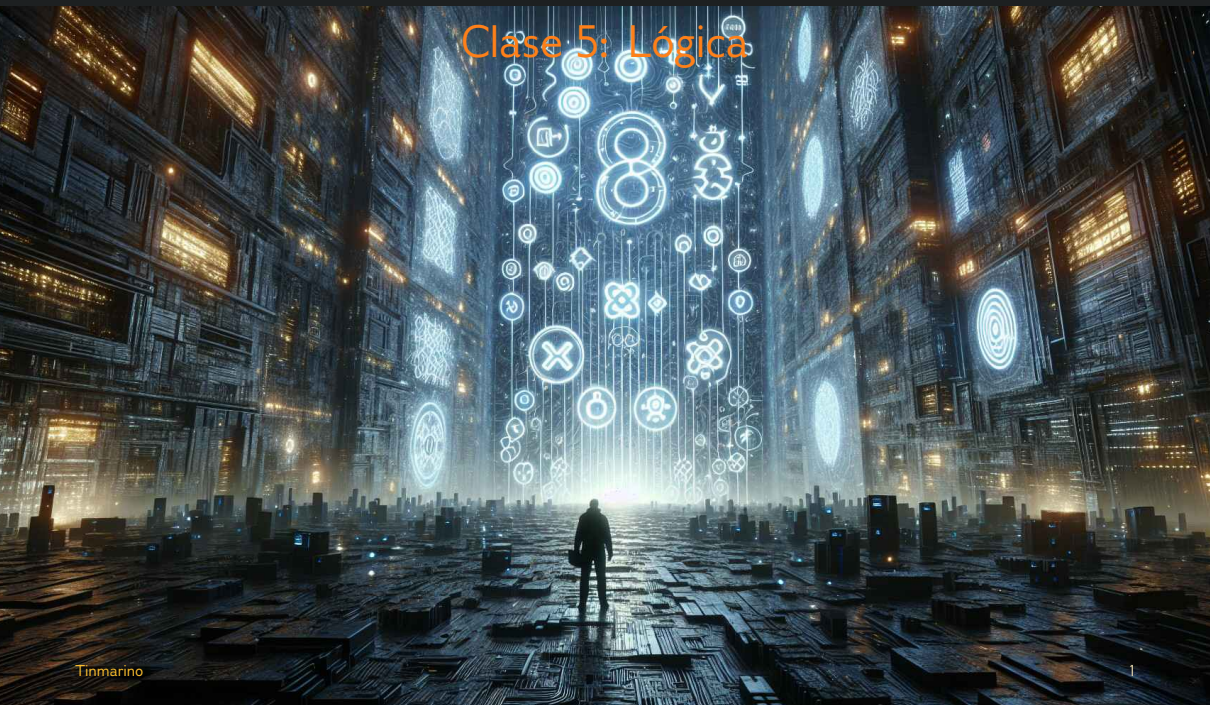
Pentest Web

Clase 5: Lógica



Clase 5: Lógica

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

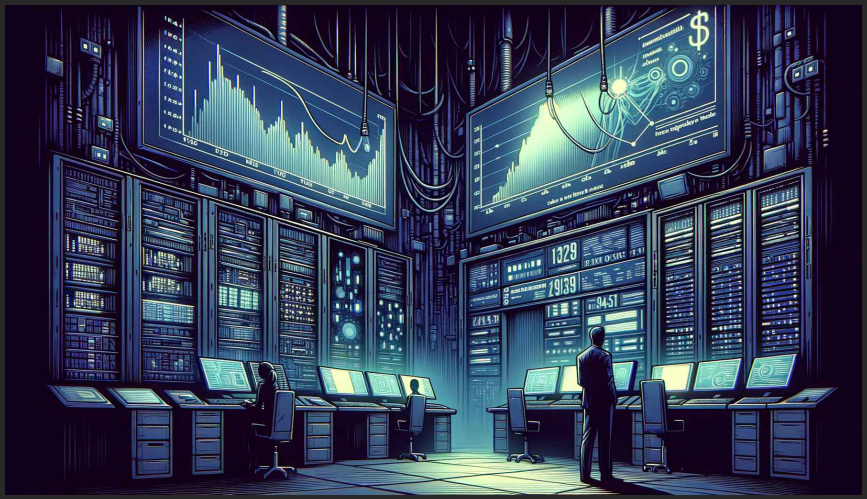
Clase 5: Lógica

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 5: Lógica

M	Nombre	Descripción
1	Negocio	Análisis de errores en función de los objetivos de negocio.
2	Diseño	Identificación de fallas en la arquitectura del sistema.
3	Flujo	Estudio de bypass en etapas críticas del proceso.
4	Aritmética	Errores en cálculos matemáticos y sus implicaciones.

Módulo 1: Negocio



Sesión 1: Marco

(Módulo 1: Negocio)

Nivel 2

(de los campeones)

Nivel 1

Hay que saber atacar para poder defender.

Nivel 2

(de los campeones)

Nivel 1

Hay que saber atacar para poder defender.

Nivel 2

Hay que saber defender para poder atacar bien.

Vulnerabilidades de lógica de negocio

- **Definición**: Fallos en el diseño de la aplicación que permiten un comportamiento no intencionado.
- **Otros nombres:** Errores de lógica, **vulnerabilidades de lógica de aplicación**.
- **Causas**: Resultan de suposiciones erróneas sobre el comportamiento del usuario.
- **Impacto**: Pueden ser explotadas para manipular funcionalidades legítimas.
- **Nota**: Las vulnerabilidades de lógica de negocio son difíciles de detectar porque no se manifiestan en el uso normal de la aplicación.
- **Consejo**: Realizar pruebas exhaustivas para identificar comportamientos inesperados.

¿Cómo surgen los errores de lógica?

- **Complejidad del sistema**: Sistemas complicados pueden llevar a malentendidos.
- **Suposiciones erróneas**: Los desarrolladores asumen cómo interactuarán los usuarios.
- **Validación inadecuada**: La falta de validación de entradas puede permitir ataques.
- **Consejo**: Como pentester, si no logras entender cómo funciona la aplicación, es probable que el desarrollador tampoco lo logré.
- **Consejo 2**: **Busca la complejidad**.

Sesión 2: Comunicacional

(Módulo 1: Negocio)

Chat

1. Cambiar el nombre del emetor.
2. Acceso a mensajes borrados.
3. Visualización de estado de conexión.
4. Acceso a chats privados.
5. Manipulación de archivos adjuntos.
6. Cambio de roles en grupos.
7. Envío de mensajes y/o notificaciones no deseadas.

