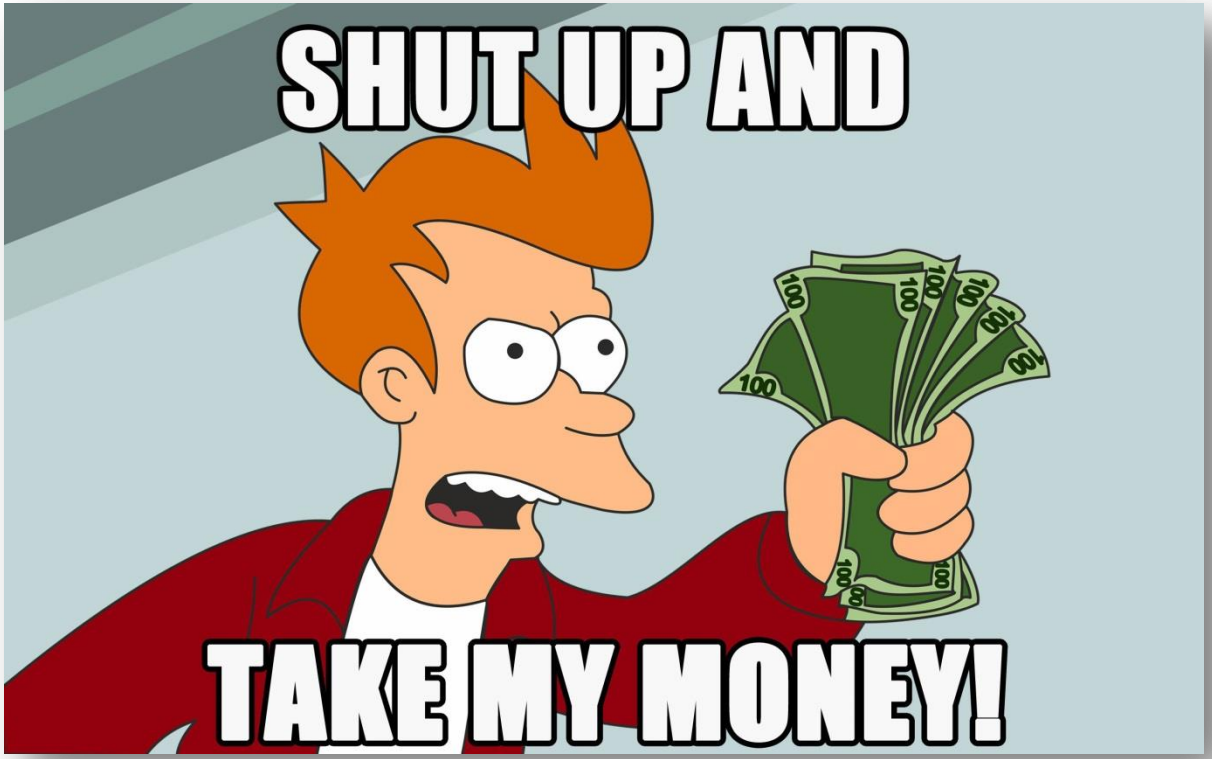
VULNERABILIDADES EN GATEWAYS DE PAGO

/ME

- Experto en seguridad en aplicaciones Web
- Desarrollador (Python!)
- Open Source evangelist
- w3af project leader
- Founder of Bonsai Information Security
- Founder and developer of TagCube SaaS

