Sprint 1 Requirements Artifacts

Linux Kernel Security Monitor (LKSM)

Team Number:	Group 32
Team Members:	Brett Balquist, Kaden Huber, Jamie King, Dustin Le, Max Biundo, Hart Nurnberg
Sprint:	1 (Weeks 1-2)
Total Story Points:	13

Req ID	Description	Story Points
1	Set up GitHub repository	2
2	Set up development dependencies	3
3	Implement kprobe registration and hook infrastructure	5
4	Create initial architecture document	3

Requirement 1: Set Up GitHub Repository

Description:

Set up GitHub repository with branch protection, CI/CD workflow templates, .gitignore for C/Python, and README with contribution guidelines

Story Points: 2

Priority: 1

Acceptance Criteria:

- GitHub repository is created under team organization or member account with appropriate name (e.g., lksm)
- Repository is accessible to all team members with appropriate permissions (write access)
- Branch protection rules are enabled on 'main' branch requiring pull request reviews
- A .gitignore file exists with appropriate entries for C projects (*.o, *.ko, *.mod.c) and Python (*.pyc, __pycache__, venv/)
- README.md includes: project description, team members, setup instructions placeholder, and contribution guidelines
- GitHub Actions workflow file (.github/workflows/) exists with basic CI template for building the kernel module
- Repository has appropriate directory structure: /src for kernel module, /python for analysis tools, /docs for documentation
- CONTRIBUTING.md file outlines branching strategy and commit message conventions

Deliverables:

- GitHub repository URL
- Screenshot of branch protection rules
- README.md with team info and contribution guidelines

Requirement 2: Set Up Development Dependencies

Description:

Set up development dependencies: VM with kernel headers, GCC, Make, Python venv with required packages, and document installation steps

Story Points: 3

Priority: 2

Acceptance Criteria:

- A virtual machine (Ubuntu 22.04/24.04 LTS) is configured and accessible for development
- Linux kernel headers are installed matching the running kernel (verified by: ls /usr/src/linux-headers-$(uname -r))
- GCC compiler version 11+ is installed (verified by: gcc --version)
- GNU Make is installed and functional (verified by: make --version)
- Python 3.10+ is installed with pip (verified by: python3 --version && pip3 --version)
- Python virtual environment is created in project directory with activation script
- requirements.txt file lists all Python dependencies needed for the project
- INSTALL.md or setup section in README documents all installation steps with exact commands
- A setup script (setup.sh) automates the installation of all dependencies
- All team members can successfully run the setup script and build a test module

Deliverables:

- INSTALL.md with detailed setup instructions
- setup.sh automation script
- requirements.txt for Python dependencies
- Screenshot showing successful dependency installation

Requirement 3: Implement Kprobe Registration and Hook Infrastructure

Description:

Implement kprobe registration and hook infrastructure in kernel module to intercept system calls and capture basic event metadata

Story Points: 5

Priority: 3

Acceptance Criteria:

- Kernel module source file (lksm.c) implements kprobe infrastructure using Linux kprobes API
- Module successfully registers at least one kprobe on a system call (e.g., do_sys_open, sys_execve)
- Pre-handler function captures basic event metadata: timestamp, PID, process name (comm)
- Module compiles without errors or warnings using provided Makefile
- Module loads successfully with 'sudo insmod lksm.ko' and registers probes
- dmesg output shows probe registration success message with target function name
- When hooked system call is triggered, handler executes and logs event to kernel ring buffer (printk)
- Module unloads cleanly with 'sudo rmmod lksm' and unregisters all probes
- No kernel warnings, oops, or panics occur during load/unload/trigger cycle
- Code includes comments explaining kprobe structure and handler logic

Deliverables:

- lksm.c - Kernel module with kprobe implementation
- Makefile - Build configuration
- Screenshot of dmesg showing probe registration and event capture
- Brief documentation of which syscall is hooked and why

Requirement 4: Create Initial Architecture Document

Description:

Create initial architecture document with system diagrams, component descriptions, data flow, and technology stack overview

Story Points: 3

Priority: 4

Acceptance Criteria:

- PDF document includes team number (Group 32) and all team member names
- Project name is present (1-3 words): LKSM
- Project synopsis is present (1-25 words) describing the system purpose
- Document contains at least 3 distinct illustrations/diagrams representing different aspects
- Diagram 1: Layered architecture showing kernel space, communication layer, and user space
- Diagram 2: Component interaction diagram showing data flow between modules
- Diagram 3: Event processing pipeline or technology stack visualization
- Narrative text explains how the software works (1000-1500 words)
- Each diagram serves a clear purpose and is referenced in the narrative
- Document is professionally formatted with consistent styling

Deliverables:

- LKSM_Initial_Architecture_Document.pdf
- Source files for diagrams (optional: Mermaid, draw.io, etc.)