Breached Labs

OSEP Study Guide.

2025. A Breached Labs Resource.

What is the OSEP

The Offensive Security Experienced Penetration Tester (OSEP) certification represents advanced proficiency in enterprise-level penetration testing. Unlike more foundational certifications, OSEP focuses on sophisticated attack chains that simulate real-world threat actors operating in complex network environments. The course and exam (PEN-300) challenge students to demonstrate advanced persistence, lateral movement, and privilege escalation techniques against hardened targets.

At its core, OSEP embraces an assumed breach mindset, rather than focusing solely on initial access, it emphasizes what happens after gaining a foothold. The certification validates your ability to evade modern defenses, move laterally through segmented networks, and establish dominance in Active Directory environments. These skills reflect the realities of modern adversaries who employ multi-stage attack methodologies to achieve their objectives.

OSEP significantly differs from OSCP in both scope and depth. Where OSCP validates foundational penetration testing skills and single-system exploitation, OSEP requires chaining multiple techniques to bypass security controls like antivirus, application whitelisting, and endpoint protection.

Why OSEP

OSEP provides concrete validation of your ability to conduct sophisticated network intrusions that mirror real-world adversary tactics. This certification demonstrates to employers and clients that you can execute complex attack chains, adapt to defensive countermeasures, and think creatively when standard approaches fail, all critical skills for advanced security roles.

The certification opens doors to specialized career paths including dedicated Red Team positions, Senior Penetration Testing roles, Security Research opportunities, and Adversary Simulation specialists. These positions typically offer greater technical challenge, higher compensation, and more strategic involvement in security programs compared to entry-level security testing roles.

Mindset

Developing the right mindset is perhaps more important than technical skills for OSEP success:

Persistence & Adaptability: OSEP challenges frequently require trying multiple avenues when initial attempts fail. You must be willing to pivot approaches, adapt techniques to specific environments, and persevere through repeated blocking of your attack paths. This resilience mirrors real advanced adversaries who may spend weeks working around specific defenses.

Situational Awareness: After gaining initial access, understanding the environment becomes critical. What defenses are in place? Where are you in the network topology? What privileges do you have? Who else is logged in? This contextual understanding shapes your subsequent actions and helps avoid detection.

Attacker Simulation: Successful OSEP students think like goal-oriented adversaries rather than vulnerability scanners. Instead of trying to find every possible weakness, focus on pathways that advance your objectives most efficiently. This approach emphasizes practicality over exhaustiveness.

Methodical Approach: While creativity is valuable, systematic enumeration and exploitation remain essential. Develop structured processes for post-exploitation reconnaissance, privilege escalation attempts, and lateral movement to ensure thoroughness despite complex environments.

Defense Awareness: Understanding modern protection mechanisms, including how antivirus engines analyze files, how EDR solutions monitor processes, how AppLocker enforces policies, and how AMSI inspects scripts, allows you to develop effective evasion strategies.

Ideal path to OSEP

OSCP Certification/Equivalent Knowledge: Consider OSCP a minimum baseline for OSEP preparation. You should be comfortable with exploitation fundamentals, managing multiple targets simultaneously, and adapting public exploits to specific scenarios. Without this foundation, OSEP's advanced material will likely prove overwhelming.

Strong Networking Fundamentals: Beyond basic connectivity, understand how traffic flows through enterprise networks, including routing between subnets, firewall rule evaluation, proxy configurations, and network segmentation. This knowledge is critical for pivoting between network segments and evading network-based controls.

Windows Internals & Administration: Develop working knowledge of Windows architecture including processes, services, permissions models (especially privileges), registry structure, Windows Management Instrumentation (WMI), and PowerShell functionality. You should be comfortable navigating and manipulating Windows systems without relying on GUI tools.

Active Directory Fundamentals: Before attempting OSEP, familiarize yourself with Active Directory concepts including domains, forests, trust relationships, Kerberos authentication flow, LDAP queries, Group Policy Objects, and common AD objects. Understanding the relationships between these components provides the foundation for advanced AD attacks.

Scripting Proficiency: PowerShell proficiency is non-negotiable for OSEP success, it's both an administrative tool and an attack platform. Additionally, basic Python skills support custom tool development, while C# knowledge enables creating specialized payloads designed to evade defenses. The ability to read, modify, and create scripts separates OSEP from more tool-focused certifications.

