SXAL/MBAL • Threefish • Treyfer • UES • xmx • XXTEA • Zodiac
Design	Feistel network • Key schedule • Lai–Massey scheme • Product cipher • S-box • P-box • SPN • Confusion and diffusion • Round • Avalanche effect • Block size • Key size • Key whitening (Whitening transformation)
Attack (cryptanalysis)	Brute-force (EFF DES cracker) • MITM (Biclique attack • 3-subset MITM attack) • Linear (Piling-up lemma) • Differential (Impossible • Truncated • Higher-order) • Differential-linear • Distinguishing (Known-key) • Integral/Square • Boomerang • Mod *n* • Related-key • Slide • Rotational • Side-channel (Timing • Power-monitoring • Electromagnetic • Acoustic • Differential-fault) • XSL • Interpolation • Partitioning • Rubber-hose • Black-bag • Davies • Rebound • Weak key • Tau • Chi-square • Time/memory/data tradeoff
Standardization	AES process • CRYPTREC • NESSIE • NSA Suite B • CNSA
Utilization	Initialization vector • Mode of operation • Padding

Cifrado de César (-50)

Aspecto	Descripción
Origen	Nombrado en honor a Julio César (-50), quien utilizaba este método para enviar mensajes secretos.
Descripción	Cifrado por sustitución donde cada letra se desplaza un número fijo en el alfabeto (ej. 'A' a 'D').
Uso	Simple, pero vulnerable a ataques de fuerza bruta y análisis de frecuencia.

Cifrado de César (-50)

```
def caesar(s, k, decode = False):
    if decode: k = 26 - k
    return "".join([
        " " if i == " "
        else chr((ord(i) - ord('a') + k) % 26 + ord('a'))
        for i in s])

caesar("acuerdate de amar", 3)
Out: 'dfxhugdwh gh dpdu'

caesar("dfxhugdwh gh dpdu", 3, decode=True)
Out: 'acuerdate de amar'
```

Cifrado de Vigenère (1553)

Aspecto	Descripción
Origen	Desarrollado por Blaise de Vigenère en el siglo XVI.
Descripción	Utiliza una palabra clave para determinar el desplazamiento de cada letra en el texto.
Uso	Considerado seguro durante siglos, pero vulnerable a ataques de análisis de frecuencia.

Cifrado de Vigenère (1553)

```python
from itertools import starmap, cycle

def vigenere(msg, key, decrypt=False):
  return ''.join(starmap(
    lambda c, k: ' ' if c == ' ' else (
      (lambda shift: chr(((ord(c) + shift) % 26) + ord('a')))(
        -ord(k) if decrypt else ord(k) - 2 * ord('a'))),
    zip(msg, cycle(key))))
vigenere("en la duda reinicia", "clavesecretalarga")
# Out: 'gy ge hwue rpieockl'
vigenere("gy ge hwue rpieockl", "clavesecretalarga",
 ↪  decrypt=True)
# Out: 'en la duda reinicia'
```

Cifrado de Vigenère (1553)

El cifrado de Vigenère, aunque más seguro que el cifrado de César, también presenta desventajas:

1. **Vulnerabilidad a análisis de frecuencia**: Con suficientes datos, los patrones de la clave pueden ser descubiertos, lo que podría permitir a los atacantes romper el cifrado.
2. **No Maneja la punctuación**: Los espacios en el texto plano se ignoran durante el cifrado, lo que significa que se mantienen en su lugar en el texto cifrado.
3. **Requiere clave segura**: La seguridad depende de la longitud y aleatoriedad de la clave; claves cortas o repetidas comprometen la seguridad.
4. **Dificultad para paralelizar**: La naturaleza del cifrado de Vigenère hace que sea complicado implementar el cifrado y descifrado en paralelo, lo que puede afectar el rendimiento.
5. **No es perfectamente seguro**: Aunque más seguro que métodos simples, no es infalible y puede ser descifrado con técnicas adecuadas.

Libreta de único uso (1882)

Aspecto	Descripción
Origen	Descrito por Frank Miller en 1882. Patentado por Gilbert Vernam en 1917. Demostrado seguro por Claude Shanonpor en 1945.
Descripción	Realiza la operación XOR entre el texto plano y la clave.
Uso	Militares durante la Primera Guerra Mundial.

Libreta de único uso (1882)

```python
from itertools import starmap, cycle

def one_time_pad(msg, key):
    return b''.join(starmap(
        lambda c, k: (c ^ k).to_bytes(),
        zip(msg, key)
    ))

key = b"\x03\x86\x62\xE1\x11\xB6\x0C\x1A\xA1\xE8\xDA\x9F\x63"
one_time_pad(b"asi de simple", key)
# Out: b'b\xf5\x0b\xc1u\xd3,i\xc8\x85\xaa\xf3\x06'
one_time_pad(b'b\xf5\x0b\xc1u\xd3,i\xc8\x85\xaa\xf3\x06', key)
# Out: b'asi de simple'
```

Libreta de único uso (1882)

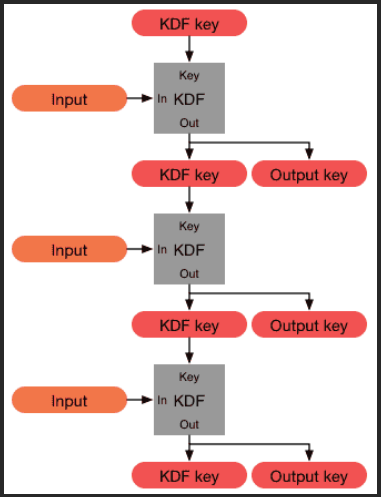
El texto cifrado resultante es **imposible de romper** si se cumplen las siguientes cuatro condiciones:

1. La clave debe ser al menos tan larga como el texto plano.
2. La clave debe ser verdaderamente aleatoria.
3. La clave no debe ser reutilizada, ni en su totalidad ni en parte.
4. La clave debe mantenerse completamente en secreto por las partes que se comunican.

