MalTrak

COM Objects P.1: The Hidden Backdoor in Your System

Amr Thabet Follow

May 26 · 5 min read

In the last few years, attackers have abused COM Objects to craft their Fileless attacks, evade defenses, bypass whitelisting, and even move laterally inside the organization using Microsoft RPC protocol.

In this series of blog posts, we will dive deeper into COM objects, how to utilize them in your red teaming engagement and how to detect and protect your organization from them if you are on the blue team side.

But before we start, let us first get to understand, what is a COM Object and why it's there in the first place.

What is a COM Object?

According to Microsoft, "The Microsoft Component Object Model (COM) is a platform-independent, distributed, object-oriented system for creating binary software components that can interact. COM is the foundation technology for Microsoft's OLE (compound documents), ActiveX (Internet-enabled components), as well as others."

In simple words (I know you need that), COM is a system that allows software components to communicate with one another and interact together. You can communicate with the Firewall using INetFwMgr COM Object, you can communicate with Outlook, Word, Excel, Internet Explorer and other applications.

This communication can help you open a URL in internet explorer and receive the output (in the hidden window of course), or open outlook and receive an email … etc.

As well, these objects can be executed remotely (Distributed COM or DCOM) on another machine using RPC (Remote Procedure Call) protocol. So, you can use someone else's machine.

COM is an old model but it's the foundation of so many other models/systems like ActiveX, OLE, and others. Also, it's considered obsolete with the appearance of .NET and WCF. But don't worry, with Windows backward compatibility, COM is not going away any time soon.

COM Attacks: COM Objects for Initial Access:

For an attacker to get his foot inside any organization's internal network, he needs to have a payload delivery method (ex: phishing email with a malicious document with a macro) and a payload (ex: VBS code).

COM Objects' power relies on the payload phase. Using what's called COM Scriptlet, the attacker can download and execute a malicious javascript or VBScript using windows legitimate applications.

You can then deliver this payload (which is mostly a command line) in a LNK file inside a zip file, in a malicious macro (to bypass defenses), or a document with an OLE Embedded.

COM Scriptlets: Your Payload Free Delivery Service

COM Objects can use a DLL to execute its functionality or a script. COM Objects that use scripts (instead of DLLs) are named Scriptlets and saved in *.sct files in XML format.

The structure of a scriptlet file is like this:

Basic SCT File (basic.sct)

So, in the registration section, the ClassID is a unique identifier for this COM Object and it's used to find the object, load it and execute its code. ProgID is a human-friendly identifier for this COM Object that can be used.

And last, in this registration section, you got your script. This script will be executed in the registration phase allowing your malware to execute by just trying to register such COM object.

The **CDATA** Section is known in XML Schema. This section is not parsed by the XML parser so you can avoid using **&lt;, &gt;,** and **&amp;** to escape/replace different parsable characters in XML like < or >. It comes in handy while writing javascript code inside an XML.

Luckily enough, there are so many windows legitimate applications that can register and execute such scriptlets. Even download them from the internet and automatically execute your JavaScript or VBScript code inside.

Before we show you all your options, we will dive into the javascript/VBScript code and how you can abuse this section to execute your 2nd stage backdoor

Scripting in COM Objects:

Access to The Shell:

The simplest script you can use is this:

Powershell Execution in Javascript (in SCT File)

With this simple script, we can execute any commands in command prompt (Shell) using an ActiveX Object (a bit modern version of COM objects) called WScript.Shell. This object allows us to execute any command we want.

With this, you can execute powershell.exe, you can use curl, cmd, rundll.exe, or whatever you want to execute. Or even download Netcat using curl and execute it to provide you with a reverse shell (it will be detectable very easily) or a cobalt strike beacon if you have Cobalt Strike.

In VBScript, it’s even easier. You can just run the command “Shell” with the application you want to run:

Powershell Execution in VBScript (in SCT File)

This is the power of the COM scriptlets.