Linux Competency: While OSEP emphasizes Windows environments, Linux skills remain important for managing attack platforms, pivoting through compromised Linux systems, and utilizing Linux-based attack tools. Comfort with command-line operations, SSH tunneling, and basic Linux services is expected.

Familiarity with Common Pentesting Tools: Understand tools like Metasploit, Nmap, and Burp Suite, but recognize that OSEP requires using them differently than in OSCP, often in more targeted ways, through proxies, or with custom modifications to avoid detection.

Pre-Course Preparation

Solidify OSCP Concepts: Ensure you've mastered fundamental exploitation techniques including buffer overflows, web application attacks, and basic exploit modification. These skills provide the technical foundation upon which OSEP's more advanced content builds.

Deep Dive into Windows & AD: Build a home lab with a basic Active Directory implementation (Domain Controller plus several workstations/servers). Practice creating and managing users, groups, and permissions. Familiarize yourself with basic AD enumeration tools like

PowerView and BloodHound to understand relationship visualization and attack path identification.

Learn/Improve C#: Focus on understanding how to leverage .NET framework capabilities, particularly Platform Invocation Services (PInvoke) for direct Windows API calls and reflection for dynamic code execution. These techniques are fundamental to creating AV-evasive payloads and custom tools.

Master PowerShell: Move beyond basic commands to understand PowerShell's scripting capabilities, remoting functionality, and methods for interacting with .NET classes and WMI. Practice creating PowerShell scripts that can run entirely in memory without touching the disk. Learn PowerShell's security features and how they can be bypassed.

AV Evasion Fundamentals: Study basic concepts of antivirus evasion including obfuscation techniques, in-memory execution, and signature avoidance. Follow security blogs and conference talks that discuss modern evasion methods to understand the cat-and-mouse game between attackers and defenders.

Practice AD Machines: Complete Active Directory-focused challenges on platforms like Hack The Box (especially dedicated AD tracks and Pro Labs such as Dante) and TryHackMe AD learning paths. These environments provide practical experience with common AD misconfigurations and attack techniques in controlled environments.

The Community

The OffSec community provides invaluable support during your OSEP journey. The official forums and Discord channels connect you with fellow students and alumni who understand the challenges you're facing. When engaging with these communities you should seek hints rather than direct solutions when stuck, and ask about general approaches rather than specific answers.

Remember that while community support is valuable, OffSec's learning philosophy emphasizes personal problem-solving. Use these resources to overcome technical obstacles or clarify concepts, but rely primarily on your own efforts to develop the independent thinking required for the exam.

Course Structure

The PEN-300 course fundamentally differs from entry-level penetration testing material by focusing on advanced adversary simulation within defended environments. Unlike OSCP, which emphasizes individual vulnerability exploitation, OSEP teaches you to operate as a sophisticated threat actor, chaining techniques to navigate through modern security controls.

Beyond basic phishing, you'll develop custom macro payloads designed to bypass endpoint protection. This includes crafting malicious Office documents (.docm/.docx), browser exploits, and executable payloads that evade detection while establishing reliable initial access. You'll learn to blend social engineering with technical evasion to create truly effective attack vectors.

Understanding Windows process architecture becomes crucial as you learn multiple techniques for injecting malicious code into legitimate processes. This includes classic DLL injection, reflective loading, process hollowing, and more sophisticated techniques that leave minimal forensic evidence.

Moving beyond simple signature evasion, you'll develop payloads that evade heuristic and behavioral detection. This involves understanding how modern AV engines analyze files and processes, then systematically circumventing these detection mechanisms through code obfuscation, in-memory execution, and strategic manipulation of execution behaviors.

As organizations implement stricter execution controls like AppLocker and WDAC policies, you'll learn multiple techniques to execute code within these constraints. This includes leveraging trusted utilities, LOLBins (Living Off the Land Binaries), signed script abuse, and other methods that work within the confines of restrictive execution policies.

The course expands significantly on traditional lateral movement techniques, teaching you to move through networks using encrypted channels, alternative authentication mechanisms, and novel command execution methods. Both Windows and Linux environments are covered, with emphasis on bypassing network segmentation and endpoint restrictions.

