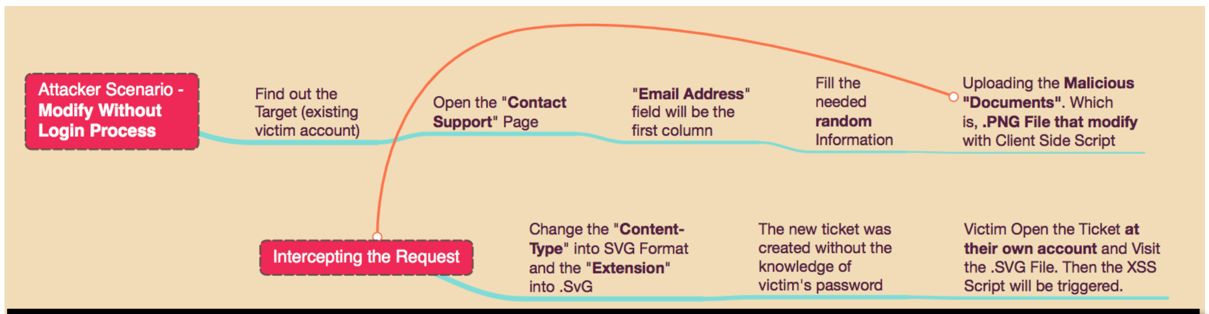
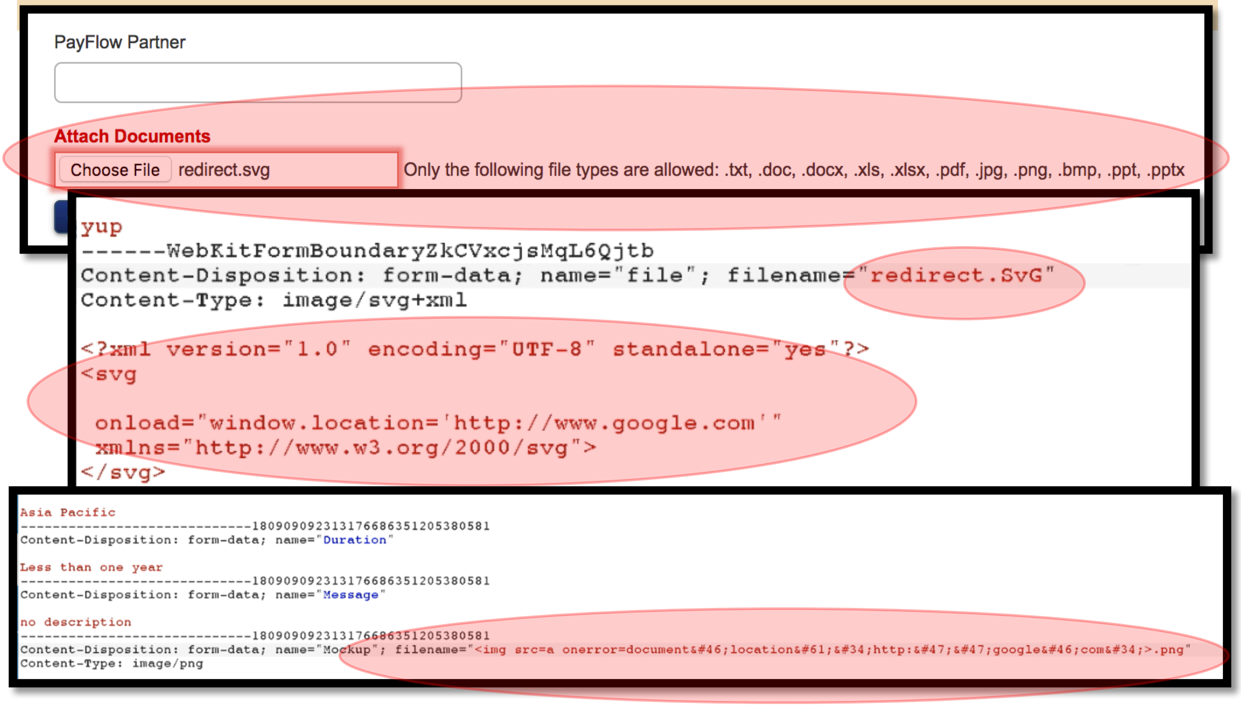
Turning Self-XSS into non-Self Stored-XSS via Authorization Issue by “Using Malicious SVG File” at “PayPal Tech-Support Portal”

(**Guest Star**: PayPal Brand Central:

“Turning Self-XSS into non-Self Stored-XSS at “**PayPal Brand Central**” via Authorization Issue by Using “**File Name Injection**”)


Attacker Scenario - Modify Without Login Process
Find out the Target (existing victim account)
Open the "Contact Support" Page
"Email Address" field will be the first column
Fill the needed random Information
Uploading the Malicious "Documents". Which is, .PNG File that modify with Client Side Script
Intercepting the Request
Change the "Content-Type" into SVG Format and the "Extension" into .SvG
The new ticket was created without the knowledge of victim's password
Victim Open the Ticket at their own account and Visit the .SVG File. Then the XSS Script will be triggered.


PayFlow Partner

Attach Documents

Choose File redirect.svg Only the following file types are allowed: .txt, .doc, .docx, .xls, .xlsx, .pdf, .jpg, .png, .bmp, .ppt, .pptx

```
yup
------WebKitFormBoundaryZkCVxcjsMqL6Qjtb
Content-Disposition: form-data; name="file"; filename="redirect.SvG"
Content-Type: image/svg+xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<svg

 onload="window.location='http://www.google.com'"
 xmlns="http://www.w3.org/2000/svg">
</svg>
```

```
Asia Pacific
-----------------------------180909092313176686351205380581
Content-Disposition: form-data; name="Duration"

Less than one year
-----------------------------180909092313176686351205380581
Content-Disposition: form-data; name="Message"

no description
-----------------------------180909092313176686351205380581
Content-Disposition: form-data; name="Mockup"; filename="<img src=a onerror=document&#46;location&#61;&#34;http:&#47;&#47;google&#46;com&#34;>.png"
Content-Type: image/png
```

Nov 13th, 2017

@YoKoAcc (yk@firstsight.me)

[English Version]

