

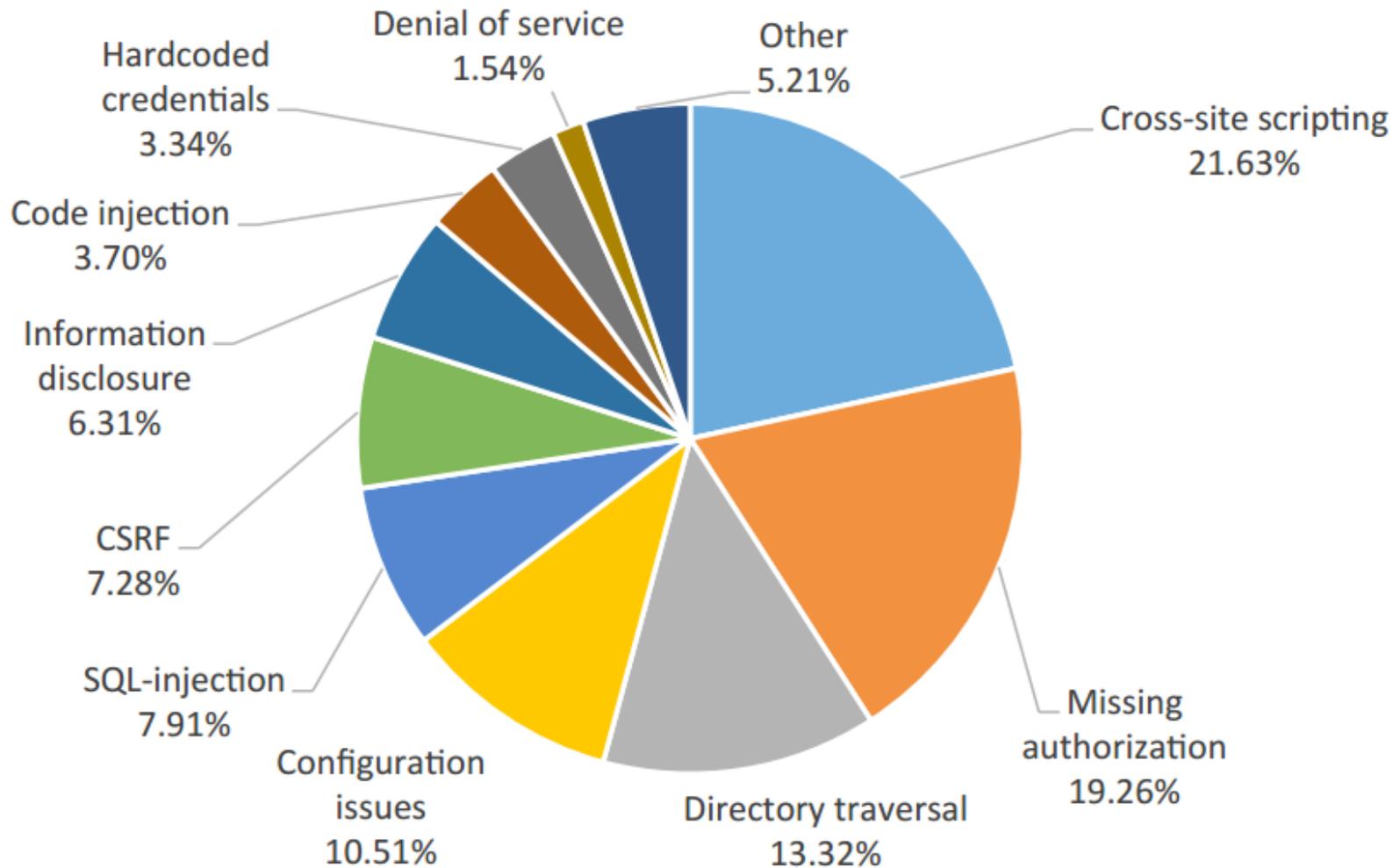
About Web Vulnerabilities

And how to protect your GEN web applications

Ruben Mijwaart, Sr. consultant API management



ca[®]
technologies

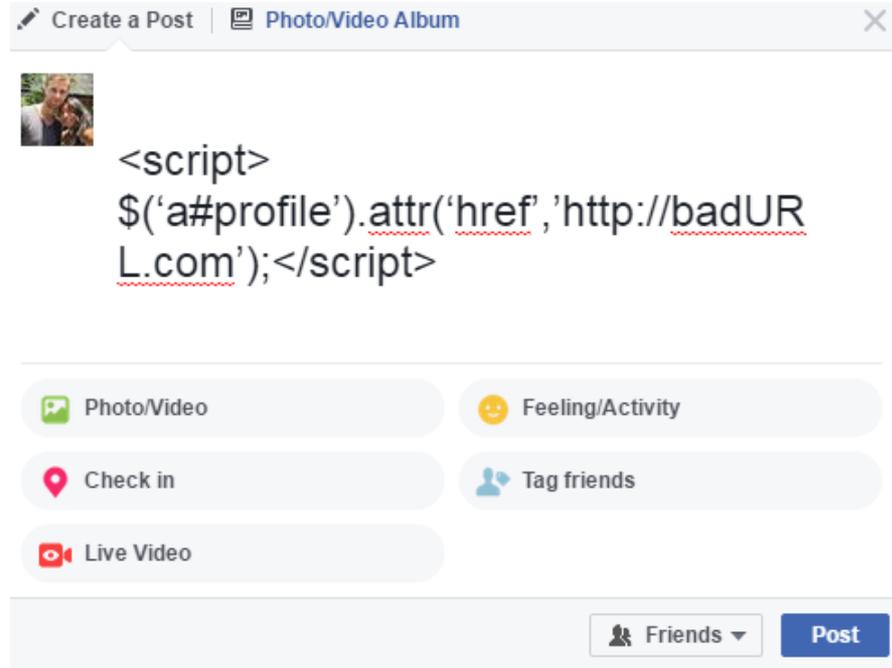




Example: Cross-site Scripting (XSS)

Find a form input and inject your malicious JavaScript code.

p.s. this only works if people other than you will load the injected code in their browser.

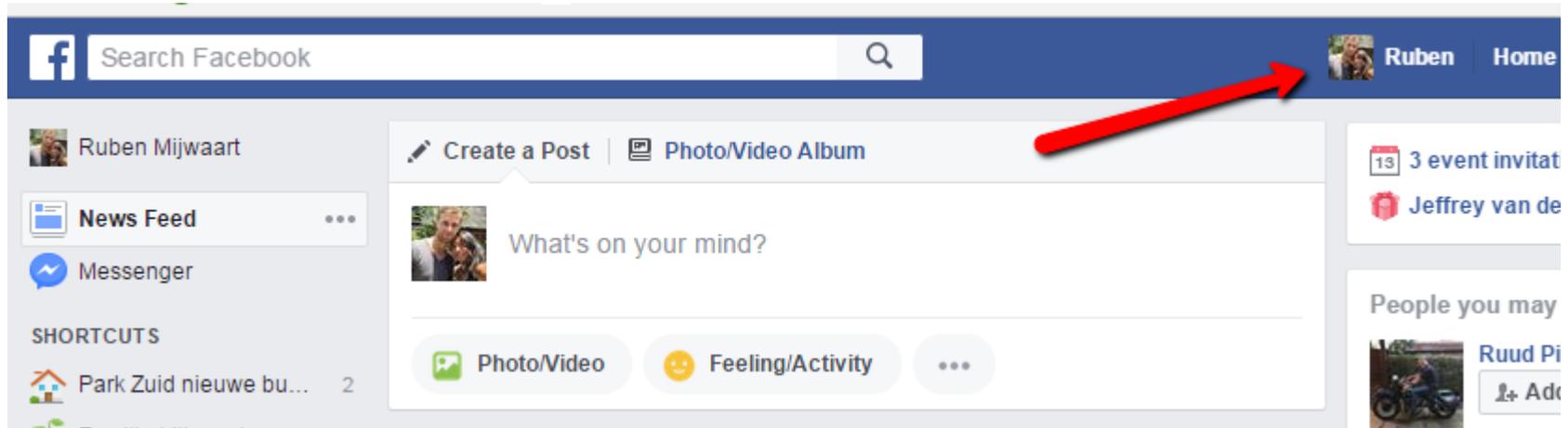


The image shows a screenshot of a Facebook 'Create a Post' form. The form is titled 'Create a Post' and 'Photo/Video Album'. It features a profile picture placeholder, a text input field containing a malicious JavaScript payload, and several interactive buttons: 'Photo/Video', 'Feeling/Activity', 'Check in', 'Tag friends', and 'Live Video'. At the bottom right, there is a dropdown menu for privacy settings (set to 'Friends') and a 'Post' button.