Building on basic AD attacks, you'll learn sophisticated techniques for compromising domain controllers, exploiting trust relationships, manipulating domain objects, and achieving persistent domain control. This includes advanced Kerberos attacks, ACL manipulation, and sophisticated privilege escalation chains unique to enterprise AD environments.

The Lab

The lab simulates different network zones with varying security postures, requiring you to adapt techniques as you move deeper into the environment. Each segment introduces new security controls that invalidate previously successful techniques. Unlike more basic labs, you'll encounter functioning security tools including antivirus, application whitelisting, network monitoring, and endpoint protection. These aren't simplified versions but realistic implementations of common security controls.

The lab design forces incremental learning by gradually introducing more sophisticated defensive measures. Techniques that work in initial networks will fail in later segments, requiring you to evolve your approach. Each network segment teaches specific lessons about enterprise penetration testing, from initial access vectors through privilege escalation chains to domain compromise scenarios.

Lab Time

Your lab access represents the most valuable learning resource for OSEP preparation. Don't be satisfied with simply running provided scripts or commands. Take time to understand each technique's underlying mechanics, how they work, why they evade specific defenses, and under what conditions they might fail. Modify examples to test boundary conditions and defensive limits.

For each successful technique, analyze what security mechanisms were bypassed. For failed attempts, investigate why they were detected or blocked. This defense-aware perspective develops crucial intuition about what will likely succeed in new environments.

As you progress through the lab, build a personalized collection of tools, scripts, and methodology notes that fit your approach. It’s heavily recommended to use Obsidian to compile a repository of notes.

You should complete all six challenges within the lab environment before sitting the exam, as to understand what the exam will ask of you exactly.

Initial Access

In real-world operations, the initial foothold phase often serves as your entryway into the target environment, but the OSEP frequently assumes you've already achieved this access to focus on more advanced techniques.

Office Macro Exploitation: Creating sophisticated VBA macros that blend legitimate functionality with malicious code execution while evading detection. This includes understanding Office's trust model, macro security settings, and document properties that influence execution.

HTML Applications (HTA): Developing HTA files that leverage mshta.exe execution to bypass application whitelisting while appearing as legitimate web content to users.

LNK File Abuse: Crafting malicious shortcut files that execute commands through cmd.exe or powershell.exe while displaying innocuous icons and file properties.

JScript/VBScript Payloads: Utilizing Windows Script Host to execute code through .js or .vbs files, often combined with obfuscation techniques to evade detection.

Once inside, comprehensive enumeration becomes your foundation for all subsequent actions. You should evaluate a complete picture of your target environment:

User Context Assessment: `whoami /all` reveals not just your username but critical group memberships, privileges, and SIDs that might enable privilege escalation paths.

Network Configuration Analysis: Beyond basic `ipconfig` output, understanding subnet layouts, routing tables, and firewall configurations through `netstat -ano`, `route print`, and `netsh` commands helps identify potential pivot points.

Process & Service Enumeration: Identifying running processes, services, and their permissions provides exploitation targets and potential defense mechanisms to avoid.

Environment Reconnaissance: Analyzing environment variables, installed software, patch levels, and scheduled tasks reveals information about the system's purpose and potential weaknesses.

Engagements often hinge on discovering stored credentials and sensitive information to accelerate attack progression. Memory-based extraction techniques using Mimikatz provide immediate access to plaintext passwords, NTLM hashes, and Kerberos tickets through commands like `sekurlsa::logonpasswords` or `sekurlsa::tickets`.

DPAPI abuse allows decryption of credentials stored in Windows Credential Manager and browser vaults. Implementation of malicious Security Support Providers via Mimikatz enables capture of credentials from subsequent logons.

Thorough file system and registry hunting for configuration files, connection strings, and application settings frequently reveals cleartext credentials or reused passwords.

Browser credential stores represent additional high-value targets, accessible through direct database manipulation or specialized extraction tools.

Evasion

For PowerShell AMSI bypasses, you should focus on memory modification techniques that target AMSI's internal components. These techniques typically involve using reflection to access non-public fields and methods within the AMSI infrastructure, then modifying them to disable scanning functionality.

The most common approach is to force AMSI initialization to fail by setting internal flags that cause the scanning engine to believe it has already encountered an error.