Revision Detail

Version	Date	Detail
0.1	Nov 13th, 2017	-

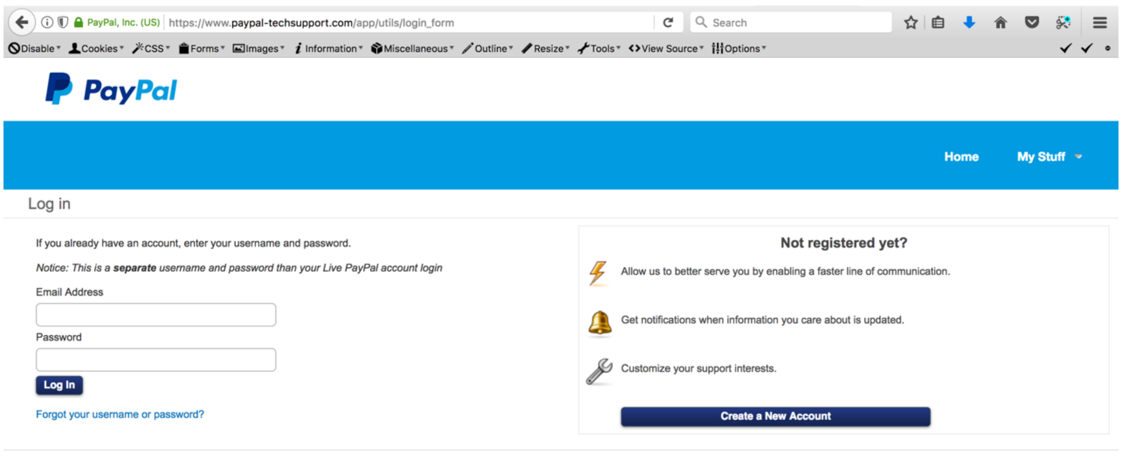
Table of Contents

Table of Figures

List of Table

I. ABSTRACT

As a part for delivering the technical support to all of PayPal's merchant, PayPal providing the portal (located at: https://www.paypal-techsupport.com) for their merchant to communicate each other with PayPal such as discussing about the integration, send a feedback about the needed new feature, or any technical issue that could be face by PayPal's merchant.


PayPal, Inc. (US) https://www.paypal-techsupport.com/app/utils/login_form

Log in

If you already have an account, enter your username and password.

Notice: This is a ***separate*** *username and password than your Live PayPal account login*

Email Address

Password

Forgot your username or password?

Not registered yet?

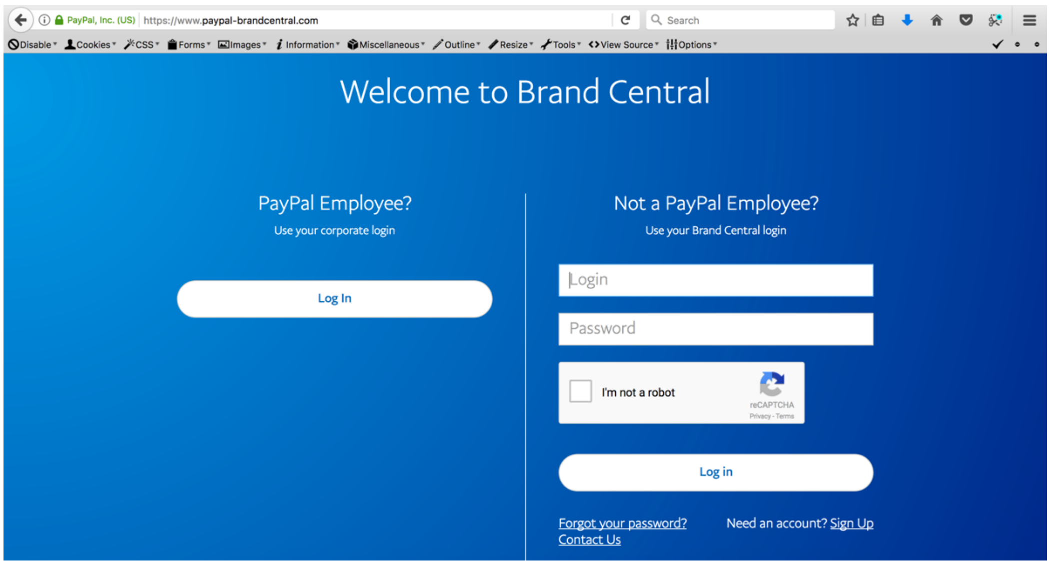
Allow us to better serve you by enabling a faster line of communication.

Get notifications when information you care about is updated.

Customize your support interests.

Figure 1 PayPal Technical Support Portal

In other side, for completing the support to all of entities that has use PayPal in their daily activity, PayPal also providing the site that could be used specifically by their customer to asking a permission about the use of PayPal's logo at their business (located at: https://www.paypal-brandcentral.com).


PayPal, Inc. (US) https://www.paypal-brandcentral.com

Welcome to Brand Central

PayPal Employee?

Use your corporate login

Not a PayPal Employee?

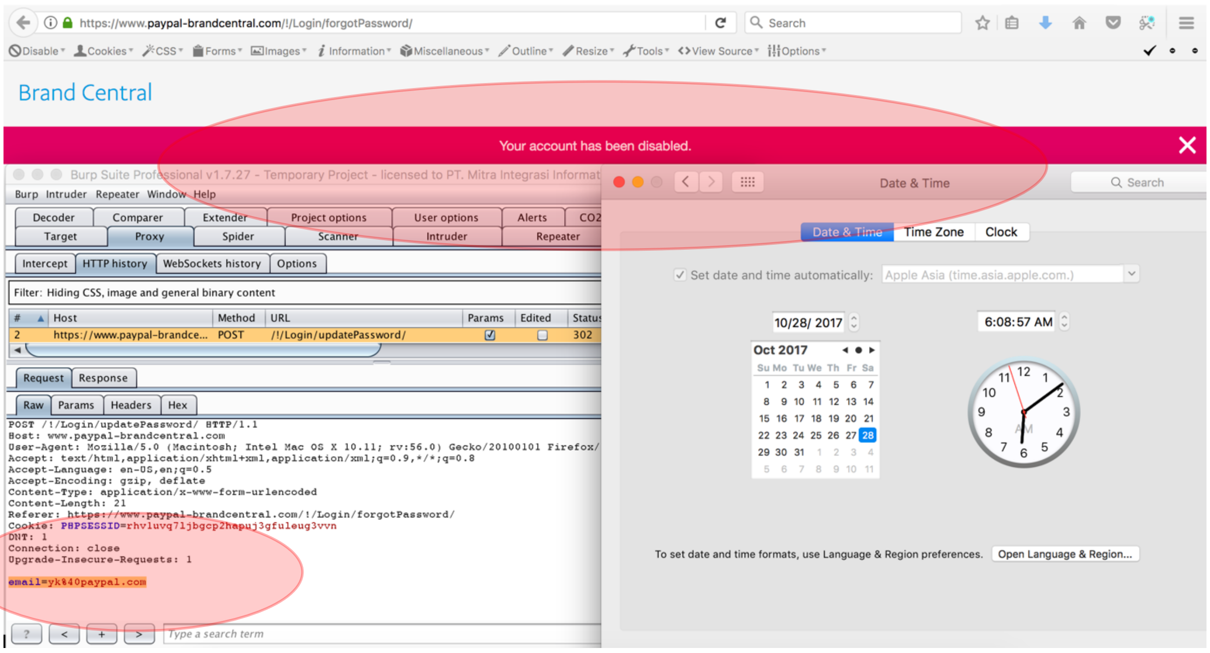
Use your Brand Central login

Figure 2 PayPal Brand Central Portal

Both of site has been supported with the **file upload feature** so either merchant or customer could try to upload the permitted file that could be used by PayPal for looking the situation that faced by them each other.

