

Forum: [Hack Forums](#)

Thread: [IRC Botnet \[Python 3.5\]](#)

05317832892 - February 12, 2017, 11:55 am

I've been off school with half term + weekend recently so I write a nice and simple IRC botnet. Commented this one out so it's easier to follow.

Compile with pyinstaller:

Code pyinstaller -F -w -i icon.ico bot.pyFlags explained:

-F Compiles to a single standalone .exe, no .dlls and config shit

-w Runs in a hidden window

-i specifies a path for an icon

bot.py name/location of your python script

Features:

-Copies itself to startup

-UDP Flood

-FUD

-Only accepts commands from the bot owner

-High compatability

-Download and execute

-Ping your bots

As usual with python malware, my favourite type of malware, completely FUD!

Scan:

Spoiler [\(Click to View\)](#)

Code:

Spoiler [\(Click to View\)](#)

Screenshots:

Spoiler [\(Click to View\)](#)

PotatoChipz - February 12, 2017, 12:15 pm

This code is fucking trash, I guess that's why it's free.

SIM-3063187 - February 12, 2017, 12:18 pm

(Feb 12, 2017 01:15 PM)PotatoChipz Wrote: [\[CITE\]](#) This code is fucking trash, I guess that's why it's free.

Please man. Be happy he gave this out for free. A little compliment is well earned even when you don't like the code common.

I don't use botnets but I'm very happy that people from HF are contributing to the community!
Good job!!

05317832892 - February 12, 2017, 12:25 pm

(Feb 12, 2017 01:15 PM)PotatoChipz Wrote: [\[CITE\]](#) This code is fucking trash, I guess that's why it's free.

Open to suggestions.

(Feb 12, 2017 01:18 PM)sixert Wrote: [\[CITE\]](#) Please man. Be happy he gave this out for free. A little compliment is well earned even when you don't like the code common.

I don't use botnets but I'm very happy that people from HF are contributing to the community!
Good job!!
Thanks!

PotatoChipz - February 13, 2017, 5:46 am

(Feb 12, 2017 01:25 PM)?Cam? Wrote: [\[CITE\]](#) Open to suggestions.

Thanks!

Make it OOP

Your variable names are inconsistent Make the variable names more descriptive so rather than saying "irc" call it "ircSocket"

You shouldn't actually need a delay 0.5.

* Rather than calling send directly on the socket create a wrapper around the socket and use that.

However because I'm cool I wrote a better IRC botnet.

You can get it from here: <http://pastebin.com/nsDGFJhi> - Python 2.7.12

To use it you just run it, and then authenticate yourself.

!authenticate super_secret_password

You can then use !ping or !ip.

Feel free to add on to it.

Note: You will need the "requests" library which can be downloaded by running "pip install requests".

Infamia - February 13, 2017, 8:17 am

(Feb 13, 2017 06:46 AM)PotatoChipz Wrote: [\[CITE\]](#) * Make it OOP Your variable names are inconsistent

Make the variable names more descriptive so rather than saying "irc" call it "ircSocket" You shouldn't actually need a delay 0.5.

Rather than calling send directly on the socket create a wrapper around the socket and use that.

However because I'm cool I wrote a better IRC botnet.

You can get it from here: <http://pastebin.com/nsDGFJhi> - Python 2.7.12

To use it you just run it, and then authenticate yourself.

```
!authenticate super_secret_password
```

You can then use !ping or !ip.

Feel free to add on to it.

Note: You will need the "requests" library which can be downloaded by running "pip install requests".

yeah, making it a library would be better tbh, but you can't really make it oop, you can just implement oop into it.

you can usually detect the newbie python coders from the way they format their code and by how they use classes/functions or even lack thereof.

but yes, as he said your code definitely is pretty ugly and needs either a main class to store the functions and have everything done easier code-wise and debug-wise.

just he worded it a bit meaner.

also you write legit the same style as me @PotatoChipz

+1

Goodies - February 13, 2017, 8:39 am

> 3.5

> Using multi-threading

I just shat myself.