Another effective technique involves breaking suspicious strings into fragments that are reassembled at runtime, preventing signature detection of the bypass code itself. You are given a fairly simple-to- apply AMSI bypass in the course work, use it.

Signature-based detection identifies malicious code through pattern matching. Obfuscation changes your code's appearance without altering functionality through string manipulation, control flow alterations, and variable randomization, as a result preventing signature recognition while maintaining operability.

Encryption takes this further by completely transforming payloads into unreadable formats until execution time. **XOR** encoding is commonly used for its simplicity.

Reflective DLL loading bypasses standard Windows loading mechanisms, while process hollowing creates legitimate-appearing processes before replacing their memory with malicious code. Both techniques avoid the file system operations that typically trigger security alerts.

Direct syscalls bypass API hooking by directly interacting with the Windows kernel instead of using monitored API functions. By understanding the distinction between Windows Native API and Win32 API, you can implement syscalls that evade detection mechanisms.

PInvoke allows C# code to access low-level system functionality, with dynamic resolution techniques preventing static analysis from identifying suspicious API usage patterns.

Living Off The Land Binaries[1] and Scripts (LOLBAS/LOBAS) are legitimate system tools that can be repurposed for offensive operations while bypassing application whitelisting. MSBuild.exe executes C# code embedded within XML project files, while Regsvr32.exe can run scripts through Scriptlet registration.

InstallUtil.exe executes malicious code through installation callbacks, and CertUtil.exe provides unexpected functionality for encoding and file downloads. All these binaries typically remain whitelisted due to their legitimate system purposes.

PowerShell alternatives offer execution paths even in restricted environments. Runspaces can create isolated execution environments with potentially different restrictions than the main console. The Add-Type cmdlet compiles and executes C# code from within PowerShell scripts, while WMI providers offer execution capabilities that may circumvent standard restrictions.

[1] https://lolbas-project.github.io/

PowerShell Constrained Language Mode (CLM) restricts PowerShell functionality but has several bypass vectors. Environment-based CLM can sometimes be circumvented by modifying variables before PowerShell initialization.

Similar to the AMSI bypass, you can reuse a fairly simple CLM bypass which is provided to you in the course work.

Active Directory

BloodHound revolutionized AD enumeration by visually mapping domain relationships in a graph database. SharpHound collectors gather information that BloodHound processes to reveal connections between users, computers, groups, and permissions.

Understanding BloodHound's node types, edge relationships, and query language allows quick identification of privilege escalation paths that would be nearly impossible to discover manually. Use it, a lot.

Mimikatz extracts credentials from LSASS memory using modules like sekurlsa::logonpasswords for local access and lsadump::dcsync for remote password data replication.

When direct memory access triggers alerts, alternatives like procdump capture memory for offline analysis. SAM/SYSTEM hive extraction provides local account hashes when domain credentials aren't accessible.

Kerberoasting exploits the Kerberos protocol by requesting service tickets for accounts with Service Principal Names (SPNs), then extracting and cracking their password hashes offline. Tools like GetUserSPNs and Rubeus automate this process.

AS-REP Roasting targets accounts with Kerberos preauthentication disabled, allowing attackers to request authentication data for offline cracking without prior authentication.

Pass-the-Ticket reuses extracted Kerberos tickets for authentication without knowing the original password. Over-Pass-the-Hash converts NTLM hashes to Kerberos tickets, combining hash extraction with Kerberos authentication to bypass restrictions that block traditional hash passing techniques.

Pass-the-Hash authenticates with NTLM hashes without requiring plaintext passwords. Impacket tools, Mimikatz, and Invoke-TheHash offer different implementations for various scenarios.

LLMNR/NBT-NS poisoning captures authentication attempts through fallback name resolution protocols. When combined with SMB Relay, these credentials can be forwarded to target systems for lateral movement. AD Certificate Services exploits include template misconfiguration and enrollment protocol weaknesses. Print Spooler vulnerabilities enable remote code execution through printer driver manipulation.

Forest trusts connect separate Active Directory infrastructures, often with incomplete security boundaries. Understanding trust types and security filtering helps identify cross-forest privilege escalation opportunities. MSSQL

Server instances frequently operate with excessive privileges, offering command execution through xp_cmdshell and lateral movement via linked servers.

