[P]

[P] | Prelude

Overcoming Technology Gaps of Traditional Purple Teaming

SANS Cyber Fest 2024

Matt Hand

Director of Security Research

Former red team operator with over a decade of experience focusing on testing highly secured environments.

Author of “Evading EDR: The Definitive Guide to Defeating Endpoint Detection Systems”

SPECTEROPS tenable [P] | Prelude

Agenda

01. The challenge we face
02. Finding solutions in modular testing
03. A practical application with Prelude Detect
04. What’s next
05. Questions

1

The challenge we face

Scope & manual processes are holding us back

Purple teams are being dealt a bad hand

Purple teaming exercises remain a foundational way for security teams to evaluate and augment defenses, but they face countless challenges to do so.

Time consuming, manual processes

Traditional purple teaming requires significant coordination, often spanning multiple teams, to plan, execute, and action results from exercises.

Limited scope and scalability

Exercises typically only focus on evaluating a handful of techniques across a small subset of hosts, lacking the scope or scale to comprehensively test defenses, leaving many questions unanswered.

Point-in-time

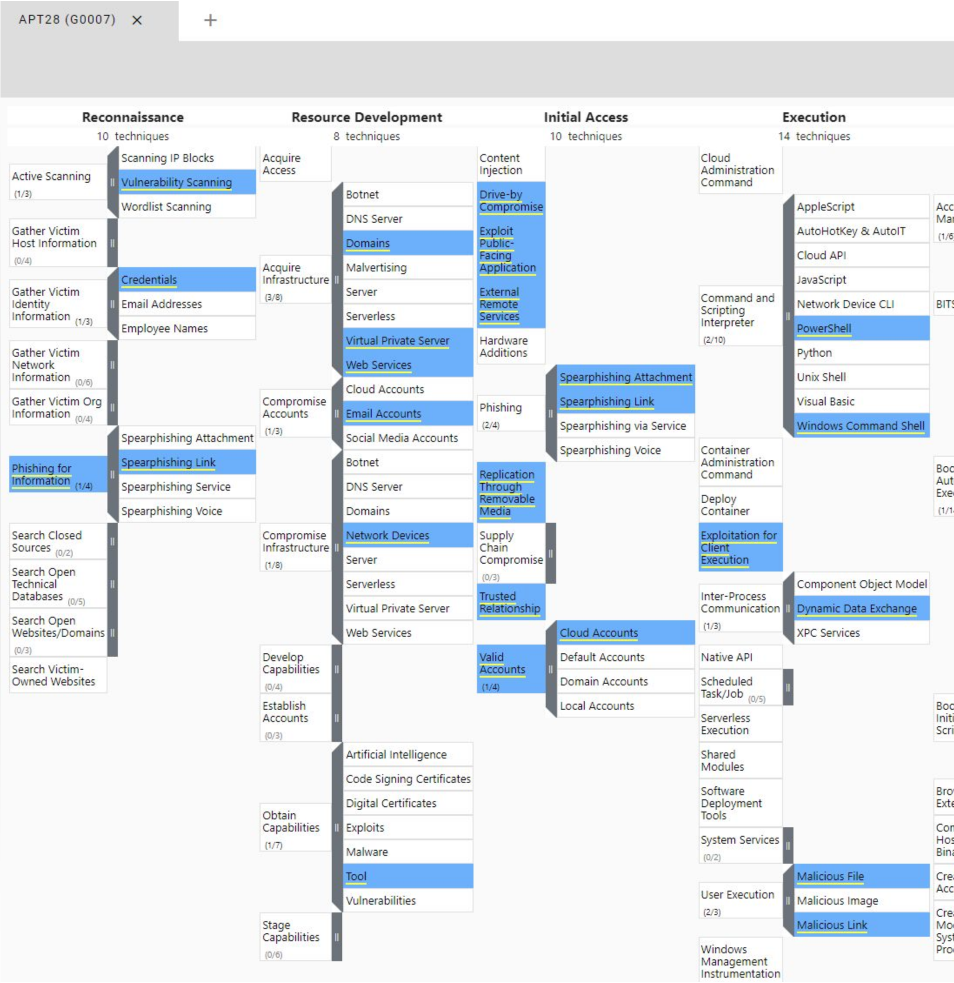
Purple team exercises are generally point-in-time assessments rather than continuous evaluations, leading to a false sense of security as controls, configurations, and behaviors naturally drift.