Otras aplicaciones comunicacionales

1. Correo electrónico
2. Videoconferencias
3. Llamada de voz
4. Redes Sociales, foros, blogs, páginas web
5. Sistemas de gestión de proyectos
6. Plataformas de streaming
7. Aplicaciones de votos, encuestas y opiniones
8. Aprendisaje en linea
9. Publicidad
10. Desarrollo de software
11. Documentación
12. Monitoreo y control
13. Intercambio de archivos
14. Calendarios
15. Video juegos

Sesión 3: Humano

(Módulo 1: Negocio)

Gestión de salarios

1. Acceso a información sensible
2. Aumento de remuneración
3. Modificación del historial
4. Cambio de cuenta beneficiaria
5. Creación de empleados
6. Corrupción de impuestos
7. Modificación de monto disponible

Otras aplicaciones humanas

1. Remuneración
2. Vacaciones
3. Reclutamiento
4. Migración
5. Policia
6. Justicia
7. Salud
8. Registro Civil
9. Transporte
10. Carceles
11. Hospitales
12. Universidades
13. Banco de sangre
14. Inmobiliario
15. Asociativo, voluntariado

Sesión 4: Financiero

(Módulo 1: Negocio)

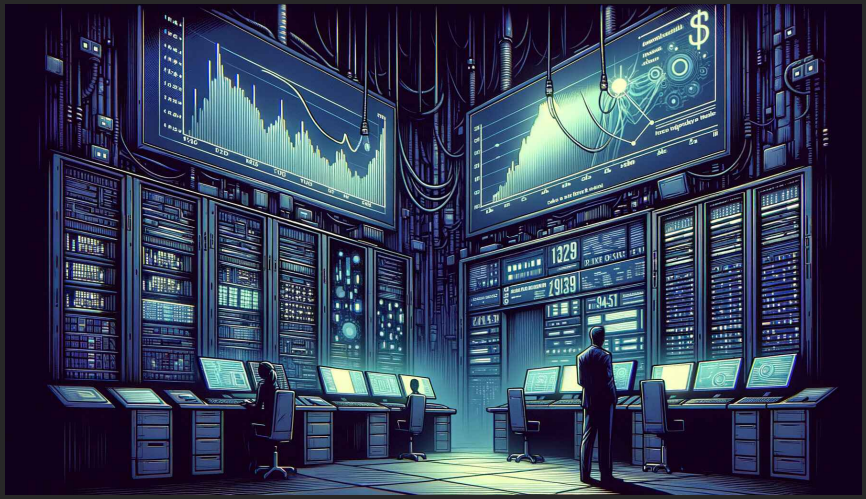
Vulnerabilidades en venta

1. Cambiar precio
2. Multiplicar productos
3. Validar compra sin pagar
4. Cambiar direción de otra compra
5. Eliminar producto
6. Manipular *sock* disponible
7. Obtener información de envio de otros usuarios
8. Generar descuento, bonos, ofertas
9. Desfigurar el sito

Vulnerabilidad en banca

1. **Extrær dinero**
2. Obtener cuenta de otro usuario (2FA)
3. Remover dinero
4. Modificar tasa de interez
5. Obtener datos de usuarios
6. Desfigurar el sitio
7. Crear un usuario zombie (inmortal)
8. **Todo lo que duele**

Módulo 2: Diseño



Sesión 1: Confianza

(Módulo 2: Diseño)

Confianza

De alguna forma todas las vulnerabilidades pueden atribuirse a un **exceso de confianza** en los usuarios remotos (fuera de la zone de confianza).

O a una falta de precauciones adecuadas.

Confianza: asunciones

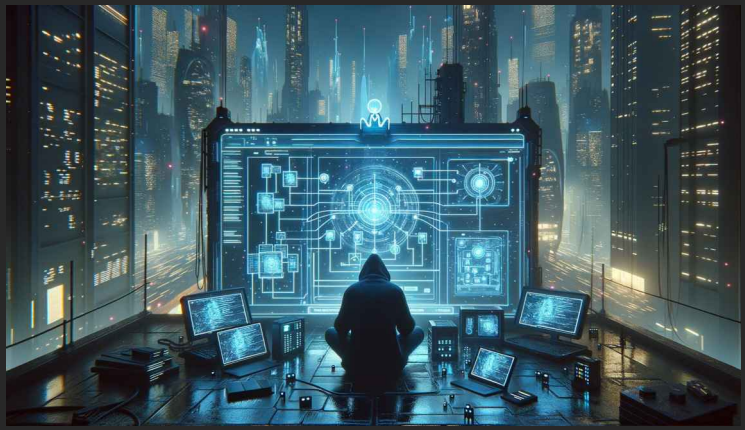
1. Uso de un navegador
2. Parámetros de HTTP dentro de lo legítimo
3. Flujo según JavaScript
4. Entradas dentro de lo rango y frecuencia “normales”
5. Sistemas internos no corumpibles
6. Authenticación necesaría
7. **Legitimidad de los usuarios**

Confianza: consecuencias

1. **¡HACKED!**
2. IDOR
3. **Parámetro de negocio accesible al usuario** (explicitamente o implicitamente)
4. Inyecciones
5. Evasión de controles (2FA)
6. Corrupción de logs
7. Divulgación masiva de datos
8. Desfiguración del sitio

Sesión 2: Validación

(Módulo 2: Diseño)



Validación

De alguna forma todas las vulnerabilidades pueden atribuirse a una mala validación de los parámetros remotos (fuera de la zona de confianza).

O a una falta de precauciones adecuadas.

Validación: lista de acceptación

Tipo	Ejemplo	Validación
Número	42	ENTERO y `0 < 42 < 100`
Alias	toto	STRING y `/^[a-z0-9]$/`
URL	http://uc.cl/user/32	STRING y `/^https:\/\/uc.cl\/[\w\.-]+$/`
Fecha	1990-01-01	STRING y `/^\d{4}-\d{2}-\d{2}$/` y 1800-01-01 < fecha válida < 2025-03-01
Imagén	logo.png	STRING y `/^[a-z0-0]\.png)$/`
Comentario	“Buen trabajo!”	STRING y `/.+/` y sanitización y `htmlspecialchars($comentario, ENT_QUOTES, 'UTF-8')`

Listas de denegaciones => problemas

Las listas de denegaciones son **imcompletas** y **complejas**.

- **Incompletitud**: No cubren todos los casos, lo que permite entradas no deseadas.
- **Complejidad**: Aumenta la dificultad de mantenimiento, rendimiento, integración, verificación mediante más patrones.

Listas de denegaciones => problemas

Las listas de denegaciones son **imcompletas** y **complejas**. Dan una **falsa sensación de seguridad**.

- **Incompletitud**: No cubren todos los casos, lo que permite entradas no deseadas.
- **Complejidad**: Aumenta la dificultad de mantenimiento, rendimiento, integración, verificación mediante más patrones.
- **Falsa sensación de seguridad**: Pueden dar una falsa impresión de protección.