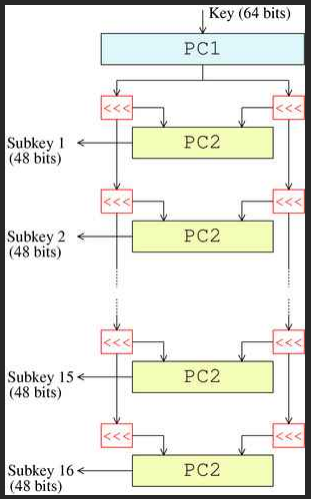
Derivación de clave (1978)

Aspecto	Descripción
Origen	Inventado por Robert Morris en 1978.
Descripción	Utiliza la salida de un bloque como nueva clave.
Uso	Alargar, acortar o diversificar claves; crear hashes y corrientes pseudoaleatorias.

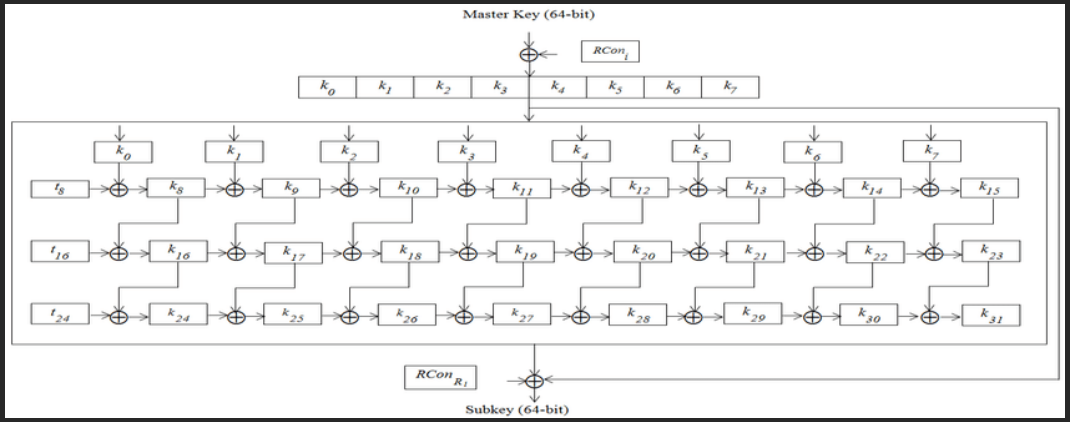
Expansión de clave



Expansión de clave

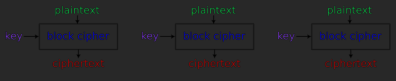


Expansión de clave

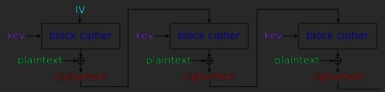


Modos de cifrado por bloques

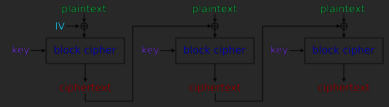
Electronic codebook (ECB)



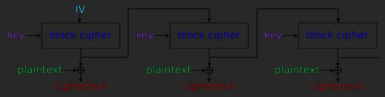
Cipher feedback (CFB)



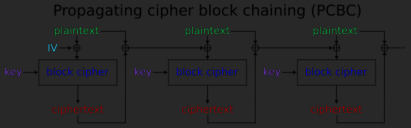
Cipher block chaining (CBC)



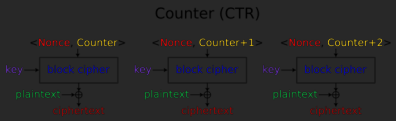
Output feedback (OFB)



Propagating cipher block chaining (PCBC)



Counter (CTR)



El pinguino ECB

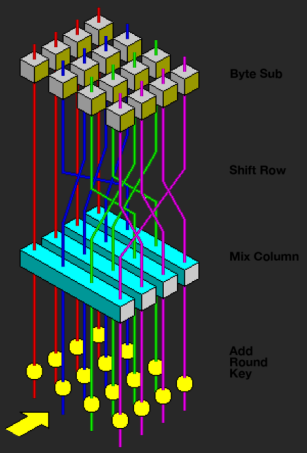
El pinguino ECB

```
# First convert the Tux to PPM with Gimp
# Then take the header apart
head -n 4 Tux.ppm > header.txt
tail -n +5 Tux.ppm > body.bin
# Then encrypt with ECB (experiment with some different keys)
openssl enc -aes-128-ecb -nosalt -pass pass:"ANNA" -in body.bin
 ↪  -out body.ecb.bin
# And finally put the result together and convert to some better
 ↪  format with Gimp
cat header.txt body.ecb.bin > Tux.ecb.ppm
```

Advanced Encryption Standard (1998)

Aspecto	Descripción
Origen	Implementado por Joan Dæmen, Vincent Rijmen en 1998
Descripción	Cifrado por bloques de 128 bits. Utiliza claves de 128, 192 o 256 bits. Realiza múltiples rondas (10, 12 o 14) de: substitución, permutación, mezcla de columnas y adición de clave.
Uso	Ampliamente utilizado, por ejemplo: HTTPS, cifrado de discos y comunicaciones seguras.