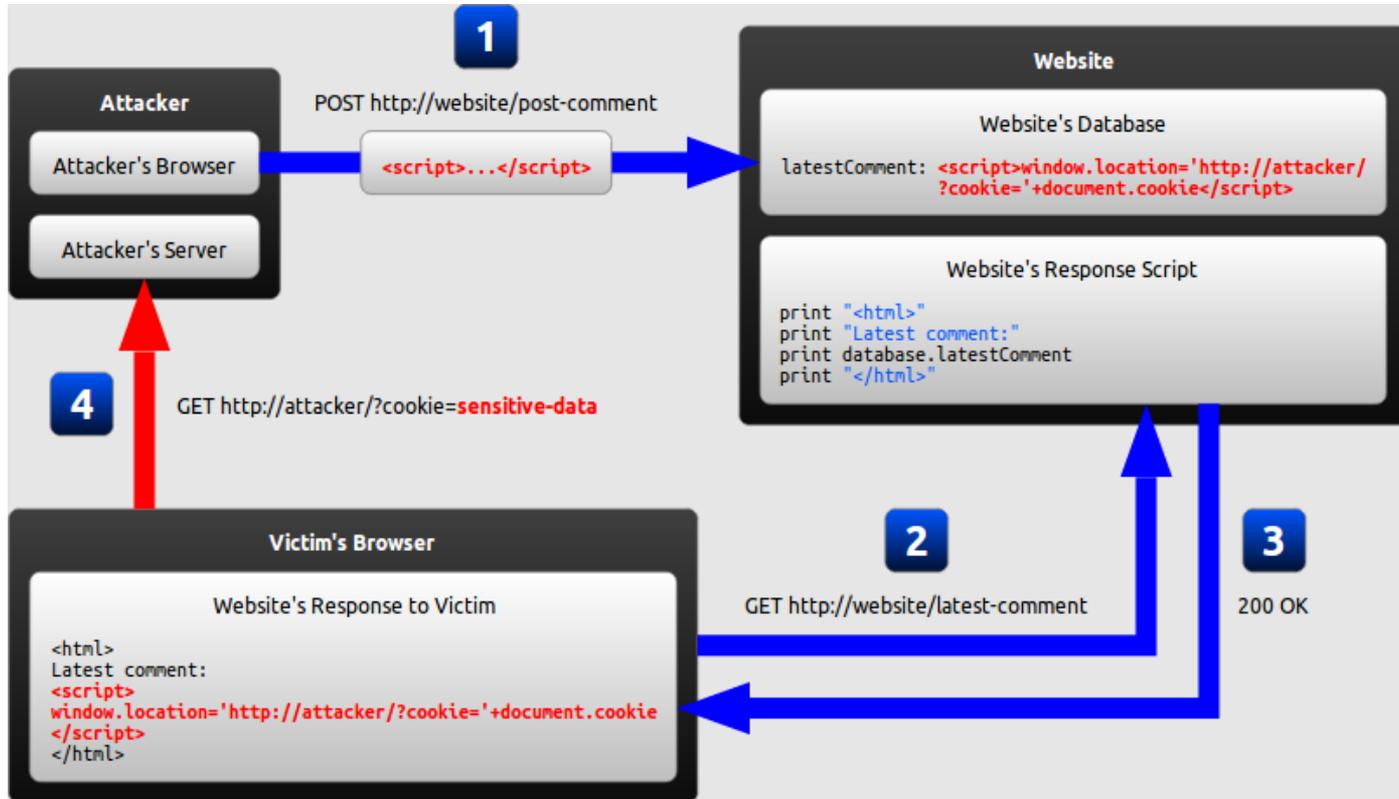
```
<script>
$( 'a#profile' ).attr( 'href', 'http://badURL.com' ); </script>
```

Example: Cross-site Scripting

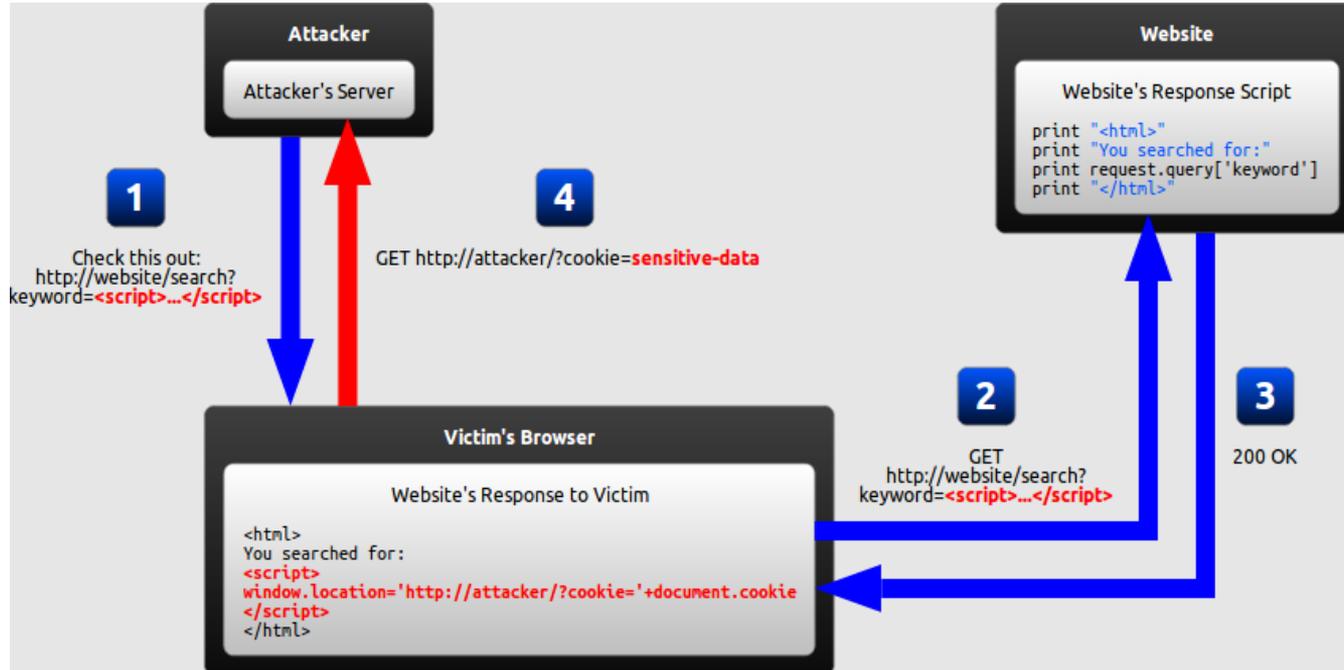
Users that see this post and click on their profile link will be redirected to a malicious website.