Validación: sumideros

Inyección SQL

```
user_input = request.GET['user_id']
query = f"SELECT * FROM users WHERE id = {user_input}"
cursor.execute(query)
```

XSS

```
<div>Comentario del Usuario: {{ user_comment }}</div>
```

Inyección de comandos

```
os.system(f"ping {user_input}")
```

Validación: sumideros

Carga de archivos

```
file.save(f"/uploads/{file.filename}")
```

Desbordamiento de Búfer

```
char buffer[50];
strcpy(buffer, user_input);
```

Manejo Inadecuado de Tipos:

```
user_age = int(request.GET['age'])
```

Validaciones vulnerables

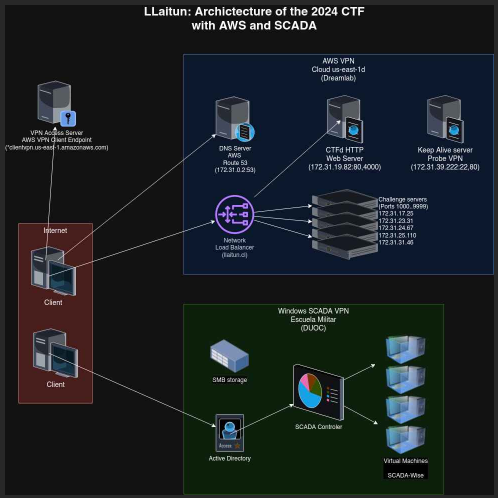
```
$file =~ s/\.\.\///g                # lfi-labs/LFI-14
$number =~ /^\d+$/                  # No verifica el rango
$url =~ /^http:\/\/[\w\.-]+$/)      # No verifica el dominio
$email =~ /^[\w\.-]+@[\w\.-]+$/     # No verifica el dominio
defined $id && $id > 0              # No verifica .. nada
$username =~ /^[a-zA-Z0-9_]+$/      # Sin limites de longitud
$phone =~ /^\d{10}$/                # No verifica que está en Chile
$email !~ /bademail\.com$/          # Lista negra
$username !~ /admin|root|superuser/
$ip !~ /192\.168\.1\.1|10\.0\.0\.1/
$file !~ /\.exe|\.bat|\.sh$/
```

Sesión 3: Depuración

(Módulo 2: Diseño)

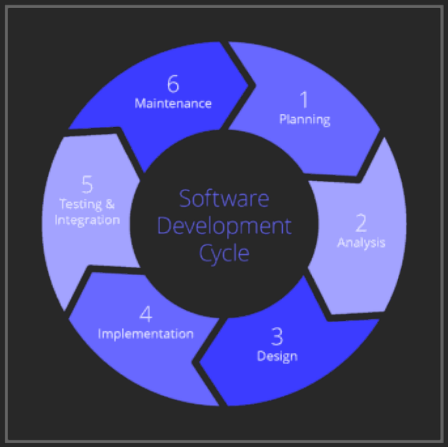
Funcionalidad de depuración

Servidor, Endpoint, Función, Parámetro



Seguridad en el proceso

La seguridad es un camino no un objetivo



A typical SDLC representation

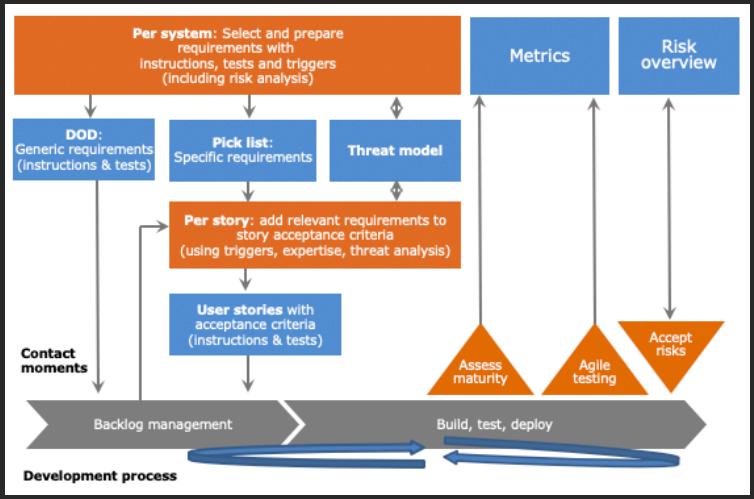
Nivel 3

(de los generales)

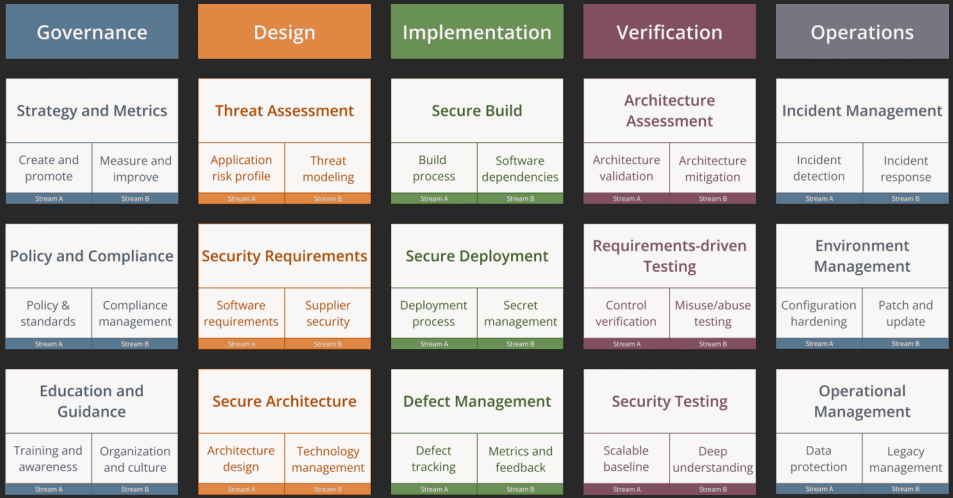
Donde manda capitán no manda marinero.

La ciberseguridad está **en los procesos**.

OWASP SAMM


Per system: Select and prepare requirements with instructions, tests and triggers (including risk analysis)
Metrics
Risk overview
DOD: Generic requirements (instructions & tests)
Pick list: Specific requirements
Threat model
Per story: add relevant requirements to story acceptance criteria (using triggers, expertise, threat analysis)
User stories with acceptance criteria (instructions & tests)
Contact moments
Assess maturity
Agile testing
Accept risks
Backlog management
Build, test, deploy
Development process

OWASP SAMM


Governance
Design
Implementation
Verification
Operations
Strategy and Metrics
Create and promote
Measure and improve
Stream A
Stream B
Threat Assessment
Application risk profile
Threat modeling
Stream A
Stream B
Secure Build
Build process
Software dependencies
Stream A
Stream B
Architecture Assessment
Architecture validation
Architecture mitigation
Stream A
Stream B
Incident Management
Incident detection
Incident response
Stream A
Stream B
Policy and Compliance
Policy & standards
Compliance management
Stream A
Stream B
Security Requirements
Software requirements
Supplier security
Stream A
Stream B
Secure Deployment
Deployment process
Secret management
Stream A
Stream B
Requirements-driven Testing
Control verification
Misuse/abuse testing
Stream A
Stream B
Environment Management
Configuration hardening
Patch and update
Stream A
Stream B
Education and Guidance
Training and awareness
Organization and culture
Stream A
Stream B
Secure Architecture
Architecture design
Technology management
Stream A
Stream B
Defect Management
Defect tracking
Metrics and feedback
Stream A
Stream B
Security Testing
Scalable baseline
Deep understanding
Stream A
Stream B
Operational Management
Data protection
Legacy management
Stream A
Stream B