ACTORES

CLIENTE



BONSAI

MERCHANT

BONSAI

GATEWAY DE PAGOS

BONSAI

FLOWS

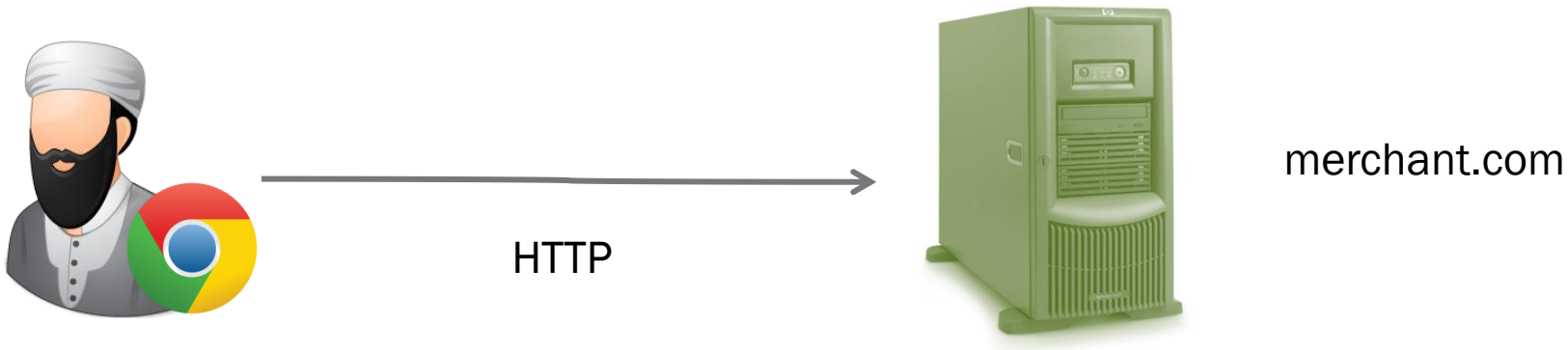
INICIO DE TRANSACCIÓN – CLICK EN PAGAR

El usuario es *redireccionado* **desde merchant.com** a **gateway.com**, donde ingresa la tarjeta de credito y confirma el pago. Es necesario que gateway.com conozca:

- El importe a pagar
- El merchant a quien acreditar el pago

BONSAI

CALLBACK – PAGO FINALIZADO



El usuario es *redireccionado* **desde gateway.com** a **merchant.com**, donde puede ver el resultado (exitoso|fallido) de su compra. Es necesario que merchant.com conozca:

- Un identificador de la compra
- El resultado de la transacción en el gateway de pagos

BONSAI

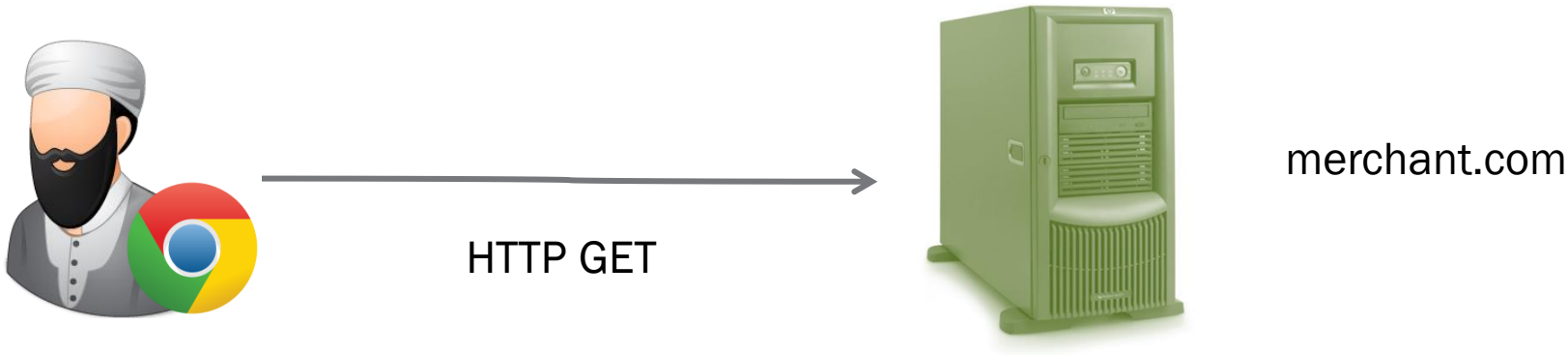
BONSAI

BETA1

INICIO DE TRANSACCIÓN

http://payments.com/pay?amount=358.00&merchant=221&pid=4982

CALLBACK



http://merchant.com/cb?pid=4982&result=success

BONSAI

BONSAI

VULNERABILIDADES @ BETA1

- Se debe utilizar HTTPS!

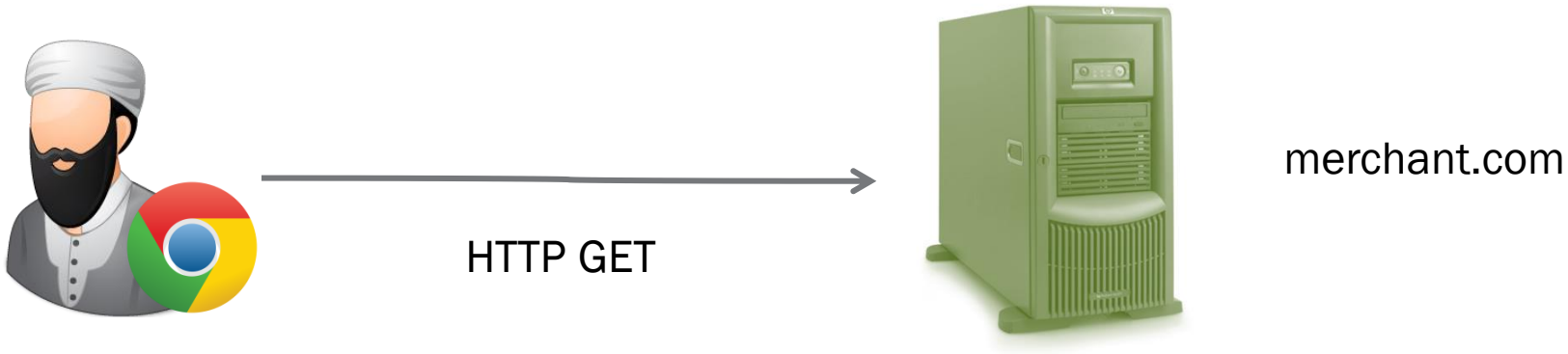
- Es posible **cambiar el importe a pagar** en el inicio de la transacción y de esa manera pagar un monto arbitrario por *pid=4982*