ÆS (1998)



ÆS (1998)

1. **KeyExpansion** – round keys are derived from the cipher key using the ÆS key schedule. ÆS requires a separate 128-bit round key block for each round plus one more.
2. **Initial round** key addition:
    2.1 **AddRoundKey** – each byte of the state is combined with a byte of the round key using bitwise xor.
3. 9, 11 or 13 rounds:
    3.1 **SubBytes** – a non-linear substitution step where each byte is replaced with another according to a lookup table.
    3.2 **ShiftRows** – a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.
    3.3 **MixColumns** – a linear mixing operation which operates on the columns of the state, combining the four bytes in each column.
    3.4 **AddRoundKey**
4. **Final round** (making 10, 12 or 14 rounds in total):
    4.1 **SubBytes**
    4.2 **ShiftRows**
    4.3 **AddRoundKey**

ÆS (1998)

```
from Crypto.Cipher import AES  # pip install pycryptodome

key = b'Random thirty-two bytes key !!!!'
msg = b'A text of 32 bytes to encrypt !!'

cipher = AES.new(key, AES.MODE_CBC)
ciphertext = cipher.encrypt(msg)
# Out: b"1\\\xd4\xa1Oz'\xccL\xcb\xce\x1b\xc0\x04h....

uncipher = AES.new(key, AES.MODE_CBC, iv=cipher.iv)
plaintext = uncipher.decrypt(ciphertext)
# Out: b'A text of 32 bytes to encrypt !!'
```

¿Como probar la criptografía?

1. Buscar **oráculo**.
2. Explorar **criptografía casera**.
3. **Modificar un número**.
4. Analizar el **tiempo** y el **contenido** de las respuestas.
5. Hacer la **retroingeniería** del algoritmo.

N.	Parámetro	Valor
1	Scheme	ÆS
2	Mode	**ECB**
3	Bytes	16
4	Pad	PKCS7
5	Salt	**N/A**

Criptografía simétrica: recordar

Advanced Encryption Standard

1. **No implementar criptografía si mismo**.
2. Utilizar **ÆS** como cifrado de bloque.
3. Utilizar **CBC** como modo de encripción.
4. Esconder la clave, es decir no almacenarla.

Sesión 3: Asimétrica

(Módulo 2: Criptografía)

¿Qué aporta la asimétria?

Permite cifrar mensajes de manera que solo el propietario de la clave privada pueda descifrarlos.

1. **Envió seguro de clave**
2. Certificados digitales
3. Firmas
4. Encriptación para destinatarios especificos

Además ofrece de todas la posibilidades que ofrece la criptografía simétrica.

Algoritmos asimétricos

Algoritmo	Fecha	Descripción
Función unidireccional	1874	Formalización del problema de factorización
Diffie-Hellman	1976	Basado en logaritmo discreto
RSA	1977	Basado en la factorización de números grandes
ECC	1985	Basado en curvas elípticas
DSA	1991	Basado en logaritmo discreto

Índice temático

V • T • E	Public-key cryptography	
Algorithms	**Integer factorization**	Benaloh • Blum–Goldwasser • Cayley–Purser • Damgård–Jurik • GMR • Goldwasser–Micali • Naccache–Stern • Paillier • Rabin • RSA • Okamoto–Uchiyama • Schmidt–Samoa
	Discrete logarithm	BLS • Cramer–Shoup • DH • DSA • ECDH (X25519 • X448) • ECDSA • EdDSA (Ed25519 • Ed448) • ECMQV • EKE • ElGamal (signature scheme) • MQV • Schnorr • SPEKE • SRP • STS
	Lattice/SVP/ CVP/LWE/SIS	BLISS • Kyber • NewHope • NTRUEncrypt • NTRUSign • RLWE-KEX • RLWE-SIG
	Others	AE • CEILIDH • EPOC • HFE • IES • Lamport • McEliece • Merkle–Hellman • Naccache–Stern knapsack cryptosystem • Three-pass protocol • XTR • SQIsign
Theory	Discrete logarithm cryptography • **Elliptic-curve cryptography** • Hash-based cryptography • Non-commutative cryptography • RSA problem • Trapdoor function	
Standardization	CRYPTREC • IEEE P1363 • NESSIE • NSA Suite B • CNSA • Post-Quantum Cryptography	
Topics	Digital signature • OAEP • Fingerprint • PKI • Web of trust • Key size • Identity-based cryptography • Post-quantum cryptography • OpenPGP card	

RSA (1977)

- Desrollado por **R**ivest, **S**hamir y **A**dleman en 1977
- Basado en la dificultad de factorizar el producto de dos números primos
- Las claves son números primos >= 2048 bits (3×10^{616})

RSA: Descripción

- **Fundamento**: La seguridad de la criptografía asimétrica se basa en problemas matemáticos complejos, como la factorización de números grandes.
- **Generación de números primos**: Se eligen dos números primos grandes, (p) y (q).
- **Producto**: Se calcula $(N = p \times q)$, que se utiliza como parte de la clave pública.
- **Resultado**: Solo la persona que conoce (p) puede factorisar (N) y reducir así reducir la complejidad de ciertos problemas matematicos de manera exponencial.