Sesión 4: Dependencias

(Módulo 2: Diseño)

Asegurar la cadena de suministros

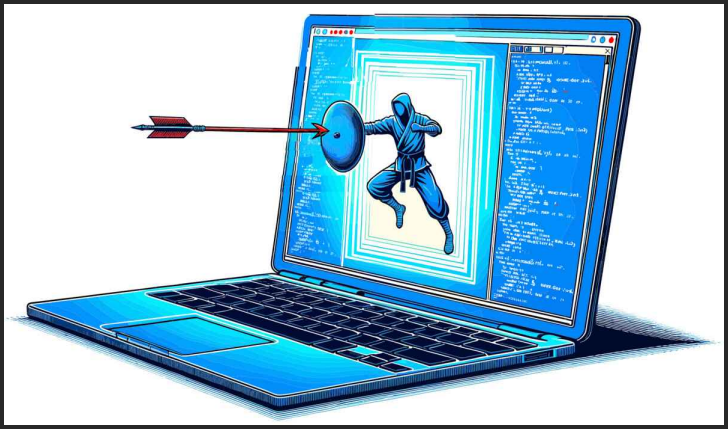
1. Dependencias de las dependencias
2. Proceso de integración
3. Versiones y parches
4. Entorno y compatibilidad

Pasos para asegurar las dependencias

1. Identificación de recursos críticos
2. Superficie de exposición
3. Control de versiones
4. Gestión de vulnerabilidades
5. Evaluacipon de proverdores
6. Proteción legal
7. Plan de respuesta a incidentes

Diseño: recordar

¡Blindate!



Módulo 3: Flujo

Sesión 1: Pasos

(Módulo 3: Flujo)

Ejemplo de registro con pasos

1. Resuelva el CAPTCHA
2. Ingresa tu RUT
3. Ingresa el número de serie del carnet
4. Ingresa tus datos
5. Confirma las condiciones
6. Ingresa tus datos bancarios

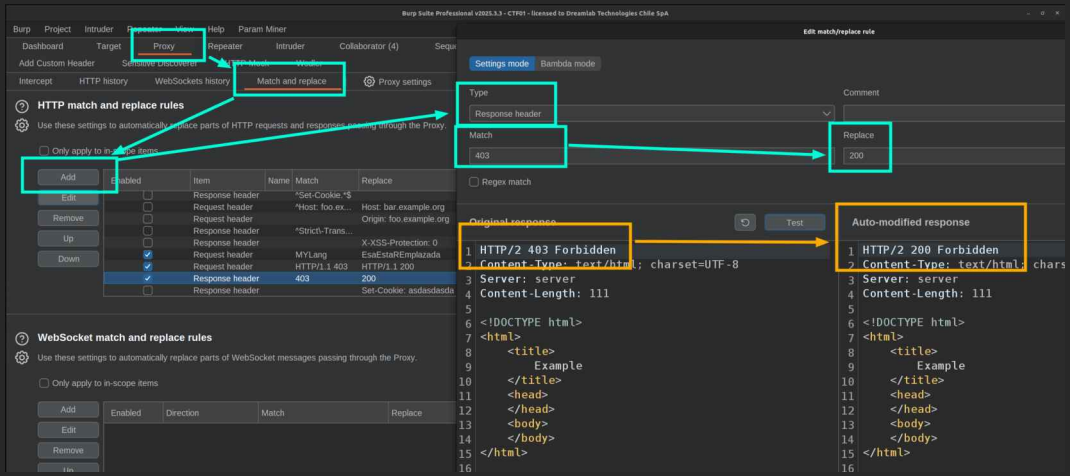
¿Como se almacenan los datos?

N.	Lugar	Vulnerable
1	Cliente	cambio de datos
2	Archivo	condición de carreras
3	PHP	denegación de servicio
4	Base de datos indexada	IDOR

Metodología de explotación

1. Recorrer todas las etapas del flujo legítimamente
   - 1.1 Anotar los *endpoints*.
   - 1.2 Anotar los datos.
   - 1.3 Utilizar el Organizer de Burp Suite (Ctrl + O).
2. Intentar saltar etapas
   - 2.1 Engañar al cliente.
   - 2.2 Enviar solicitudes directas.
   - 2.3 Remover los tokens de autenticación.
   - 2.4 Saltar etapas o volver atrás.
   - 2.5 Mezclar flujos.
   - 2.6 Buscar condiciones de carrera.
   - 2.7 Considerar qué hacer si nada de lo anterior funciona.

Lógica 200


Burp Suite Professional v2025.3.3 - CTF01 - licensed to Dreamlab Technologies Chile SpA
Burp Project Intruder Repeater View Help Param Miner
Dashboard Target Proxy Repeater Intruder Collaborator (4)
Add Custom Header Sensitive Discoverer
Intercept HTTP history WebSockets history Match and replace Proxy settings
HTTP match and replace rules
Use these settings to automatically replace parts of HTTP requests and responses passing through the Proxy.
Only apply to in-scope items
Add
Edit
Remove
Up
Down
Enabled Item Name Match Replace
Response header ^Set-Cookie.*$
Request header ^Host: foo.ex... Host: bar.example.org
Request header Origin: foo.example.org
Response header ^Strict\-Trans...
Response header X-XSS-Protection: 0
Request header MYLang EsaEstaREmplazada
Request header HTTP/1.1 403 HTTP/1.1 200
Response header 403 200
Response header Set-Cookie: asdasdasda
WebSocket match and replace rules
Use these settings to automatically replace parts of WebSocket messages passing through the Proxy.
Only apply to in-scope items
Add
Edit
Remove
Enabled Direction Match Replace
Edit match/replace rule
Settings mode Bambda mode
Type
Response header
Comment
Match
403
Replace
200
Regex match
Test
Auto-modified response
1 HTTP/2 403 Forbidden
2 Content-Type: text/html; charset=UTF-8
3 Server: server
4 Content-Length: 111
5
6 <!DOCTYPE html>
7 <html>
8 <title>
9 Example
10 </title>
11 <head>
12 </head>
13 <body>
14 </body>
15 </html>
16
1 HTTP/2 200 Forbidden
2 Content-Type: text/html; chars
3 Server: server
4 Content-Length: 111
5
6 <!DOCTYPE html>
7 <html>
8 <title>
9 Example
10 </title>
11 <head>
12 </head>
13 <body>
14 </body>
15 </html>
16

Ejemplo de mezcla de flujos

1. Resuelva el CAPTCHA
2. Entrar el RUT del atacante
3. Entrar el número de serie del atacante
4. Entrar el RUT de la victima mediante la solicitud del atacante (con los mismo Cookies)
5. Saltar a la etapa de los datos
6. Leer o editar los datos de la victima

Sesión 2: Carrera

(Módulo 3: Flujo)

Carrera

Las condiciones de carrera son vulnerabilidades que ocurren cuando el comportamiento de un sistema depende del orden en que se ejecutan las operaciones concurrentes. Esto puede llevar a resultados inesperados, ya que múltiples procesos pueden interferir entre sí al acceder a recursos compartidos.