[P]

And the deck you’re dealing with is too large

The MITRE ATT&CK framework is an invaluable resource to categorize and understand threats. The comprehensive nature, however, presents an overwhelming volume of potential attack vectors.

Manually or even automatically testing and detecting for such a vast array of tactics is daunting. **A one-size-fits-all approach is inefficient.**



APT28 (G0007) × +

Reconnaissance 10 techniques
- Active Scanning (1/3)
  - Scanning IP Blocks
  - Vulnerability Scanning
  - Wordlist Scanning
- Gather Victim Host Information (0/4)
- Gather Victim Identity Information (1/3)
  - Credentials
  - Email Addresses
  - Employee Names
- Gather Victim Network Information (0/6)
- Gather Victim Org Information (0/4)
- Phishing for Information (1/4)
  - Spearphishing Attachment
  - Spearphishing Link
  - Spearphishing Service
  - Spearphishing Voice
- Search Closed Sources (0/2)
- Search Open Technical Databases (0/5)
- Search Open Websites/Domains (0/3)
- Search Victim-Owned Websites

Resource Development 8 techniques
- Acquire Access
- Acquire Infrastructure (3/8)
  - Botnet
  - DNS Server
  - Domains
  - Malvertising
  - Server
  - Serverless
  - Virtual Private Server
  - Web Services
- Compromise Accounts (1/3)
  - Cloud Accounts
  - Email Accounts
  - Social Media Accounts
- Compromise Infrastructure (1/8)
  - Botnet
  - DNS Server
  - Domains
  - Network Devices
  - Server
  - Serverless
  - Virtual Private Server
  - Web Services
- Develop Capabilities (0/4)
- Establish Accounts (0/3)
- Obtain Capabilities (1/7)
  - Artificial Intelligence
  - Code Signing Certificates
  - Digital Certificates
  - Exploits
  - Malware
  - Tool
  - Vulnerabilities
- Stage Capabilities (0/6)

Initial Access 10 techniques
- Content Injection
- Drive-by Compromise
- Exploit Public-Facing Application
- External Remote Services
- Hardware Additions
- Phishing (2/4)
  - Spearphishing Attachment
  - Spearphishing Link
  - Spearphishing via Service
  - Spearphishing Voice
- Replication Through Removable Media
- Supply Chain Compromise (0/3)
- Trusted Relationship
- Valid Accounts (1/4)
  - Cloud Accounts
  - Default Accounts
  - Domain Accounts
  - Local Accounts

Execution 14 techniques
- Cloud Administration Command
- Command and Scripting Interpreter (2/10)
  - AppleScript
  - AutoHotKey & AutoIT
  - Cloud API
  - JavaScript
  - Network Device CLI
  - PowerShell
  - Python
  - Unix Shell
  - Visual Basic
  - Windows Command Shell
- Container Administration Command
- Deploy Container
- Exploitation for Client Execution
- Inter-Process Communication (1/3)
  - Component Object Model
  - Dynamic Data Exchange
  - XPC Services
- Native API
- Scheduled Task/Job (0/5)
- Serverless Execution
- Shared Modules
- Software Deployment Tools
- System Services (0/2)
- User Execution (2/3)
  - Malicious File
  - Malicious Image
  - Malicious Link
- Windows Management Instrumentation

- Acc Mar (1/6
- BITS
- Boo Aut Exe (1/1
- Boo Initi Scri
- Bro Exte
- Con Hos Bina
- Cre Acc
- Cre Mo Sys Pro

Manual efforts throw up roadblocks

Understanding where tests fail requires significant manual effort and resources—reverse engineering, tooling expertise, and extensive research—that holds up exercises or ends up passed up entirely. To even understand whether a technique was **observed**, one might have to:

Dive into opaque detection logic

Security researchers need to understand the defensive tool inside and out to consider how they are prone to log or capture the expected behavior

Reverse engineer adversary tools

Knowing what signatures are left behind in the various executions of an adversary’s tooling can often start with a full breakdown of its code.