But the problem exists when those file upload feature **has a validation issue** at their content (for technical support portal) and at their file name (for brand central portal) that could allow the Attacker to be able to store a script (client side scripting) which could be used for few attack scenarios (like redirecting a user to a phishing page or even downloading a malware – which is called Cross Site Scripting – will explain later at this report).

Additional Note: Previously, PayPal Brand Central Site has been open for public for registration and everyone could activate their selves with the activation link that provide by PayPal (and yes, we could bypass this activation and we will release the related simple paper in near future – you could try to test the forgot password feature first with account **yk@paypal.com**). But now, every registration at this portal should be approved manually by PayPal.



https://www.paypal-brandcentral.com/!/Login/forgotPassword/
Brand Central
Your account has been disabled.
Burp Suite Professional v1.7.27 - Temporary Project - licensed to PT. Mitra Integrasi Informat
Date & Time
Set date and time automatically: Apple Asia (time.asia.apple.com.)
10/28/ 2017
6:08:57 AM

#	Host	Method	URL	Params	Edited	Status
2	https://www.paypal-brandce...	POST	/!/Login/updatePassword/	✓		302

```
POST /!/Login/updatePassword/ HTTP/1.1
Host: www.paypal-brandcentral.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:56.0) Gecko/20100101 Firefox/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 21
Referer: https://www.paypal-brandcentral.com/!/Login/forgotPassword/
Cookie: PHPSESSID=rhvluvq7ljbgcp2hapuj3gfuleug3vvn
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

email=yk%40paypal.com
```

Figure 3 Bypassing the PayPal Activation Link at PayPal Brand Central

II. INTRODUCTION

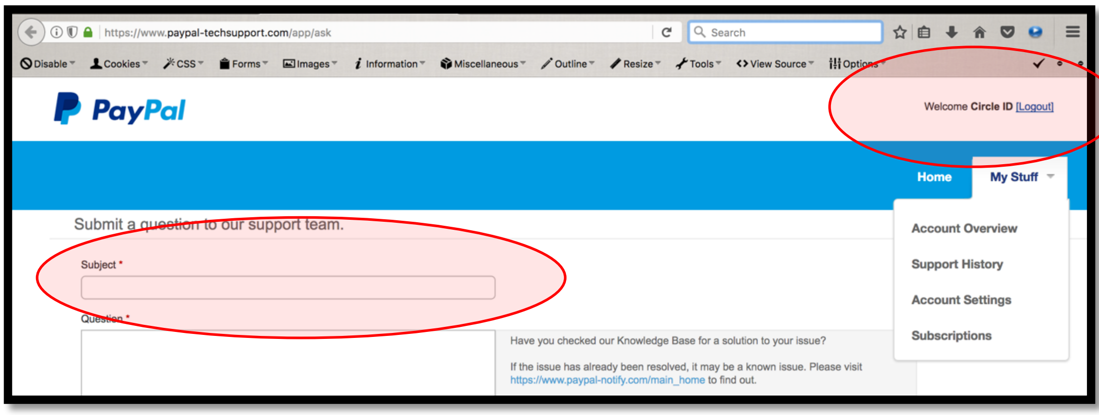
2.1. Stored Cross Site Scripting (XSS)

To put it simply, this kind of vulnerability is a vulnerability that could “let” an Attacker to be able to execute a code in the input section that hasn’t implemented filtering for special characters such as “ > < : / ; etc. In contrast to Reflected XSS that doesn’t save the “script” and required the target to visit the URL that has been “injected” by “additional contents” from an Attacker, Stored XSS save this script at the database and could be execute automatically when the target visit the affected page like normal.

2.2. **0x01 - General Ticket Submission at Tech Support Portal**

At the TechSupport Portal, PayPal allows the users to submit their issue via "Contact Support" Feature that could be visit at https://www.paypal-techsupport.com/app/ask. Generally, there are two process that could be used to user to submit the ticket into the TechSupport portal, which are:

2.2.1. The **user should login into their TechSupport Portal account** and visit the mentioned URL related "Contact Support" Feature. At this condition, then the user will directly face the **"Subject" column as the first thing** that need to be filled.


https://www.paypal-techsupport.com/app/ask

Welcome Circle ID [Logout]

Submit a question to our support team.

Subject *

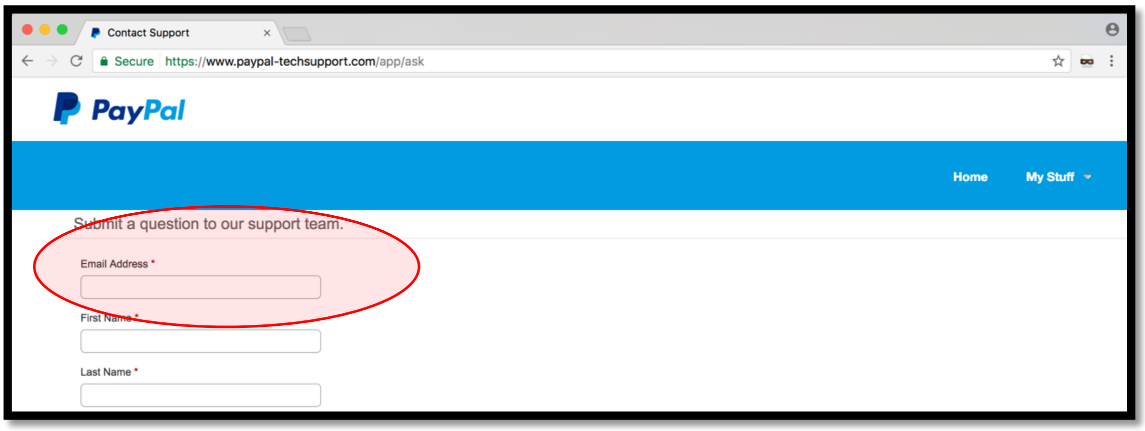
Question *

Have you checked our Knowledge Base for a solution to your issue?