Download & Execute:

Another common script you can use is the download & execute type of scripts. In this script, we will use another ActiveX object for network communication to download the

2nd stage backdoor and execute it using the “eval” command

Download & Execute in Javascript

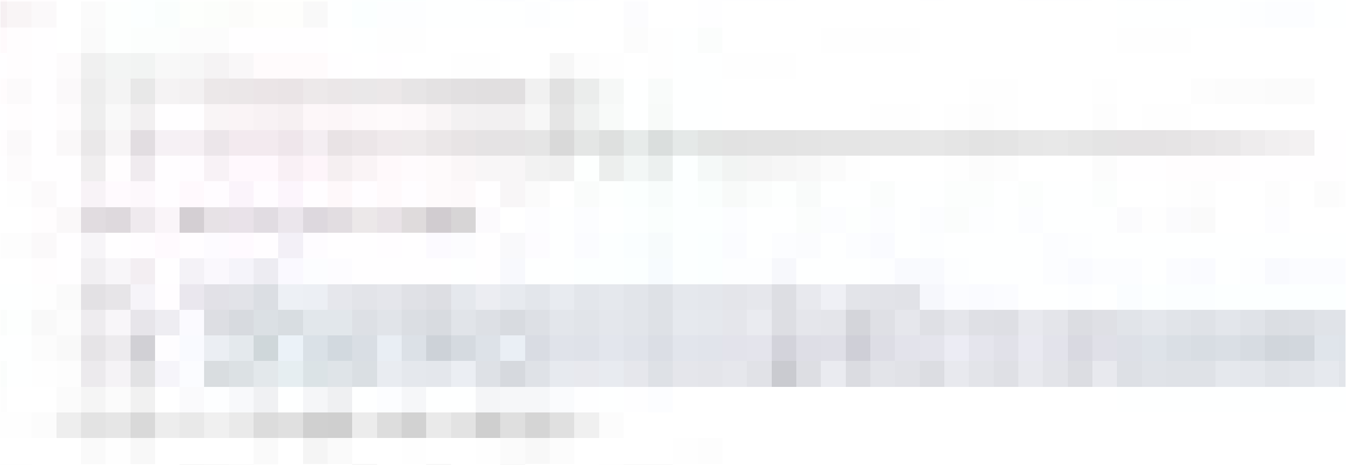
the “eval” command executes the downloaded javascript code dynamically.

.Net Reflective DLL Loader:

I won’t go to the code details of this one but in .Net, there’s a feature called Serialization. This feature allows the application to send and receive objects in binary, JSON, or XML format.

These objects are like DLLs written in C# (or any .Net language) to be dynamically loaded and their code to be executed.

These objects can be encoded in Base64 format, decoded in memory, deserialized/loaded and their main code gets executed. All of that is done using a Microsoft ActiveX Object called **“System.Runtime.Serialization.Formatters.Binary.BinaryFormatters”**



Reference: https://gist.github.com/bohops/545f92b49c4341ee2f7bf1c366fba327

Microsoft has pushed the idea of objects. Everything is an object that is sent through the network, deserialized, and executed remotely which led to reflective DLL Loaders, remote code execution, and much more threats. Now it’s considered obsolete because of the threats it imposed.

To build your own JavaScript .Net/C# loader, you can use DotNetToJScript from here: https://github.com/tyranid/DotNetToJScript and it will help you as well generate the equivalent VBScript if you need.

There’s so much more you can do with JavaScript and VBScript. So many opportunities to obfuscate and bypass AVs, EDRs, and other defenses. And for EDRs, they totally lack when it comes to scripting.

As we have covered how to write an effective Scriptlet, now it’s time to execute it.

What’s Next?

In the next part of this series, we will see the different ways to execute our COM Object payloads and the different ways we can use them to bypass detection, whitelisting, and so on. So, don’t miss our next blog, and feel free to subscribe with your email so you don’t miss the next posts.

Conclusion

COM Objects are the swiss army knife for any attacker or red teamer today. They are built with no security in mind and led to lots of opportunities to abuse them to bypass defenses, escalate privileges and move laterally inside the organization.

If you are a red teamer, learning how to abuse COM objects will get you far ahead in bypassing defenses and detections. And if you are a blue teamer, protecting your organization against such type of attacks should be one of your top priorities and would definitely elevate your organization's security posture.

In-Depth Comprehensive Red Team Training:

If you want to dive in-depth into this topic and other advanced attacks techniques, check out our training:

"Advanced Red Teaming: Weaponization & Adversary Simulation"

This is a live virtual classroom-style training that focuses on developing cyber weapons that can evade AV detection, EDR logs, and forensics traces like how advanced targeted attacks do, and provide you with insights on how to improve your organization's overall detections and security posture. Check it out from here: https://maltrak.com/redteam

References

1. https://github.com/3gstudent/SCTPersistence
2. https://github.com/tyranid/DotNetToJScript
3. https://gist.github.com/bohops/545f92b49c4341ee2f7bf1c366fba327

Pentesting Cybersecurity Ethical Hacking Mitre Attack Incident Response