Example: Cross-site Scripting



Example: Reflected Cross-site Scripting



Consider the following facts

- JavaScript can make arbitrary **modifications to the HTML** of the current page by using DOM manipulation methods.
- JavaScript can **send HTTP requests** with arbitrary content to arbitrary destinations by using XMLHttpRequest and other mechanisms.
- JavaScript has **access** to some of the user's **sensitive information**, such as cookies.

These facts combined can cause very serious security breaches....

Potential consequences

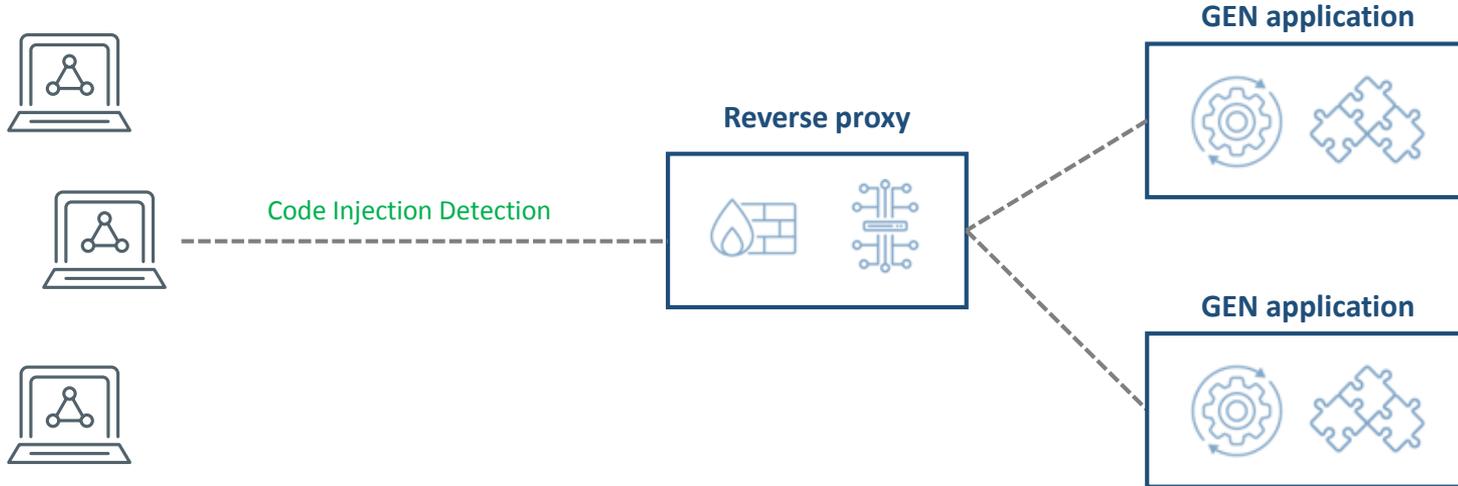
Cookie theft. The attacker can access the victim's cookies associated with the website using `document.cookie`, send them to his own server, and use them to extract sensitive information like **session IDs**.

Keylogging. The attacker can register a keyboard event listener using `addEventListener` and then send all of the user's keystrokes to his own server, potentially recording sensitive information such as **passwords** and **credit card numbers**.

