

BY — BENJAMIN HARRIS — ALIZ HAMMOND — SEP 11, 2024

We Spent \$20 To Achieve RCE And Accidentally Became The Admins Of .MOBI

 | Labs

WHOIS Adventures

We Spent \$20 To Achieve RCE And Accidentally Became The Admins Of .MOBI

Vulnerability Research

Welcome back to another watchTower Labs blog. Brace yourselves, this is one of our most astounding discoveries.

Summary

What started out as a bit of fun between colleagues while avoiding the Vegas heat and \$20 bottles of water in our Black Hat hotel rooms - has now seemingly become a major incident.

***Things are constantly
happening***

@nocturnaltrashposts



***and I would
like for it to
stop (please)***

We recently performed research that started off "well-intentioned" (or as well-intentioned as we ever are) - to make vulnerabilities in WHOIS clients and how they parse responses from WHOIS servers exploitable in the real world (i.e. without needing to MITM etc).

As part of our research, we discovered that a few years ago the WHOIS server for the .MOBI TLD migrated from whois.dotmobiregistry.net to whois.nic.mobi – and the dotmobiregistry.net domain had been left to expire seemingly in December 2023.

Putting thoughts aside, and actions first, we punched credit card details as quickly as possible into our domain registrar to acquire dotmobiregistry.net - representing much better value than the similarly priced bottle of water that sat next to us.

Our view was that as a legacy WHOIS server domain, it was likely only used by old WHOIS tools (such as phpWHOIS, which conveniently has an Remote Code Execution (RCE) CVE from 2015 for the parsing of WHOIS server responses – thus fitting our aim quite nicely).

Throwing caution into the wind and following what we internally affectionately refer to as our 'ill-advised sense of adventure' - on Friday 30th August 2024 we deployed a WHOIS server behind the whois.dotmobiregistry.net hostname, just to see if anything would actually speak to it actively.

The results have been fairly stunning since - we have identified 135000+ unique systems speaking to us, and as of 4th September 2024 we had 2.5 million queries. A brief analysis of the results showed queries from (but certainly not limited to):

- Various mail servers for .GOV and .MIL entities using this WHOIS server to presumably query for domains they are receiving email from,
- Various cyber security tools and companies still using this WHOIS server as authoritative (VirusTotal, URLSCAN, Group-IB as examples)

However, significant concern appeared on 1st September 2024 when we realised that numerous Certificate Authorities responsible for issuing TLS/SSL certificates for domains like 'google.mobi' and 'microsoft.mobi', via the 'Domain Email Validation' mechanism for verifying ownership of a domain, were using our WHOIS server to determine the owners of a domain and where verification details should be sent.

We PoC'd this with GlobalSign and were able to demonstrate that for 'microsoft.mobi', GlobalSign would parse responses provided by our WHOIS server and present 'whois@watchtowr.com' as an authoritative email address.

Effectively, we had inadvertently undermined the CA process for the entire .mobi TLD.

As is common knowledge, this is an incredibly important process that underscores the security and integrity of communications that a significant amount of the Internet relies upon. This process has been targeted numerous times before by well-resourced nation-states:

- [Hack Obtains 9 Bogus Certificates for Prominent Websites; Traced to Iran](#)
- [State-sponsored hackers in China compromise certificate authority](#)

While this has been interesting to document and research, we are a little exasperated. Something-something-hopefully-an-LLM-will-solve-all-of-these-problems-something-something.

As always, we remind everyone - if we could do this, anyone can.

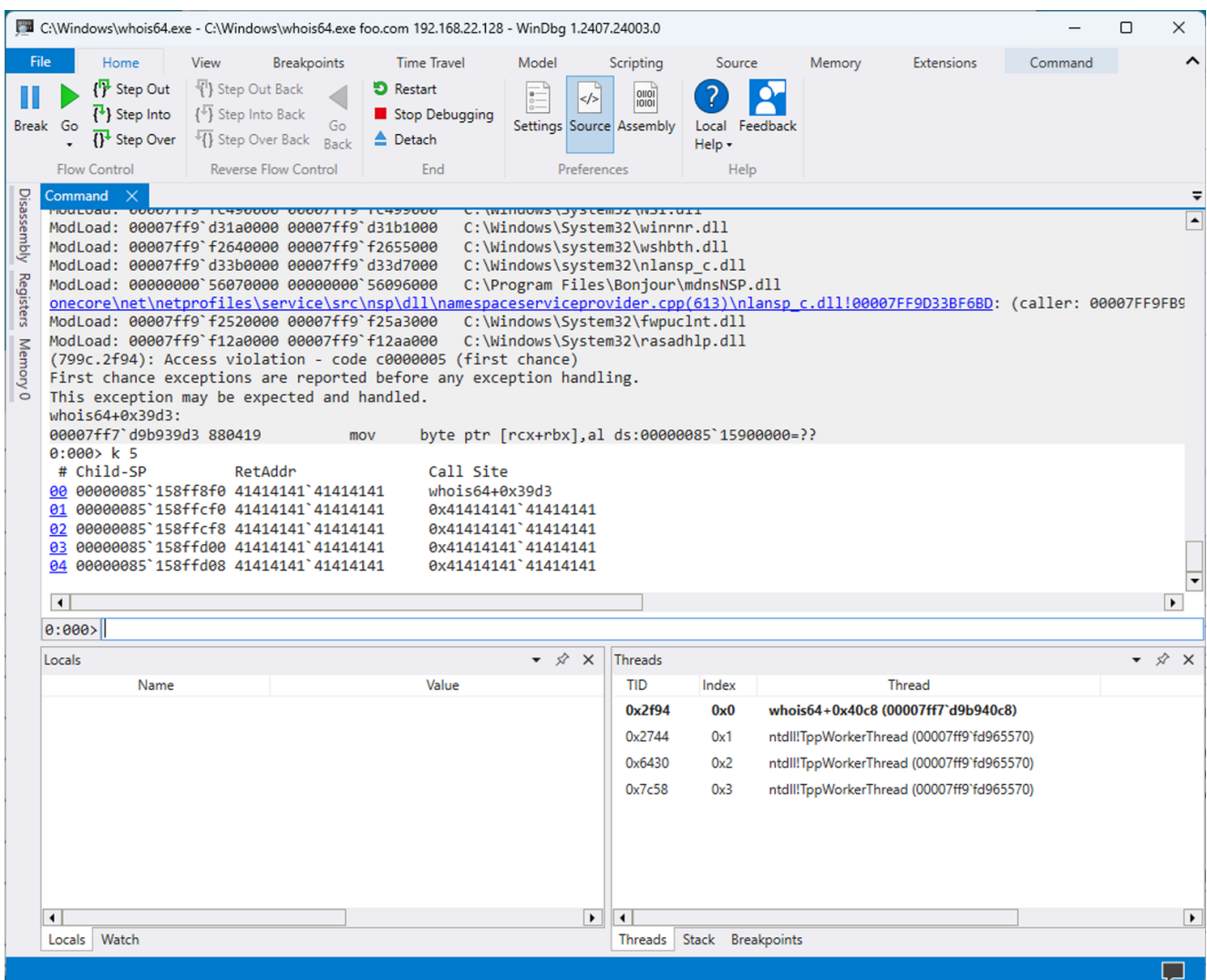
Onto the full story...

Setting The Scene

We're sure you're familiar with the old adage, 'it never rains but it pours'. That was definitely the case here, where we set out with the intention of just getting some RCE's to fling around, and ended up watching the foundation of secure Internet communication crumble before our eyes.