Por ejemplo, en un banco, si un cliente intenta retirar $80 de una cuenta con un saldo de $100 mientras otro proceso intenta retirar $50 al mismo tiempo, ambos retiros pueden ser aprobados sin actualizar el saldo. Esto puede resultar en un saldo negativo, representando una vulnerabilidad crítica en la lógica de la aplicación.

Carrera

Las condiciones de carrera son solo presentes en sistemas concurrentes.

Son muy difíciles de detectar, tanto para el desarrollador como para el pentester.

Pueden ocurrir cuando múltiples hilos acceden al mismo recurso de manera no atómica.

Escritura concurrente a archivos

```
seq 2000 | xargs -P100 -I! bash -c \
 '{ printf "Long line %06d | " {1..2000}; echo; } >> /tmp/file'
```

- **Descripción**: Acceso simultáneo a archivos de registro o configuración.
- **Escenario**: Dos procesos intentan escribir en el mismo archivo de registro al mismo tiempo.
- **Explotación**: La información puede corromperse o perderse, lo que dificulta la auditoría y el seguimiento de transacciones.

Ejemplos

- Registro: registrarse al mismo tiempo => mismo ID
- Reserva: doble reserva
- Inventario: doble retiro

Ejemplos

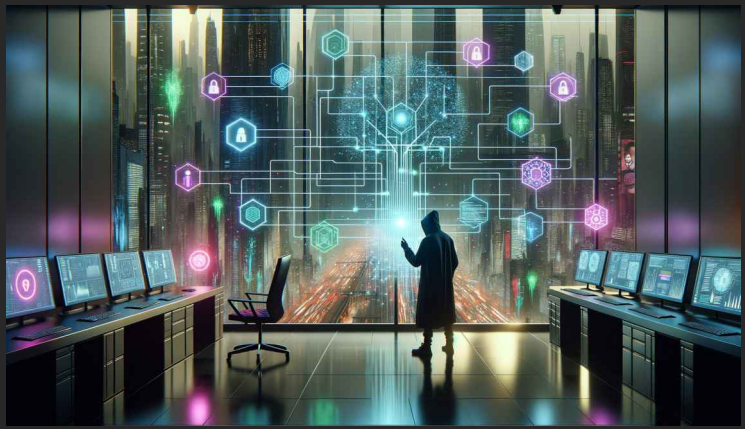
- Registro: registrarse al mismo tiempo => mismo ID
- Reserva: doble reserva
- Inventario: doble retiro

Cuando la implementación no es atómica.

1. Verificar
2. <—- Aquí se modifica el recurso
3. Utilizar

Sesión 3: Decisión

(Módulo 3: Flujo)



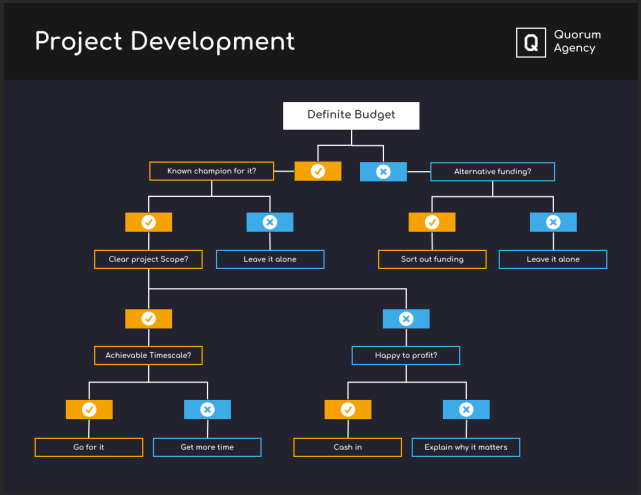
¿Qué es un árbol de decisiones?

Un árbol de decisiones es una representación gráfica de decisiones y sus posibles consecuencias.

Permite visualizar cómo las decisiones de una aplicación pueden ser manipuladas.

Comprender el árbol de decisiones ayuda a identificar y explotar las vulnerabilidades de lógica de negocio.

Árbol de decisiones



Ejemplo de árbol de decisiones

- **Escenario**: Proceso de compra en un e-commerce.
- **Decisiones**:
  - ¿El usuario está autenticado?
  - ¿El producto está en stock?
  - ¿El pago es válido?
- **Consecuencias**: Acceso a la compra, rechazo de la transacción, etc.

Ejemplo de reconocimiento

- **Técnicas de Reconocimiento**:
  - Análisis de flujos de trabajo.
  - Revisión de mensajes de error.
- **Objetivo**: Identificar cómo se toman las decisiones en la aplicación.
- **Decisiones**:
  - ¿El usuario tiene saldo suficiente?
  - ¿El usuario es el propietario de la cuenta?
- **Explotación**: Un atacante puede intentar cambiar el saldo para eludir restricciones.

Sesión 4: Errores

(Módulo 3: Flujo)

Orden de cláusulas

Acceso al sueldo de cuenta corriente arbitraria con dicotomía de errores en deposito (autenticado)). (medio)

Al hacer un deposito desde una cuenta, el servidor verifica en orden:

1. Si la cuenta existe
2. Si la cuenta tiene el saldo suficiente
3. Si la cuenta pertenece al usuario registrado

Por razones de seguridad, se recomienda invertir el orden de las cláusulas. Al realizar una solicitud mediante un número de cuenta que no le pertenece, un actor de amenazas podría exfiltrar información sensible, como la existencia de la cuenta y su saldo.

Dicotomía de errores

Es posible determinar el saldo de una cuenta arbitraria. Comenzamos con un monto inicial, por ejemplo, 20,000 pesos.

1. Si el mensaje de error indica “cuenta no le pertenece”, incrementamos el monto.
2. Si el mensaje es “monto insuficiente”, disminuimos el monto.
3. Detenemos el proceso cuando las variaciones en el saldo se vuelven mínimas.