Research previous attack patterns

Knowing the typical tactics and trends of an attacker can set researchers on the right path, but all of this work takes more time.

2

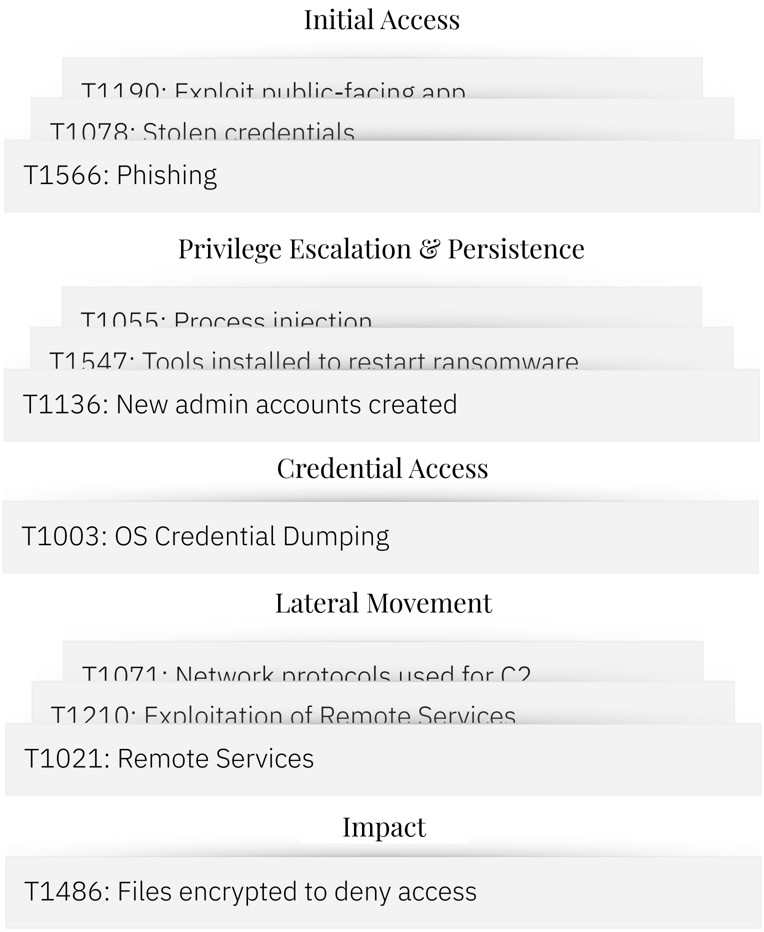
Deconstructing attacks

Finding solutions in modular-based testing

Deconstructing the deck of cards

By breaking down complex attack scenarios into discrete techniques, we can begin to address *tradecraft uncertainty* and prioritize our assessment to those which matter most in the context of our business.

Take the example here - a common ransomware attack includes a large has a large number of possible techniques permutations we would need to test to effectively gauge our defenses.