Before we get ahead of ourselves, though, let's start at the beginning, in which we decided to take a quick look at a WHOIS client. The protocol being some 50+ years old, we expected WHOIS clients to be constructed with the same brand of string as an enterprise-grade SSL VPN appliance, and so we took a naive shot and served up some A's.

```
# python3 -c "printf( 'Domain Name: ' + 'A' * 3000)" | nc -w1 -l wh
```

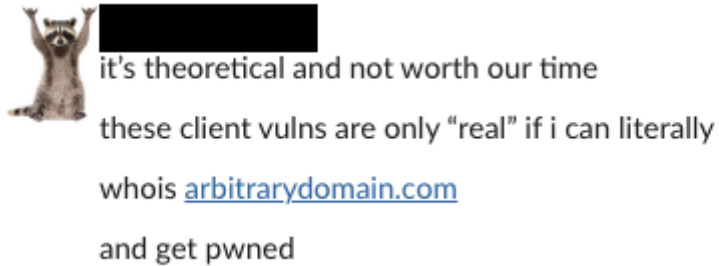


Haha, we were right. Funny.

This, at first glance, looks like an easily-exploitable crash. We were keen to find more bugs, and keenly started examining some other client implementations - but we were soon interrupted by some vocal killjoys naysayers.

They were quick to remind us that, to get to this state in our lab environment, we'd impersonated a WHOIS server, redirecting traffic from the usual server to our test server via `iptables`.

How realistic was this attack scenario, the naysayers asked?



We tried to silence the killjoy's naysayers and convince them our attack was plausible - we could find a registrar that allows us to set a Referral WHOIS value, or buy an IP range and control the range ourselves - but they suggested we spend more time doing, and less time playing academia.

The reality was that in order for an attacker to carry out an attack against a WHOIS client, they'd need one of the following:

- A Man-In-The-Middle (MiTM) attack, which requires the ability to hijack WHOIS traffic at the network layer - out of reach for all but the most advanced of APTs,
- Access to the WHOIS servers themselves, which is plausible but unlikely, or
- A WHOIS referral to a server they control.

These are effectively the preconditions of a nation-state or someone who is very comfortable compromising global TLD WHOIS servers in pursuit of exploiting clients.

You would, at this point, be forgiven for thinking that this class of attack - controlling WHOIS server responses to exploit parsing implementations within WHOIS clients - isn't a tangible threat in the real world.

We were left unsatisfied. We had located some shoddy code, but declaring it out of reach sounded like something you might bill a day rate for.

Perhaps there was another avenue for attack?

Collateral Damage In Pursuit Of RCE

The key to turning this theoretical RCE into a tangible reality is rooted in the tangled mess of the WHOIS system.

One of the biggest 'kludges' in the WHOIS system is the means of locating the authoritative WHOIS server for a given TLD in the first place.

Each TLD (the bit at the end of the domain), you see, has a separate WHOIS server, and there's no real standard to locating them - the only 'real' method being examining a textual list published by IANA. This list denotes the hostname of a server for each TLD, which is where WHOIS queries should be directed.

As you can imagine, maintainers of WHOIS tooling are reluctant to scrape such a textual list at runtime, and so it has become the norm to simply hardcode server addresses, populating them at development time by referring to IANA's list manually. Since the WHOIS server addresses change so infrequently, this is usually an acceptable solution.

However, it falls down in an ungraceful manner when server addresses change. With a little bit of legwork, we found that the WHOIS server for a particular TLD - `.mobi` - had been changed some years ago from the old domain whois.dotmobiregistry.net to a new server, at `whois.nic.mobi`.

Of course though, because the Internet is joined together by literal string and hopes/wishes at this stage, somebody had neglected to renew the old domain at `dotmobiregistry.net` meaning it was up for grabs by anyone with \$20 and an ill-advised sense of exploration.



We registered the domain, working on the theory that, while most client tooling would be updated to use `whois.nic.mobi`, most of the Internet population is still surprised when their 2011 SAP deployment gets popped, and thus WHOIS applications in production had a fairly decent chance of still referencing `whois.dotmobiregistry.net`.

Of course, this being the Internet, we got a little more than we bargained for.

Dear AWS customer,

We successfully registered the `dotmobiregistry.net` domain. We also created a hosted zone for the domain.

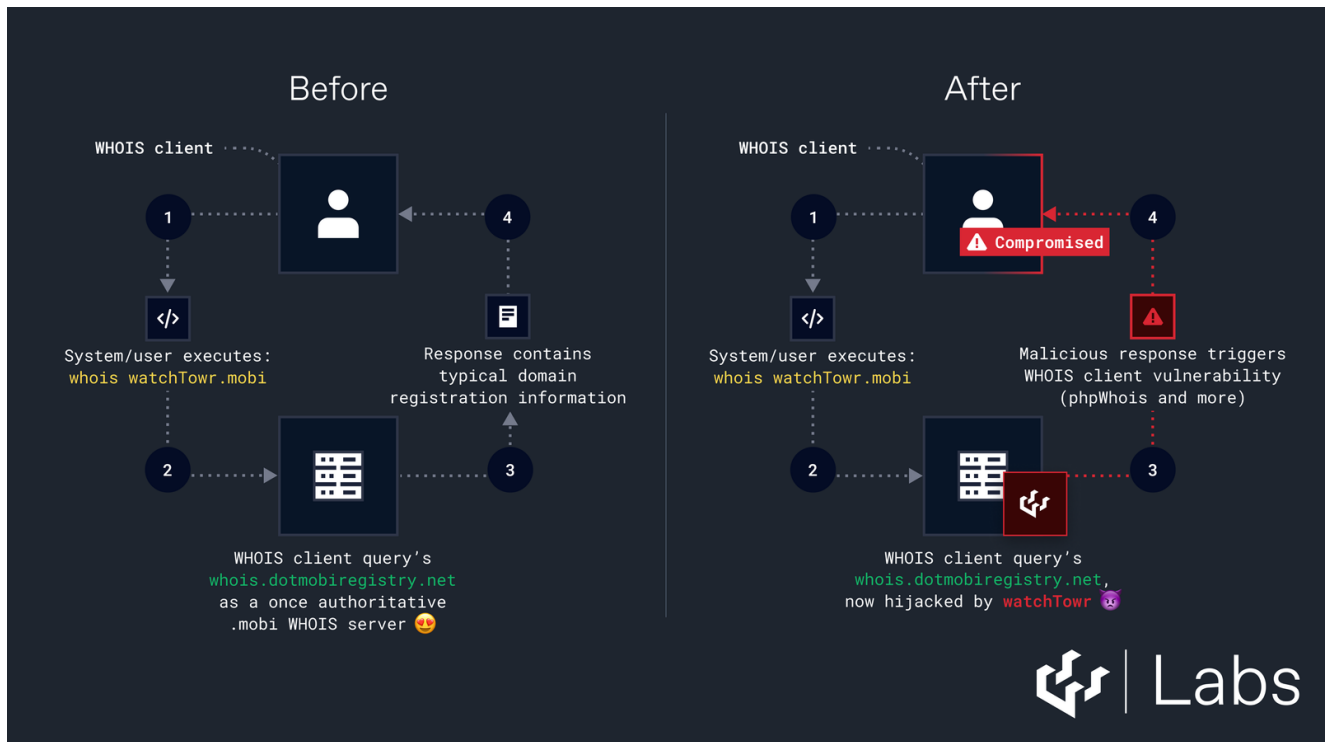
So What? It's Old

We soon realized the threat model for this attack had just changed.

Now that we control a WHOIS server, we were in the position to 'respond' to traffic sent by anyone who hadn't updated their client to use the new address (auto updates are bad, turn them off).

No longer do we require a Man-In-The-Middle attack, or some exotic WHOIS referral, to exploit a WHOIS client vulnerability - all we need to do is wait for queries to come in, and theoretically respond with whatever we want.

The pre-requisites for real-world exploitation now sat within what we deemed 'rough reality'.



Things were beginning to escalate.

We had set out to find some simple bugs in WHOIS client tooling, file for some CVEs, get them fixed.. but then we realised that once again we'd probably chewed off more than we intended and things were about to become worse - *much* worse.

Never Update, Auto-Updates And Change Are Bad

Unfortunately, there is a lot of Internet infrastructure which depends on the antiquated WHOIS protocol.

Starting off slow, we're now in a position to attack the [many websites](#) that run a WHOIS client and echo the results back to the user, injecting XSS or PHP `eval` payloads. Ethical (and legal) concerns prevent us from doing so, however - and we did not spend \$20 to get an XSS.

