Poking Servers with {Facebook, Google, Mozilla Foundation, Adobe, Yahoo Developer Network}
whoami | head

- WebAppSec Consultant, Penetration Tester, Bug Bounty Hunter for Google, Facebook, Paypal, Mozilla and other bounty programs
- Null Security Community Bangalore Chapter Lead
- Work at a Big4 and have conducted several Penetration Tests all over the world.
history | less

Started hunting for bugs on several bug bounty programs for
history | less
dpkg -i investigate.deb

Found a facebook.com URL which fetched the <title> from a URL I could control

Tested for XSS, SQLi and LFI/RFI

Setup a HTTP server with port 8080 exposed to the Internet

Used http://myserver:8080 as an argument, mistyped the port number as 808

Facebook displayed an error that hinted on the port being closed, tested with other open and closed ports
up:me
cut –d " " –f2

Realized I could port scan Internet facing servers using verbose distinct errors from facebook
cat /etc/issue

Facebook was using underlying server side code to open socket connections to remote servers to download content

Friendly error messages were being sent to the client for failed socket connections at the web application level

There was no proper data handling for non HTTP streams, which was causing the application to behave unexpectedly
mail -s 'Bug!' sec@fb.com < /dev/null

Reported the issue to Facebook who responded saying that they did not see how this was a problem
mail -s 'Bug!' sec@fb.com < /dev/null

Sent facebook a Proof of Concept python port scanner

Scanned some random servers on the Internet using the script

Facebook replied and acknowledged that this was a problem
STUMBLED UPON BUG

BUG BOUNTY!
Information for Security Researchers

If you're a security researcher, please review our responsible disclosure policy before re the Facebook Security Page for assistance.

If you believe you've found a security vulnerability on Facebook, we encourage you to l our best to quickly fix the problem.

Responsible Disclosure Policy

If you give us a reasonable time to respond to your report before making any informatio destruction of data and interruption or degradation of our service during your research, investigate you.

Thanks!

On behalf of our millions of users, we would like to thank the following people for making

- Riyaz Walikar
which category

Searched for any references to port scanning using web apps on the server side.

Searched for other attacks using this same technique.

Being the foremost knowledgebase for everything WebAppSec, searched the OWASP website as well.
XSPA – Cross Site Port Attacks

XSPA occur when a web application attempts to connect to user supplied URLs and does not validate backend responses received from the remote server.

XSPA allows an attacker to port scan servers and attack services (Internet facing as well as internal devices) while proxying the attack from another web application.
export vulnerability='XSPA'

XSPA – Cross Site Port Attacks
Consider an application that allows users to specify an external image URL.

The remote server on which the image resides has ports 22, 80 and 3306 open.

<table>
<thead>
<tr>
<th>Image File URL</th>
<th>Server Status &amp; Body Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>http://remote_server.com/image.png</td>
<td>200 OK – Image retrieved</td>
</tr>
</tbody>
</table>
export vulnerability='XSPA'

XSPA – Cross Site Port Attacks

Application displays verbose errors for failed socket connections, receives fixed length responses or delays response for a fixed length of time

Application does not verify received data from the remote server, if the connection was successful

Application does not blacklist internal IP addresses/URLs
Déjà vu
WHAT IF I TOLD YOU

XSPA = SSRF
Server
Side
Request
Forgery
comm /riyaz/xspa /deral/ssrf

Deral Heiland - Shmoocon 2008
Was able to attack internal network using web portlets
SSRF via URL parameters – GET & POSTs (mostly GETs)

Alexander Polyakov - 2012
Attacking internal networks using SAP applications
SSRF via XML eXternal Entity (XXE) attacks

OWASP - ???
No mention of SSRF, although contains references to XXE
An attacker generates an HTTP request of the form


The vulnerable server then on behalf of the attacker makes a GET request to the internet server for the image.png

The image is downloaded to the web server and then a local link to it is sent to the attacker

Image (Response)
An attacker generates an HTTP request of the form:

```
```

The vulnerable server then on behalf of the attacker makes a GET request to the locally accessible server for the img.png file. Since a GET request generates an HTTP request, any banners and/or errors in the response can be displayed raw.