Eficientemente por dicotomía (es decir, log)

```
while high - low > epsilon:
    mid = (low + high) / 2
    print(f'\nTarget [{i}]: Account: {account}: Amount {mid}')
    if query_amount(i, account, mid): low = mid
    else: high = mid
```

Retro-oráculo mediante error

Un retro-oráculo en los mensajes de error del proceso de compra [...] facilita la ingeniería inversa de una criptografía débil del parametro [encryptedRut]. (alto)

```
POST /recharge
Cookie: session=e57826d1-5652-49bc-8913-1c9229162998
Content-Type: application/json

{ "encryptedRut": "QOUwMUVCOTQxOEZGMzJGRUEzOEY2OQ==",
  "amount": 00}
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{ "error": "validación error para rut 11.111.111-1, error en
 ↪  monto"}
```

Formato de mensajes

Request

Pretty Raw Hex

```
1 POST /api/person HTTP/1.1
2 Host: ctf.tinmarino.com:9315
3 Content-Type: application/json
4 Content-Length: 7
5
6 {
      "a":1
  }
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 500 INTERNAL SERVER ERROR
2 Server: Werkzeug/3.1.3 Python/3.9.22
3 Date: Sun, 27 Apr 2025 04:55:04 GMT
4 Content-Type: application/json
5 Content-Length: 614
6 Connection: close
7
8 {
      "error":
      "Validation error: [{\"type\":\"missing\",\"loc\":[\"id\"],\"msg\"
      :\"Field required\",\"input\":{\"a\":1},\"url\":\"https://errors.p
      ydantic.dev/2.11/v/missing\"},{\"type\":\"missing\",\"loc\":[\"gen
      der\"],\"msg\":\"Field required\",\"input\":{\"a\":1},\"url\":\"ht
      tps://errors.pydantic.dev/2.11/v/missing\"},{\"type\":\"missing\",
      \"loc\":[\"region\"],\"msg\":\"Field required\",\"input\":{\"a\":1
      },\"url\":\"https://errors.pydantic.dev/2.11/v/missing\"},{\"type\
      ":\"missing\",\"loc\":[\"is_adult\"],\"msg\":\"Field required\",\"
      input\":{\"a\":1},\"url\":\"https://errors.pydantic.dev/2.11/v/mis
      sing\"}]"
  }
9
```

Módulo 4: Aritmética

Sesión 1: Redondeo

(Módulo 4: Aritmética)

Redondeo hacía par (sin sesgo)

```
printf '%0.f\n' 1.5  # Out: 2
```

Redondeo hacía par (sin sesgo)

```
printf '%0.f\n' 1.5  # Out: 2
```

```
printf '%0.f\n' 2.5  # Out: 2
```

Redondeo hacía par (sin sesgo)

```
printf '%0.f\n' 1.5  # Out: 2
```

```
printf '%0.f\n' 2.5  # Out: 2
```

Alice ($10) envia $2.5 a Bob ($11)

Redondeo hacía par (sin sesgo)

```
printf '%0.f\n' 1.5  # Out: 2
```

```
printf '%0.f\n' 2.5  # Out: 2
```

Alice ($10) envia $2.5 a Bob ($11)

Persona	Inicial	Cambio	Final	Redondeo
Alice	10	-2.5	7.5	8
Bob	11	+2.5	13.5	14
Suma	21	0	21	22

Tipos de redondeos

Directo

- Arriba
- Abajo
- Hacia cero
- Lejos de cero

Cercano

- **Arriba**
- Abajo
- Hacia cero
- Lejos de cero
- **Hacia par**
- Hacia impar

Otros

- Con semilla
- Aleatorio
- Logaritmíco
- Hacía precisión mínima

Tipos de redondeos

Comparison of approaches for rounding to an integer [edit]

Value	Functional methods											Randomized methods					
	Directed rounding				Round to nearest						Round to prepare for shorter precision	Alternating tie-break		Random tie-break		Stochastic	
	Down (toward −∞)	Up (toward +∞)	Toward 0	Away From 0	Half Down (toward −∞)	Half Up (toward +∞)	Half Toward 0	Half Away From 0	Half to Even	Half to Odd		Average	SD	Average	SD	Average	SD
+2.8	+2	+3	+2	+3	+3	+3	+3	+3	+3	+3	+2	+3	0	+3	0	+2.8	0.04
+2.5												+2.505	0	+2.5	0.05	+2.5	0.05
+2.2					+2	+2	+2	+2	+2	+2		+2	0	+2	0	+2.2	0.04
+1.8																+1.8	0.04
+1.5	+1	+2	+1	+2	+1	+1	+1	+1	+1	+1	+1	+1.505	0	+1.5	0.05	+1.5	0.05
+1.2												+1	0	+1	0	+1.2	0.04
+0.8																+0.8	0.04
+0.5	0	+1	0	+1	0	0	0	0	0	0		+0.505	0	+0.5	0.05	+0.5	0.05
+0.2												0	0	0	0	+0.2	0.04
−0.2																−0.2	0.04
−0.5	−1	0	−1	−1	−1	−1	−1	−1	−1	−1	−1	−0.495	0	−0.5	0.05	−0.5	0.05
−0.8												−1	0	−1	0	−0.8	0.04
−1.2																−1.2	0.04
−1.5	−2	−1	−1	−2	−2	−2	−2	−2	−2	−2		−1.495	0	−1.5	0.05	−1.5	0.05
−1.8												−2	0	−2	0	−1.8	0.04
−2.2																−2.2	0.04
−2.5	−3	−2	−2	−3	−3	−3	−3	−3	−3	−3	−2	−2.495	0	−2.5	0.05	−2.5	0.05
−2.8												−3	0	−3	0	−2.8	0.04

Redondeo en cambio de divisas

USD	round(USD)	CLP	round(CLP)	Hacker	Taza
1	1	969	969	Analiza	**969**
0.01	0.01	9.69	10	Vende USD	1000
0.02	0.02	19.38	19	Compra dolares	950
0.0144	0.01	**14**	14	Vende dólares	1400
0.0051	0.01	**5**	5	Compra dólares	500

Redondeo en cambio de divisas

USD	round(USD)	CLP	round(CLP)	Hacker	Taza
1	1	969	969	Analiza	**969**
0.01	0.01	9.69	10	Vende USD	1000
0.02	0.02	19.38	19	Compra dolares	950
0.0144	0.01	**14**	14	Vende dólares	1400
0.0051	0.01	**5**	5	Compra dólares	500

```
sol = 5 / 969; print(sol); print(round(sol*100)/100);
# Out: 0.00515 \n 0.01
```

¿Por qué es **pervasivo**?

La facilidad para robar dinero mediante redondeo radica en que las **rutinas informáticas** a menudo buscan **redondeos sin sesgo**.

Un actor **malicioso**, por su parte, buscará sistemáticamente el desplazamiento que **le resulte más conveniente**, a **diferencia** de lo que dictaría el **azar estadístico**.

Al igual que discutimos anteriormente, Internet fue diseñado inicialmente para facilitar el intercambio de fotos de gatos entre amigos, en lugar de garantizar transferencias seguras de dinero entre desconocidos.

