That's the final piece of the core design! Let's expand the **NexusScript** language with features necessary for complex, late-game scripting and integrate the premium functionalities exclusive to the **Architect VIP Tier**. This ensures the language supports advanced cybersecurity concepts like parallel processing, remote tunneling, and specialized exploit development.

7. Advanced NexusScript (Level 15+ & VIP Tier)

These advanced features are unlocked either through high-level progression (Level 15+) or are exclusive quality-of-life additions for the VIP Architect Tier.

A. Core Advanced Functions (High-Level Progression)

These commands are essential for complex missions requiring **Post-Exploitation** and **Pivoting** techniques.

Command/Object	Syntax Example	Core Function Logic	Learning Concept
pivot	pivot $compromised_server $target_subnet	Uses the compromised server as a proxy to launch a **scan** or **ping** command against a previously unreachable internal network ($target_subnet).	**Pivoting**: Bouncing attacks off an internal asset to bypass perimeter defenses.
persist	persist $root_access "backdoor module"	Leaves a small, low-CPU module that attempts to re-gain **root** access if the target system reboots or the connection is lost.	**Maintaining Access**: Setting up command and control (C2) or persistent shells.
cleanup	cleanup $target_server logs	Executes a script to virtually scrub logs and remove traces of the last session, reducing the risk of being discovered by a rival player or in-game AI defense.	**Covering Tracks**: Essential step in the professional pen-testing cycle.
raw	raw send $target_ip $data_packet	Allows the player to bypass the high-level **Service_Object** methods and inject raw data into the virtual network stack. The data must be perfectly formatted.	**Low-Level Programming**: Bypassing application-layer security to exploit weaknesses at the transport layer.

B. Parallel Processing & Resource Control (Hardware Dependent)

These commands directly utilize the upgraded VC hardware (CPU Cores and RAM) for

high-performance scripting.

Command/Object	Syntax Example	Functionality	VC Requirement
thread spawn	set $t1 = thread spawn "scan module" $ip_list	Executes the specified script ("scan module") as a separate, parallel process.	Requires **Multiple CPU Cores**; resource-intensive scripts may fail.
thread sync	thread sync $t1 $t2	Pauses the main script execution until two or more parallel threads have completed their tasks.	Teaches **Concurrency and Synchronization** in multi-threaded programming.
mem alloc	mem alloc 512mb "temp buffer"	Explicitly allocates a block of RAM for large operations (like cracking a massive dictionary). Failure to allocate enough causes a crash.	Teaches **Memory Management**: Critical for avoiding the low-RAM constraint of the F2P tier.

C. VIP Tier Exclusive NexusScript Functions (Architect Tier)

The Architect Tier unlocks automation and complex debugging features that save significant real-world time but do not grant access to unique "win" buttons.

VIP Function	Syntax Example	**Architect VIP Benefit**	F2P Alternative (Time Sink)
auto_deploy	auto_deploy $exploit_module $network_map	Automatically attempts the module against every vulnerable target identified in the map without manual interaction or input.	F2P must use a custom-coded **for loop** module with manual error handling for each individual target.
log.realtime	log.realtime stream $target_ip	Opens a live, streaming view of the target's virtual activity logs and error messages as the attack executes.	F2P must manually use **log.read** after an attack and often deal with **delayed log delivery**.
trace.var	trace.var $exploit_module $pass_var	Sets a silent breakpoint that monitors and prints the value of a specific variable ($pass_var) only when it changes.	F2P must manually insert print statements throughout their code and rerun the module multiple times to debug.
vc.auto_defend	vc.auto_defend on	Automatically executes the player's highest-ranked defense module when an incoming PvP attack is detected while the player is offline.	F2P must rely on their last manually activated defense script or be logged in to initiate defense.

By separating these functionalities, the game maintains its learning focus:

- **F2P Players** learn **efficiency** and **logic** by solving problems with minimal resources and writing the required loops/automation themselves.
- **VIP Players** gain **speed** and **quality-of-life** by automating the tedious parts, allowing them to focus immediately on the high-level challenges like **Zero-Day creation** and complex **multi-threading** logic.