Of course, our original goal was to find and exploit some Oday in WHOIS clients, or some other system that embeds a WHOIS client (such as a spam filter), similar to the trivial memory corruption we found earlier.

Our biggest hurdle here - as alluded to above - was the simplicity of the WHOIS protocol itself, which is a simple text-based TCP data stream. With so little complexity, there seemed very little room for developers to make errors.

Ha.

Prior Art

To fully understand and look to leverage our new capability and adjusted threat model, we decided to examine the area's 'prior art' in exploitation, looking at historic attacks on WHOIS clients.

We were somewhat surprised that a search for [relevant CVE data](#) yielded relatively few results, which we attributed to the area being under-researched - the search return 26 CVE records.

[Once we discount the irrelevant results, we are left with only three bugs that are triggered by malformed WHOIS responses.](#)

This small number - three bugs since 1999 - makes it obvious to us that very little research has been done - likely due to the perception that any real-world exploitation comes with difficult prerequisites, such as control of a TLD WHOIS server.

But, there have been some interesting cases - just to give you a taste of where this is going.

phpWHOIS (CVE-2015-5243)

The first bug that our retrospective found was [CVE-2015-5243](#). This is a monster of a bug, in which the prolific phpWhois library simply *executes* data obtained from the WHOIS server via the PHP 'eval' function, allowing instant RCE from any malicious WHOIS server.

The vulnerable code snippet:

```
foreach ($items as $match => $field) {
    $pos = strpos($val, $match);

    if ($pos !== false) {
        if ($field != '') {
            $var = '$r' . getvarname($field);
            $itm = trim(substr($val, $pos + strlen($match)));

            if ($itm != '')
                eval($var . '=' . str_replace('"', '\\\\"', $itm))
        }

        if (!$scanall)
            break;
    }
}
```

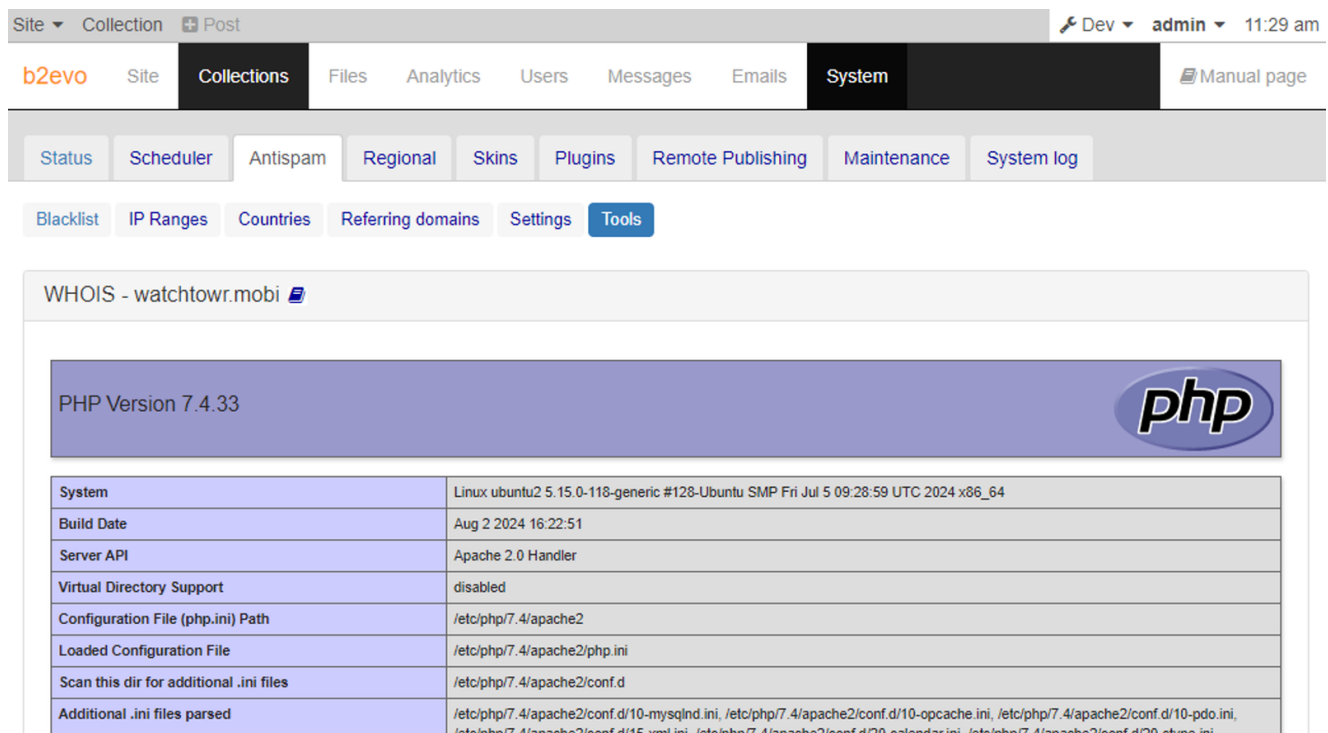
What's going on here?

The important item is the juicy `eval` statement in the middle of the snippet, which is fed data returned from the WHOIS server.

While it attempts to escape this data before it evaluates it, it does so imperfectly, only replacing " with the escaped form, \\" . Because of this, we can sneak in our own PHP code, which is then executed for us.

[Netitude's blogpost](#) lays out all the details, and even provides us with exploitation code - `";phpinfo();//` - is enough to spawn a `phpinfo` page.

We tried this out on an application that uses `phpWhois`, purely to demonstrate, and it worked swimmingly:



WHOIS - watchtowr.mobi

PHP Version 7.4.33

System	Linux ubuntu2 5.15.0-118-generic #128-Ubuntu SMP Fri Jul 5 09:28:59 UTC 2024 x86_64
Build Date	Aug 2 2024 16:22:51
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/apache2
Loaded Configuration File	/etc/php/7.4/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/apache2/conf.d
Additional .ini files parsed	/etc/php/7.4/apache2/conf.d/10-mysqld.ini, /etc/php/7.4/apache2/conf.d/10-opcache.ini, /etc/php/7.4/apache2/conf.d/10-pdo.ini, /etc/php/7.4/apache2/conf.d/15-xml.ini, /etc/php/7.4/apache2/conf.d/20-calendar.ini, /etc/php/7.4/apache2/conf.d/20-ctype.ini

Clearly this is a powerful bug - the best part being that `phpWhois` hardcodes our newly found `whois.dotmobiregistry.net` in vulnerable versions (it's old, but at a cursory glance no-one appears to have ever updated `phpWhois`).

What other historic artefacts could we find, though?

Fail2Ban (CVE-2021-32749)

As we continued to examine historic client-side bugs, we came across [CVE-2021-32749](#). This one is again a pretty nasty bug, this time in the ever-popular `fail2ban` package. It's a command injection vulnerability, a vulnerability class keenly sought by attackers due to its power and ease of exploitation.

As you may know, if you have administered a `fail2ban` server, the purpose of `fail2ban` is to monitor failed login attempts, and prevent bruteforce or password-guessing attacks by blocking hosts which repeatedly fail to log in.

Being the polished package it is, it also includes the ability to email an administrator when an IP address is banned, and - very helpfully - when it does so, it will enrich the email with information about who owns the banned IP address.

This information is gleaned from - yeah, you guessed it! - our friend WHOIS.

Unfortunately, for some time, the output of the WHOIS client wasn't correctly sanitized before being passed to the `mail` tool, and so a command injection bug was possible.

```
102 102 # Notes.: Your system mail command. Is passed 2 args: subject and recipient
103 103 # Values: CMD
104 104 #
105 105 - mailcmd = mail -s
105 105 + mailcmd = mail -E 'set escape' -s
106 106
107 107 # Option: mailargs
108 108 # Notes.: Additional arguments to mail command. e.g. for standard Unix mail:
```

Fortunately - or unfortunately, if you're an attacker - because `fail2ban` runs a WHOIS query on the *IP address* rather than, for example, a *domain name specified in the PTR record of an IP address* of blocked hosts - this attack is not within reach still based on our newly found capability.