Initial Access

T1190: Exploit public-facing app

T1078: Stolen credentials

T1566: Phishing

Privilege Escalation & Persistence

T1055: Process injection

T1547: Tools installed to restart ransomware

T1136: New admin accounts created

Credential Access

T1003: OS Credential Dumping

Lateral Movement

T1071: Network protocols used for C2

T1210: Exploitation of Remote Services

T1021: Remote Services

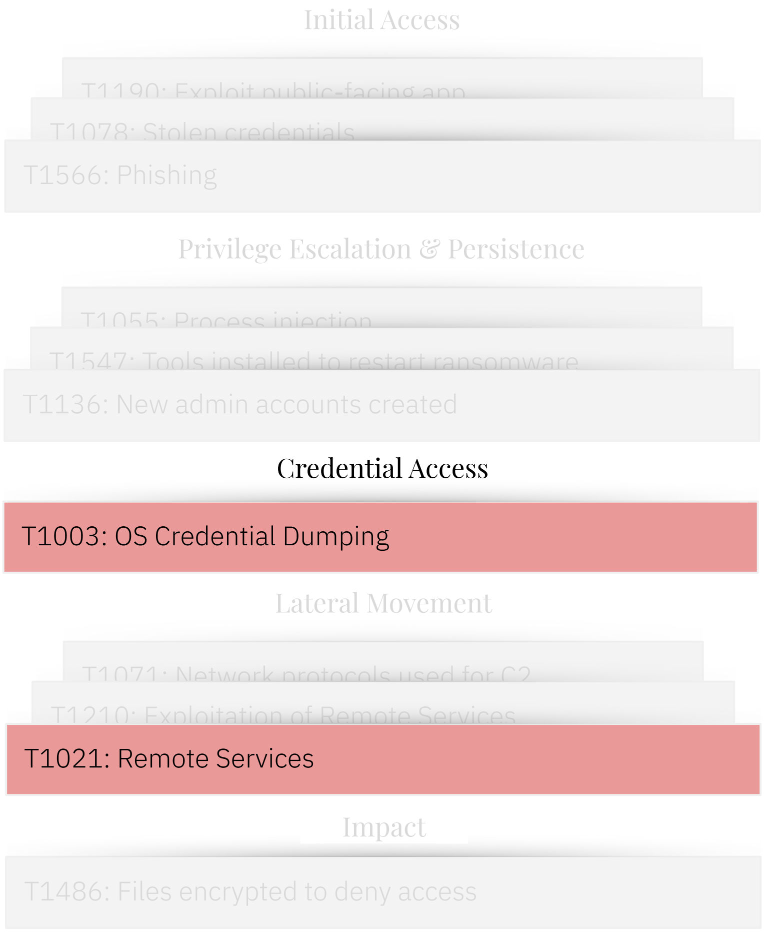
Impact

T1486: Files encrypted to deny access

Prioritizing the techniques that matter

Where testing and writing detections for each of the permutations of these techniques might overwhelm detection needs, we can streamline to specific **chokepoints** that are essential for an attacker by assessing which techniques and procedures are mandatory for a given attack path.

Where attackers can switch from phishing to brute-forcing credentials, they can't avoid extracting credential material, where we can focus more efficient detection & response efforts.



