FORTRA

DATASHEET (OFFENSIVE SECURITY)

BOF, UDRL & Sleepmask Development

Author: Alex Reid
Levels: Certified
Study time: 14 hours

ZERO POINT SECURITY

Get the 'BOF Development & Tradecraft' and 'URDL & Sleepmask Development' courses, created by Alex Reid and Zero-Point Security, in a single purchase, at a discounted price.

The BOF Development & Tradecraft course teaches how to write and unit test Beacon Object Files (BOFs) for use in Cobalt Strike and other C2 frameworks.

The UDRL and Sleepmask Development course teaches students how to apply low-level Windows knowledge and offensive tradecraft in the writing and development of Cobalt Strike's User-Defined Reflective Loader and Sleepmask components.

Course content is kept aligned with the latest Cobalt Strike release, keeping students equipped with the most up-to-date tradecraft with the inclusion of new modules on BeaconGate configuration, novel call stack spoofing, detection avoidance for library loading.

Training Content

BOF Development & Tradecraft

Getting Started

- Welcome
- Author's Note
- Software Requirements
- Windows Environment Setup
- Linux Environment Setup
- Resources

Introduction to BOF Development

- Background and Basics
- Windows API
- COFFLoader
- BOF Development on Linux
- BOF Development on Windows
- Aggressor Scripting

Practical 1: Ransomware Simulator

- Introduction
- Initial Setup
- Finding the Desktop folder
- Code Download
- Changing the Wallpaper and Leaving the Ransom Note
- Code Download
- Renaming Files
- Code Download
- Aggressor Script
- Code Download
- Closing

Practical 2: Iscsipl.exe UAC Bypass

- Introduction
- Initial Setup
- Code Review, Testing, and Analysis
- Initial Port of Code
- Code Download
- Replacing Resource Functionality
- Code Download
- Offensive Tradecraft
- Code Download
- Code Cleanup
- Code Download
- Aggressor Script
- Code Download
- Closing
- Resources

Practical 3: TGT Auto-Harvester

- Introduction
- Initial Setup
- Introduction to Stardust
- Calling Beacon APIs from Stardust
- Code Download
- Integrating Stardust into the BOF
- Code Download
- Monitoring for New Logins
- Code Download
- Dumping TGTs Automagically
- Code Download
- Patching BOF Arguments
- Code Download
- Teardown and Cleanup
- Code Download
- Aggressor Script
- Code Download
- Dancing with Sleep Mask
- Code Download
- Closing
- Resources

Update 1: BOFPatcher

- Background
- Design Process
- Code Download

Course Completion

- Course Evaluation
- 3 questions
- Certificate of Course Completion

UDRL and Sleepmask Development

Welcome

- Introduction
- Author's Note
- Software Requirements

Introduction to UDRL and Sleepmask Development

- Defining the Problem
- Component Requirements
- Project setup
- Build Automation
- Testing Payloads

UDRL: Extending Stardust

- Introduction to Stardust
- API Resolution Via Macros
- Debug Output
- Supporting Forwarded APIs
- Pointer Arithmetic
- Custom User Data
- Global Variables Without NtProtectVirtualMemory
- Removing Padding

UDRL: Basic Reflective Loading

- Parsing PE Headers
- Mapping Sections
- Populating the Import Address Table
- Processing Relocations
- Transferring Execution

Sleepmask: Basic Ekko Implementation

- Integrating Common Assets
- Preparing sleep_mask
- Masking Heap Memory
- Understanding Ekko
- Implementing Ekko

Sleepmask: Through the BeaconGate With Ekko

- Understanding the Problem
- Passing Stack Parameters
- Storing Return Values
- Integrating BeaconGate

Lightweight BeaconGate and Runtime Configuration BOF

- Introduction to Abbreviated Obfuscation
- Implementing Obfuscation with a BOF
- Toggling Between Processes

BeaconGate Support for BOFs

- Understanding Beacon APIs
- Routing Windows APIs
- Adding BeaconGate Functionality

UDRL: Module Stomping

- Introduction to Module Stomping
- Identifying Sacrifical DLLs
- Basic Implementation

UDRL: Advanced Module Stomping

- Background Research
- Manipulating Section Handles
- Replacing LoadLibraryEx
- Mapping Beacon's Unwind Info
- Finding and Fixing Sleepmask and BOF Unwind Info

Common: Control Flow Guard

- Understanding Control Flow Guard
- Enumerating CFG and Modifying the Bitmap

Evasion: Call Stack Spoofing

- Introduction to Hunt Sleeping Beacons and Call Stack Detection
- Exposing Timer Functionality to the UDRL
- Beacon Entry via NtContinue
- Concealing the Main Thread's Call Stack During Timer Execution
- Disguising Timer Stacks with ROP and JOP
- Suspicious Timers and Where to Hide Them

Evasion: Loading Sensitive Libraries Via Proxy

- Detecting Suspicious Library Loads
- Leveraging Call-Stack Stack Spoofing
- Implementing Proxied Library Loading

Evasion: Cleaning the LitterBox

- Introduction to LitterBox and Underlying Tools
- Patriot: Hiding Suspicious CONTEXTs
- Moneta: Freeing the Initial Allocation
- YARA: Addressing Static Signatures
- PE-sieve: Avoiding Entropy Checks

Alpharius: A CET-Compliant Stack Spoofing and Sleep Obfuscation Technique

- Explaining Intel Control-Flow Enforcement Technology
- CETs Impact on Tooling
- Introduction to Fibers as Spoofing Technique
- Designing the Alpharius Technique

Evasion: Assorted Tradecraft and Code Cleanup

- Extending BeaconGate for Unsupported APIs
- Exception Handling via Wow64PrepareForException
- Exiting Beacon Gracefully
- Ekko via Threadpool Timers
- Code Cleanup and Bug Fixes

Course Completion

- Areas for Future Exploration
- Closing
- Course Evaluation
- Certificate of Completion