For those that control a WHOIS server that is queried for IP addresses, though, exploitation is simple - simply attempt to unsuccessfully authenticate to a server via SSH a few times to trigger a ban, and once `fail2ban` queries the WHOIS server for information on your IP address - serve a payload wrapped in backticks.

Reality check

So, the burning question on our minds - can we actually exploit these bugs, *right now*?

Well, at this stage, our view was fairly pessimistic in terms of achieving real-world impact. We saw the following pre-requisites:

- The WHOIS client must be querying an old authoritative .MOBI WHOIS server and thus by definition, has not been working for *quite a while*
- To achieve client-side code execution (i.e. compromise) via a WHOIS client vuln - the only public option available to us was disclosed in 2015 and appears to have been rectified in 2018 - likely due to the perceived lack of real-world exploitation mechanisms.

Meh. Our gut feeling remained that most of the Internet and those in the sane world would logically be querying the new .mobi authoritative WHOIS server `whois.nic.mobi`, rather than the decommissioned dotmobiregistry.net (which we now controlled).

“Surely no large organisations would still reference the old domain”, we thought to ourselves.

Kill WHOIS With Fire

Without skipping a beat and really not considering the consequences, we set up a WHOIS server beneath our new domain at `whois.dotmobiregistry.net`, and logged incoming requests. We specifically focused on two things:

- Source IPs (so we can perhaps begin to work out who exactly was querying an outdated server), and,
- The queried domain (because again, this may give off some clues).

We threw together the [lglass](#) server to respond to WHOIS requests that found their way to our WHOIS server, and returned:

- ASCII art (we were relatively refrained here, but it was a priority)
- Fake WHOIS details indicating watchTower as the owner for every queried entity.

As this was our private server, we included a request for queries to cease (after all, they were unauthorised).

A quick test directly to our new WHOIS server showed that all was working as expected, with the following response provided for a query about `google.mobi`:

```
$ whois -h whois.dotmobiregistry.net google.mobi
```

```

%
%
%
%
%      .....
%      ...:##@*-. ....
%      .##@#@#@#@#@#-...
%      :@#@#@#@#@*:*:...
%      :@#@#@#@+...
%      ..... :@#@#@#@#:. ....
%      ...:=%@%+... :@#@#@#@#:. ....:....
%      ...+%@#@#@#@#@*:*:..-@#@#@#@#:. ....-*@#@#@#=. ....
%      .-@#@#@#@#@#@*:*:..+@#@#@#@#@#@#:. ....##@#@#@#@#@#@*:*:..
%      .-@#@#@#@#@#=#..-@#@#@#@#@#@#:. ....@#@#@#@#@#@*:-.....
%      .-@#@#@#@+..-@#@#@#@#@#=. ....@#@#@#@#=. ....
%      .-@#@#@#@+..-@#@#@#-.....:....@#@#@#@#=. ....
%      .-@#@#@#@+..-*:.....#@#@#@*-. ....@#@#@#@#=. ....
%      .-@#@#@#@+.. ..:;%@#@#@#@#@#@#@+..-@#@#@#@#=. ....
%      .-@#@#@#@*.....:..@#@#@#@#@#@*:*:..=%@#@#@#@#@#=. ....
%      .-@#@#@#@#@#=#..:..@#@#@#@#-.....:@#@#@#@#@#@#@+...
%      .-@#@#@#@#@#@#@#@#@*:*:..@#@#@#@#:. ....@#@#@#@#@%+...
%      ...:;%@#@#@#@#@#@#@*:*:..:..@#@#@#:. ....:@#@%-. ....
%      ...:=%@#@#@#@#@#@#@#@*:*:..@#@#@#@#:. ....-.....
%      .....-##@#@#@#@#@#@#@#@#@#:. ....
%      .....-*@#@#@#@#@#@#@#@#@#:. ....
%      .....+%@#@#@#@#@#@#:. ....
%      .....:##@#@#@#:.
%      .....:###:.
%      .....
%
%
%
%
% This is a private server operated by watchTowr. Please stop querying this system unless you are authorized to do so.
%
% The objects are in RPSL format.

% Information related to 'primary'

role:          primary
primary:       watchTowr
domain:        google.mobi
zone-c:        whois@watchtowr.com
admin-c:       whois@watchtowr.com
tech-c:        whois@watchtowr.com
abuse-c:       whois@watchtowr.com
mnt-by:       whois@watchtowr.com
nserver:       127.0.0.1
created:       2024-08-30T11:32:18Z
nic-hdl:       watchtowr
last-modified: 2024-09-03T04:57:26Z
source:       1

% Information related to '1.3.3.7/0'

ip_network:    1.3.3.7/0
role:          primary
primary:       watchTowr
zone-c:        whois@watchtowr.com
admin-c:       whois@watchtowr.com
tech-c:        whois@watchtowr.com
abuse-c:       whois@watchtowr.com
mnt-by:       whois@watchtowr.com
nserver:       1.3.3.7
created:       2024-08-30T11:32:18Z
nic-hdl:       idk
last-modified: 2024-09-01T06:08:17Z
source:       1

```

Nice.

Uh.....

Well, it's 2024 - absolutely no one has the ability to exercise patience, including ourselves.

So, we began just looking around the Internet for obvious locations that could be sending queries our way. Surely, we thought - *surely!* - the broken clients using an outdated server address wouldn't be in anything major, that we use every day?

- A significant number of domain registrars and WHOIS-function websites
 - domain.com
 - godaddy.com
 - who.is
 - whois.ru
 - smallseo.tools
 - seochekei.net
 - centralops.net
 - name.com
 - webchart.org

etc (you get the idea)

Search the WHOIS Database

google.mobi

Search

google.mobi is taken
We still might be able to get it for you. [See How](#)

Broker Service Fee
SG\$160.99  [Add to Cart](#)

WHOIS search results

```

%
%
%
%
% .....
% ...=#@*~.....
% ..#@@@@@@@@@#~...
% :@@@@@@@@@%*~...
% :@@@@@@@@@+~...
% ..... :@@@@@# : ...
% ...=%@%+~... :@@@@@# : .....
% ..+@@@@@@@@@*~...-@@@@@# : ...*@@@@@%= ....
% ..=@@@@@@@@@%+~...+@@@@@@@@@# : ..#@@@@@@@@@*~...
% =@@@@@@@@@%~...-@@@@@@@@@# : .@@@@@@@@@*~...
% =@@@@@@@@@+~...-@@@@@@@@@%~... ..@@@@@%~...
% =@@@@@@@@@+~...-@@@@@#~... :@@@@@%~...
% =@@@@@@@@@+~...-~#@@@@@~...@@@@@%~...
% =@@@@@@@@@+~... :%@@@@@@@@@+~...-@@@@@%~...
% =@@@@@@@@@*~... :%@@@@@@@@@*~...=%@@@@@@@@@%~...
% ..=@@@@@@@@@%~... :@@@@@#~...:@@@@@@@@@+~...
% ..=@@@@@@@@@%+~...:@@@@@#~...:@@@@@%+~...
% ....+%@@@@@@@@@*~...:@@@@@# : :@%-:~...
% ....=%@@@@@@@@@%~...%@@@@@# : :~...
% ....-#@@@@@@@@@*~... : ...
% ....-~@@@@@@@@@# : ...
% ....+%@@@@@# : ..
% ....:#@# : ..
% .....
%
%
%
%
% This is a private server operated by watchTowr. Please stop querying this system unless
you are authorized to do so.
%
% The objects are in RPSL format.
% Information related to 'primary'
role: primary
primary: watchTowr
domain: google.mobi
zone-c: whois@watchtowr.com
admin-c: whois@watchtowr.com
tech-c: whois@watchtowr.com
abuse-c: whois@watchtowr.com
mnt-by: whois@watchtowr.com
nserver: 127.0.0.1

```

Take a look at these alternate options:

Sorry, we don't have any alternative domain options available.