RSA: Generación de claves

1. **Elegir dos números primos (privados) (p y q):** $(p = 61)$ y $(q = 53)$
2. **Calcular el producto (N) :** $(N = p \times q = 61 \times 53 = 3233)$
3. **Calcular la función totiente ($\phi(N)$):** $(\phi(N) = (p-1)(q-1) = 60 \times 52 = 3120)$.
4. **Elegir una clave pública (e) menor que $\phi(N)$ que es coprima con $\phi(N)$:** $(e = 17)$.
5. **Calcular la clave privada (d), el inverso multiplicativo de (e) módulo** $(\phi(n))$: $(d = 2753)$ porque $2753 \times 17 \equiv 1 \pmod{3120}$

RSA: Cifrado y descifrado

- **Cifrado**: Para cifrar un mensaje (m), se utiliza la clave pública (N y e): $c \equiv m^e \pmod{N}$
- **Descifrado**: Para descifrar el mensaje (c), se utiliza la clave privada (N y d): $m \equiv c^d \pmod{N}$

RSA: Ejemplo demostrativo

Número	Verbo	Descripción
p = 61	Elejir	1.er n.º primo privado
q = 53	Elejir	2.º n.º primo privado
N = 3233	Calcular	producto ($p \times q$)
e = 17	Elejir	exponente público
d = 2753	Calcular	exponente privado ($d \times e \equiv 1 \pmod{\phi(N)}$)
m = 42	Elejir	mensaje
c = 2557	Calcular	cifrado ($m^d \equiv N$)

Clave RSA: Test de primalidad

1. División por tentativa
2. Hipótesis China (1860)
3. Test de primalidad de Fermat (1613) (rosetacode) => pseudoprimos
4. Test de primalidad de Miller-Rabin (1980) (rosetacode)

Pseudoprimo (Fermat)

Se dice que «n» es pseudoprimo respecto la base «b» si es **compuesto** y además verifica la congruencia.

$$b^{n-1} \equiv 1 \pmod{n}$$

Probablemente primos (Miller)

Se dice que «N» es probable primo fuerte respecto la base «b» si cumple alguna de las siguientes relaciones de congruencia.

$$N^d \equiv 1 \pmod{n}$$
$$a^{2^r \cdot d} \equiv -1 \pmod{n} \text{ para algún } 0 \leq r < s$$

Probablemente primo (Miller)

```
import sys, random
def is_probably_prime(p):
    # p => (2**s | d) + 1
    def shift(s, d): return (s, d) if (d % 2 != 0) else
     ↪  shift(s+1, d>>1)
    s, d = shift(0, N-1)

    # Test p on one base
    def trial_composite(a):
        if pow(a, d, p) == 1: return False
        return all(not pow(a, 2**i * d, p) == p - 1 for i in
         ↪  range(s))

    return all(not trial_composite(random.randrange(2, p)) for _
     ↪  in range(8))
```

RSA: Ejemplo práctico

```
# 1. Generar clave privada
openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:2048

# 2. Extraer clave pública
openssl rsa -pubout -in private_key.pem -out public_key.pem

# 3. Crear archivo de mensaje
echo "Este es un mensaje secreto." > message.txt

# 4. **Cifrar el mensaje
openssl pkeyutl -encrypt -inkey public_key.pem -pubin -in message.txt -out
 ↪  encrypted_message.bin

# 5. Descifrar el mensaje
openssl pkeyutl -decrypt -inkey private_key.pem -in encrypted_message.bin -out
 ↪  decrypted_message.txt

# 6. Verificar el mensaje descifrado
cat decrypted_message.txt
```

RSA: Ejemplo real

```
ssh-keygen
ssh-copy-id -p 8022 u0_a103@192.168.1.84
ssh u0_a103@192.168.1.84  # Thanks
```

Evita implementar tu propia criptografía.

Criptografía asimétrica: recordar

1. Sirve para enviar claves.
2. Utiliza dos claves: pública y privada.
3. Facilita la autenticación mediante firmas digitales y certificados.
4. Proporciona encriptación específica para destinatarios.
5. Basada en problemas matemáticos complejos, como la factorización.

Sesión 4: Claves

(Módulo 2: Criptografía)

Definición de clave

Las claves son secuencias de bits utilizadas en algoritmos de cifrado. Su gestión es crucial para la seguridad de los sistemas criptográficos.

Problematicas

Aspecto	Descripción
Creación	Las claves deben generarse con alta entropía para ser difíciles de adivinar.
Almacenamiento	Las claves deben guardarse de forma segura en hardware protegido.
Distribución	La distribución de claves debe ser controlada para evitar compromisos.
Rotación	Rotar las claves periódicamente reduce el riesgo de exposición.

Generación de aleatorio

N.	Técnica	Limitación
1	Rodar la cabeza sobre el teclado	Dependiente del humano
2	`echo $RANDOM`	Genera números con baja entropía
3	`cat /dev/urandom`	Entropía no garantizada
4	`openssl rand 1000000000`	Sin limitaciones conocidas

```
for i in {1..13}; do printf '\\%02X' $((RANDOM%256)); done  #
 ↪  One-time pad key
```

Almacenamiento de clave

1. Almacenar el cifrado o hash de la clave, no el texto plano
2. Almacenar claves en hardware seguro
3. Utilizar salting y hashing
4. Limitar el acceso a las claves
5. Utilizar entornos separados

Clave: recordar

1. Utilizar claves con bastante entropia (16 bytes).
2. Almacenar solo el *hash* de las claves.
3. Utilizar sal.
4. Rotar las claves periódicamente.
5. Implementar autenticación multifactor.
6. Limitar el acceso a las claves.