Database instances **often** provide alternative paths to domain compromise when traditional vectors are secured.

Windows Lateral Movement

Windows environments offer numerous lateral movement options using built-in protocols. SMB/RPC-based techniques include PsExec for remote command execution through admin shares, while Impacket's Python implementations (psexec.py, smbexec.py, wmiexec.py) provide similar functionality with greater flexibility.

WinRM enables PowerShell remoting through Invoke-Command when properly configured, offering native encrypted communications. Scheduled tasks can execute commands on remote systems through schtasks.exe or Task Scheduler COM objects. WMI provides alternate execution through both command-line (wmic) and PowerShell interfaces.

RDP sessions can also be hijacked by compromising the target system, allowing attackers to take over existing authenticated sessions.

Linux Lateral Movement

SSH provides primary remote access in Linux environments. Key reuse exploits administrators' habit of using identical keys across systems, allowing movement to any server with the same key. Agent hijacking targets the SSH agent's in-memory keys by accessing socket files, while forwarding enables pivoting through intermediate systems with preserved authentication.

Similar to the OSCP, misconfigured sudo permissions can also grant elevated access to specific commands. Identifying these permissions through sudo -l and exploiting command features (such as shell escapes or file read/write capabilities) can provide privilege escalation.

Cron jobs with improper permissions may enable lateral movement through scheduled task manipulation.

The Exam

The OSEP exam spans 47 hours and 45 minutes, providing ample time to tackle a challenging objective-based penetration test within a simulated corporate network. Resist the urge to rush forward; instead, methodically document each compromised system's configuration, permissions, and network connections.

Map the environment as early as possible using both automated tools (BloodHound when available) and manual enumeration. Understanding the network topology and domain structure provides context for each compromise and helps prioritize targets based on potential value.

Document continuously throughout the exam, treating this documentation as your report draft. Capture commands with their output, take screenshots of significant findings, and maintain a chronological narrative.

Be prepared to modify course scripts and tools to bypass specific defenses. Small adjustments to evasion techniques or payload delivery methods often overcome environmental restrictions. Having a library of modifiable scripts ready before the exam saves valuable time.

The Report

OffSec expects comprehensive, professional reports that document your methodology and findings. Structure your report with clear sections: executive summary, methodology, findings, and recommendations. Reports must demonstrate reproducibility, providing sufficient detail for others to validate your findings. Include command syntax, tool configurations, and specific parameters used during testing. Avoid **ambiguity** in your explanations of critical attack paths.

Provide concrete evidence for each required objective. Screenshots should capture both the execution of critical commands and their output, particularly showing system identifiers that prove specific target compromise. Include terminal output demonstrating privilege levels, access to protected resources, and successful exploitation of key vulnerabilities.

For domain compromise, evidence typically includes proof of Domain Admin group membership, access to critical domain systems, and the ability to extract sensitive domain data.

Don't forget to submit your flags into the portal, and take the necessary screenshots. You may be liable to fail your exam if you fail to do so.

Recommended Practice

HackTheBox Pro Labs provides realistic environments for OSEP preparation. RastaLabs and Offshore closely mirror OSEP complexity, featuring Active Directory, defensive measures, and multi-stage attack paths. Other AD-focused labs like Dante and Cybernetics offer additional practice with varying defensive configurations and network topologies.

Reminders

Master Active Directory enumeration with both automated tools and manual techniques. BloodHound provides the big picture, but understanding raw LDAP queries and PowerView commands enables enumeration in restricted environments. Combine approaches to develop a comprehensive understanding of domain relationships and security weaknesses.

Develop deep understanding of authentication protocols, particularly Kerberos and NTLM attack vectors. Know when each attack applies and how defensive measures might impact success. Practice transitioning between authentication types as environmental restrictions change.

Practice evasion techniques continuously against increasingly sophisticated defenses. Modern environments rarely permit unmodified payloads or tools to execute successfully.

Resources

Each resource has a degree of necessity. The degree of necessity is based on its importance when it comes to the lab challenges, and subsequently the exam.

A red label, marks high importance, whilst a green label, marks something of lesser importance, that can just be read for further understanding of a given, particular topic.