A screenshot of each WHOIS tool would become repetitive, but you get the idea.

- urlscan.io - "A sandbox for the web" - used our WHOIS server for .mobi, too. You can see the results by browsing to a page representing any .mobi domain ([like this one](#)).

bbc.mobi

BBC BBC Internet Services, UK, GB

Not observed on urlscan.io

Live Screenshot Hover to expand



Attention: This is a live snapshot of this website, we do not host or control it!

General Info

[Open in Search](#)

Geo	United Kingdom (GB) -
Created	May 11th, 2006
AS	AS2818 - BBC BBC Internet Services, UK, GB <small>Note: An IP might be announced by multiple ASs. This is not shown.</small>
Registrar	RIPENCC
Route	212.58.224.0/19 <small>(Route of ASN)</small>
PTR	bbc-vip144.lbh.bbc.co.uk <small>(PTR record of primary IP)</small>
IPv4	212.58.249.206 212.58.244.129 212.58.249.207 212.58.244.210
IPv6	2001:41c1:4007::bbc:6 2001:41c1:4007::bbc:7 2001:41c1:4007::bbc:8 2001:41c1:4008::bbc:5 2001:41c1:4008::bbc:6 2001:41c1:4007::bbc:5

No direct hits

Nothing is hosted on this domain

No incoming hits

Nothing talked to this domain

Disclaimer

The websites, domains, and other information displayed on this page are not managed by urlscan. Each website and its contents are solely the responsibility of their respective owners. urlscan can remove specific website scans from our platform, but urlscan does not bear responsibility for their content. If you have been a victim of fraud by any of the websites listed, please contact your local law enforcement to report it.

WHOIS for bbc.mobi

Updated Date:	31337-05-11T00:00:00Z
Creation Date:	2006-05-11T21:08:42Z
Registrar Registration Expiration Date:	31337-05-11T00:00:00Z
Registrar:	watchTowr
Registrar IANA ID:	470
Domain Status:	clientDeleteProhibited https://www.icann.org/epp#clientDeleteProhibited
Domain Status:	clientTransferProhibited https://www.icann.org/epp#clientTransferProhibited
Domain Status:	clientUpdateProhibited https://www.icann.org/epp#clientUpdateProhibited
Registry Registrant ID:	watchTowr
Registrant Name:	watchTowr
Registrant Organization:	watchTowr
Registrant Street:	watchTowr
Registrant City:	watchTowr
Registrant State/Province:	watchTowr
Registrant Postal Code:	watchTowr
Registrant Country:	watchTowr
Registrant Phone:	+1.123456789
Registrant Phone Ext:	

- [VirusTotal](#), the popular malware-analysis site, was querying us! A tool dedicated to the analysis of hostile code seemed like an opportunity for enjoyment.

virustotal.com/gui/domain/bbc.mobi/details

bbc.mobi

0 / 94
Community Score

No security vendors flagged this domain as malicious

Reanalyze Similar Graph API

bbc.mobi
news

Registrar: watchTower
Creation Date: 18 years ago
Last Analysis Date: 6 days ago

DETECTION DETAILS RELATIONS COMMUNITY

Categories

Comodo Valkyrie Verdict news

Last DNS records

Record type	TTL	Value
A	3600	212.58.249.207
A	3600	212.58.249.206
A	3600	212.58.244.129
A	3600	212.58.244.210
AAAA	3600	2001:41c1:4007:bbc:5
AAAA	3600	2001:41c1:4008:bbc:5
AAAA	3600	2001:41c1:4008:bbc:6
AAAA	3600	2001:41c1:4007:bbc:6
AAAA	3600	2001:41c1:4007:bbc:8
AAAA	3600	2001:41c1:4007:bbc:7

Whois Lookup

```
Tech City: watchTower
Tech Country: US
Tech Email: e4b791e2779e83d1s@watchtowr.com
Tech Organization: watchTower
Tech Postal Code: 31337
Tech State/Province: WA
Updated Date: 31337-05-11T00:00:00Z
abuse-c: e4b791e2779e83d1s@watchtowr.com
admin-c: e4b791e2779e83d1s@watchtowr.com
created: 2024-08-30T11:32:18Z
domain: bbc.mobi
last-modified: 2024-09-01T06:08:17Z
last-modified: 2024-09-01T06:08:39Z
mnt-by: e4b791e2779e83d1s@watchtowr.com
nic-hdl: idk
nserver: 1.3.3.7
source: 1
tech-c: e4b791e2779e83d1s@watchtowr.com
zone-c: e4b791e2779e83d1s@watchtowr.com
```

Sadly, VirusTotal doesn't render our ASCII art properly, but as you can see - VirusTotal is querying our makeshift WHOIS server for this global .TLD and presenting back the results. We were also pleased to see that VirusTotal updated their records of who owns `bbc.mobi` :

Passive DNS Replication (6) ⓘ			
Date resolved	Detections	Resolver	IP
2019-11-30	0 / 94	VirusTotal	212.58.244.210
2019-11-30	0 / 94	VirusTotal	212.58.244.129
2019-11-30	0 / 94	VirusTotal	212.58.249.206
2019-11-30	0 / 94	VirusTotal	212.58.249.207
2018-07-11	0 / 94	VirusTotal	212.58.246.226
2018-07-11	0 / 94	VirusTotal	212.58.246.227

Subdomains (1) ⓘ			
www.bbc.mobi	0 / 94	212.58.249.206	212.58.244.129

Historical Whois Lookups (5) ⓘ	
Last Updated	Registrar
+ 2024-09-01	watchTower
+ 2022-08-18	NOM-IQ Ltd dba Com Laude Nom-iq Ltd. dba COM LAUDE
+ 2022-01-06	NOM-IQ Ltd dba Com Laude Nom-iq Ltd. dba COM LAUDE
+ 2021-01-31	NOM-IQ Ltd dba Com Laude Nom-IQ Ltd DBA Com Laude
+ 2019-11-30	NOM-IQ Ltd dba Com Laude Nom-IQ Ltd DBA Com Laude

Graph Summary ⓘ

For anyone that has ever worked in offensive security, you occasionally get a sinking feeling where you realize something may be a little larger than expected, and you begin to wonder.. “what have we broken?”.

(Editors note: Technically, this should be ‘what *was* broken’, because people were querying our WHOIS server without authorisation and we’re very upset - get off our lawn!).

Well, with our WHOIS server clearly working - we figured we’d come back in a few days and see if anything at all reached out to us - giving us a good excuse to stare at a separate PSIRT response indicating a 2 year lead time to resolve a vulnerability.

Being insatiable and generally finding it hard to focus on anything longer than a TikTok video of a dog in a hat, we took a look to see how many unique IPs had queried our new WHOIS server after a few hours:

```
$ sqlite3 whois-log-copy.db "select source from queries" | sort | uniq |  
76085
```

Uh. Yes, that's correct - this is 76,000+ unique source IP addresses that have sent queries to our WHOIS server in just a couple of hours.

We were somewhat dismayed when, after leaving our server running for around two days, the poor little SQLite DB containing the logs ballooned to some 1.3 million queries! Clearly, we'd stumbled into something more major than we'd anticipated.

We threw the list of IPs at ZDNS and just sat back, as a relatively feeble way of doing attribution:

```
$ cat whois-src.txt | ./zdns PTR > ptr.txt
```

Anyway, the results were curious.

```
$ grep gov ptr.txt | {magic} | sort | uniq  
.gov-east-1.compute.amazonaws.com. "  
.gov.ar. "  
.gov.bd. "  
.gov.br. "  
.gov.il. "  
.gov.in. "  
.gov.ph. "  
.gov"
```

Great. We'd inadvertently *done a thing*.

Some other highlights of source hosts (not exhaustive, but just to give you some idea of just how bad this trash fire appeared to be):

- Mail servers! Lots and lots of mail servers.