Infamia - February 13, 2017, 8:45 am

(Feb 13, 2017 09:39 AM)Goodies Wrote: [\[CITE\]](#) > 3.5

> Using multi-threading

I just shat myself.

true, he should use 2.7.6 and threading.

it's better than 3.x's asynio or whatever that shit was that you constantly asslick

05317832892 - February 13, 2017, 8:47 am

(Feb 13, 2017 09:39 AM)Goodies Wrote: [\[CITE\]](#) > 3.5

> Using multi-threading

I just shat myself.

Lmao it's not that bad, I started on 3.5 and can't really bring myself to downgrade to 2.7

Goodies - February 13, 2017, 9:26 am

(Feb 13, 2017 09:45 AM)Infamia Wrote: [\[CITE\]](#) it's better than 3.x's asynio or whatever that shit was

that you constantly asslick

Ha! There is nothing better than asyncio, you mongoloid.

Infamia - February 13, 2017, 9:29 am

(Feb 13, 2017 10:26 AM)Goodies Wrote: [\[CITE\]](#) Ha! There is nothing better than asyncio, you mongoloid.

old is gold

Goodies - February 13, 2017, 9:34 am

(Feb 13, 2017 09:47 AM)?Cam? Wrote: [\[CITE\]](#) Lmao it's not that bad, I started on 3.5 and can't really bring myself to downgrade to 2.7

No, don't downgrade... never downgrade. [\(Click to View\)](#)

Learn to use and love asyncio.

05317832892 - February 13, 2017, 10:30 am

(Feb 13, 2017 10:34 AM)Goodies Wrote: [\[CITE\]](#) No, don't downgrade... never downgrade. [\(Click to View\)](#)

Learn to use and love asyncio.

Of course.

PotatoChipz - February 13, 2017, 1:04 pm

(Feb 13, 2017 10:34 AM)Goodies Wrote: [\[CITE\]](#) No, don't downgrade... never downgrade. [\(Click to View\)](#)

Learn to use and love asyncio.

I can write in Python 3.x, it's just that I prefer Python 2.x.

Old is gold, new is ew.

deadsh0t - February 17, 2017, 2:46 am

hi im new to programming python

and think a script like this is very interesting however i had a few questions, this line:

```
botnick="BOT_" + str(random.randint(10000000, 99999999))
```

since this script is ran again at reboot the bot will get a new nickname each reboot right ?

And second it generates a random name for every bot but wouldnt that mean that if a bot somehow got the same name as another bot it would fail to connect?

you could also use this code `botnick="BOT_" + urllib.request.urlopen('http://ip.42.pl/raw').read()`

This should give the bot BOT_(their ip) as a nickname

However im not home to test it

05317832892 - February 17, 2017, 8:17 am

(Feb 17, 2017 03:46 AM)deadsh0t Wrote: [\[CITE\]](#) hi im new to programming python

and think a script like this is very interesting however i had a few questions, this line:

```
botnick="BOT_" + str(random.randint(10000000, 99999999))
```

since this script is ran again at reboot the bot will get a new nickname each reboot right ?

And second it generates a random name for every bot but wouldn't that mean that if a bot somehow got the same name as another bot it would fail to connect?

you could also use this code `botnick="BOT_" + urllib.request.urlopen('http://ip.42.pl/raw').read()`

This should give the bot BOT_(their ip) as a nickname

However im not home to test it

Didn't think about doing it that way, it will also act as a muted because it will fail to connect twice under the same ip.