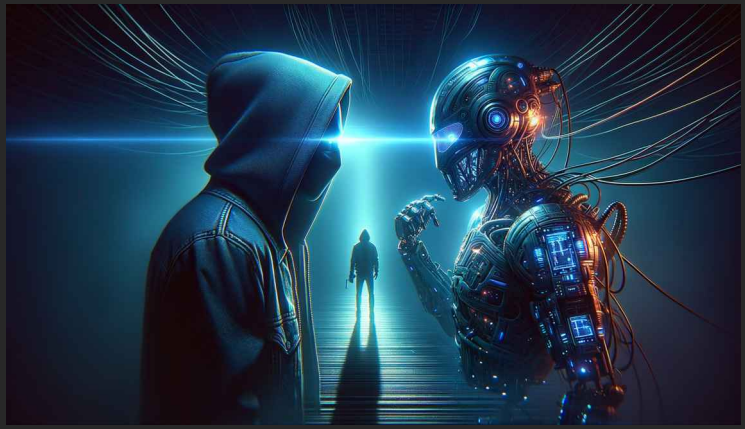
Criptografía: otros temas

1. Función hash
2. Ataque de colisión
3. Arquitectura
4. Criptografía en casa
5. Función de generación de clave
6. Criptografía cuántica
7. Algorithms
8. Cifrado homomórfico
9. Criptomonedas
10. Firmas digitales
11. Curvas elípticas
12. Análisis de vulnerabilidades criptográficas
13. Ataque de cumpleaños
14. Cifrado de flujo
15. Esteganografía

Módulo 3: Tecnología

Sesión 1: CAPTCHA

(Módulo 3: Tecnología)



¿Qué es un CAPTCHA?

Definición: CAPTCHA (Completely Automated Public Turing test to Tell Computers and Humans Apart) es una prueba diseñada para diferenciar entre humanos y programas automáticos (bots).

Función: Se utiliza para proteger servicios en línea, evitando que los bots realicen acciones no autorizadas, como el envío de spam o la creación de cuentas fraudulentas.

Tipos de CAPTCHA

Tipo	Descripción
Basados en texto	Presentan caracteres distorsionados que los humanos debe
Basados en imágenes	Requieren que el usuario seleccione imágenes que cumpla
Basados en preguntas	Preguntas de cultura general o lógica que los humanos pue
Basados en audio	Proporcionan una secuencia de números o letras que el us

Historia de los CAPTCHAs

El término CAPTCHA fue introducido en el año **2000** por Luis von Ahn, Manuel Blum, Nicholas J. Hopper de la Universidad Carnegie Mellon, y John Langford de IBM. Inicialmente, los CAPTCHAs consistían en que los usuarios debían **introducir correctamente un conjunto de caracteres distorsionados que aparecían en una imagen**.

La idea era que **las máquinas no pudieran resolver estos desafíos, mientras que los humanos sí**.

Con el tiempo, los algoritmos de reconocimiento óptico de caracteres (OCR) han mejorado, lo que ha llevado a la evolución de los CAPTCHAs. Se han desarrollado diferentes tipos, como los basados en **imágenes**, **preguntas lógicas**, **movimientos del ratón** y **audio**, para adaptarse a los avances en la inteligencia artificial y los métodos de resolución automatizada.

A pesar de su efectividad inicial, los CAPTCHAs continúan enfrentando desafíos en términos de accesibilidad y seguridad.

Pentest de CAPTCHA

1. Remover el CAPTCHA de la solicitud
2. Reutilisar el CAPTCHA de una solicitud
3. Utilizar Selenium para automatizar
4. Desarrollar scripts personalizados para imitar el comportamiento humano

Sesión 2: Sesiones

(Módulo 3: Tecnología)

Token de sesión

```
POST /login HTTP/2
Host: 0acf00aa032f8ac880f635d100ac0058.web-security-academy.net
Cookie: session=05wC9bolCVNhabfcl7HjeEgRT8gkDUts
Content-Length: 29
Content-Type: application/x-www-form-urlencoded
User-Agent: Tinmarino

username=§carlos§&password=
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

Token de sesión

```php
<?php
# Leer PHPSESSION (cookie) y establecer $_SESSION (PHP)
session_start();

# Leer variables de sesión arbitrarias
$user = $_SESSION['user'];

# Escribir variables de sesión arbitrarias
$_SESSION['active'] = True;
?>
```

Flujo: Autenticación (al final)

```php
<?php session_start();
$username = $_POST['username'] ?? exit();
$password = $_POST['password'] ?? exit();
$error = False;

// Validate credentials
if ($username !== $valid_username) { error = True; }
if ($password !== $passwords[$username]) { $error = True; }
if ($error){ header("Location: /login.php"); exit(); }

// Set magic session variable ONLY WHEN registered
$_SESSION['username'] = $username;
?>
```

Flujo: Autorisación (centralizada)

```php
<?php // File: session_utils.php included by utils.php
function authorize() { session_start();
    if (!isset($_SESSION['username'])) {
        exit('No username found in session.'); }
    return $_SESSION['username'];
} ?>
```

```php
<?php // File: dashboard.php
include 'session_utils.php';