De manera similar, la aritmética computacional ha sido desarrollada principalmente para organizar datos estadísticos, en lugar de realizar cálculos aritméticos exactos.

Sumisteros de las fallas de redondeo

Language	Código
JavaScript	`usd = Math.round(usd * 100) / 100`
Python	`usd = round(usd*100) / 100`
PHP	`round($usd, 2)`
Java	`usd = Math.round(usd * 100.0) / 100.0`
SQL	`usd DECIMAL(10, 2) DEFAULT`

Suelen presentarse en conjunto. Busca el Javascript estáticamente (mediante regex en el código fuente) y el SQL dinámicamente (mediante prueba y error).

Falacia clásica (1)

Eso solo pasa con los montos pequeños.

Falacia clásica (1)

Eso solo pasa con los montos pequeños.

También pasa con montos grandes.

```
sol=1_000_004 / 969; print(sol, round(sol*100)/100);
# Out: 1031.9958 1032.0
```

Sin embargo, se podría extrær un máximo de 0.005 dólares por transacción.

Falacia clásica (2)

Un robo de **medio centavo** (0.005) de dólares por transacción no tiene un impacto significativo.

Falacia clásica (2)

Un robo de **medio centavo** (0.005) de dólares por transacción no tiene un impacto significativo.

Cierto. Sin embargo, **cien millones** (100M) de transacciones de medio centavo (0.005) podrían tener un impacto significativo.

```
for i in {1..100000000}; do
  curl -X POST https://bancopenca.com/convert \
    -H "Content-Type: application/json" \
    -d '{"amount": 5, "from": "CLP", "to": "USD"}'
done

# Resultado: 480 mil dólares extraídos
# -- Ref: (969 - 500) * 0.01 * 100_000_000 / 969 = 484004
```

Falacia clásica (3)

En caso de que ocurra, lo detectaremos y lo bloquearemos.

Falacia clásica (3)

En caso de que ocurra, lo detectaremos y lo bloquearemos.

Lo reporté hace dos semanas y lo exploté el 3 de abril.

¿Pueden garantizarme que nadie más, aparte de mí, haya explotado esta vulnerabilidad?

Falacia clásica (...)

- 2FA, WAF, Firewall (defensa en profundida mal implementada).
- Los otros lo hacen igual (generalización apresurada).
- Siempre lo hicimos así (apelación a la tradición).
- Nadie me lo reportó (apelación a la autoridad).
- La explotación requiere autenticación; solicitaremos un reembolso (ignorancia del riesgo).
- Existen prioridades más relevantes (desviación).
- No hay evidencía, estadisticamente no generaría perdidas (causa falsa).

Falacia clásica (...)

- 2FA, WAF, Firewall (defensa en profundida mal implementada).
- Los otros lo hacen igual (generalización apresurada).
- Siempre lo hicimos así (apelación a la tradición).
- Nadie me lo reportó (apelación a la autoridad).
- La explotación requiere autenticación; solicitaremos un reembolso (ignorancia del riesgo).
- Existen prioridades más relevantes (desviación).
- No hay evidencía, estadisticamente no generaría perdidas (causa falsa).
- ...

Falacia clásica (...)

- 2FA, WAF, Firewall (defensa en profundida mal implementada).
- Los otros lo hacen igual (generalización apresurada).
- Siempre lo hicimos así (apelación a la tradición).
- Nadie me lo reportó (apelación a la autoridad).
- La explotación requiere autenticación; solicitaremos un reembolso (ignorancia del riesgo).
- Existen prioridades más relevantes (desviación).
- No hay evidencía, estadisticamente no generaría perdidas (causa falsa).
- ...

Lo que el defensor menosprecia, el atacante lo explota.

Reación esperada

- Gracias por informarme (empatía).
- No lo sabía (**humildad**, curiosidad).
- Al final, es un error de validación de entrada (atención al detalle).
- Implementaremos una validación más estricta (disciplina).

Reación esperada

- Gracias por informarme (empatía).
- No lo sabía (**humildad**, curiosidad).
- Al final, es un error de validación de entrada (atención al detalle).
- Implementaremos una validación más estricta (disciplina).

El hackeo es un estado de espíritu.

Redondeo: Recordar

- El redondeo es un proceso **complejo**.
- La representación de fracciones en binario presenta **dificultades**.
- El *casting* de tipos puede resultar en comportamientos **indefinidos**.
- **Cada división** puede resultar en un número fraccionario (con decimales).
- Nuestros antepasados implementaron primitivas de redondeo para evitar sesgos **en general**.
- El pentester busca **fuentes** que conducen a vulnerabilidades (sumisteros) relacionadas con la división y el redondeo, y luego prueba diferentes números de entrada para **obtener un resultado favorable** (para él).

Sesión 2: Flotantes

(Módulo 4: Aritmética)

Flotantes: Imprecisión

```
0.01 + 0.02 - 0.03
```

Flotantes: Imprecisión

```
0.01 + 0.02 - 0.03
```

```
# Out: 0.0
```

Flotantes: Imprecisión

```
0.01 + 0.02 - 0.03
```

```
# Out: 0.0
```

```
0.1 + 0.2 - 0.3
```

Flotantes: Imprecisión

```
0.01 + 0.02 - 0.03
```

```
# Out: 0.0
```

```
0.1 + 0.2 - 0.3
```

```
# Out: 5.55[...]e-17
```

Representación de punto flotante

Los números de punto flotante se representan en el hardware de la computadora como fracciones en base 2 (binarias).

- Decimal: $0.625 = \frac{6}{10} + \frac{2}{100} + \frac{5}{1000}$
- Binario: $0.101 = \frac{1}{2} + \frac{0}{4} + \frac{1}{8}$

Problema de precisión

- La mayoría de las fracciones decimales no pueden representarse exactamente como fracciones binarias.
- Los números de punto flotante decimales ingresados son solo aproximados por los números de punto flotante binarios almacenados en la máquina.

Metafora con base 10

Entendiendo con Base 10 - Ejemplo: $\frac{1}{3}$ aproximado en base 10: - $0.3, 0.33, 0.333, \ldots$ - No importa cuántos dígitos escribas, nunca será exactamente $\frac{1}{3}$.

Representación en Base 2 - De manera similar, 0.1 no puede representarse exactamente en base 2. - En base 2, $\frac{1}{10}$ es la fracción que se repite infinitamente: - $0.00011001100110011\ldots$

Flotantes: Subesbordamiento

```
1e-300
# Out: 1e-300

1e-300  * 1e-300
# Out: 0.0
```

Flotantes: Desbordamiento

```
1e300
# Out: 1e-300

1e300  * 1e300
# Out: inf
```

Sesión 3: Límites

(Módulo 4: Aritmética)

El infinito

El infinito es el número que, al sumarle uno, sigue siendo igual a sí mismo.