Very nice though, it like it.

deadsh0t - February 18, 2017, 5:33 am

and also to update your bots for whatever reason like moving to another irc server or changing to another username

you could recompile your script with `/AppData/Roaming/Microsoft/Windows/Start Menu/Programs/Startup/pythonfile" + ".exe"` to `/AppData/Roaming/Microsoft/Windows/Start Menu/Programs/Startup/pythonfile1" + ".exe"`

and add this line at the top

`os.remove(/AppData/Roaming/Microsoft/Windows/Start Menu/Programs/Startup/pythonfile" + ".exe")`

and then compile the exe and use download and execute from the current exe so that the old exe will be removed and the new 1 will be ran

deadsh0t - February 20, 2017, 5:52 am

how can i get pyinstaller to use 3.5 instead of 3.6 since 3.6 fails because its not supported by pyinstaller

05317832892 - February 20, 2017, 10:24 am

(Feb 20, 2017 06:52 AM)deadsh0t Wrote: [\[CITE\]](#) how can i get pyinstaller to use 3.5 instead of 3.6 since 3.6 fails because its not supported by pyinstaller

Reinstall Python, install 3.5

Your not loosing anything significant.

obby - February 20, 2017, 11:05 am

(Feb 13, 2017 10:26 AM)Goodies Wrote: [\[CITE\]](#) Ha! There is nothing better than asyncio, you mongoloid.

Correction, I'm sorry that you're unfamiliar with the GIL but the GIL (Global Internal Limiter) limits python to an extreme so they made the multithreading and multiprocessing modules for users to attempt a bypass on this. ASYNCIO will only use one core and is very slow. Making the script simply async will not achieve optimal performance, using a process per core and threads per process with async will. Using threads without async is also more user friendly, and more optimal in terms of performance.

(Feb 13, 2017 10:29 AM)Infamia Wrote: [\[CITE\]](#) old is gold

Fortunately the syntax is close enough for it to not really matter but I also use python2.7 where possible as python3 support still isn't as big as I'd like it to be. Proper code is usually 1:1 translation anyways. I do also agree with you that he needs to stop asslicking asyncio.

deadsh0t - February 21, 2017, 9:58 am

so i got pyinstaller to work however it connects to the irc server succesfully but it wont join the channel

EDIT 1: I fixed it for people having this problem paste the join line right after while true: this is because with some irc servers the handling can be too fast so you need to wait until its fully

connected

Goodies - February 21, 2017, 10:59 am

(Feb 20, 2017 12:05 PM)obiwankanobi Wrote: [\[CITE\]](#) Correction, I'm sorry that you're unfamiliar with the GIL but the GIL (Global Internal Limiter) limits python to an extreme so they made the multithreading and multiprocessing modules for users to attempt a bypass on this.

I'm overtly familiar with the GIL (L = Lock, not Limiter, by the way). I program in C++ with the Python API, bud, and I'm plenty familiar with how it works in the backend. The GIL is just a binary semaphore and a condition variable. And judging by the fact that you think the multithreading module is an attempt to bypass the GIL, I'd say I have a much better understanding of it than you do lol. Asyncio DOES bypass the GIL for I/O bound code because the GIL is released upon blocking I/O operations (such as network activity). Multithreading more or less does the same thing, but far less efficiently. In CPython, only one thread can access the Python API at any given time. Multithreading works by switching between threads every `sys.getcheckinterval()` ticks, OR by releasing the GIL when doing anything I/O related. If you are running CPU-bound pure Python code with multiple threads in CPython, you are guaranteed to be slower than if you only use one thread (due to the GIL).

Example, this code will always run slower in the multithreaded version in CPython so long as there is a GIL:

Spoiler ([Click to View](#))

Quote:ASYNCIO will only use one core and is very slow.

I don't think you get how this whole async thing works... Over 90% of the time, the thread is just waiting on an I/O operation. That's not an exaggeration. It really is over 90% in a huge number of networking applications. While it's waiting for the socket to respond (which often takes several hundred milliseconds, sometimes more) it can do plenty of other things (such as spawning a new socket connection, or even dozens of concurrent ones) all without affecting the speed of the others. This DOES provide maximum efficiency. Adding multiple threads will do nothing. Especially since asyncio can only have one event loop per thread/process. If you added more processes, you could handle more connections, but you also have to add IPC and process synchronization and safety. Typically, though, the bottleneck with asyncio is the network. More often than not, asyncio can handle way more active connections than your NIC can reliably support. You'll start to get socket timeouts before you'll run out of available connections.

