Automatic Privilege Escalation with Deep Reinforcement Learning

DISCLAIMER: Summarized by AI

Problem they are trying to solve / Purpose of method

What are the previous problems that need to be solved?

- Existing attack automation tools (e.g., Metasploit, Cobalt Strike) are script-based and require human configuration tailored to specific environments. These generate predictable patterns, making them easier to detect.
- Machine learning-based defensive systems require large, realistic datasets of attacks for training, but collecting such data is a challenge.
- Current research has focused on automating simple or narrowly defined cyberattack tasks. Automating complex attack stages, like **local privilege escalation**, remains a difficult challenge due to long action sequences, partial observability, and vast action spaces.

Why is the method introduced/needed?

- To demonstrate that **deep reinforcement learning (DRL)** can be used to automate **local privilege escalation**, a critical post-exploitation step.
- To provide a **realistic, intelligent red teaming agent** capable of learning from environment interaction, and to generate high-fidelity training data for defensive systems.
- To explore the potential for malicious use of DRL in cybersecurity, thus helping defenders anticipate future threats.

How does it differ from other methods?

Compare what is different from other methods:

- Prior work (e.g., DeepExploit) focused on **initial access** and treated **lateral movement** simplistically.
- Some DRL agents (like Maeda & Mimura's) addressed **lateral movement**, but with limited state/action complexity.

What makes this method unique?

- First DRL agent to tackle **local privilege escalation** in a realistic Windows 7 environment.
- Uses a **rich, high-dimensional state space** (thousands of variables) and a **high-level action space** (38 atomic, manually designed actions).
- The environment is partially observable, with a large variety of possible system configurations and vulnerabilities (12 types).
- Training is done in a **custom simulator**, but the model transfers successfully to **real-world VMs** without retraining.

- Agent generalizes to **arbitrary numbers of services, DLLs, and tasks**, using neural network modules with shared parameters.

How the method works

Simple overview:

- The authors use a **Deep Reinforcement Learning (A2C - Advantage Actor-Critic)** agent to learn how to perform **local privilege escalation** in a Windows environment.
- The agent interacts with a simulated environment that models Windows 7, including services, tasks, and DLLs, with randomized vulnerabilities in each episode.
- Once trained, the agent is evaluated in a **real VM** to verify generalization and effectiveness.

More details:

- **Reward Function:** Sparse reward; agent gets +1 if privilege escalation is successful, 0 otherwise.
- **Action Space:** 38 high-level actions (e.g., analyze DLLs, reconfigure service, overwrite binaries). Actions are atomic and independent of low-level implementation.
- **State Representation:** Encodes info about services, DLLs, scheduled tasks, autoruns, credentials, and system state using binary/trinary attributes.
- **Neural Network Architecture:** Modular processing for varying numbers of services and components; uses max-pooling to combine component states.
- **Training:** Performed on a fast Python simulator (to avoid VM slowness), using A2C with shared weights for value and policy functions.
- **Testing:** Deployed in real Windows 7 VMs using SSH, successfully escalated privileges across all tested vulnerabilities. Outperformed random and stochastic policies, closely matched expert rule-based policies.