Agenda / Timetable

Endpoint Security Insights: Shellcode Loaders & Evasion Fundamentals

Contents

Day 1: Fundamentals

Module	Module Details	Timetable
Official Course Begin	▪ Kick off for day 1	08:15 am
Course Introduction	▪ An overview of the topics that will be covered over the next 4 days. ▪ Course objectives and expectations. ▪ An introduction to the tools, code, etc. that will be used during the course.	08:30 am – 09:00 am
Windows Internals Basics & Endpoint Security a Primer	▪ A brief introduction to Windows internals ▪ A brief introduction to antivirus and EDR ▪ Architecture of a modern EDR ▪ Differences between antivirus (AV) and EDR ▪ Introduction to relevant endpoint security mechanisms.	09:00 am – 10:00 am
	Coffee Break	10:00 am – 10:15 am
Deep-Dive Shellcode Loader	▪ Main components of a shellcode loader ▪ Technical comparison of shellcode from different C2 frameworks ▪ Evasion options from a shellcode perspective ▪ Evasion options from a loader perspective	10:15 am – 11:00 am
Staged vs Non-Staged Shellcode	▪ Staged vs. non-staged shellcode introduction ▪ Theory: Meterpreter shellcode staged/ non-staged ▪ Practice: How to create staged/ non-staged shellcode with Meterpreter ▪ Limitations of non-staged shellcode in the context of Visual Studio.	11:00 am – 11:30 am
Hands-on: A base - Win32 Classic Loader	▪ A deep dive int to a Win32 classic loader, which builds the basic for all loaders later in this course ▪ Build the Win32 classic loader in Visual Studio ▪ Debugging the loader with x64dbg, memory allocation, copying shellcode, and executing shellcode.	11:30 pm – 12:00 pm

	▪ Weaknesses of the Win32 loader and how to build an evasive shellcode loader step-by-step	
	Lunch Break	12:00 pm – 01:00 pm
Hands-on: A base - Win32 Classic Loader	▪ A deep dive int to a Win32 classic loader, which builds the basic for all loaders later in this course ▪ Build the Win32 classic loader in Visual Studio ▪ Debugging the loader with x64dbg, memory allocation, copying shellcode, and executing shellcode. ▪ Weaknesses of the Win32 loader and how to build an evasive shellcode loader step-by-step	01:00 pm – 02:30 pm
Hands-on: Shellcode in PE	▪ An introduction to sections in portable executable structure ▪ Building a Win32 loader that stores shellcode in the .data section ▪ Building a Win32 loader that stores shellcode in the resource section (.rsrc) ▪ Debugging both loaders, debug shellcode memory allocation, shellcode position etc.	02:30 pm – 04:00 pm
	Coffee Break	04:00 pm – 04:15 pm
	Summary and Q&A for day 1	04:15 pm – 05:00 pm

Day 2: Memory Manipulation & Shellcode Enc/Dec

Module	Module Details	Timetable
Official Course Begin	▪ Kick off for day 2	08:15 am
Hands-on: Memory Protection	▪ An introduction to memory protection constants in Windows and how to use them in context of shellcode loaders ▪ Building a Win32 loader which allocates rw-memory, changes memory protection to rx-memory etc. ▪ Debugging the loader, debug position, rw allocated memory, changing of memory etc.	08:30 am – 10:00 am
	Coffee Break	10:00 am – 10:15 am
Hands-on: Shellcode Encoding	▪ Deep dive into shellcode encoding types like base64, double base64, MACs, UUIDs, shellcode-as-words ▪ Introduction to the CodeFuscation tool, which will be used throughout the course to encode shellcode. ▪ Building a Win32 loader which supports double base64 encoded shellcode, decoding at runtime etc. ▪ Building a Win32 loader which supports shellcode-as-words encoded shellcode, decoding at runtime etc. ▪ Debugging the loader with x64dbg, debug position of encoded shellcode, shellcode decoding, memory allocation, etc.	10:15 am – 12:00 pm
	Lunch Break	12:00 pm – 01:00 pm
Hands-on: Shellcode Encryption	▪ Deep dive into shellcode encoding types like XOR, RC4 and AES ▪ Introduction to the CodeFuscation tool, which will also be used throughout the course encrypt shellcode.	01:00 pm – 02:30 pm

	▪ Building a Win32 loader which supports RC4 encrypted shellcode, decoding at runtime etc. ▪ Debugging the loader with x64dbg, debug position of encrypted shellcode, shellcode decryption, memory allocation, etc.	
	Coffee Break	02:30 pm – 02:45 pm
Hands-on: Shellcode on Web Server	▪ How to store shellcode outside of loader (PE) and instead host it on webserver or cloud service like GitHub and Microsoft Azure ▪ Learn how to store shellcode on GitHub in a private repository and make it accessible via Privat Access Token (PAT) ▪ Learn how to store shellcode in a private azure blob by using a SAS-URL. ▪ Building a Win32 loader which supports download encrypted shellcode from private GitHub repository ▪ Building a Win32 loader which supports download encrypted shellcode from private Azure blob storage ▪ Debugging both loaders, debug for position from GitHub PAT, Azure SAS-URL, shellcode decryption etc.	02:45 pm – 04:45 pm
	Summary and Q&A for day 2	04:45 pm – 05:15 pm

