

Forum:[Hack Forums](#)

Thread: [Need some advice on a dropper](#)

PulseLink - April 5, 2024, 1:15 pm

I am coding a dropper, right now its 100% native, working on x32 and x64 bit systems, passed all checks on scanner.to as well as EDR.

Here is my issue, the file is about 9kb with UPX compression and currently:

- dynamically finds a drop folder in tmp or roaming, creates a new folder or adds to a small array of popular existing folders (Adobe, Microsoft, etc)
- in-memory execution after injection into explorer.exe and executing first without dropping to disk
note: we have also used direct memory execution where the shellcode uses network apis to download the exe and manually map into memory, similar to manual PE loading. In-memory execution with delays has proven to be the most successful against AV/EDR.
- Using the injection into explorer, it helps reduce suspicion as its a legitimate windows process.

I am now adding a few anti-debug/anti-vm methods, runtime encryption, and the file that is eventually dropped after execution (delay for EDR/AV), the file will be added to registry.

I would like to add a small web-panel to this to track downloads and geolocation, since i plan on selling builds, having a web panel potentially tracking 3-4 files and showing statistics on the downloads and geo information would be useful. As well as a way to automatically update the file and an av-scanner API so the user knows when the file should be rotated.

I just would like to hear some feedback, thoughts, features i should remove/add, and overall my question is: I personally was coding this for my own use to achieve a higher execution rate and track downloads, as well as easily rotate files. I think as a service this could be very useful to users and easy for them to setup.

I may release some of it open source, but i would like to hear some feedback on the general idea. I have found a stager has increased execution rates for my loaders a lot. Given the sub size and compatibility ive seen 20-30% increase in execution rate when loading bots from my traffic sources (landing pages, torrents, google ads, cloaking, and some other methods, but 100% unique, so a 20-30% increase for an already expensive loader is crazy)

here is a gif of it in action so far:

<https://imgur.com/a/r0LDqJL>

not much, but it surely has helped me

catLover78 - April 5, 2024, 1:51 pm

talked with developer and he showed me some of the code, windows cloud bypass as well as some of the functions are impressive.

for example the geo map, is an essential that every dropper nowadays should have, you can see how many clients are from what are, the speed is also amazing. tasks get executed nearly instantly, I was impressed.

also the webpage does have an amazing interface (if someone cares)
definitely gonna keep my eyes on the progress

good luck with the future development, I think this can be something like we have not seen before
tbh

PulseLink - April 5, 2024, 1:53 pm

(Apr 5, 2024 01:51 PM)catLover78 Wrote: [\[CITE\]](#) talked with developer and he showed me some of the code, windows cloud bypass as well as some of the functions are impressive.
for example the geo map, is an essential that every dropper nowadays should have, you can see how many clients are from what are, the speed is also amazing. tasks get executed nearly instantly, I was impressed.
also the webpage does have an amazing interface (if someone cares)
definitely gonna keep my eyes on the progress

good luck with the future development, I think this can be something like we have not seen before
tbh

ty for sharing your thoughts, right now i am working on some polymorphic-like string/class encryption on compilation to make sure bins continue to stay fud and bypass WD cloud.

Fludems - April 7, 2024, 2:03 pm

(Apr 5, 2024 01:53 PM)PulseLink Wrote: [\[CITE\]](#) ty for sharing your thoughts, right now i am working on some polymorphic-like string/class encryption on compilation to make sure bins continue to stay fud and bypass WD cloud.

Even though you don't want to hear it I'd use Laravel. You can implement all of this without ofcourse it'd just be a lot faster to ship and easier to maintain and update.

would like to add a small web-panel to this to track downloads and geolocation

Here I'd create a Rest API endpoint that accepts a HWID.

Assuming by dropping into the registry we meaning it has persistency I'd do the send a POST request to this endpoint with the HWID (Add more data if you need to for creating a record of a 'download')

Edit: I'm also assuming this is not like a 'Botnet SaaS' so we dont need to specify which customer this belongs to.

We won't resolve Geo Location on the client machine but instead on the Web Panel Backend as it's easier to do this there. The client machine may have more accurate data in some circumstances if we do more than just rely on the IP Geo location but it's a lot of work for not much value.

To ensure you don't get fucked by someone we'll implement Rate Limiting by IP and we'll have to have some way of making sure the machine doesn't register it's self as a download multiple times within a short time span if we don't want to.

We can just use a Model for the download in laravel and they have the option to add rate limiting built in already.