If the issue has already been resolved, it may be a known issue. Please visit https://www.paypal-notify.com/main_home to find out.

Figure 4 Contact Support After Login

2.2.2. The second one is the user just directly visit the "Contact Support" Feature **without any login activity**. At this condition, then the user will directly face the "**Email Address" column as the first thing** that need to be filled.


Contact Support

Secure https://www.paypal-techsupport.com/app/ask

Submit a question to our support team.

Email Address *

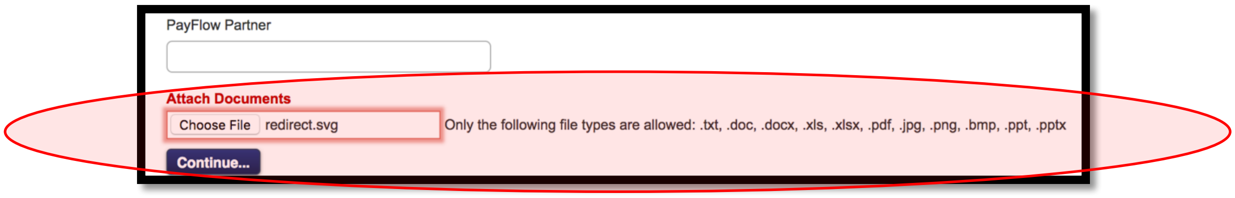
First Name *

Last Name *

Figure 5 Contact Support Page without Login

At those ticket submission activity, the users can upload the supported documents that could be used to explains their issue more completely. In this situation, there are specific extension of the files that could be submitted by user, which are .txt, .doc, .docx, .xls, .xlsx, .pdf, .jpg, .png, .bmp, .ppt, and .pptx.

In simple, **.svg** file format is not allowed by system to be upload at the portal.


PayFlow Partner

Attach Documents

Choose File redirect.svg Only the following file types are allowed: .txt, .doc, .docx, .xls, .xlsx, .pdf, .jpg, .png, .bmp, .ppt, .pptx

Continue...