Spam filters will often do WHOIS lookups on sender domains. We saw a bunch of these, ranging from the aptly-named [cheapsender.email](#) through to [mail.bdcustoms.gov.bd](#) - which appears to be part of the Bangladeshi government's infrastructure. Yikes! Theoretically, we could cause mayhem by serving responses indicating that the sending domain was a known spammer - and even more mayhem-worthy to start fuzzing the WHOIS parsing code to pop RCE on the mail servers themselves.

(We didn't)

- Leading on from that thought, what other [.gov](#) apparatus have we been queried by? Well, we found Brazil in our logs multiple times - for example, [antispam.ap.gov.br](#) and [master.aneel.gov.br](#) , and Brazil was not alone. We also found [.gov](#) addresses belonging to (but again not limited to):
 - Argentina,
 - Pakistan,
 - India,
 - Bangladesh,
 - Indonesia,
 - Bhutan,
 - Philippines,
 - Israel,
 - Ethiopia,
 - Ukraine,
 - USA.

Neat.

- Militaries (.mil)
 - Swedish Armed Forces, for example
- Universities (.edu)
 - All of them
- We even saw cyber security companies - [hey Group-IB, Detectify!](#) - query our WHOIS server (presumably doing *threat intel things* for .mobi domains).
 - We saw Censys query us for '[google.com](#)' and wondered if we'd get an APT number and a threat intel report shout-out if we'd been actively delivering payloads. Maybe we did? Check your boxen. (We didn't. Or did we?)

We're still trying to determine what software solutions are in play here/configured to query this WHOIS server for .mobi - let us know if you have any ideas.

Those who are nefariously minded likely realised what we saw as well - with .gov and other mail servers querying us each time they received an email from a .mobi domain - we could begin to passively determine who may be in communication.

This is not ideal. How do we fix this? Well, hold that thought - [IT GETS WORSE.](#)

Tales of TLS

TLS/SSL. Everyone knows it - it's that friendly little padlock icon in the address bar that assures you that your connection is secure. It's powered by the concept of *certificates* - sometimes used for HTTPS, sometimes used for signing your malware.

For example, say you're the owner of [watchTowr .mobi](#) . You want to secure communications to your web server by speaking TLS/SSL , so you go off to your favourite Certificate Authority and request a certificate (let's also pretend you haven't heard of LetsEncrypt).

The Certificate Authority will verify that you own the domain in question - [watchTowr .mobi](#) - and will then sign a private certificate, attesting to your identity as the owner of that domain. This is then used by the browser to ensure your communications are secure.

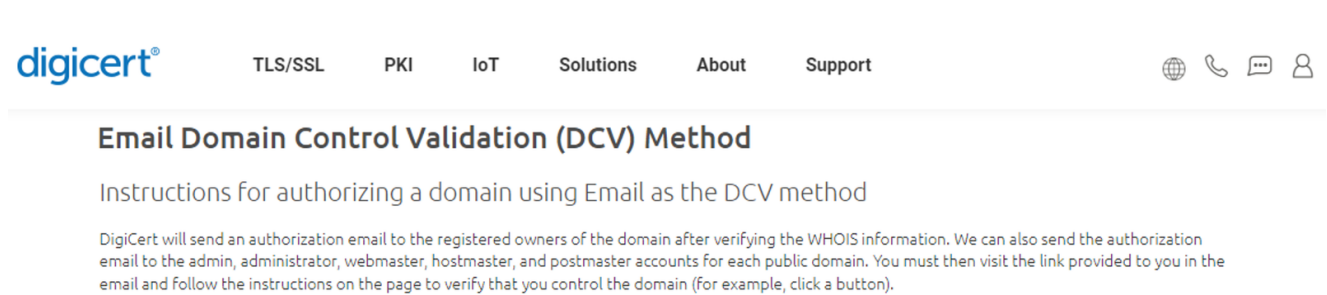
Speaking of LetsEncrypt, this thread is interesting - <https://community.letsencrypt.org/t/why-doesnt-lets-encrypt-use-whois-information-for-domain-validation/46287>). In this thread, forum posters detail why LetsEncrypt doesn't validate domains via WHOIS. Seems paranoid.

Anyway, what does this have to do with WHOIS, and what does it have to do with us?!

Well, it turns out that a number of TLS/SSL authorities will verify ownership of a domain by parsing WHOIS data for your domain - say `watchTowr.mobi` - and pulling out email addresses defined as the 'administrative contact'.

The process is to then send that email address a verification link - once clicked, the Certificate Authority is convinced that you control the domain that you are requesting a TLS/SSL cert for and they will happily mint you a certificate.

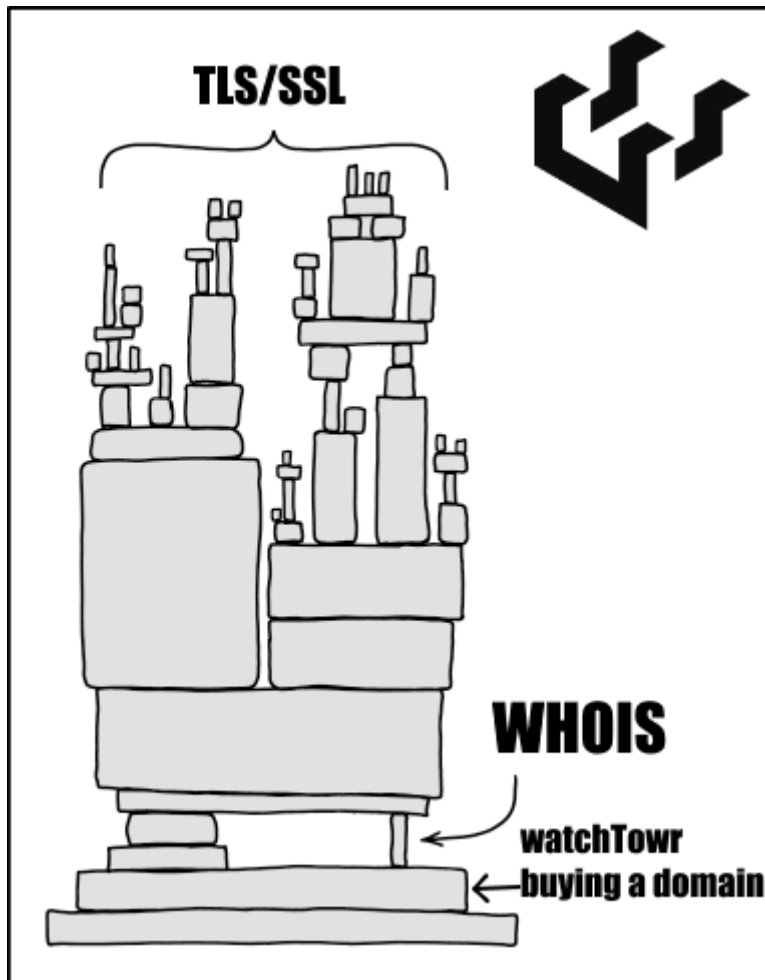
For example:



The screenshot shows the DigiCert website navigation bar with links for TLS/SSL, PKI, IoT, Solutions, About, and Support. On the right, there are icons for a globe, a phone, a chat bubble, and a user profile. Below the navigation bar, the page title is "Email Domain Control Validation (DCV) Method" and the subtitle is "Instructions for authorizing a domain using Email as the DCV method". The main content area contains a paragraph: "DigiCert will send an authorization email to the registered owners of the domain after verifying the WHOIS information. We can also send the authorization email to the admin, administrator, webmaster, hostmaster, and postmaster accounts for each public domain. You must then visit the link provided to you in the email and follow the instructions on the page to verify that you control the domain (for example, click a button)."

Perhaps you can see where we're going with this? *sobs*

If a TLS/SSL certificate authority is using our WHOIS server for `.mobi` domains, we can likely provide our own email address for this "Email Domain Control Validation" method.



Uh-oh. Is this a fringe feature supported only by two-bit, poor-quality certificate authorities?