El infinito

El infinito es el número que, al sumarle uno, sigue siendo igual a sí mismo.

```
2e308   # Out: inf

2e308 - 2e308  # Out: nan

2e308 - 1e308 - 1e308 # Out: inf

# Referencia
import sys; sys.float_info.max
# Out: 1.8e+308  (approx)
```

Importancia de los límites

- **Verificación de Límites**: Cada número debe ser verificado con un límite **inferior y superior**. (2 verificaciones)
- **Riesgos**: No verificar límites puede llevar a errores críticos en cálculos y lógica de negocio.

Verificación de rango

low < amount < high

```python
def transfert(amount: int):
    # Clause: Check type
    # -- Explicit is better than implicit
    # -- Ref: python -c 'import this'
    if not isinstance(amount, int)
        return "Error: El monto ingresado no es un número entero válido."

    # Clause: Check low
    if amount < 0:
        return "Error: El monto no puede ser negativo."

    # Clause: Check high
    if amount > 1000:
        return "Error: El monto no puede superar los 1000 dólares."

    # ---
```

Debordamiento de enteros

```c
#include <stdio.h>

int main() {
    // Declarar el saldo inicial
    int balance = 100;
    printf("Saldo inicial:
    ↪ %d\n", balance);

    // Transferir 2.2G
    balance -= 2200000000;
    printf("Saldo después:
    ↪ %d\n", balance);

    return 0;
}
```

```
gcc underflow.c ; ./a.out
# Saldo inicial: 100
# Saldo después: 2094967396
```

Uno nunca es feliz más que en la felicidad que se da.

Dar es recibir.

(Abbé Pierre (un monje francés))

Lista de tipos

Bits	Name	Byte	SLimit	ULimit
8	char	1	127	255
16	short	2	32_767	65_535
32	long	3	2e9	4e9
64	long long	4	9.2e18	1.8e19

Lista de tipos

char: 8

Type	Explanation	Size (bits)	Format specifier	Range	Suffix for decimal constants
`bool`	Boolean type, added in C23.	1 (exact)	`%d`	`[false, true]`	—
`char`	Smallest addressable unit of the machine that can contain basic character set. It is an integer type. Actual type can be either signed or unsigned. It contains CHAR_BIT bits.[3]	≥8	`%c`	`[CHAR_MIN, CHAR_MAX]`	—
`signed char`	Of the same size as *char*, but guaranteed to be signed. Capable of containing at least the $[-127, +127]$ range.[3][a]	≥8	`%c` [b]	`[SCHAR_MIN, SCHAR_MAX]` [6]	—
`unsigned char`	Of the same size as *char*, but guaranteed to be unsigned. Contains at least the $[0, 255]$ range.[7]	≥8	`%c` [c]	`[0, UCHAR_MAX]`	—

Lista de tipos

short: 16

`short`
`short int`
`signed short`
`signed short int`
	Short signed integer type. Capable of containing at least the [−32 767, +32 767] range.[3] [a]	≥16	`%hi` or `%hd`	`[SHRT_MIN, SHRT_MAX]`	—

`unsigned short` `unsigned short int`	*Short* unsigned integer type. Contains at least the [0, 65 535] range.[3]	≥16	`%hu`	`[0, USHRT_MAX]`	—

`int`
`signed`
`signed int`
	Basic signed integer type. Capable of containing at least the [−32 767, +32 767] range.[3] [a]	≥16	`%i` or `%d`	`[INT_MIN, INT_MAX]`	none[8]

`unsigned` `unsigned int`	Basic unsigned integer type. Contains at least the [0, 65 535] range.[3]	≥16	`%u`	`[0, UINT_MAX]`	u or U[8]

Lista de tipos

long: 32 y long long: 64

`long`
`long int`
`signed long`
`signed long int`
	Long signed integer type. Capable of containing at least the [−2 147 483 647, +2 147 483 647] range.[3][a]	≥32	`%li` or `%ld`	`[LONG_MIN, LONG_MAX]`	`l` or `L`[8]

`unsigned long` `unsigned long int`	*Long* unsigned integer type. Capable of containing at least the [0, 4 294 967 295] range.[3]	≥32	`%lu`	`[0, ULONG_MAX]`	both `u` or `U` and `l` or `L`[8]

`long long`
`long long int`
`signed long long`
`signed long long int`
	Long long signed integer type. Capable of containing at least the [−9 223 372 036 854 775 807, +9 223 372 036 854 775 807] range.[3][a] Specified since the C99 version of the standard.	≥64	`%lli` or `%lld`	`[LLONG_MIN, LLONG_MAX]`	`ll` or `LL`[8]

`unsigned long long` `unsigned long long int`	*Long long* unsigned integer type. Contains at least the [0, 18 446 744 073 709 551 615] range.[3] Specified since the C99 version of the standard.	≥64	`%llu`	`[0, ULLONG_MAX]`	both `u` or `U` and `ll` or `LL`[8]

Lista de tipos

float: 32 y double: 64

`float`	Real floating-point type, usually referred to as a single-precision floating-point type. Actual properties unspecified (except minimum limits); however, on most systems, this is the IEEE 754 single-precision binary floating-point format (32 bits). This format is required by the optional Annex F "IEC 60559 floating-point arithmetic".		Converting from text: [d] `%f` `%F` `%g` `%G` `%e` `%E` `%a` `%A`		f or F
`double`	Real floating-point type, usually referred to as a double-precision floating-point type. Actual properties unspecified (except minimum limits); however, on most systems, this is the IEEE 754 double-precision binary floating-point format (64 bits). This format is required by the optional Annex F "IEC 60559 floating-point arithmetic".		`%lf` `%lF` `%lg` `%lG` `%le` `%lE` `%la` `%lA` [e]		none

Lista de tipos

long double: 128

`long double`	Real floating-point type, usually mapped to an extended precision floating-point number format. Actual properties unspecified. It can be either x86 extended-precision floating-point format (80 bits, but typically 96 bits or 128 bits in memory with padding bytes), the non-IEEE "double-double" (128 bits), IEEE 754 quadruple-precision floating-point format (128 bits), or the same as double. See the article on long double for details.		`%Lf` `%LF` `%Lg` `%LG` `%Le` `%LE` `%La` `%LA` [e]		`l` or `L`

Sesión 4: Entropía

(Módulo 4: Aritmética)

¿Dónde se espera una entropía alta?

1. Contraseña de uso único (OTP)
2. Segundo factor de autenticación (2FA)
3. Tokens de sesión
4. Claves de API
5. Preguntas de “seguridad”
6. UUID
7. URL

Ejemplos de falta de entropía

1. Semilla con fecha de hoy
2. Semilla con tiempo desde el inicio del sistema
3. Uso de datos de usuario como claves
4. Contraseñas por defecto
5. Criptografía casera para generar UUID
6. Diseño inseguro (por ejemplo, backdoor mediante cripto sin semilla y oráculo).

Nivel 4

(de los heroes)

Decepción cibernética: el arte de atrær a los atacantes a una trampa.