Quote:Making the script simply async will not achieve optimal performance, using a process per core and threads per process with async will.

Again, you completely misunderstand what async means. "Asynchronous" is just the overarching umbrella term. It does not mean "single-threaded asynchronous." It just means you're doing more than one thing at a time. True multithreading, multiprocessing, parallel processing, etc... these are all forms of asynchronous execution.

Also, what you've recommended actually really doesn't provide any benefit in this case. Generally, what I do is have a `concurrent.futures.ProcessPoolExecutor` to which I offload any necessary CPU-bound processing, and the main thread simply runs the asyncio event loop, registers socket file descriptors, etc... Asyncio is so efficient BECAUSE it only uses one thread (except for things like DNS lookups in which case it does use another thread, but the overhead is very small overall). You don't need to incorporate thread synchronization or any extra safety. If you did, that would only make it slower.

Quote:Using threads without async is also more user friendly, and more optimal in terms of

performance.

No. No it so is completely not. You're wrong about literally everything you said. If you use threads, and you don't want to be an idiotic skid about it, you have to implement thread safety. Obviously there are APIs like ``multiprocessing.pool.ThreadPool`` and ``concurrent.futures.ThreadPoolExecutor`` that can help, but you still need thread safety if you want to access the same mutable structure or something like that.

Quote:I do also agree with you that he needs to stop asslicking asyncio.

If I'm being honest, I've never eaten ass before. Doesn't appeal to me. If a girl wants to do it to me, that's fucking awesome, but I've never done it. That being said, if asyncio was a girl, I'd munch on the brown star as long as she wants.

05317832892 - February 21, 2017, 1:06 pm

(Feb 21, 2017 11:59 AM)Goodies Wrote: [\[CITE\]](#) If I'm being honest, I've never eaten ass before. Doesn't appeal to me. If a girl wants to do it to me, that's fucking awesome, but I've never done it. That being said, if asyncio was a girl, I'd munch on the brown star as long as she wants.
Lmao, brilliant.

obby - February 21, 2017, 6:35 pm

(Feb 21, 2017 11:59 AM)Goodies Wrote: [\[CITE\]](#) I'm overtly familiar with the GIL (L = Lock, not Limiter, by the way). I program in C++ with the Python API, bud, and I'm plenty familiar with how it works in the backend. The GIL is just a binary semaphore and a condition variable. And judging by the fact that you think the multithreading module is an attempt to bypass the GIL, I'd say I have a much better understanding of it than you do lol. Asyncio DOES bypass the GIL for I/O bound code because the GIL is released upon blocking I/O operations (such as network activity). Multithreading more or less does the same thing, but far less efficiently. In CPython, only one thread can access the Python API at any given time. Multithreading works by switching between threads every ``sys.getcheckinterval()`` ticks, OR by releasing the GIL when doing anything I/O related. If you are running CPU-bound pure Python code with multiple threads in CPython, you are guaranteed to be slower than if you only use one thread (due to the GIL).

Example, this code will always run slower in the multithreaded version in CPython so long as there is a GIL:

Spoiler [\(Click to View\)](#)

I don't think you get how this whole async thing works... Over 90% of the time, the thread is just waiting on an I/O operation. That's not an exaggeration. It really is over 90% in a huge number of networking applications. While it's waiting for the socket to respond (which often takes several hundred milliseconds, sometimes more) it can do plenty of other things (such as spawning a new socket connection, or even dozens of concurrent ones) all without affecting the speed of the others. This DOES provide maximum efficiency. Adding multiple threads will do nothing. Especially since asyncio can only have one event loop per thread/process. If you added more processes, you could handle more connections, but you also have to add IPC and process synchronization and safety. Typically, though, the bottleneck with asyncio is the network. More often than not, asyncio can handle way more active connections than your NIC can reliably support. You'll start to get socket timeouts before you'll run out of available connections.