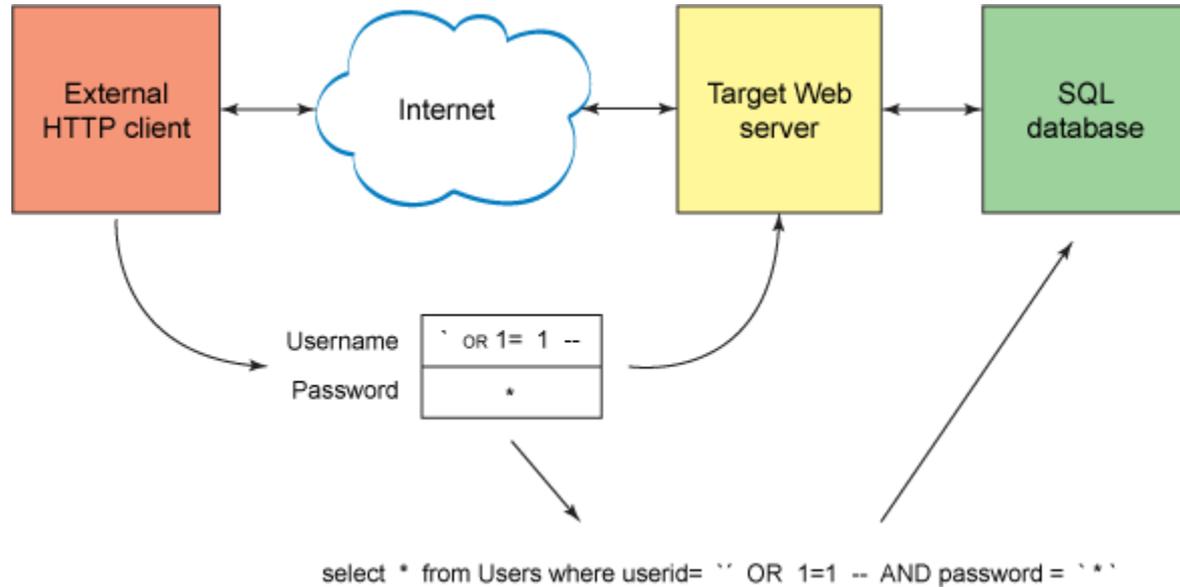
Phishing. The attacker can insert a **fake login form** into the page using DOM manipulation, set the form's action attribute to target his own server, and then trick the user into submitting sensitive information.

Solution

Route all HTTP(s) traffic trough reserve proxy

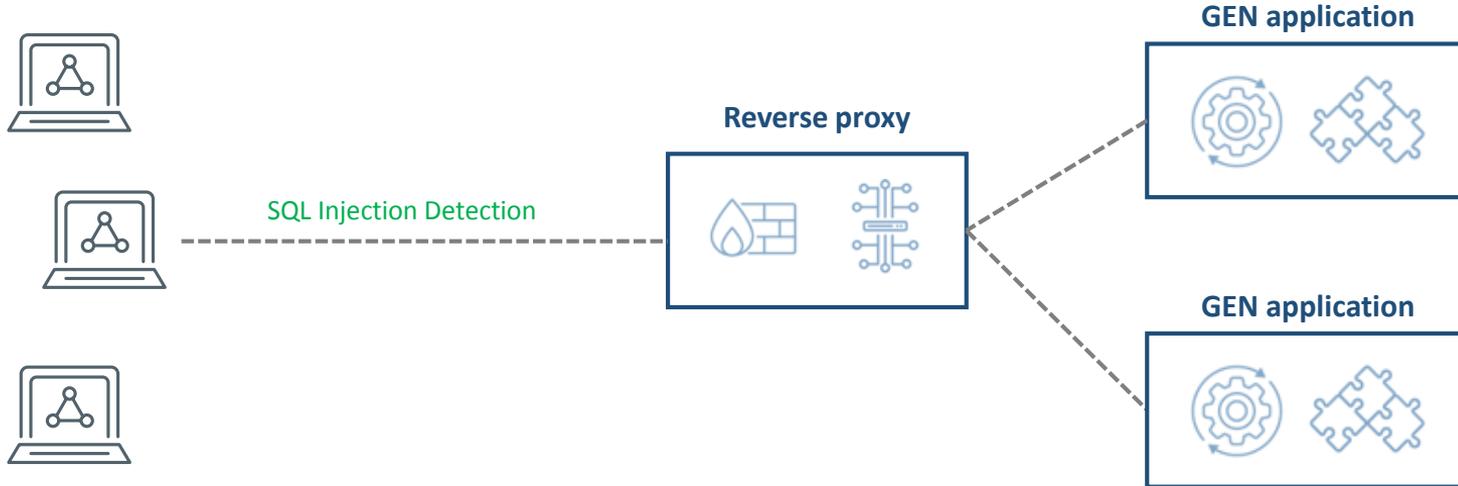


Example: SQL injection (SQLI)



Solution

Route all HTTP(s) traffic trough reserve proxy



Example: ClickJacking

Objective:

- Quickly spread marketing content via facebook.