Initial Access

T1190: Exploit public-facing app

T1078: Stolen credentials

T1566: Phishing

Privilege Escalation & Persistence

T1055: Process injection

T1547: Tools installed to restart ransomware

T1136: New admin accounts created

Credential Access

T1003: OS Credential Dumping

Lateral Movement

T1071: Network protocols used for C2

T1210: Exploitation of Remote Services

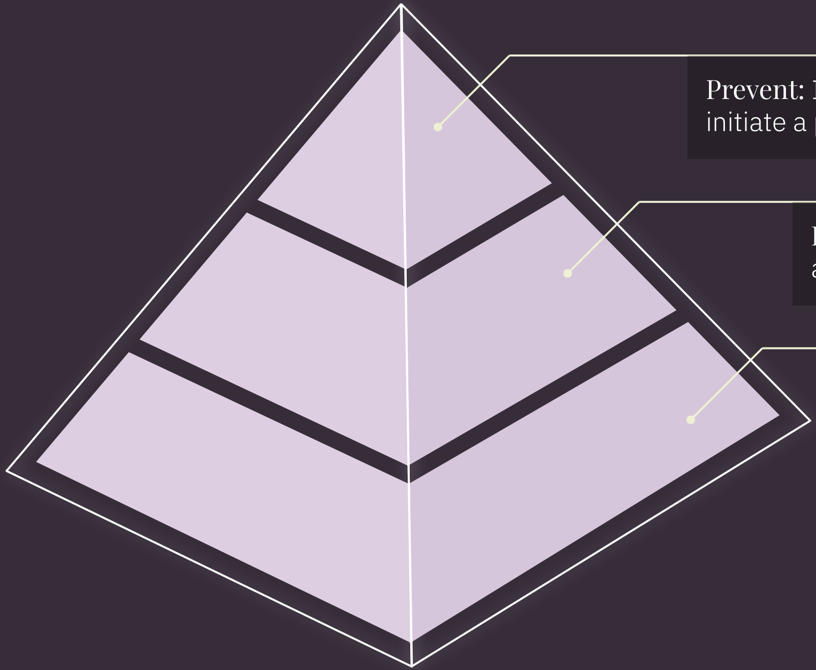
T1021: Remote Services

Impact

T1486: Files encrypted to deny access

A modular approach to testing critical techniques

As we look at threats in a modular format, we can do the same with the data we look for to better understand where our remediation efforts need to be targeted. The goal is to enable attacking and defending teams to **focus their efforts.**



Prevent: Is our detection appropriately tuned to automatically initiate a prevention like quarantining the device?

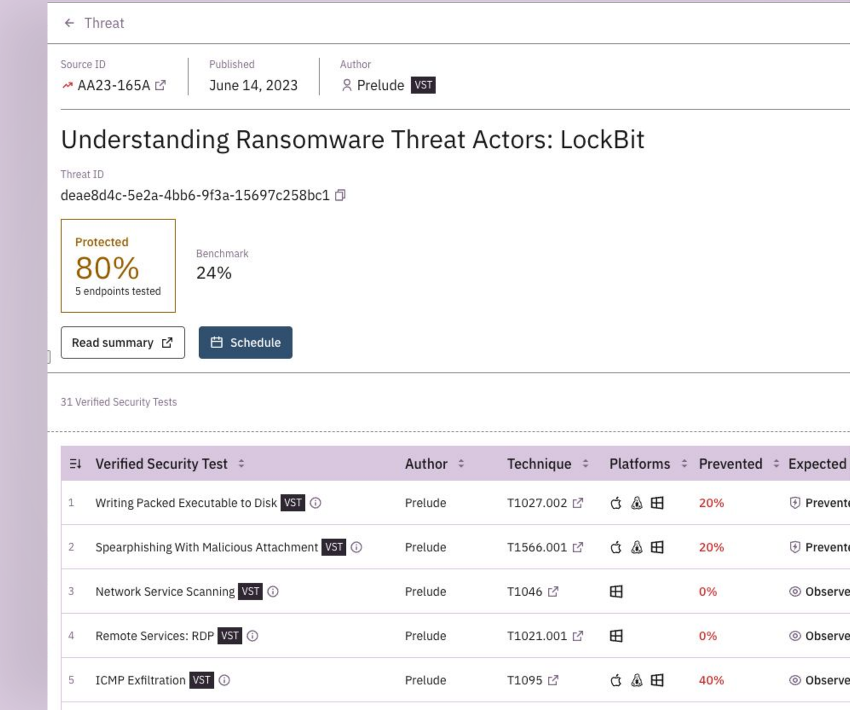
Detected: Did we have accurate detections to classify and alert about that execution?

Observed: Did our EDR observe the raw telemetry of the technique, such as a registry key creation?

3

A practical application

Using Prelude Detect for modular purple teaming



← Threat

Source ID: AA23-165A
Published: June 14, 2023
Author: Prelude VST

Understanding Ransomware Threat Actors: LockBit

Threat ID
deae8d4c-5e2a-4bb6-9f3a-15697c258bc1

Protected
80%
5 endpoints tested

Benchmark
24%

31 Verified Security Tests

	Verified Security Test	Author	Technique	Platforms	Prevented	Expected
1	Writing Packed Executable to Disk VST	Prelude	T1027.002	🍎 🐧 ⊞	20%	Prevent
2	Spearphishing With Malicious Attachment VST	Prelude	T1566.001	🍎 🐧 ⊞	20%	Prevent
3	Network Service Scanning VST	Prelude	T1046	⊞	0%	Observe
4	Remote Services: RDP VST	Prelude	T1021.001	⊞	0%	Observe
5	ICMP Exfiltration VST	Prelude	T1095	🍎 🐧 ⊞	40%	Observe