Again, you completely misunderstand what async means. "Asynchronous" is just the overarching umbrella term. It does not mean "single-threaded asynchronous." It just means you're doing more

than one thing at a time. True multithreading, multiprocessing, parallel processing, etc... these are all forms of asynchronous execution.

Also, what you've recommended actually really doesn't provide any benefit in this case. Generally, what I do is have a ``concurrent.futures.ProcessPoolExecutor`` to which I offload any necessary CPU-bound processing, and the main thread simply runs the asyncio event loop, registers socket file descriptors, etc... Asyncio is so efficient BECAUSE it only uses one thread (except for things like DNS lookups in which case it does use another thread, but the overhead is very small overall). You don't need to incorporate thread synchronization or any extra safety. If you did, that would only make it slower.

No. No it so is completely not. You're wrong about literally everything you said. If you use threads, and you don't want to be an idiotic skid about it, you have to implement thread safety. Obviously there are APIs like ``multiprocessing.pool.ThreadPool`` and ``concurrent.futures.ThreadPoolExecutor`` that can help, but you still need thread safety if you want to access the same mutable structure or something like that.

If I'm being honest, I've never eaten ass before. Doesn't appeal to me. If a girl wants to do it to me, that's fucking awesome, but I've never done it. That being said, if asyncio was a girl, I'd munch on the brown star as long as she wants.

No one said anything about thread safety, so I'm not going to tackle these as it can be done in many ways and was never a topic to begin with. Continuing from there asyncio will allow you to get more work done, and purely using multiprocessing (side-stepping GIL) opposed to pure asyncio coroutines will result in higher performance when the task is primarily CPU-bound, however in primarily I/O-bound tasks yes asyncio will out perform both and I will concede on my argument of threading and multiprocessing being a more efficient approach and for something like an IRC bot all would provide very similar results but asyncio would out perform. I apologize for coming across as rude and I get mixed up lock / limiter often.

Black?jack - February 26, 2017, 10:42 am

This code is not great. Read the RFCs for the IRC protocol, learn it, and then learn how to properly do socket programming. You're doing both of those things wrong.

05317832892 - February 26, 2017, 1:51 pm

(Feb 26, 2017 11:42 AM)Black?jack Wrote: [\[CITE\]](#) This code is not great. Read the RFCs for the IRC protocol, learn it, and then learn how to properly do socket programming. You're doing both of those things wrong.

I'm well aware, the protocol is not amazingly documented. I have read up.

Might redo this when I have time, maybe using an IRC module instead of trying to do the socket communication manually.

micahmonday - March 1, 2017, 6:54 am

Hi, I'm just wondering... couldn't a whole `CountDown()` thread be replaced by something like:

```
Code global timePeriod #or "timePeriod = q.get()" using Queue module
while True:
    if timePeriod:
        lastClock = time.clock()
        while (time.clock() - lastClock) < timePeriod:
            #do stuff
```

timePeriod = 0
placed within Attack() function?

05317832892 - March 1, 2017, 11:30 am

(Mar 1, 2017 07:54 AM)michalmonday Wrote: [\[CITE\]](#) Hi, I'm just wondering... couldn't a whole Countdown() thread be replaced by something like:

```
Code global timePeriod #or "timePeriod = q.get()" using Queue module
while True:
    if timePeriod:
        lastClock = time.clock()
        while (time.clock() - lastClock) < timePeriod:
            #do stuff
        timePeriod = 0
    placed within Attack() function?
```

That's quite a good idea, cuts down on threads and might help with variable safety.

The2G - March 6, 2017, 7:25 am

Hi Cam,

I'm developing a variant of your given code, encrypting the messages, generating real-looking nicknames etc.. But When I try to connect to the IRC Server I get the following error:
ConnectionRefusedError: [WinError 10061]

I tried with different IRC servers and all give me that error. Why isn't connecting to the server?