Disclaimer:

- Works only with facebook users
- And only the ones that choose to 'stay logged in'

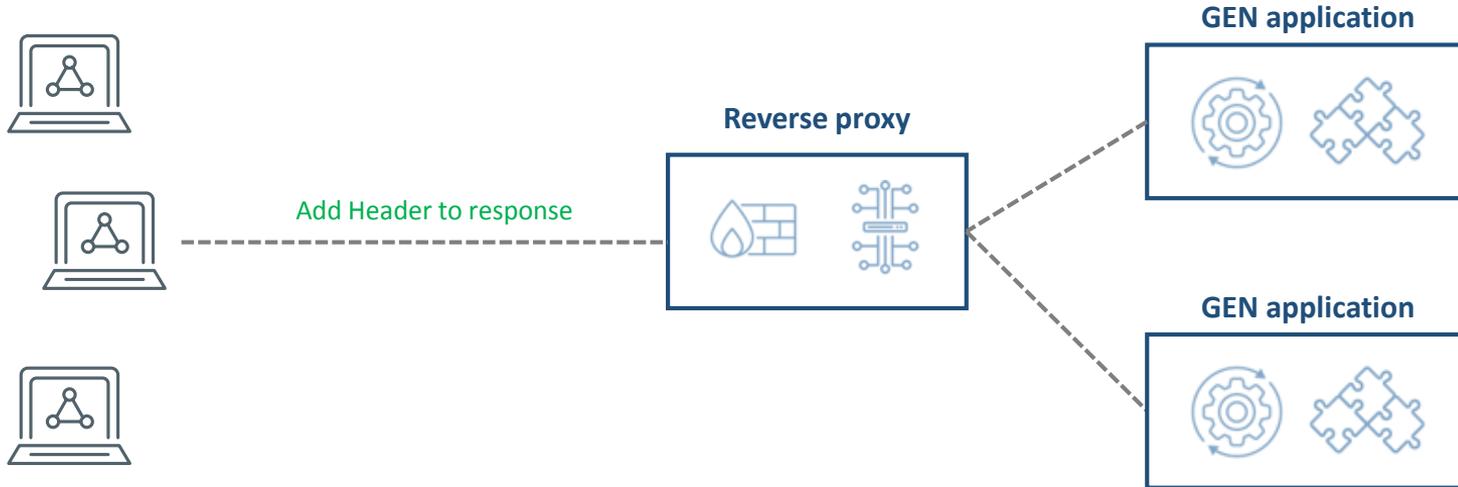
Example: ClickJacking

1. Create a 'malicious' web page with misleading call to action e.g. 'WIN'.
2. Use a transparent iframe to load the facebook page we would like to promote.
3. Make sure the 'share' button overlays the 'WIN' button.
4. Now all logged-in FB users that click 'WIN' shared your FB fan page on their timeline.



Solution

Route all HTTP(s) traffic trough reserve proxy



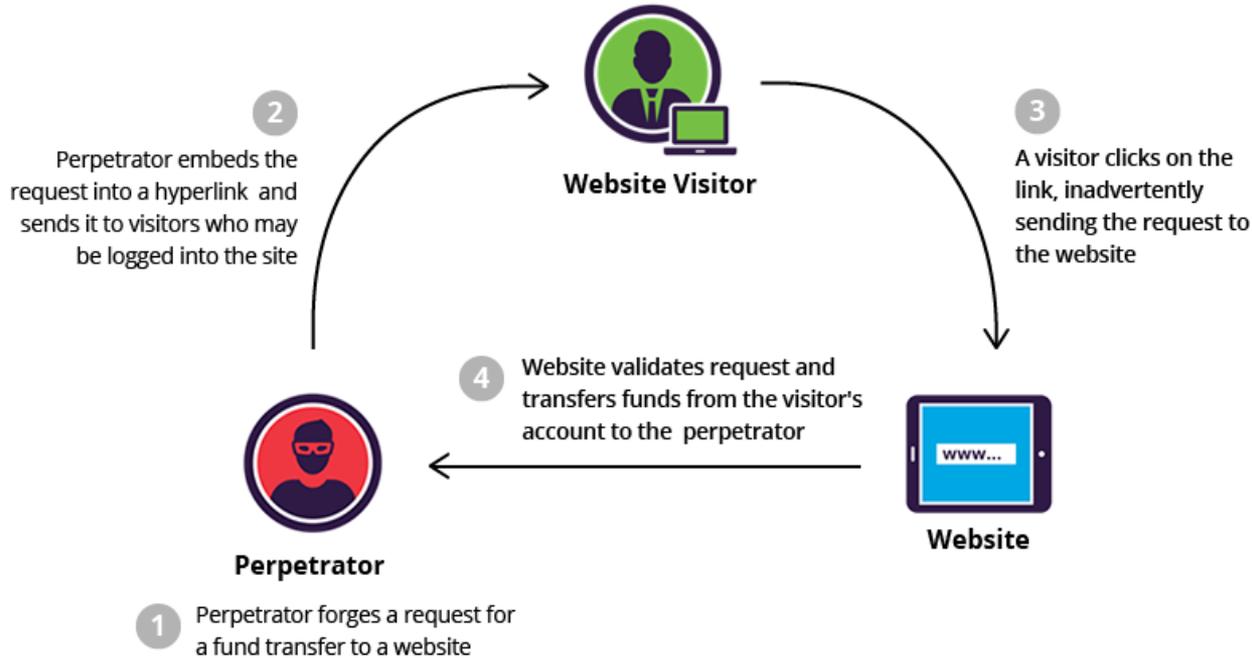
Add X-Frame-Options headers to HTTP response to prevent the client browser from loading your content into an iframe.