We'll also implement an API Key but it is not going to secure your Web Panel as someone could always Proxy the request and see the variables sent anyway. This is why we'll make sure not to leak any information via the API or return any data that we don't want anyone else seeing.

I'd also decouple the API URL from the Web Panel URL, the last thing you want is someone discovering the endpoint by proxying this connection and the domain also serving the backend. If someone see's this endpoint by accident for example if it appears somewhere on a client machine

like a firewall popup we don't want them visiting it and announcing to them they just stumbled upon a control centre.

If you spread your client process enough and it draws some attention you may want to cycle the API Endpoints domain so it can't just be blocked as it'll just break the whole solution you've created regardless of whatever cloaking things you've put in.

For the AV related features I can't really say I've tackled this before. Likely some service exists online that doesn't provide the AV's used in the scan with your file as a sample to look at which causes detection as a result. One of these probably has an API endpoint you can fire using Guzzle which is not specific to Laravel.

If this isn't the case I'd look into it more but I guess a solution if you can't find anything is to have a Windows Machine or Machines somewhere with the most popular AVs installed, try to cover as much market share as you can without trying to support 30 AVs, 3-5 is probably good enough max.

You would need to always sell this as a paid service to people because it will cost you money, and I can't really think of a way of easily allowing people to deploy all of the stuff needed for checking AVs if there isn't a solution online you can leverage. Each customer would have their API key to ensure they paid for this stuff and you'll have to lock it down to an IP or something to ensure it's not shared.

Their web panel would send a request to yet another API with an executable to scan. How the rest of that process goes is up to you. The panel can routinely check if its finished via a GET Request, you could register webhooks etc.

PulseLink - April 8, 2024, 5:06 pm

(Apr 7, 2024 02:03 PM)Onyx Digital Wrote: [\[CITE\]](#) Even though you don't want to hear it I'd use Laravel. You can implement all of this without ofcourse it'd just be a lot faster to ship and easier to maintain and update.

would like to add a small web-panel to this to track downloads and geolocation

Here I'd create a Rest API endpoint that accepts a HWID.

Assuming by dropping into the registry we meaning it has persistency I'd do the send a POST request to this endpoint with the HWID (Add more data if you need to for creating a record of a 'download')

Edit: I'm also assuming this is not like a 'Botnet SaaS' so we dont need to specify which customer this belongs to.

We won't resolve Geo Location on the client machine but instead on the Web Panel Backend as it's easier to do this there. The client machine may have more accurate data in some circumstances if we do more than just rely on the IP Geo location but it's a lot of work for not much value.

To ensure you don't get fucked by someone we'll implement Rate Limiting by IP and we'll have to have some way of making sure the machine doesn't register it's self as a download multiple times within a short time span if we don't want to.

We can just use a Model for the download in laravel and they have the option to add rate limiting built in already.

We'll also implement an API Key but it is not going to secure your Web Panel as someone could

always Proxy the request and see the variables sent anyway. This is why we'll make sure not to leak any information via the API or return any data that we don't want anyone else seeing.

I'd also decouple the API URL from the Web Panel URL, the last thing you want is someone discovering the endpoint by proxying this connection and the domain also serving the backend. If someone sees this endpoint by accident for example if it appears somewhere on a client machine like a firewall popup we don't want them visiting it and announcing to them they just stumbled upon a control centre.

If you spread your client process enough and it draws some attention you may want to cycle the API Endpoints domain so it can't just be blocked as it'll just break the whole solution you've created regardless of whatever cloaking things you've put in.

For the AV related features I can't really say I've tackled this before. Likely some service exists online that doesn't provide the AV's used in the scan with your file as a sample to look at which causes detection as a result. One of these probably has an API endpoint you can fire using Guzzle which is not specific to Laravel.

If this isn't the case I'd look into it more but I guess a solution if you can't find anything is to have a Windows Machine or Machines somewhere with the most popular AVs installed, try to cover as much market share as you can without trying to support 30 AVs, 3-5 is probably good enough max.

You would need to always sell this as a paid service to people because it will cost you money, and I can't really think of a way of easily allowing people to deploy all of the stuff needed for checking AVs if there isn't a solution online you can leverage. Each customer would have their API key to ensure they paid for this stuff and you'll have to lock it down to an IP or something to ensure it's not shared.

Their web panel would send a request to yet another API with an executable to scan. How the rest of that process goes is up to you. The panel can routinely check if its finished via a GET Request, you could register webhooks etc.

I know how i would create it, if you pm me i will show you the loader in development right now.