Launching an exercise in Prelude Detect

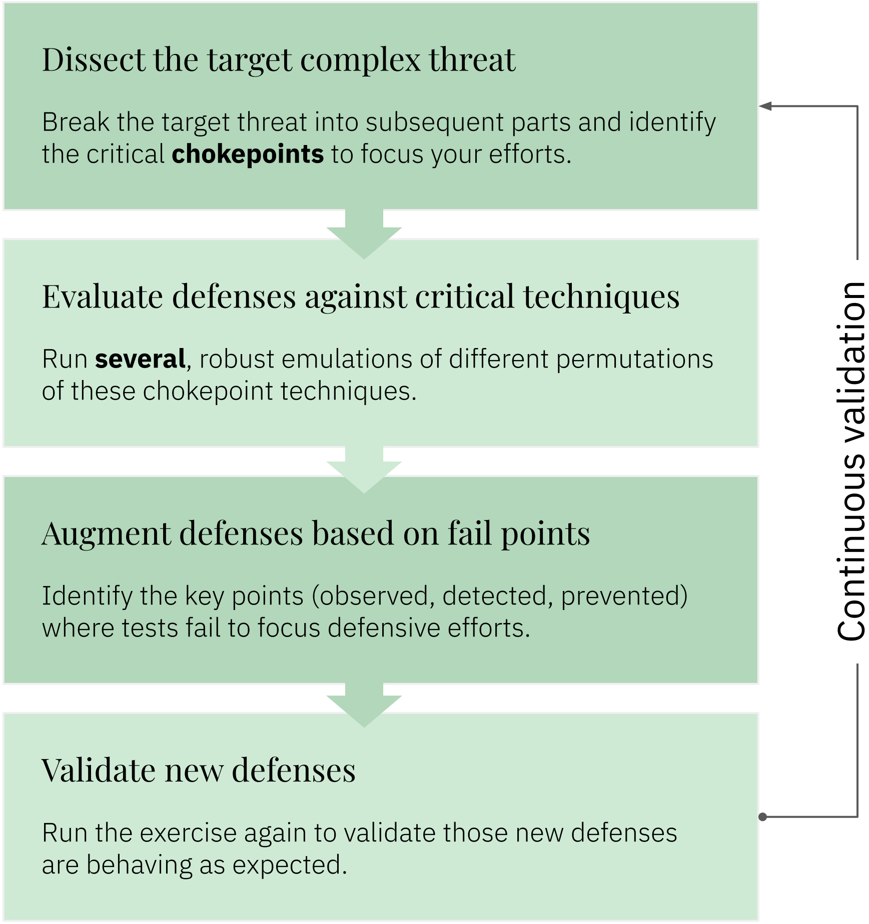
At Prelude, we're building a simple, automated way to simulate those critical threat techniques and evaluate an organization's defensive capabilities.

Every technique is evaluated against an expected outcome to help defensive teams prioritize their response efforts.

Let's run through a testing exercise.

A tech-powered approach to purple teaming

If traditional purple team exercises are caught in a slog of manual and lengthy efforts, the goal is to automate and streamline that process to enable **continuous improvement**.



Dissect the target complex threat

Break the target threat into subsequent parts and identify the critical **chokepoints** to focus your efforts.

Evaluate defenses against critical techniques

Run **several**, robust emulations of different permutations of these chokepoint techniques.

Augment defenses based on fail points

Identify the key points (observed, detected, prevented) where tests fail to focus defensive efforts.

Validate new defenses

Run the exercise again to validate those new defenses are behaving as expected.

[P]

4

What’s next?

Running continuous purple team exercises

Making the most of what you're given

Purple team exercises don't need to get caught in manual, arduous processes to be successful.

Test the techniques that actually matter

Spreading your efforts across a broad spectrum of techniques only holds you back. Focus on the techniques critical to attackers to achieve their goals—chokepoints.

Test them continuously

With manual efforts, purple team exercises can be few and far between, forcing you to often start from behind. With automation, continuously test and validate your efforts for a ongoing feedback loop.

Test them at scale

Because of the work required, these exercises are often limited in scope. With automation, we can scale our efforts to all the relevant segments of OS, policy configurations, and extraneous variables for a full picture.

5

Questions?

What’s on your mind

Thank you!

www.preludesecurity.com