Figure 6 .SVG in not allowed to be Upload

But the problem exists when the Attacker could manipulate the process of ticket submission to conduct a client side Attack to targeted account. In this case, Attacker could use the "Ticket submission **without** login" process and **bypassing** the protection of **.svg upload**. Also, **the best thing in this scenario** is PayPal Technical Support Portal didn't validate the content of the .svg file too so Attacker could modify the content with the client side script that they would like to inject. Here is the example of the content that we use as the PoC (will explain later):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<svg onload="window.location='http://www.google.com'"
xmlns="http://www.w3.org/2000/svg"> </svg>
```

Table 1 URL Redirection to Another Site Payload

Special note: Once more, very thanks to @BruteLogic for his very cool research related this thing that could be found at: https://brutelogic.com.br/blog/file-upload-xss/

2.3. 0x01 – Mind Map of Ticket Submission at PayPal Tech Support Portal

For completing to understanding the process of ticket submission, here are the simple mind map with few little explanation that could be help:

2.3.1. The **figure** below shown the normal process of ticket submission at TechSupport Portal. In this case, the scenario is "**Ticket submission with Login**". Everything looks normal just like the common flow of support portal.

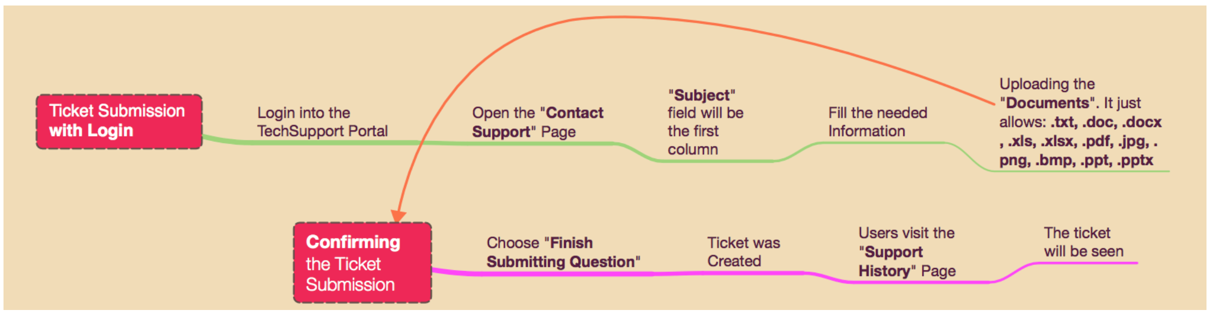

Ticket Submission with Login
Login into the TechSupport Portal
Open the "Contact Support" Page
"Subject" field will be the first column
Fill the needed Information
Uploading the "Documents". It just allows: .txt, .doc, .docx, .xls, .xlsx, .pdf, .jpg, .png, .bmp, .ppt, .pptx
Confirming the Ticket Submission
Choose "Finish Submitting Question"
Ticket was Created
Users visit the "Support History" Page
The ticket will be seen

Figure 7 Ticket Submission with Login

As has been described earlier, at those part, users are allow to upload the white list extension as their supported documents.

2.3.2. And the figure below also shown the process of ticket submission at TechSupport Portal with different condition. In this case, the scenario is "**Ticket submission without Login**". So, at this part, the users should fill the email address that they would like to use to interact with the submitted ticket.

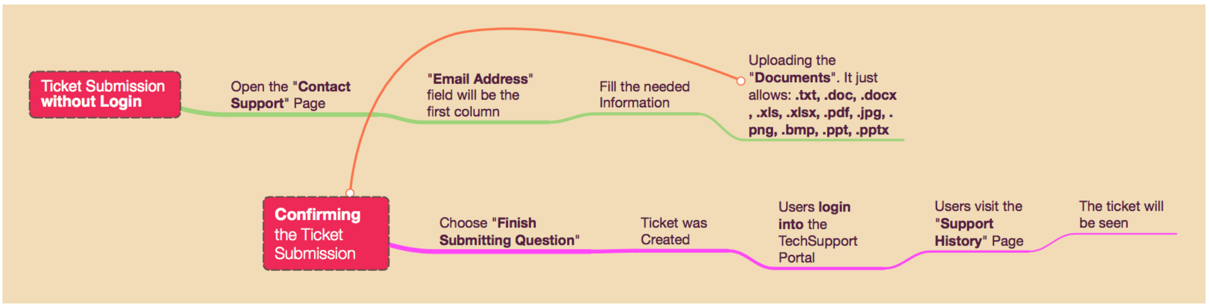

Ticket Submission without Login
Open the "Contact Support" Page
"Email Address" field will be the first column
Fill the needed Information
Uploading the "Documents". It just allows: .txt, .doc, .docx , .xls, .xlsx, .pdf, .jpg, . png, .bmp, .ppt, .pptx
Confirming the Ticket Submission
Choose "Finish Submitting Question"
Ticket was Created
Users login into the TechSupport Portal
Users visit the "Support History" Page
The ticket will be seen

Figure 8 Ticket Submission without Login

After the question has been submitted, then the user itself should login into the portal to interact with their own question.

III. SUMMARY OF ISSUE

As it has been described before, the security problem in this report is related with a vulnerability that "allows" an Attacker to be able to "store" an injected script (client side scripting) which could be used for few attack scenarios (like redirecting a user to a phishing page or even downloading a malware). The problem exist since there is a **validation issue** at their content checking (for technical support portal) and at their file name (for brand central portal). This condition also supported with **authorization issue at both of application.**

IV. PROOF OF CONCEPT

The proof of concept related this one is a little tricky since basically this XSS issue was a self-XSS. For completing the explanation, we do try to give simple mind map that could explain the attack method.

4.1. **0x01 – Proof of Concept at Technical Support Portal**

4.1.1. When we back into the **point #2.3.2** at this report, we will realize if there is an authorization issue at this portal. In other words, any anonymous user could try to submit a ticket to the registered user at the portal. All they need just **"know" the registered email ID**.

So basically, we could use this authorization issue to create a new ticket and upload the malicious .svg file to trigger the XSS at the target. Since the kind of XSS is stored XSS, then the Attacker just need to wait until the target open the created ticket.

Another interesting part is, since there is no CAPTCHA at this ticket submission feature, then the Attacker could automatically brute the email field with the list of emails that they want. (yes, there is a token that will expired in some of period, but at least we could use it).

Note: Probably this method is one of the business process that has been thinking deeply by PayPal so they didn't fix this kind of ticket submission method (All PayPal do is fixed the XSS problem).

4.1.2. At this step, we should learn the flow of the attack first before directly go to the proof of concept. The figure below shown the method that used by Attacker to tricks the victim. In this case, the Attacker should uses the process of "**Ticket submission without Login**" and "**bypass the file upload protection"** to upload the malicious .svg file

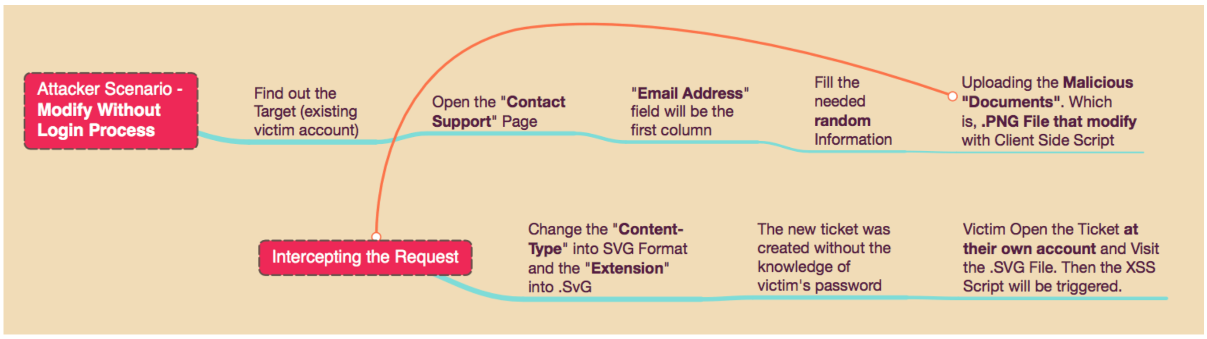

Attacker Scenario - Modify Without Login Process
Find out the Target (existing victim account)
Open the "Contact Support" Page
"Email Address" field will be the first column
Fill the needed random Information
Uploading the Malicious "Documents". Which is, .PNG File that modify with Client Side Script
Intercepting the Request
Change the "Content-Type" into SVG Format and the "Extension" into .SvG
The new ticket was created without the knowledge of victim's password
Victim Open the Ticket at their own account and Visit the .SVG File. Then the XSS Script will be triggered.

Figure 9 Mind Map of Attack

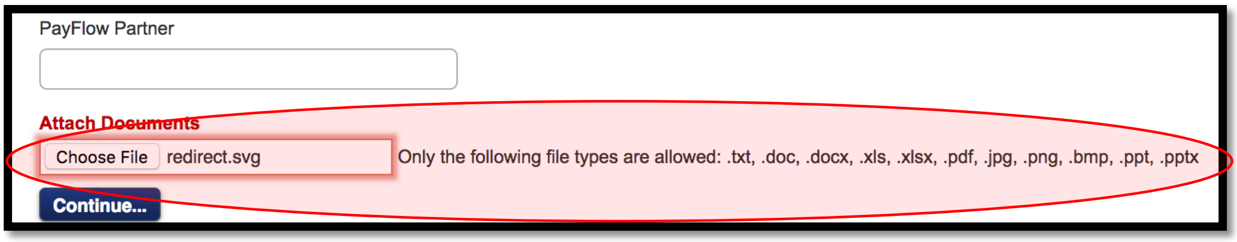
So, just like the flow of "ticket submission **without** login" part, the email address that should be fulfilled by Attacker is the **existing victim's email account** at TechSupport Portal (see the point #4.1.1).

4.1.3. After everything was filled, then the next thing that Attacker should do is upload the malicious .svg file that contains the client side scripting. Here is the sample of the client script that used as a PoC purpose:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<svg onload="window.location='http://www.google.com'"
xmlns="http://www.w3.org/2000/svg"> </svg>
```

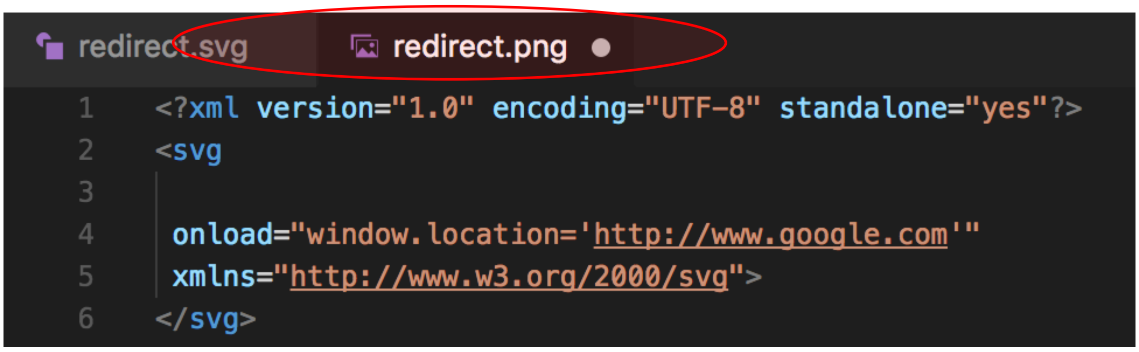
Table 2 URL Redirection to Another Site Payload

Please kindly note, we still need to do a bypass of the client side protection that explained previously (see the figure below), then the Attacker should change the extension of the .svg file to the allowed one such as .png.