$username = authorize(); // Get username or exit

// Use $username to display user-specific information
echo "Hello, " . htmlspecialchars($username)"; ?>
```

Sesión 3: Cookies

(Módulo 3: Tecnología)

¿Que es una cookie HTTP?

Una cookie HTTP es un pequeño bloque de datos que un servidor web crea y almacena en el dispositivo del usuario a través del navegador.

Se utiliza para recordar información sobre la sesión del usuario.

Ejemplo de cookie

Servidor

Cuando un servidor envía una cookie al navegador, lo hace a través del encabezado `Set-Cookie`.

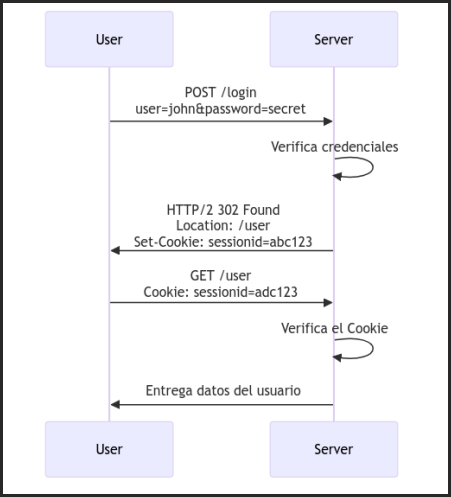
```
HTTP/1.1 200 OK
Content-Type: text/html
Set-Cookie: sessionid=abc123;
 ↪  Expires=Wed, 09 Jun 2025
 ↪  10:18:14 GMT; Path=/;
 ↪  Domain=example.com; Secure;
 ↪  HttpOnly
Set-Cookie: x-id=123
```

Cliente

Posteriormente a eso, cuando el cliente realiza una solicitud al servidor, las cookies se envían en el encabezado `Cookie`

```
GET /user HTTP/1.1
Host: example.com
Cookie: sessionid=abc123; x=123
```

Autenticación mediante Cookie



HTTP Redirect

```
HTTP/2 302 Found
Location: /user
Content-Type: text/html
Set-Cookie: sessionId=abc123; Expires=Wed, 09 Jun 2025 10:18:14
 ↪  GMT; Path=/; Domain=example.com; Secure; HttpOnly;
 ↪  SameSite=Lax

<html>
<head> <title>Redirecting...</title> </head>
<body> <p>If you are not redirected automatically, follow this
 ↪  <a href="/user">link</a>.</p> </body>
</html>
```

Atributos de Set-Cookie

```
Set-Cookie: sessionId=abc123; Domain=example.com;  Path=/;
 ↪  Max-Age=3600; Secure; HttpOnly; SameSite=Lax
```

Atributo	Ejemplo	Descripción
Nombre	`sessionId`	Nombre de la cookie.
Valor	`abc123`	Valor asociado a la cookie.
Dominio	`example.com`	Dominio para el cual es válida la cookie.
Ruta	`/`	Ruta en el servidor donde es válida la cookie.
Expira	`Wed ...`	Fecha y hora de expiración de la cookie.
Max-Age	3600 (1 hora)	Duración en segundos antes de que expire.

Atributos de seguridad de Set-Cookie

