Project TajMahal IOCs and Registry Data Decrypter

github.com/TheEnergyStory/malware_analysis/tree/master/TajMahal

TheEnergyStory

TheEnergyStory/
malware_analysis

Malware analyses and helpful scripts

1 Contributor | 0 Issues | 28 Stars | 6 Forks

Last year in April, Kaspersky released an article about a new complex malware framework dubbed TajMahal: https://securelist.com/project-tajmahal/90240/

Later the same year, one of the stated samples from Kaspersky report was uploaded to Virustotal: https://www.virustotal.com/gui/file/0b74fc2594b25987841a7897aff323f4165519e6c26d679256cb0d282a6f0147/

The sample is a `.NET assembly` with obfuscated strings but self explaining code after decompilation (dnSpy or ILSpy). As the names of the namespace and classes imply, the sample's main purpose is to manage the Windows service persistency.

This repository contains the `decompiled code with decoded strings` (I left the deobfuscation method in the code), `extracted IOCs` and a `registry config data decrypter`.

Decompiled and Deobfuscated code

After decompilation and string decoding we have a perfect readable C# code. See: Chaperone

IOCs

Type	Details
Scheduled task with name	MaintenancePolicy
Registry values	Key: HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AdaptiveDisplayBrightness
	Value: seg0 (unknown content)
	Value: seg4 (contains encrypted service config data)

Code oddities

The code contains two errors in the `Tools` class, not sure if they are caused by the decompilation engine or are actual coding errors.

1) While the Enum reg key is created for the Windows service (`text2`), it uses the before created registry path of the service itself to set the values (`text`)

```
text2 = @"SYSTEM\CurrentControlSet\Services\";
text2 += this.regdata.srvName;
text2 += @"\Enum";
using (RegistryKey registryKey3 = Registry.LocalMachine.CreateSubKey(text))
{
        registryKey3.SetValue("0", @"Root\LEGACY_SRVC\0000",
RegistryValueKind.String);
        registryKey3.SetValue("Count", 1, RegistryValueKind.DWord);
        registryKey3.SetValue("NextInstance", 1, RegistryValueKind.DWord);
        registryKey3.Close();
}
```

2) There is a method named `DeleteTask` which uses `Security` as a file name to delete a scheduled task. Maybe there is a custom executable named `Security.exe` in the same directory with such functionality, but it's more likely that the author wanted to use `schtasks.exe` according to the used arguments.

```
public int DeleteTask()
{
        this.RunProgram("Security", @"/Delete /TN
\Microsoft\Windows\Maintenance\MaintenancePolicy /F", true);
        return 0;
}

public int RunProgram(string exe, string cmd, bool waitfor)
{
        Process process = new Process();
        process.StartInfo.FileName = exe;
```

Registry config data decrypter

The sample tries to get RC4 encrypted data from a registry key (see IOCs) and use that data for the service set up. I have coded a config data decryptor for those who are infected. See: tajmahal_regdata_decrypter.py

Config data description:

Data	Explanation
DisplayName	Persistence service display name value under "SYSTEM\CurrentControlSet\Services<Name>"
Description	Persistence service description value under "SYSTEM\CurrentControlSet\Services<Name>"
Name	Persistence service name key under "SYSTEM\CurrentControlSet\Services"
entryPoint	Name in "ServiceMain" under "SYSTEM\CurrentControlSet\Services<Name>\Parameters" which describes the service DLL's entry point (export function name)
srvPath	Unknown, likely service DLL path
markerPath	Unkown, likely infection validation file
task_name	Unkown, likely scheduled task name
ttl	Date of self destruction