Only the following file types are allowed: .txt, .doc, .docx, .xls, .xlsx, .pdf, .jpg, .png, .bmp, .ppt, .pptx

Figure 10 Restricted Files Only - .SVG File is not Allowed

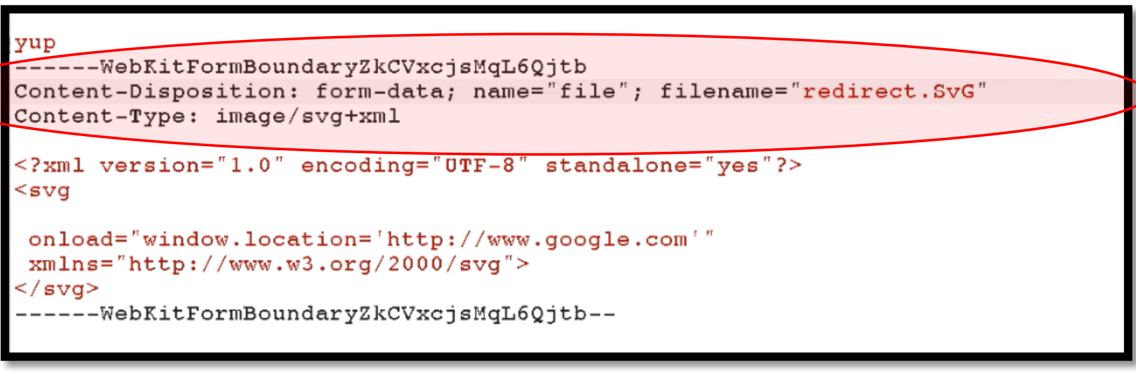


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<svg

 onload="window.location='http://www.google.com'"
 xmlns="http://www.w3.org/2000/svg">
</svg>
```

Figure 11 Change the Extension into the .png File Format

4.1.4. After we choose the malicious .png file that shipped with the client side script code, then the next thing is intercept the request before we send it to the server. In the interception request, then we should change the content-type and extension into the .svg file format and .svg extension.



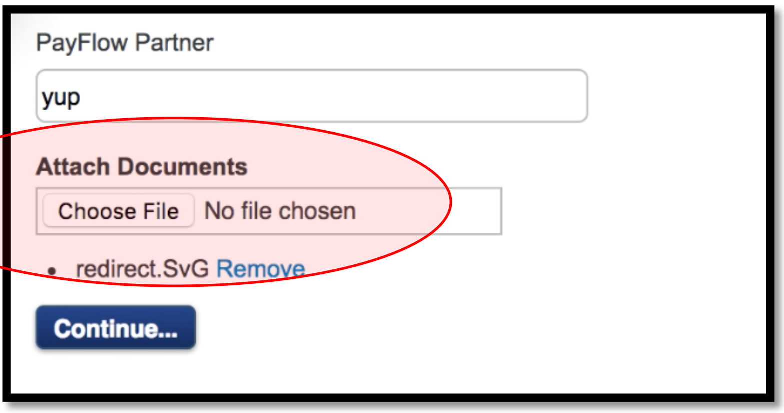
```
yup
------WebKitFormBoundaryZkCVxcjsMqL6Qjtb
Content-Disposition: form-data; name="file"; filename="redirect.SvG"
Content-Type: image/svg+xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<svg