```
Set-Cookie: sessionId=abc123; Domain=example.com;  Path=/;
 ↪  Max-Age=3600; Secure; HttpOnly; SameSite=Lax
```

Atributo	Descripción
Secure	Solo se envía a través de HTTPS.
HttpOnly	No accesible desde JavaScript.
SameSite	Controla el envío en solicitudes de terceros.
SameSite=Strict	Solo se envía en el mismo sitio.
SameSite=Lax	Se envía en solicitudes de navegación seguras.
SameSite=None	Permite el envío en solicitudes de terceros.

Historia de las cookies

- **Origen**: Introducidas por Lou Montulli en **1994** mientras trabajaba en Netscape, las cookies fueron **diseñadas para resolver el problema de mantener el estado en HTTP**, un protocolo sin estado.
- **Evolución**: La especificación inicial fue desarrollada en respuesta a la necesidad de implementar carritos de compras virtuales, para permitir a los servidores recordar la información del usuario entre sesiones.
- **RFC 6265**: La especificación actual que define su uso en la web moderna.

¿Antes de las cookies?

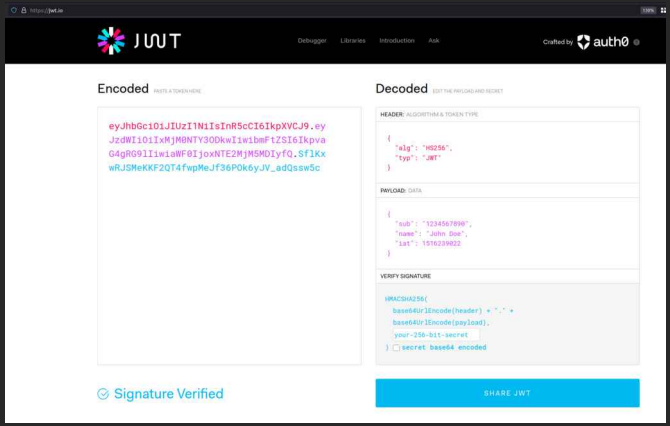
- **Campo de autoridad en la URL**: Usuario y contraseña en la URL.
  - `http://user:password@example.com`
- **Cadenas de consulta**: Identificadores de sesión en la URL.
  - `http://example.com/dashboard?sessionId=abc123`
- **Campos ocultos en formularios** Identificadores de sesión en formularios.
  - `<input type="hidden" name="sessionId" value="abc123">`
- **Autenticación básica**: Envío de usuario y contraseña en encabezados HTTP.
  - `Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=`
- **Gestión de sesiones en el servidor**: Estado de sesión mantenido en el servidor (por IP).

Vulnerabilidades y riesgos

- **Robo de cookies**: Las cookies pueden ser robadas a través de ataques de intermediario (man-in-the-middle) o XSS, lo que pood ia permitir a un actor de amenzas suplantar sesiones.
- **Secuestro de Sesiones**: Un actor de amenazas que obtiene una cookie de sesión puede acceder a la cuenta del usuario sin necesidad de credenciales.
- **Cookies de Terceros**: Utilizadas para el seguimiento, pueden comprometer la privacidad del usuario y son objeto de regulaciones como el GDPR en Europa.

Sesión 4: JWT: JSON Web Token

(Módulo 3: Tecnología)



JWT: Formato

JWT <= Header.Payload.Signature

Header

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9

{“typ”:“JWT”,“alg”:“HS256”}

Payload

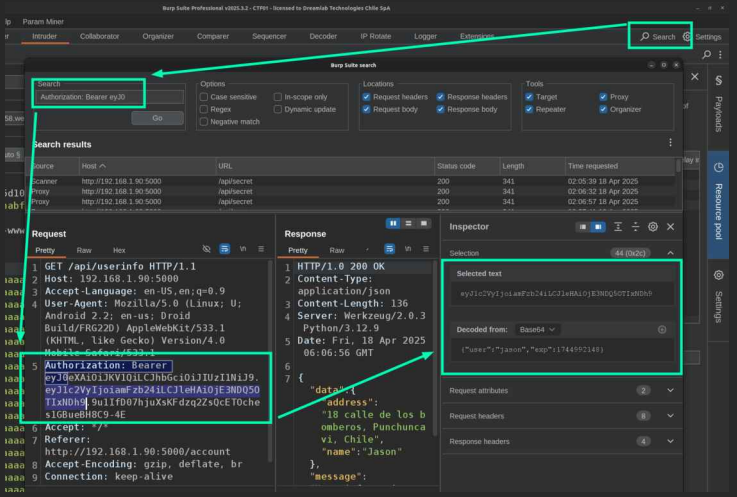
eyJ1c2VyIjoiamFzb24iLCJleHAiOjE3NDQ5OTIxNDh9

{“user”:“jason”,“exp”:1744992148}

Signature

9u1IfD07hjuXsKFdzq2ZsQcETOches1GBueBH8C9-4E

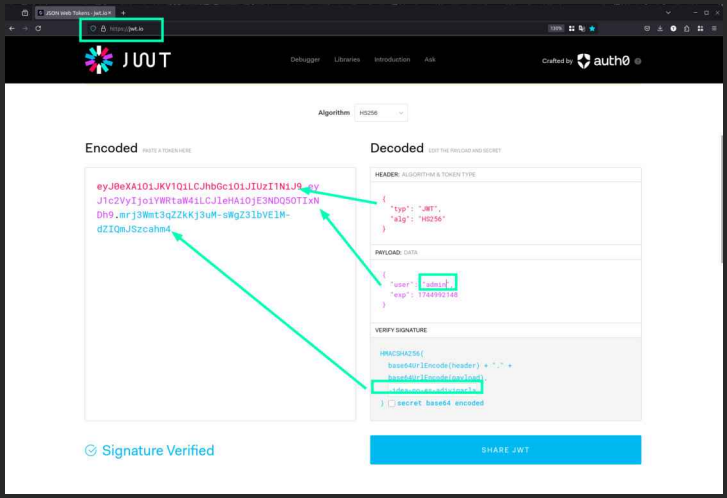
JWT: Buscar


Burp Suite Professional v2025.3.2 - CTF01 - licensed to DreamLab Technologies Chile SpA
Param Miner
Intruder
Collaborator
Organizer
Comparer
Sequencer
Decoder
IP Rotate
Logger
Extensions
Search
Settings
Burp Suite search
Search
Authorization: Bearer eyJ0
Go
Options
Case sensitive
Regex
Negative match
In-scope only
Dynamic update
Locations
Request headers
Request body
Response headers
Response body
Tools
Target
Repeater
Proxy
Organizer
Search results
Source
Host
URL
Status code
Length
Time requested
Scanner http://192.168.1.90:5000 /api/secret 200 341 02:05:39 18 Apr 2025
Proxy http://192.168.1.90:5000 /api/secret 200 341 02:06:32 18 Apr 2025
Proxy http://192.168.1.90:5000 /api/secret 200 341 02:06:57 18 Apr 2025
Payloads
Resource pool
Settings
Request
Pretty
Raw
Hex
1 GET /api/userinfo HTTP/1.1
2 Host: 192.168.1.90:5000
3 Accept-Language: en-US,en;q=0.9
4 User-Agent: Mozilla/5.0 (Linux; U; Android 2.2; en-us; Droid Build/FRG22D) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1
5 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyIjoiamFzb24iLCJleHAiOjE3NDQ5OTIxNDh9.9u1IfD07hjuXsKFdzq2ZsQcETOchesIGBueBH8C9-4E
6 Accept: */*
7 Referer: http://192.168.1.90:5000/account
8 Accept-Encoding: gzip, deflate, br
9 Connection: keep-alive
Response
Pretty
Raw
1 HTTP/1.0 200 OK
2 Content-Type: application/json
3 Content-Length: 136
4 Server: Werkzeug/2.0.3 Python/3.12.9
5 Date: Fri, 18 Apr 2025 06:06:56 GMT
6
7 {
"data": {
"address": "18 calle de los bomberos, Punchunca vi, Chile",
"name":"Jason"
},
"message":
Inspector
Selection
44 (0x2c)
Selected text
eyJ1c2VyIjoiamFzb24iLCJleHAiOjE3NDQ5OTIxNDh9
Decoded from:
Base64
{"user":"jason","exp":1744992148}
Request attributes
2
Request headers
8
Response headers
4

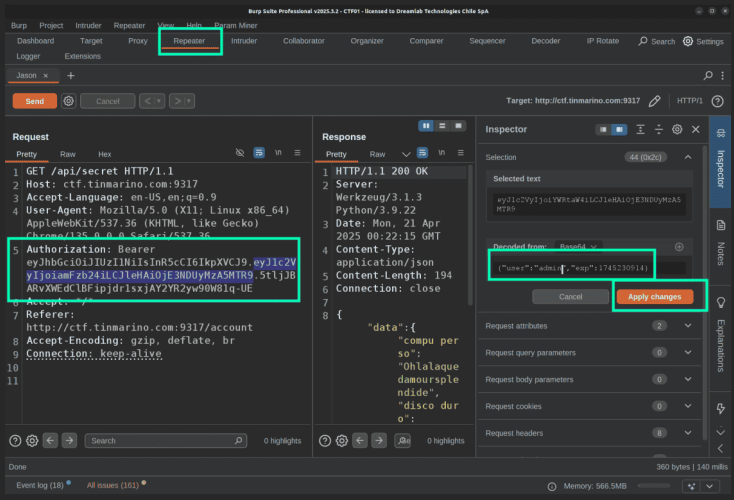
JWT: Buscar