PowerView | Powershell-based tool for Active Directory Enumeration
https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/PowerView.ps1

FodHelper | Powershell-based UAC bypass
https://github.com/winscripting/UAC-bypass/blob/master/FodhelperBypass.ps1

PrivescCheck | Powershell-based Privilege Escalation Vector Enumeration Tool
https://github.com/itm4n/PrivescCheck

OSEP-1 Repo | Variety of useful tools & commands
https://github.com/ksecurity45/OSEP-1

VBA Bin Runner | FUD VBA Generator
https://github.com/ScriptIdiot/vba_bin_runner

ADCheatGen Repo | Useful Active Directory Exploitation Tools & Commands
https://github.com/ksecurity45/ADCheatgen

HollowGhost | C# Process Hollowing Runner
https://github.com/Logan-Elliott/HollowGhost

SharpUp | The C# variant of PowerUp
https://github.com/GhostPack/SharpUp

Bypass-UAC.ps1 | A one-for-all solution for any UAC bypass.
https://github.com/FuzzySecurity/PowerShell-Suite/blob/master/Bypass-UAC/Bypass-UAC.ps1

Windows Privilege Escalation | Notes necessary to escalate your permissions to the system user.
https://book.hacktricks.xyz/windows-hardening/windows-local-privilege-escalation

DACL Attacks | How to identify & exploit DACL (Discretionary Access Control Lists)
https://www.bordergate.co.uk/dacl-attacks/

RBCD Exploitation | How to abuse resource-based constrained delegation in an AD environment.
https://medium.com/@offsecdeer/a-practical-guide-to-rbcd-exploitation-a3f1a47267d5

Active Directory Cheat Sheet | Every command you'll need to enumerate an AD.
https://github.com/S1ckB0y1337/Active-Directory-Exploitation-Cheat-Sheet

PowerView Cheat Sheet | List of PowerView commands you can hammer against a domain.
https://netwerklabs.com/powerview-cheat-sheet/

Pentesting MSSQL Microsoft SQL Server | Enumeration + exploitation of MSSQL services.
https://book.hacktricks.xyz/network-services-pentesting/pentesting-mssql-microsoft-sql-server

Attacking MSSQL | Exploitation of MSSQL services.
https://www.bordergate.co.uk/attacking-mssql/

ReadLapsPassword | How to read LAPS passwords from within Kali.
https://www.thehacker.recipes/ad/movement/dacl/readlapspassword

LDAP CrackMapExec | Utilizing CrackMapExec to enumerate a user account.
https://crackmapexec.popdocs.net/protocols/ldap-crackmapexec

Golden Ticket | Crafting Golden Tickets
https://book.hacktricks.xyz/windows-hardening/active-directory-methodology/golden-ticket

Bloodhound | How to use BloodHound to collect domain information.
https://book.hacktricks.xyz/windows-hardening/active-directory-methodology/bloodhound

OSEP Reference | Detailed cheatsheet containing a variety of different commands in OSEP context.
https://github.com/In3x0rabl3/OSEP/blob/main/osep_reference.md

Linux Shellcode Loaders | Useful Shellcode Loaders for Linux-based hosts
https://github.com/chvancooten/OSEP-Code-Snippets/tree/main/Linux%20Shellcode%20Loaders

PowerMad.ps1 | Leverage MachineAccountQuota and Create Machines on a given domain.
https://github.com/Kevin-Robertson/Powermad/blob/master/Powermad.ps1

LolBas Project | Evade detection without the need for external tools.
https://lolbas-project.github.io/#

FakeLib.sh | Generate shared libraries for linux quickly.
https://github.com/eblazquez/fakelib.sh

CSWin32 | A more modern way of referencing Win32 API
https://fermiparadox.gitbook.io/fermiparadox/ethicalhackingarticles/coding/using-cswin32-instead-of-pinvoke.net-signatures

NiShang | Powerful Powershell-based Payload Framework
https://github.com/samratashok/nishang/

ForSec OSEP Notes | A Complete OSEP Cheatsheet
https://forsec.nl/osep.html

Ending Notes

Embrace the challenges, learn from obstacles, and maintain curiosity about both offensive techniques and defensive countermeasures. Good luck on your journey to becoming an Offensive Security Experienced Penetration Tester!