 onload="window.location='http://www.google.com'"
 xmlns="http://www.w3.org/2000/svg">
</svg>
------WebKitFormBoundaryZkCVxcjsMqL6Qjtb--
```

Figure 12 Change the Content-Type and Extension of the File

4.1.5. After everything was setup, then forward the request. When the request has been forwarded, then we will get a success result if the .SvG extension has been uploaded.


PayFlow Partner

yup

Attach Documents

Choose File No file chosen

- redirect.SvG Remove

Figure 13 Success to Upload the .SvG File

4.1.6. The latest part is of course the Attacker just need to wait the victim open the ticket and opening the .svg file that has been uploaded previously by Attacker. When the .svg file is open, then the XSS will be triggered.

4.2. <u>**0x02 – Proof of Concept at Brand Central Portal**</u>

4.2.1. Similar with the **point #2.3.2** at this report, there is an authorization issue at this portal. In other words, any anonymous user could try to submit a ticket to the registered user at the portal. All they need just **"know" the registered email ID**.

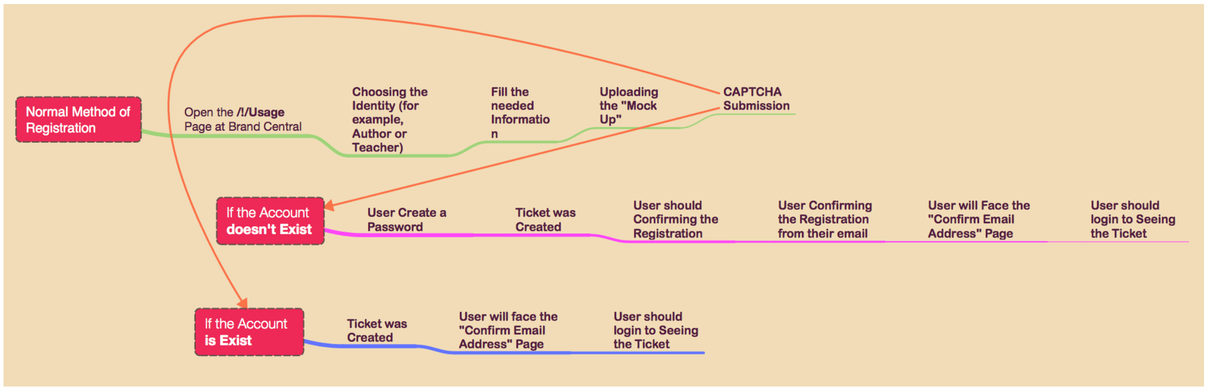

Normal Method of Registration
Open the /!/Usage Page at Brand Central
Choosing the Identity (for example, Author or Teacher)
Fill the needed Informatio n
Uploading the "Mock Up"
CAPTCHA Submission
If the Account doesn't Exist
User Create a Password
Ticket was Created
User should Confirming the Registration
User Confirming the Registration from their email
User will Face the "Confirm Email Address" Page
User should login to Seeing the Ticket
If the Account is Exist
Ticket was Created
User will face the "Confirm Email Address" Page
User should login to Seeing the Ticket

Figure 14 Normal Method of Registration / Ticket Submission at Brand Central Portal

The above shown the process of registering the account at BrandCentral. In this case, there are both scenario, which is: "**If the account doesn't exist**" and "**If the account is exist**";

- If the account exists, then the registering process will turn their function into the "**Ticket Submission without the knowledge of User's Password**";
- If the account doesn't exist, then the registering process will stand as normal function (which is registration).

4.2.2. At this step, we will use the same flow just like the Technical Support Portal. The figure below shown the method that used by Attacker to tricks the victim. In this case, the Attacker should use the process of "**Ticket submission without Login**" and "**upload any file name to be change later"** to trigger the XSS.

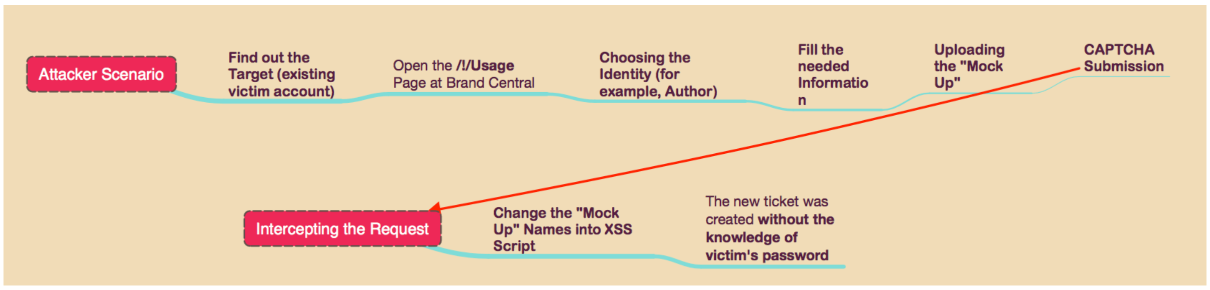
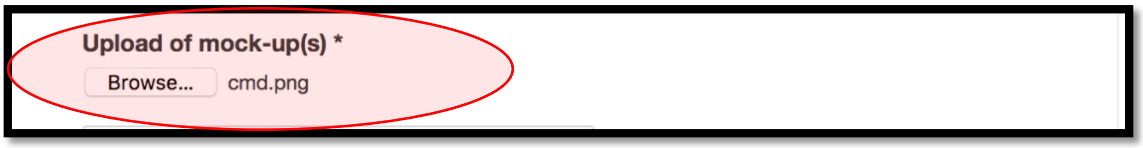

Attacker Scenario
Find out the Target (existing victim account)
Open the /l/Usage Page at Brand Central
Choosing the Identity (for example, Author)
Fill the needed Informatio n
Uploading the "Mock Up"
CAPTCHA Submission
Intercepting the Request
Change the "Mock Up" Names into XSS Script
The new ticket was created without the knowledge of victim's password

Figure 15 Mind Map of Attack – PayPal Brand Central

So, just like the flow of "ticket submission **without** login" part, the email address that should be fulfilled by Attacker is the **existing victim's email account** at Brand Central Portal (see the point #4.2.1).


Upload of mock-up(s) *

Browse... cmd.png

Figure 16 Uploading the File with "cmd" as the name

4.2.3. After everything was filled, then the next thing that Attacker should do before sending the request completely to server is intercepting the request to change the file name into the malicious script that we would like to trigger at the victim's side. For a sample proof to know this issue is works or not, we could use this simple pop up alert payload:

```
><img src=x onerror=prompt(1)>
```

Table 3 Pop Up Alert Payload

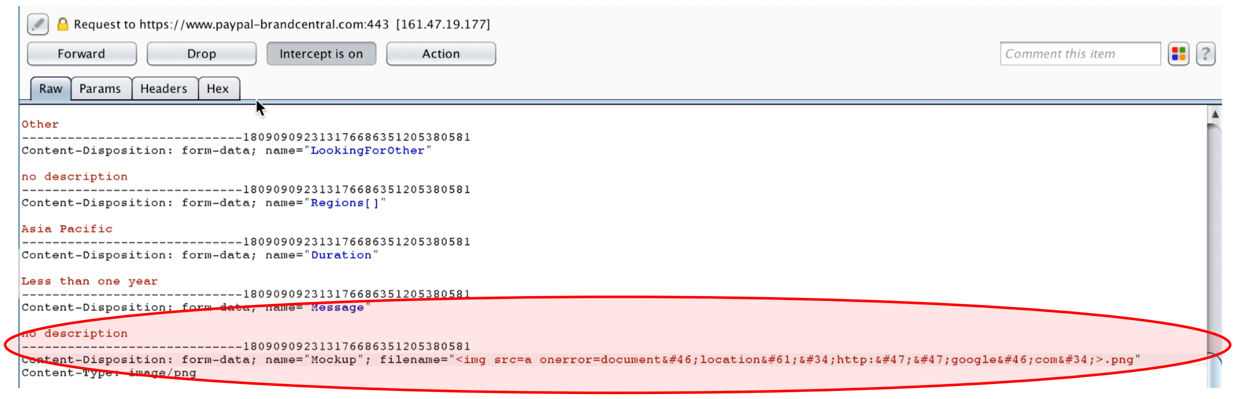
But since this pop up alert payload is not good enough to rising the risk level of issue, then we should try to change the payload (at least to the URL Redirection or downloading a program). At this situation, we found another issue, which is, **the file name at this portal was restrict from dot, quote, equal, and slash character → . " / =**

So we should try to bypass it by changing the payload into this one:

```
<img src=a onerror=document&#46;location&#61;&#34;http:&#47;&#47;google&#46;com&#34;>
```

Table 4 URL Redirection Payload with JS

And here is the final setup when we intercept the request:



```
Other
-----------------------------180909092313176686351205380581
Content-Disposition: form-data; name="LookingForOther"

no description
-----------------------------180909092313176686351205380581
Content-Disposition: form-data; name="Regions[]"