BETA2

INICIO DE TRANSACCIÓN

https://payments.com/pay?amount=358.00&merchant=221&pid=4982

CALLBACK



https://merchant.com/cb? amount=358.00&pid=4982&result=success

BONSAI

VULNERABILIDADES @ BETA2

- Aún verificando en https://merchant.com/cb que el importe pagado por cada **pid** sea el esperado, el problema es que **"cb" no tiene manera de verificar si el mensaje proviene de payments.com** , o sí un usuario navegó directamente a esta URL.

- De la misma manera, payments.com no tiene forma de verificar que los datos enviados a https://payments.com/pay hayan sido realmente generados por merchant.com

BETA3

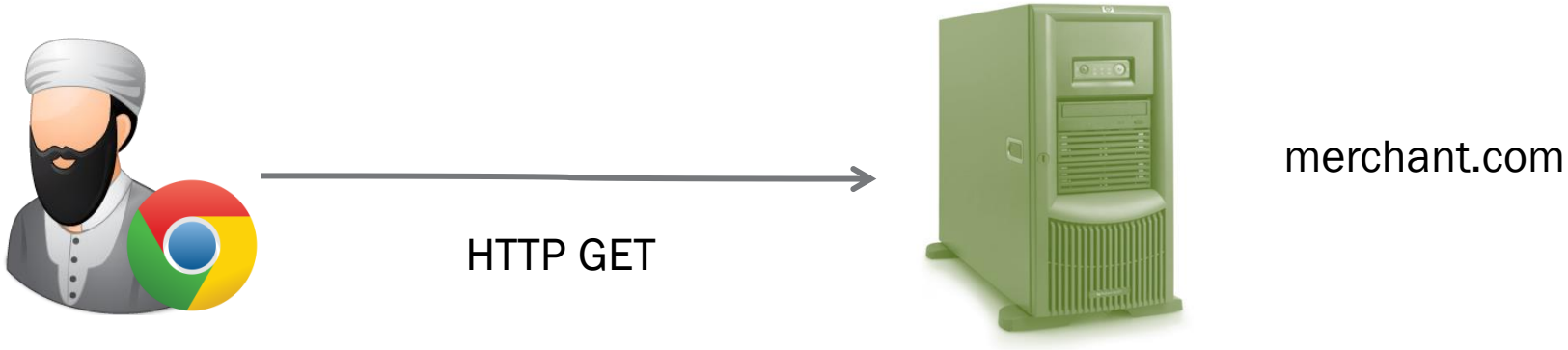
INICIO DE TRANSACCIÓN

https://payments.com/pay?amount=358.00
&merchant=221
&pid=4982
&sig=8b1a9953...8c47804d7

```
secret = 'long-pre-shared-secret'
data = qs['amount'] + qs['merchant'] + qs['pid']
sig = sha1(secret + data)
```

BONSAI

CALLBACK



https://merchant.com/cb?amount=358.00
&result=success
&pid=4982
&sig=8b1a9953...8c47804d7

```
secret = 'long-pre-shared-secret'
data = qs['amount'] + qs['result'] + qs['pid']
sig = sha1(secret + data)
```

BONSAI

BONSAI

HASH LENGTH EXTENSION ATTACK

Utilizando la vulnerabilidad de hash length extension presente en numerosos algoritmos de hashing es posible **agregar datos** a la información original y obtener una firma válida **sin tener acceso al secreto compartido**:

```
$ hashpump
Input Signature: 99180b25a0c8a2b4e4981165a7223a8b
Input Data: pid=4982&amount=890.99&result=success
Input Key Length: 17
Input Data to Add: amount=1.00
c685c55aaa1da2097873fca8c5cdf72b
pid=4982&amount=890.99&result=success\x80\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00`\x01\x00\x00\x00\x00\x00\x00
amount=1.00
```