Thanks!

05317832892 - March 6, 2017, 11:01 am

(Mar 6, 2017 08:25 AM)Rubba Wrote: [\[CITE\]](#) Hi Cam,

I'm developing a variant of your given code, encrypting the messages, generating real-looking nicknames etc.. But When I try to connect to the IRC Server I get the following error:
ConnectionRefusedError: [WinError 10061]

I tried with different IRC servers and all give me that error. Why isn't connecting to the server?

Thanks!

Try the Kiwi IRC servers, IRC.FREENODE.NET I think it was.

Make sure the IP and port are correct.

The2G - March 7, 2017, 6:47 am

(Mar 6, 2017 12:01 PM)?Cam? Wrote: [\[CITE\]](#) Try the Kiwi IRC servers, IRC.FREENODE.NET I think it was.

Make sure the IP and port are correct.

Thanks Cam. It was my net blocking that port. I connected the server over VPN and all works!!

captn_nemo - March 8, 2017, 4:39 pm

I have some problems with pyinstaller.
I runnig pyinstaller under linux is this a problem?

05317832892 - March 9, 2017, 2:12 pm

(Mar 8, 2017 05:39 PM)captn_nemo Wrote: [\[CITE\]](#) I have some problems with pyinstaller.
I runnig pyinstaller under linux is this a problem?
Probably, considering py installer is for making .exes

captn_nemo - March 9, 2017, 3:28 pm

(Mar 9, 2017 03:12 PM)?Cam? Wrote: [\[CITE\]](#) Probably, considering py installer is for making .exes
That's what I want to do but under Linux.

Folks - March 10, 2017, 10:05 am

(Mar 9, 2017 04:28 PM)captn_nemo Wrote: [\[CITE\]](#) That's what I want to do but under Linux.
PyInstaller doesn't support cross-compiling. It only compiles for the operating system it was used on.

hunter24957 - March 10, 2017, 11:21 am

(Mar 9, 2017 04:28 PM)captn_nemo Wrote: [\[CITE\]](#) That's what I want to do but under Linux.
cx_Freeze works wonders my friend ?

Folks - March 10, 2017, 12:29 pm

(Mar 10, 2017 12:21 PM)hunter24957 Wrote: [\[CITE\]](#) cx_Freeze works wonders my friend ?
This has the same problem as PyInstaller in that while it is cross platform (so is PyInstaller), it can only create an executable for whatever OS you are currently using it on.

hunter24957 - March 10, 2017, 3:03 pm

(Mar 10, 2017 01:29 PM)Folks Wrote: [\[CITE\]](#) This has the same problem as PyInstaller in that while it is cross platform (so is PyInstaller), it can only create an executable for whatever OS you are currently using it on.

Oh right, didn't realize he wanted to create an executable on Linux. In that case; it's not easily possible, well it's not simply done anyways. Because Linux is Linux there's not really anything PyInstaller can use to build the exe, I'm sure a bit of google won't hurt, I found this, check it out: <https://sourcecontribute.com/2015/05/02/compiling-python-to-standalone-executables-on-linux-using-nuitka/> Take care! ?

michalmonday - March 11, 2017, 7:40 am

not FUD anymore...

For example avast classifies a program consisting of 'print "test"' as a malware if it's compiled by pyinstaller using -w parameter. It feels like Santa Claus came back a week after Christmas to take presents back...

Folks - March 11, 2017, 10:18 am

(Mar 11, 2017 08:40 AM)michalmonday Wrote: [\[CITE\]](#) not FUD anymore...

For example avast classifies a program consisting of 'print "test"' as a malware if it's compiled by pyinstaller using -w parameter. It feels like Santa Claus came back a week after Christmas to take presents back...

Well it has been a month since release, and it is open source. Not sure what you were expecting.

michalmonday - March 11, 2017, 1:22 pm

I thought pyinstaller was popular before the release too for similar purposes...

