# The Universal XSS PDF Vulnerability

Ofer Shezaf

OWASP IL Chapter leader

CTO, Breach Security

## What Is It?

- A bug in Adobe Acrobat Reader
- Enables running malicious scripts on a victims computer when (nearly any) browser executed such a link and uses acrobat in embedded mode:

  http://host/file.pdf#anyname=javascript:your_code_here
- (Nearly) every computer has Acrobat Reader installed.
- (Nearly) every web site has PDF files
- Now Site is vulnerable to every imaginable XSS scenarios without having a local XSS vulnerablity

## How can it be exploited?

- Attacker gets victim to activate a link:
  - With user action required:
    - Sending link by e-mail (lame but useful)
    - Link on a malicious site
  - Without user interaction:
    - <Iframe> on malicious site
  - Obfuscated:
    - TinyURL
  - Persistent
    - But unlike XSS, can be planted in any site, just like comments spam.
- Script in link works in the context of the referenced site, and every XSS technique would work.

BREACH™

## How Can You Solve that?

- Client side solutions:
  - Patch upgrade or replace Adobe Acrobat
  - Foxit highly recommended (http://www.foxitsoftware.com)
  - Change action for PDF to “save file” or “run default application”, avoiding plug in
  - Upgrade the client (IE7 is not vulnerable).
- Server side solutions:
  - Is it our responsibility at all?
  - The attack is totally invisible to the server as the fragment, which is the part after the “#” is not sent to the server.
  - As a result, we need to change every PDF request to the site

# Sever Side Solutions - Defeat the plug-in

- Set PDF type to application/octet-stream:
  - Can be done using normal configuration in most web servers
  - Defeated by IE which ignores Content-Type header and check content type according to magic bytes.
  - Change the way the application behaves.
- Set Content-Disposition Header to “attachment;”
  - Harder, but still can be done using normal web server configuration.
  - Defeats IE sleazy mechanism.
  - Still changes the way the application behaves.

## Server Side Solutions – Defeat the fragment

How will I make the browser forget the fragment he holds?

Redirect every PDF request to one with a new fragment

But how will we know after the redirection that the new request should not be redirected? After all the fragment is not sent to the server.

Add a parameter to the redirected request indicating it has already been redirected.

But how will we prevent the hacker from including the parameter in his original attack?

Make the token value cryptographic: encrypt a local secret, the source IP and time stamp to create the parameter value, making it harder to the attacker to pre-generate it.

## Problems

- Dynamically generated PDFs
  - Get parameters
  - Post parameters
- Reliance on IP address:
  - Hoping source IP such as AOL
  - Shared source IP such as NAT

## Amit Klein’s algorithm

```
IF the URL doesn't contain token_query, then:
  calculate X=encrypt_with_key(server_time, client_IP_address)
  redirect to file.pdf?token_query=X

   ELSE IF the URL contains token_query, and
  decrypt(token_query).IP_address==client_IP_address and
  decrypt(token_query).time>server_time-10sec
  serve the PDF resource as an in-line resource with a none
      attack fragment.

   ELSE
  serve the PDF resource as a "save to disk" resource
      using headers tricks.
```

## Implementations

- OWASP Java filter
  - https://www.owasp.org/index.php/PDF_Attack_Filter_for_Java_EE
- F5 iRules
  - http://devcentral.f5.com/Default.aspx?tabid=29&articleType=ArticleView&articleId=70
- Mod_Rewrite (partial)
  - http://www.owasp.org/index.php/PDF_Attack_Filter_for_Apache_mod_rewrite
- .Net handler
  - http://www.techplay.net/
- ModSecurity
  - Will be published shortly
  - The only one to address dynamically generated PDFs

BREACH

## Some solutions I’ve seen

We got a support request to convert the following rules to ModSecurity 2:

```
SecFilterOutput OUTPUT "javascript:" "deny, status:500"
SecFilterOutput OUTPUT  "<\s*script.*?\s*>"  "deny, status:500"
```

Someone else suggested publicly this rule:

```
SecRule REQUEST_URI_RAW: ".*\.pdf[^A-Za-z0-9._?&%-]" \
    "deny,log,status:502,id:951004,severity:2,msg:'DOM attack on static content'"
```

BREACH

# Thank You!

Ofer Shezaf

ofers@breach.com