BONSAI

HASH LENGTH EXTENSION ATTACK

Al recibir la información firmada el servidor payments.com **verifica la firma y procede con la transacción**:

```
>>> import hashlib
>>> secret = 'areallylongsecret'
>>> data =
'product_id=321&price=890.99\x80\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00`\x01\x00\x00\x00\x00\x00\x00price=1.00'
>>> received_signature = 'c685c55aaa1da2097873fca8c5cdf72b'
>>> if hashlib.md5(secret + data).hexdigest() ==
received_signature:
...     print 'Signature is correct'
... else:
...     print 'Reject transaction'
...
Signature is correct
>>>
```

BONSAI

HMAC – FIRMA DIGITAL

```
>>> import hmac
>>> digest_maker = hmac.new('long-secret-here', hashlib.sha1)
>>> digest_maker.update(block)
>>> digest_maker.hexdigest()
'cc4a5ce1b3df48aec5d22d1f16b894a0b894eccc'
```

Evitamos la vulnerabilidad de hash length extension de las funciones de hash SHA1, MD5 y otras, la cual puede ser facilmente explotada para agregar informacion a la data firmada.

PROD1

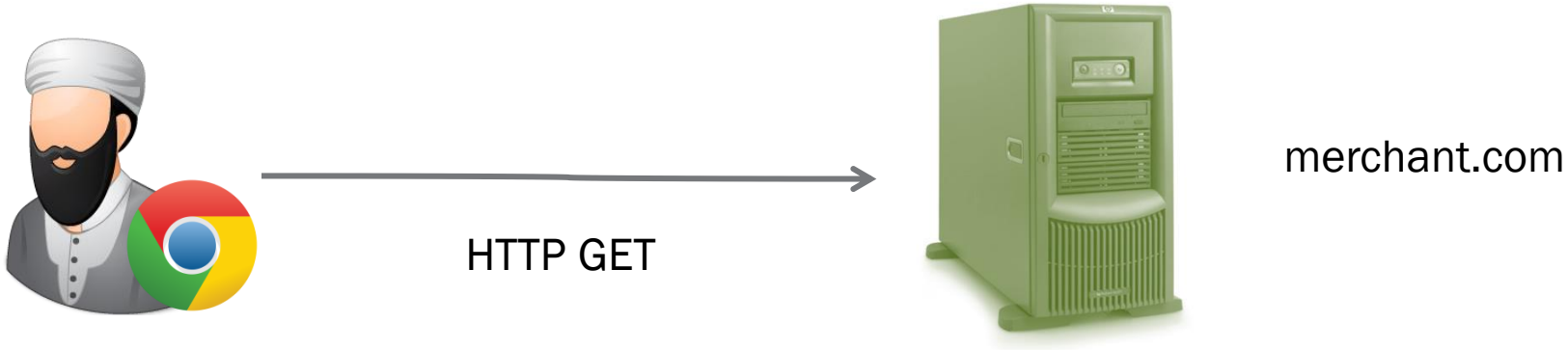
INICIO DE TRANSACCIÓN

https://payments.com/pay?amount=358.00
&merchant=221
&pid=4982
&sig=8b1a9953...8c47804d7

```
secret = 'long-pre-shared-secret'
data = qs['amount'] + qs['merchant'] + qs['pid']
sig = hmac_sha1(secret, data)
```

BONSAI

CALLBACK



https://merchant.com/cb?amount=358.00
&result=success
&pid=4982
&sig=8b1a9953...8c47804d7

```
secret = 'long-pre-shared-secret'
data = qs['amount'] + qs['result'] + qs['pid']
sig = hmac_sha1(secret, data)
```

BONSAI

POTENCIALES ATAQUES

- **Fuerza bruta** al secreto compartido. Si se permite al usuario ingresar el secreto compartido, el mismo debe ser de al menos 16 caracteres conteniendo >4 numeros >4 lower >4 upper
- **Generación deterministica del secreto** compartido. En un conocido gateway de pagos el secreto compartido se generaba al activar el usuario y era igual al site *(merchant.com en los ejemplos)*
- Se deben **firmar todos los datos,** que ocurre en los ejemplos si en el futuro se agrega el parametro *currency*?

BONSAI

CONCLUSIONES

- Aplicando este conocimiento es posible incrementar considerablemente la seguridad de los gateways de pago

- *Existen muchas otras cosas a revisar*, esta es una charla con objetivos de aprendizaje!

BONSAI

andres@bonsai-sec.com

@w3af