Asia Pacific
-----------------------------180909092313176686351205380581
Content-Disposition: form-data; name="Duration"

Less than one year
-----------------------------180909092313176686351205380581
Content-Disposition: form-data; name="Message"

no description
-----------------------------180909092313176686351205380581
Content-Disposition: form-data; name="Mockup"; filename="<img src=a onerror=document&#46;location&#61;&#34;http:&#47;&#47;google&#46;com&#34;>.png"
Content-Type: image/png
```

Figure 17 Intercepting the Request and Change the File Name

4.2.4. The latest part is of course the Attacker just need to wait the victim open the ticket. When the ticket is open, then the page will execute the script that injected at the file name.

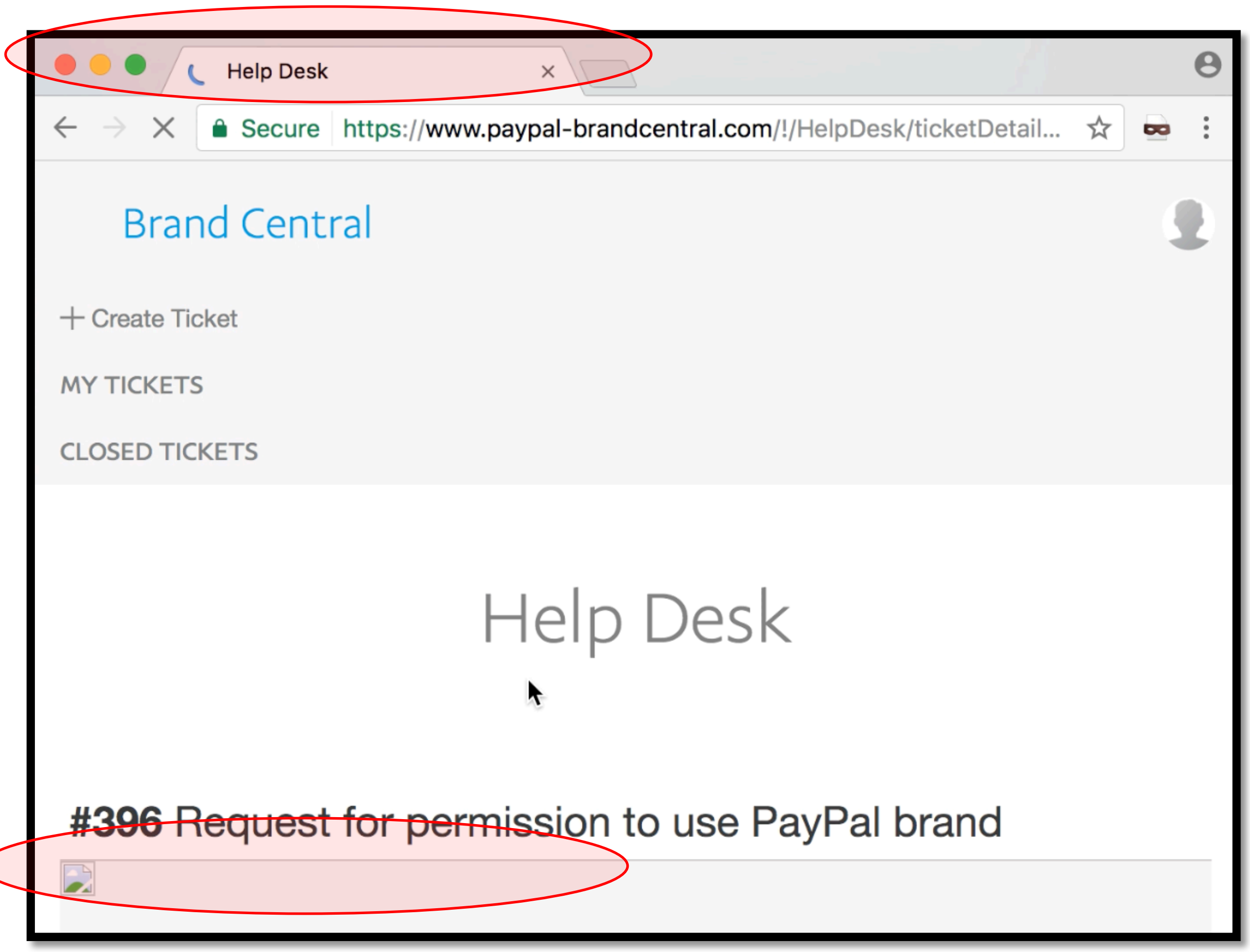

Help Desk
Secure
https://www.paypal-brandcentral.com/!/HelpDesk/ticketDetail...
Brand Central
+ Create Ticket
MY TICKETS
CLOSED TICKETS
Help Desk
#396 Request for permission to use PayPal brand

Figure 18 The Page will Execute the Script that Injected at the File Name

V. RECOMMENDATION

There are several recommendations that could be implemented for preventing these issues (completely):

5.1. Protecting the ticket submission process by disabling the ticket submission without login;

5.2. Checking the uploaded file **not only** at the client side part, but also when the request has been send into the server;

5.3. Checking the file name that uploaded into the server.

VI. ADDITIONAL INFORMATION

For completing the explanation, we upload the ~~unlisted~~ video at Youtube for both of scenario:

6.1. Stored XSS at Technical Support Portal: https://youtu.be/7Au-As7jrQs

6.2. Stored XSS at Brand Central Portal: https://youtu.be/XwynfNOxlIl

VII. LESSON LEARNED

7.1. Always try to find a way to triggering your self-XSS into the non-self XSS. Even the file couldn't be access for public (just like both of this case), then find a way to triggering it to the public or at least into the targeted victim.

Once more, @BruteLogic has released his very cool research related this that could be found at: https://brutelogic.com.br/blog/leveraging-self-xss/

7.2. PayPal has rewarded the issue that reported at Technical Support Portal and decided to mark the issue at Brand Central as "Not Actionable". The main issue at this situation is there is a lack of information that we sent to PayPal related the Brand Central (which is not with the mind map of process and mind map of Attack just like we did at the Technical Support Portal).

My point of view in this case are: Since there are so much thing that should be protected, the Security Team doesn't always know the process at each application that they have. Also, we should know if they will receive so many report from any researcher around the world. From both of those consideration, then the reporter always should create a complete report (explaining the process) or at least create the PoC video with the detail of step by step.

Please kindly note, since PayPal change the registration process at Brand Central, then the issue at Brand Central is not applicable anymore.