Possible values: **DENY**, SAMEORIGIN, or ALLOW-FROM.

Example: Cross-site Request Forgery (CSRF)

- an attack that **forces** an end user **to execute unwanted actions** on a web application in which they're **currently authenticated**.
- CSRF attacks specifically target state-changing requests, not theft of data, since the attacker has no way to see the response to the forged request.

Example: Cross-site Request Forgery (CSRF)

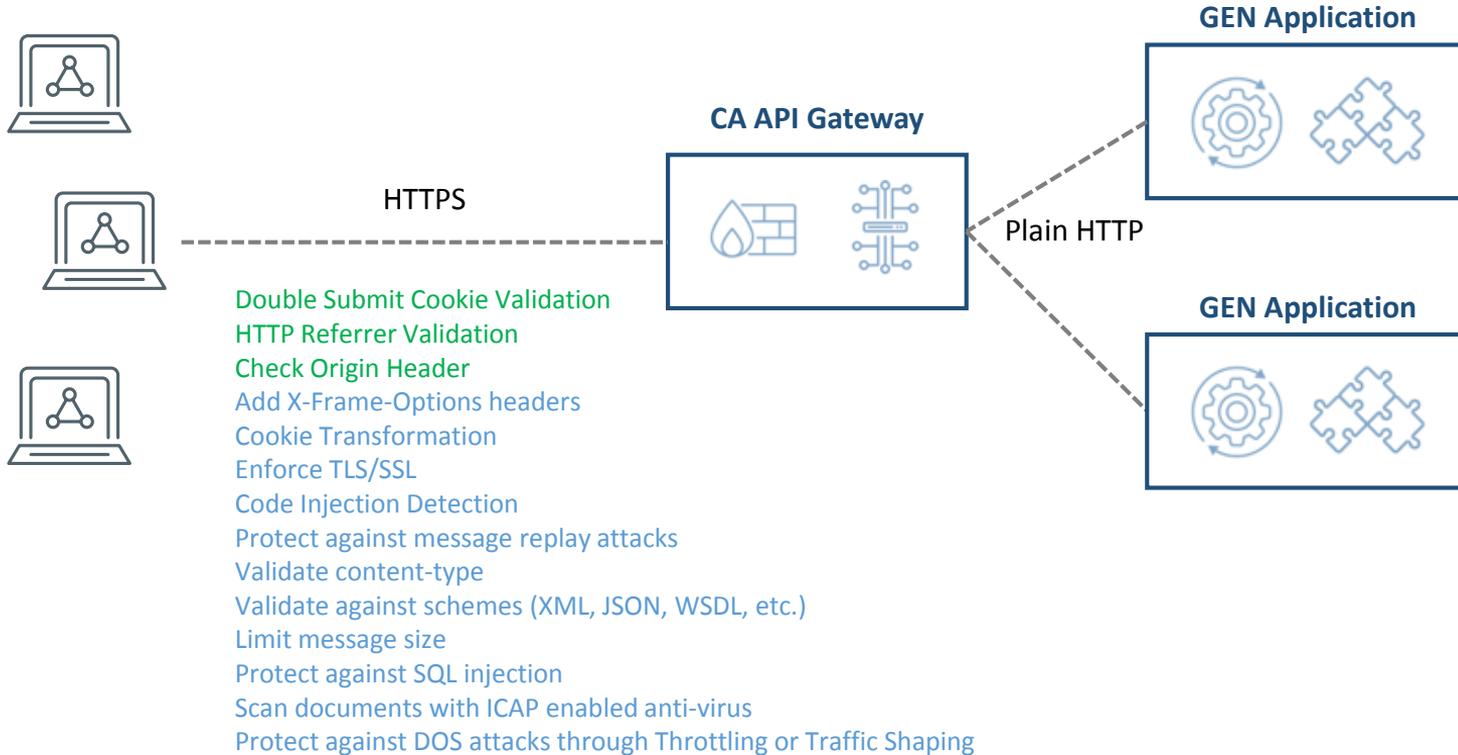


How to prevent Cross-site Request Forgery?