05317832892 - March 12, 2017, 11:40 am

(Mar 11, 2017 08:40 AM)michalmonday Wrote: [\[CITE\]](#) not FUD anymore...

For example avast classifies a program consisting of 'print "test"' as a malware if it's compiled by pyinstaller using -w parameter. It feels like Santa Claus came back a week after Christmas to take presents back...

<https://www.virustotal.com/en/file/e1768ee987e60f27ab6d4675c430e0e7e8ef1dddae552529010684d95f35b90e/analysis/1489343521/>

^ That's just a print statement, it looks like he really has taken all the presents ;(

target_ - March 13, 2017, 5:33 am

Why is this happening? :

Exception in thread Thread-2:

Traceback (most recent call last):

File "/usr/lib/python3.5/threading.py", line 914, in bootstrap_inner
self.run()

File "/usr/lib/python3.5/threading.py", line 862, in run
self._target(*self._args, self._kwargs)

File "bot.py", line 44, in Attack
UdpSocket.sendto(data,(ip,port))

OSError: [Errno 90] Message too long

(Mar 13, 2017 05:33 AM)target Wrote: [\[CITE\]](#) Why is this happening? :

Exception in thread Thread-2:

Traceback (most recent call last):

File "/usr/lib/python3.5/threading.py", line 914, in _bootstrap_inner
self.run()

File "/usr/lib/python3.5/threading.py", line 862, in run
self._target(*self._args, self._kwargs)

File "bot.py", line 44, in Attack
UdpSocket.sendto(data,(ip,port))

OSError: [Errno 90] Message too long

Nvm.. Fixed it!

Folks - March 13, 2017, 1:15 pm

(Mar 11, 2017 02:22 PM)michalmonday Wrote: [\[CITE\]](#) I thought pyinstaller was popular before the release too for similar purposes...

My bad friend, I misread your original post and meant that the actual program was detected when -w flag was used, not all the time.

05317832892 - March 14, 2017, 12:21 am

(Mar 13, 2017 01:15 PM)Folks Wrote: [\[CITE\]](#) My bad friend, I misread your original post and meant that the actual program was detected when -w flag was used, not all the time.

After some testing I'm pretty sure -w has no effect, everything gets detected.

michalmonday - March 14, 2017, 1:16 am

Are you sure? I tested 2 the same "print" statement files and got different results:

with window (3-5 / 60 on [virustotal](https://www.virustotal.com))

without (13-15 / 60 on [virustotal](https://www.virustotal.com))

05317832892 - March 14, 2017, 10:44 am

(Mar 14, 2017 01:16 AM)michalmonday Wrote: [\[CITE\]](#) Are you sure? I tested 2 the same "print"

statement files and got different results:

with window (3-5 / 60 on [virustotal](#))

without (13-15 / 60 on [virustotal](#))

Are you using Python 2 or 3?

I checked on 3.

The only difference was one of the detections the malicious confidence went from 89% to 94% I think.

zXuS - March 15, 2017, 12:28 pm

This is pretty cool thanks!, I am going to test it later, great idea!

But why are people using virustotal !! they distribute the hashes!

Then people are moaning about detection, just use nodistribute !

micthalmunday - March 15, 2017, 2:31 pm

idk about others but I tested it on virustotal after it has been detected by an antivirus. Btw if they distributed only hashes would it be a problem? Couldn't you change 1 line of code and the hash would be completely different?

hunter24957 - March 15, 2017, 2:36 pm

(Mar 15, 2017 02:31 PM)micthalmunday Wrote: [CITE](#) idk about others but I tested it on virustotal after it has been detected by an antivirus. Btw if they distributed only hashes would it be a problem? Couldn't you change 1 line of code and the hash would be completely different?

I believe it's detected buddy, if you know Python or any other related language it's not hard to make your own.

EnigmaF0x - March 17, 2017, 10:39 am

Hi check out my thread

Code <https://hackforums.net/showthread.php?tid=5577567>

I need a RAT (multiplatform)

Can you do this? If interested let me know in PM.