Web applications may then generate specific error messages or display raw errors received. For example:

```
```
find . -print | xargs grep 'logic'
<?php
if (isset($_POST['url']))
{
$content = file_get_contents($_POST['url']);
$filename = './images/'.rand().'img1.jpg';
file_put_contents($filename, $content);
echo $_POST['url']."</br>";
$img = "<img src=""'.$filename.'""/>";
}
echo $img;
?>
<?php

function GetFile($host, $port, $link)
{
    $fp = fsockopen($host, intval($port), $errno, $errstr, 30);
    if (!$fp) {
        echo "\$errstr (error number \$errno)\n";
    } else {
        $out = "GET \$link HTTP/1.1\n";
        $out .= "Host: \$host\n";
        $out .= "Connection: Close\n\r\n";
        $out .= "Accept-Language: en-us, en; q=0.5\n";
        $out .= "\r\n";
        fwrite($fp, $out);
        $contents='';
        while (!feof($fp)) {
            $contents.= fgets($fp, 1024);
        }
        fclose($fp);
        return $contents;
    }
}
?>
sudo demo &
cat /xspa/other_attacks

Attackers can access internal applications and perform URL based attacks (SQLi, Parameter manipulation etc.)

Since the GET /<data> part is controlled by the attacker, it would be possible to attack services and execute code on internal systems

Denial of service attacks on internal services
sudo demo &
cat popular_servers | ./poke

Found XSPA/SSRF in

Facebook
Google
Adobe
Pinterest
Yahoo!
Apigee
Mozilla
Face.com
The first finding

Application specific response for open port above 1024
cat facebook

The first finding

Application specific response for open port below 1024
cat facebook

The first finding

Application specific response for closed port
cat Google

Google Webmasters – XSPA/SSRF

We weren’t able to verify your site: http://scanme.nmap.org/

We couldn’t find the verification meta tag.

Recommended: HTML tag

Add a meta tag to your site’s home page.

Application specific response for open HTTP Port
cat Google

Google Webmasters – XSPA/SSRF

We weren’t able to verify your site: http://scanme.nmap.org:22/
Your server returned an invalid response.

Recommended: HTML tag
Add a meta tag to your site’s home page.

Application specific response for open non-HTTP Port
cat Google

Google Webmasters – XSPA/SSRF

We weren't able to verify your site: http://scanme.nmap.org:24/

Recommended: HTML tag

Add a meta tag to your site's home page.

Application specific response for closed port
Application specific response for open HTTP port
Application specific response for open non HTTP port
Application specific response for closed port
cat yahoo_developer_network

HTML Page content received from remote server on Open HTTP Port
Non HTTP Service responds with banner – Open non HTTP Port
cat yahoo_developer_network

Non HTTP Service responds with banner – Open non HTTP Port
ls adobe*.flv | xargs vlc
patch -p1 < /var/xspa/fixes

Response Handling - implement server side validation of responses received from remote resources

Error handling and messages - Display generic error messages to the client in case something goes wrong.

Restrict connectivity to HTTP based ports - restrict connections to HTTP ports on the server

Blacklist IP addresses - Internal IP addresses, localhost specifications and internal hostnames should be blacklisted
cat /xspa/reading

• http://spl0it.wordpress.com/2010/12/02/internal-port-scanning-via-crystal-reports/
• http://www.shmoocon.org/2008/presentations/Web%20portals,%20gateway%20to%20information.ppt
• http://media.blackhat.com/bh-us-12/Briefings/Polyakov/BH_US_12_Polyakov_SSRF_Business_WP.pdf
• http://anantshri.info/articles/web_app_finger_printing.html
• http://www.sectheory.com/intranet-hacking.htm
• http://ha.ckers.org/weird/xhr-ping-sweep.html
• http://www.w3.org/Protocols/rfc2616/rfc2616.html

All images are the property of their respective creators.
A big thanks to Jeremiah, @makash, @w3bd3vil, @abh1sek, @prajalkulkarni, Alexander Polyakov and everybody at the @null0x00 community for their help with testing, verifying, building exploits and just being so cool!
Riyaz Ahemed Walikar
@riyazwalikar
http://www.riyazwalikar.com