No! Here's a sample of large TLS/SSL Certificate Authorities/resellers that support WHOIS-based ownership verification:

- Trustico
- Comodo
- SSLs
- GoGetSSL
- GlobalSign
- DigiSign
- Sectigo

Going through the normal order flow, we began cautiously - by generating a CSR (Certificate Signing Request) for the fictitious domain `watchTowr.mobi` - the logic being that as long as our WHOIS server was queried, whether or not the domain was real was irrelevant because we respond positively to absolutely every request including domains that don't actually exist.

```
# sudo openssl req -new -key custom.key -out csr.pem
```

```
You are about to be asked to enter information that will be incorpo  
into your certificate request.
```

```
What you are about to enter is what is called a Distinguished Name
```

```
There are quite a few fields but you can leave some blank
```

```
For some fields there will be a default value,
```

```
If you enter '.', the field will be left blank.
```

```
-----
```

```
Country Name (2 letter code) [AU]:SG
```

```
State or Province Name (full name) [Some-State]:Singapore
```

```
Locality Name (eg, city) []:Singapore
```

```
Organization Name (eg, company) [Internet Widgits Pty Ltd]:watchTow
```

```
Organizational Unit Name (eg, section) []:
```

```
Common Name (e.g. server FQDN or YOUR name) []:watchtowr.mobi
```

```
Email Address []:
```

```
Please enter the following 'extra' attributes
```

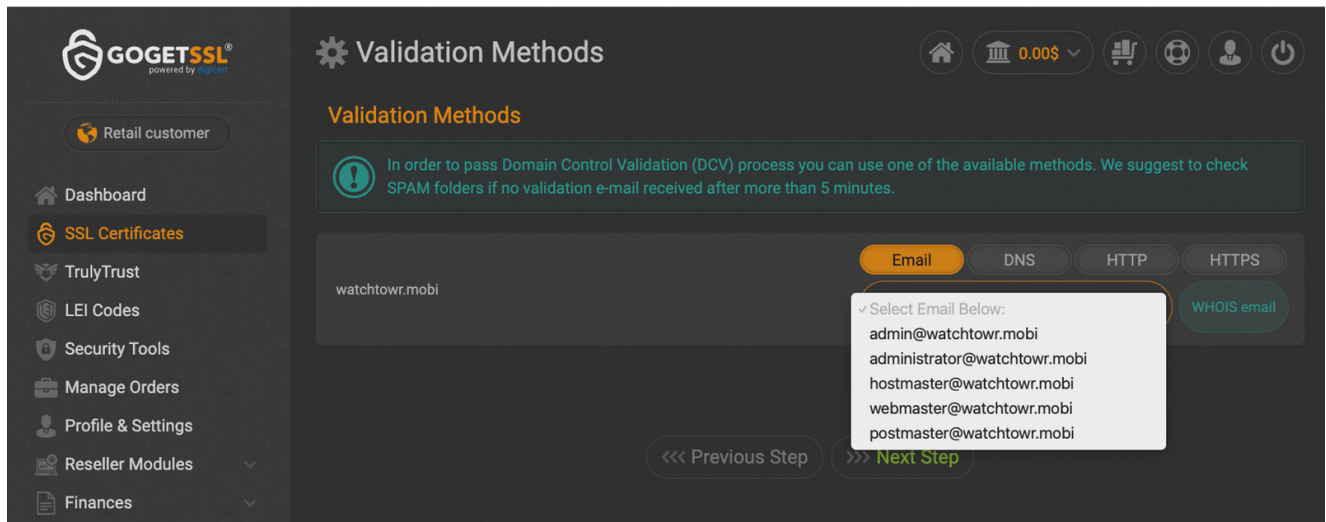
```
to be sent with your certificate request
```

```
A challenge password []:
```

```
An optional company name []:
```

We're not going to walk through each provider - for the purposes of illustration, we'll use GoGetSSL.

Once we upload our [watchTowr.mobi](#) CSR to GoGetSSL, it is parsed, and we continue. The indication of these placeholder email addresses indicates that WHOIS was *not* successful - instead of the email address that our WHOIS server is configured to respond with (`whois@watchtowr.com`), we're presented with only `@watchtowr.mobi` domains.



That's something of a relief.

The Certificate Authority has correctly determined that the domain watchTowr.mobi does not exist and thus if WHOIS is working as expected, no email addresses will be returned. We concluded that our newly set up WHOIS server was not being queried by the provider.

At least the world isn't ending. Right? (spoiler: *it actually was*)

We carried on trying a few other providers until a thought occurred.

The WHOIS protocol is extremely simple. Essentially it is a string blob returned in various formats depending on the TLD serving it. Each provider implements parsing in their own way. Perhaps, before we write off our theory, we should make sure this verification mechanism is actually working as it is supposed to.

So, we began again - choosing `microsoft.mobi` as a `.mobi` domain that appeared to follow a fairly typical WHOIS format (when using the current `.mobi` WHOIS server).

The screenshot below shows that the legitimate WHOIS record for `microsoft.mobi` was correctly parsed at Entrust, as the only email addresses available for validation were at the microsoft.com domain:

The screenshot shows the Entrust Certificate Services Starter interface. The top navigation bar includes 'Upgrade', 'Buy', and 'Messages' buttons. The main navigation menu has 'Home', 'Create', 'Certificates', 'Administration', and 'Help' options. On the left, a sidebar lists domain management options: 'Add Domains', 'Pending Domains', 'Active Domains', 'Expiring Domains', 'Inactive Domains', and 'All Domains'.

The main content area features a form for requesting new domains. It includes a text input for domain names (up to 500, separated by commas), a dropdown for 'Select a client', and another dropdown for 'Select a verification method'. Below the form is a table with columns for 'Domain', 'Client', 'Visibility (Private / Public)', and 'Verification Method'. A single row is visible for 'microsoft.mobi' with client 'watchTower' and visibility 'Public'.

A modal dialog titled 'Review email addresses for email verification method' is open over the table. It contains the following text:

For domain microsoft.mobi, select a single email address. An email will be sent, asking you to validate the domain. If none of these email addresses are suitable, select a different verification method.

 The dialog lists three categories of email addresses:

- Domain Administrator email addresses from your DNS record:**
 - azuredns-hostmaster@microsoft.com (SOA)
- Domain Administrator email addresses from your WHOIS record:**
 - domains@microsoft.com
 - msnhst@microsoft.com
- CA/Browser Forum standard domain permission email addresses:**
 - admin@microsoft.mobi
 - administrator@microsoft.mobi
 - webmaster@microsoft.mobi
 - hostmaster@microsoft.mobi
 - postmaster@microsoft.mobi

 At the bottom, there is a section for 'Choose a different verification method:' with a dropdown menu currently set to 'Email'. 'Cancel' and 'Submit Request' buttons are located at the bottom right of the modal.

While the WHOIS record for watchTower . mobi was not being parsed at all (indicating that Entrust was using the correct WHOIS server, and not ours):

The screenshot shows the Entrust Certificate Services Starter interface. The top navigation bar includes the Entrust logo, 'CERTIFICATE SERVICES Starter', and links for 'Upgrade', 'Buy', 'Messages', and 'Trusted'. Below this is a secondary navigation bar with 'Home', 'Create', 'Certificates', 'Administration', and 'Help'. On the left, a sidebar lists domain categories: 'Add Domains', 'Pending Domains', 'Active Domains', 'Expiring Domains', 'Inactive Domains', and 'All Domains'. The main content area features a form to request new domains, with a text input field for domain names (up to 500, separated by commas), dropdown menus for 'Select a client' and 'Select a verification method', and an 'Add to' button. Below the form is a table with columns for 'Domain', 'Client', 'Visibility (Private / Public)', and 'Verification Method'. A modal dialog titled 'Review email addresses for email verification method' is open, displaying instructions and a list of email addresses for the domain 'watchTowr.mobi'. The modal includes sections for 'Domain Administrator email addresses from your DNS record', 'Domain Administrator email addresses from your WHOIS record', and 'CA/Browser Forum standard domain permission email addresses'. The list of email addresses includes: admin@watchTowr.mobi, administrator@watchTowr.mobi, webmaster@watchTowr.mobi, hostmaster@watchTowr.mobi, and postmaster@watchTowr.mobi. At the bottom of the modal are 'Cancel' and 'Submit Request' buttons.