```
function setAuthHeader() {
    const token = localStorage.getItem('token');
    if (token) {return{ 'Authorization': 'Bearer ' + token };}
    return {};
}
// ...

fetch('/login', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json',
        ...setAuthHeader(), },
    body: JSON.stringify({ username, password })
})
```

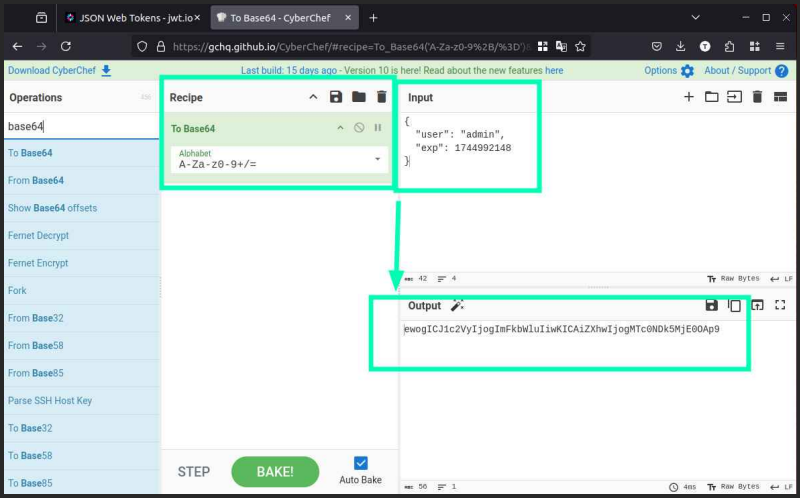
JWT: Leer y editar: JWT.io



JWT: Leer y editar: BurpSuite



JWT: Leer y editar: CyberChef



JWT: Leer y editar: Shell

```
echo -n 'eyJ1c2VyIjoiamFzb24iLCJleHAiOjE3NDQ5OTIxNDh9' | base64
 ↪  -d
# Out: {"user":"jason","exp":1744992148}


echo -n '{"user":"admin","exp":1844992148}' | base64
# Out: eyJ1c2VyIjoiYWRtaW4iLCJleHAiOjE4NDQ5OTIxNDh9
```

JWT: Implementar

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    # ...
    token = jwt.encode({
        'user': username, 'exp': datetime.datetime.utcnow()
            + datetime.timedelta(minutes=600)
    }, SECRET_KEY, algorithm='HS256')
    return jsonify({'token': token})

@app.route('/api/userinfo', methods=['GET'])
def userinfo():
    token = request.headers.get('Authorization').split(" ")[1]
    decoded = jwt.decode(token, SECRET_KEY,
 ↪  algorithms=['HS256'])
```

JWT: Explotar

1. Eliminar el JWT
2. Modificar la firma
3. Cambiar el algoritmo
4. Consultar PayloadAllTheThings JWT

Módulo 4 IDOR

¡Que es una IDOR?

Insecure Direct Object Reference (IDOR) es una vulnerabilidad de seguridad que permite a un atacante acceder a recursos o datos que no deberían estar disponibles para ellos, simplemente manipulando los parámetros de la solicitud.

Referencias directas a indices de base de datos, por ejemplo:

```
/user?id=6543
```

Problemáticas de las IDORs

1. Son **vulnerabilidades**.
2. Son **pervasivas**, errores difíciles de evitar.
3. **No** respetan el principio de **mínimos privilegios**.
4. Muchos **desarrolladores ignoran** la construcción de un sistema de control de acceso.
5. Pueden ser **codificadas** o encriptadas (Base64, hash, permutaciones).
6. Se pueden **explotar en cadena**.

Ejemplos

Referencias directas a:

Elemento	Ejemplo
Indices de base de datos	uc.cl/user?id?6543
Archivos estaticos	uc.cl/pdf/3122.pdf
Recursos de API	uc.cl/api/v1/resource/123
Llamada de funciones	uc.cl/foreach?callback=status

Ejemplos

1. Parametro GET
2. Parámetro POST
3. Cabecera (incluyendo JWT)

Casos

- Facebook
- Instagram
- Twitter

¿En práctica?

Simplemente cambiar sistmaticamente un ID.

Herramientas

- Repeater
- Intruder
- Extensión
- Param Miner