Day 3: Advanced Memory Techniques

Module	Module Details	Timetable
Official Course Begin	▪ Kick off for day 3	08:15 am
Hands-on: Heap Memory Allocation	▪ Deep dive into memory allocation via heap allocation, difference between VirtualAlloc and HeapAlloc etc. ▪ Build a Win32 loader that supports UUID encoded shellcode in PE, allocating memory via HeapAlloc, etc. ▪ Debugging the loader, debug position from shellcode, position from heap object, change from memory protection etc.	08:30 am – 10:00 am
	Coffee Break	10:00 am – 10:15 am
Hands-on: Mapped Memory	▪ Deep dive into mapped memory/memory mapping, why should we use mapped memory from an evasion perspective? ▪ Building a Win32 loader which supports double Base64 encoded shellcode in PE and mapped memory ▪ Debugging the loader, debug loading from module, base address from .text section, shellcode stomping etc.	10:15 am – 12:00 pm
	Lunch Break	12:00 pm – 01:00 pm
Hands-on: Module Stomping	▪ Deep dive into module stomping, why should we use module stomping, how to choose a module for stomping etc. ▪ Building a Win32 loader which supports shellcode-as-words encoded shellcode in PE, module stomping etc. ▪ Debugging the loader, debug loading from module, base address from .text section, shellcode stomping etc.	01:00 pm – 02:30 pm

	Coffee Break	02:30 pm – 02:45 pm
Hands-on: Function Stomping	▪ Deep dive into function stomping, why should we use function stomping, how to choose a function in a module for function stomping etc. ▪ Building a Win32 loader which supports double base64 encoded shellcode in PE, function stomping etc. ▪ Debugging the loader, debug loading from module, base address from .text section from the targeted function inside the module, change of memory protection, shellcode stomping etc.	02:45 pm – 04:30 pm
	Summary and Q&A for day 3	04:30 pm – 05:15 pm

Day 4: Shellcode Execution Techniques & Fine Tuning

Module	Module Details	Timetable
Official Course Begin	Kick off for day 4	08:15 am
Hands-on: Asynchronous Procedure Calls (APCs)	▪ Introduction into APCs, why should we use APCs for shellcode execution etc. ▪ Building a Win32 loader which supports UUID encoded shellcode in PE, function stomping, shellcode execution via APCs in context of SleepEx and NtTestAlert function. ▪ Debugging the loader, debug loading from module, base address from .text section from the targeted function inside the module, change of memory protection, shellcode stomping, shellcode execution via APCs etc.	08:30 am – 10:00 am
	Coffee Break	10:00 am – 10:15 am
Hands-on: Callback Functions	▪ Introduction into callback functions, why should we use Callback functions for shellcode execution etc. ▪ Building a Win32 loader which supports shellcode-as-words encoded shellcode in PE, function stomping, shellcode execution via callback function etc. ▪ Debugging the loader, debug loading from module, base address from .text section from the targeted function inside the module, change of memory protection, shellcode stomping, shellcode execution via callback function etc.	10:15 am – 12:00 pm
	Lunch Break	12:00 pm – 01:00 pm
Hands-on: Thread pools	▪ Introduction into thread pools in local execution context, why should we use threadpools for shellcode execution etc. ▪ Building a Win32 loader which supports shellcode-as-words encoded shellcode in PE,	01:00 pm – 02:30 pm

	module stomping, shellcode execution via threadpools etc. ▪ Debugging the loader, loading from module, base address from .text section from the targeted ▪ function inside the module, change of memory protection, shellcode stomping, shellcode execution via threadpools etc.	
	Coffee Break	02:30 pm – 02:45 pm
Hands-on: Loader Finishing	▪ We want to tweak and polish our loaders, learn compilation tips for Visual Studio, how to improve the entropy of your loader, how to apply legit metadata, how to hide the console window, how to apply fake certificates, how to implement EDR specific OPSEC gadgets to further improve the stealthiness of loaders, etc.	02:45 pm – 04:15 pm
	Summary and Q&A for day 4	04:15 pm – 05:15 pm

Bonus Material - Homework

Module	Module Details	Timetable
Import Address Table Hiding	▪ Introduction to the Import Address Table (IAT), how to implement IAT hiding using custom functions, and how to implement IAT hiding in the loaders we build during this course.	
API Hashing	▪ What is API hashing, why is it a useful addition to IAT hiding, how to implement it CRC32 hashing or combine it with IAT hiding.	

The agenda and content of this material are continuously updated and refined to maintain relevance and accuracy. Each iteration of the course may vary slightly depending on factors such as participants experience levels, the volume of questions, and other dynamic elements. These adjustments ensure the most valuable and up-to-date learning experience.