ENTRUST CERTIFICATE SERVICES Starter Upgrade Buy Messages Trusted

Home Create Certificates Administration Help

Add Domains Pending Domains Active Domains Expiring Domains Inactive Domains All Domains

To request new domains, enter or paste the new domain names below. After clicking **Submit Request**, you will be redirected to the **Pending Domains** page. Follow the instructions in the **Next Step** column. For more information, click [here](#).

Enter up to 500 domain(s), separated by commas Select a client Select a verification method Add to

Domain	Client	Visibility (Private / Public)	Verification Method
watchTowr.mobi	watchTowr	Public	Email

Review email addresses for email verification method

For domain watchTowr.mobi, select a single email address. An email will be sent, asking you to validate the domain. If none of these email addresses are suitable, select a different verification method.

Domain Administrator email addresses from your DNS record:

We did not find any domain administrator addresses in your domain DNS record. If you add these addresses, it may take 12 hours before we discover them.

Domain Administrator email addresses from your WHOIS record:

We did not find any domain administrator addresses in your domain WHOIS record.

CA/Browser Forum standard domain permission email addresses:

- admin@watchTowr.mobi
- administrator@watchTowr.mobi
- webmaster@watchTowr.mobi
- hostmaster@watchTowr.mobi
- postmaster@watchTowr.mobi

Choose a different verification method:

Cancel Submit Request

Looks good you think?



WRONG.

We skipped and hopped over to the next provider, GlobalSign. GlobalSign reported that they were unable to parse the WHOIS record of `microsoft.mobi` :

Select a Domain Verification Method

Please select from the following methods for **demonstrating domain control**.

Email Verification

We send an email to one of the addresses displayed below and you follow the instructions inside.

WHOIS Email Addresses

We couldn't find any email addresses in the WHOIS record for this domain. If this is an error and you'd like to verify using an email address in the WHOIS record, please choose any email address below and then contact our support team to change the email address.

Constructed Domain Email Addresses

- admin@microsoft.mobi
- administrator@microsoft.mobi
- hostmaster@microsoft.mobi
- postmaster@microsoft.mobi
- webmaster@microsoft.mobi

HTTP Verification

We provide a Domain Verification Code (DVC) and you place that DVC in a text file in a specific location on your website.

- Use HTTP verification

DNS Verification

We provide a Domain Verification Code (DVC) and you create a DNS record containing the DVC.

- Use DNS verification

Back

Continue

At this point, something clicked in our minds. Perhaps GlobalSign WAS querying our new WHOIS server - but the string returned by our WHOIS server was incompatible with GlobalSign's parsing?

We copied the `microsoft.mobi` output from the legitimate WHOIS server, made it our own, and loaded it into our own WHOIS server - updated to look like the following:

Select a Domain Verification Method

Please select from the following methods for **demonstrating domain control**.

Email Verification

We send an email to one of the addresses displayed below and you follow the instructions inside.

WHOIS Email Addresses

whois@watchtowr.com

Constructed Domain Email Addresses

- admin@microsoft.mobi
- administrator@microsoft.mobi
- hostmaster@microsoft.mobi
- postmaster@microsoft.mobi
- webmaster@microsoft.mobi

HTTP Verification

We provide a Domain Verification Code (DVC) and you place that DVC in a text file in a specific location on your website.

Use HTTP verification

DNS Verification

We provide a Domain Verification Code (DVC) and you create a DNS record containing the DVC.

Use DNS verification

Back

Continue

We want to be explicitly clear that we stopped at this point and did not issue any rogue TLS/SSL certificates to ourselves. This would undoubtedly create an incident, and require significant amounts of work by many parties to revoke and roll back this action.

Success!

The GlobalSign TLS/SSL certificate WHOIS domain verification system had queried our WHOIS server, parsed `whois@watchTowr.com` from the result, and presented it as a valid email address to send a verification email to, allowing us to complete verification and obtain a valid TLS/SSL certificate.

This is then blindingly simple:

- Set up a rogue WHOIS server on our previously authoritative hostname, responding with our own email address as an 'administrative contact'
- Attempt to purchase a TLS/SSL certificate for a .mobi domain we want to target (say, `microsoft.mobi`)
- A Certificate Authority will then perform a WHOIS lookup, and email *us* instead of the real domain owners [theory]
- We click the link, and.. [theory]
- ... receive an TLS/SSL cert for the target domain! [theory]

Now that we have the ability to issue a TLS/SSL cert for a .mobi domain, we can, in theory, do all sorts of horrible things - ranging from intercepting traffic to impersonating the target server. It's game over for all sorts of threat models at this point.

While we are sure some may say we didn't 'prove' we could obtain the certificate, we feel this would've been a step too far — so whatever.

One Last Thing

Please stop emailing us..

Subject:

General Inquiry

Message:

You have a scammer in your fold: This domain links back to you.

Freedom Mobile Billing Alert: Your account has reached 85% of its credit limit. To avoid any disruptions to your service, please make a payment at <https://frdm.mobi/myaccy>

Here We Go Again..

We hope you've enjoyed (and/or been terrified by) today's post, in which we took control of a chunk of the Internet's infrastructure, opened up a big slab of juicy attack surface, and found a neat way of undermining TLS/SSL - the fundamental protocol that allows for secure communication on the web.

We want to thank the UK's [NCSC](#) and the [ShadowServer](#) Foundation for rapidly working with us ahead of the release of this research to ensure that the 'dotmobiregistry.net' domain is suitably handled going forwards, and that a process is put in place to notify affected parties.

The dotmobiregistry.net domain, and whois.dotmobiregistry.net hostname, has been pointed to sinkhole systems provided by ShadowServer that now proxy the legitimate WHOIS response for .mobi domains.

We released this blog post to initially share our process around making the unexploitable exploitable and highlight the state of legacy infrastructure and increasing problems associated with abandoned domains - but inadvertently, we have shone a spotlight on the continuing trivial loopholes in one of the Internet's most vital encryption processes and structures - TLS/SSL Certificate Authorities. Our research has demonstrated that trust placed in this process by governments and authorities worldwide should be considered misplaced at this stage, in our opinion.

We continue to hold concern around the basic reality - we found this on a whim in a hotel room while escaping the Vegas heat surrounding Black Hat, while well-resourced and focused nation-states look for loopholes like this every day. In our opinion, we are not likely to be the last to find inexcusable flaws in such a crucial process.

Although subverting the CA verification process was by far the most devastating of impacts that we uncovered, it was by no means the limit of the opportunity available to us as we also found everything from memory corruptions to command injections. Our 'honeypot' WHOIS server gave us some interesting statistics, revealing just how serious the issue is, and a large amount of Internet infrastructure continues to query us instead of the legitimate WHOIS servers.

We do not intend to call out any specific organization or maintainer here - the prevalence of this issue and the statistics on hand show that this is not a pure-negligence or competence related issue - but a fundamental flaw in how these processes work together.

It's worth noting that all the above attacks that we were able to orchestrate given our takeover are also possible by any entity that is able to carry out MITM attacks - such as entities that control or can influence transit backbones. It would be very easy for an attacker with such access to fake WHOIS data for any domain, and thus obtain valid TLS/SSL certificates. Of course, there has been an insurmountable level of effort by major players to add transparency to this process over the years, and thus, 'pulling off' a heist of this scale has its operational hurdles.

At [watchTower](#), we passionately believe that continuous security testing is the future and that rapid reaction to emerging threats single-handedly prevents inevitable breaches.

With the watchTower Platform, we deliver this capability to our clients every single day - it is our job to understand how emerging threats, vulnerabilities, and TTPs could impact their organizations, with precision.

If you'd like to learn more about the [watchTower Platform, our Attack Surface Management and Continuous Automated Red Teaming solution](#), please get in touch.

PREVIOUS POST

Veeam Backup & Response - RCE With Auth, But Mostly Without Auth (CVE-2024-40711)