1. Check if the Origin header is matching the target Origin
2. When the Origin header is missing, check if the Referer header is matching the target Origin
 - Referer and Origin headers are on the forbidden header list meaning that it cannot be changed using JavaScript. (i.e. not vulnerable to XSS attacks)
3. If both are missing, or if target and origin do not match: block the request

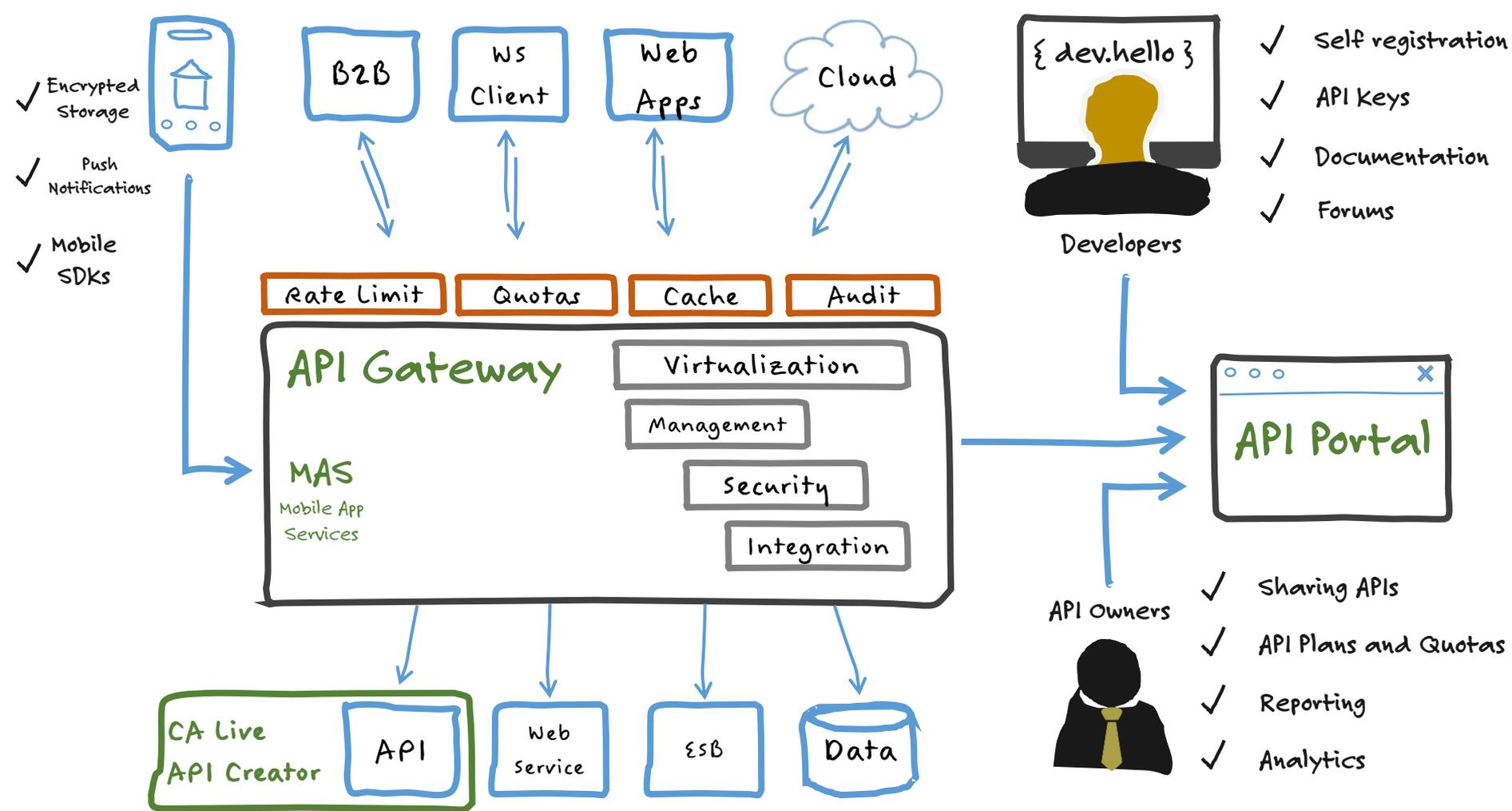
Protect GEN applications

Route all HTTP(s) traffic for GEN apps through the CA API Gateway acting as a reversed proxy



Certifications CA API Management

- Common criteria EAL4+
- FIPS 140-2
- PCI DSS
- US STIG Vulnerability Tested
- Joint DoD/IC Service Security Working Group (JSSWG)
- Joint DoD/IC Enterprise Service Monitoring
- HSPD12 Backend Attribute Exchange (BAE)



Thank you

