

Grimlock: Guarding High-Agency Systems with eBPF and Attested Channels

Qiancheng Wu*, Wenhui Zhang*, Gan Fang, Sheng Mao, Biao Gao, David Levitsky, Shawna Murphy
Butterworth, Rob Cameron
Roblox

{qwu, wenhuizhang, gfang, smao, bgao, dlevitsky, sbutterworth, rcameron}@roblox.com

Abstract

Agentic systems increasingly run user-authored orchestration code that invokes tools, spawns subtasks, and delegates work across machines and clouds. Although this high agency is productive, it creates a security problem: identity, authorization, provenance, and delegation are often pushed into application code, where they become difficult to enforce consistently and difficult to audit.

We present *Grimlock*, an *Agent Guard* that restores separation of concerns by moving trust enforcement into the sandbox substrate while leaving agent code unchanged. Grimlock uses *eBPF-enforced traffic interception* to ensure that sandbox communication passes through a guard, and combines it with *post-handshake attestation* bound to standard TLS 1.3 channel bindings. After a channel is established, the guard authorizes communication and mints short-lived, channel-bound *scope tokens* that capture least-privilege delegation. At the receiving side, the destination guard re-validates identity, scope, and channel binding, terminates TLS, and releases plaintext to the destination sandbox only after policy checks succeed. kTLS provides an efficient dataplane for protected communication.

As a result, Grimlock offers a path toward transparent, auditable, and scope-bound agent-to-agent communication across heterogeneous multi-cloud environments, using commodity Linux primitives and without requiring changes to user-layer orchestration code.

CCS Concepts: • **Computer systems organization** → *Cloud computing*; • **Security and privacy** → *Network security*; **Systems security**.

Keywords: agent OS, high agency, sandboxing, eBPF, attested TLS, post-handshake attestation, remote attestation, confidential computing, workload identity, provenance, service mesh, multi-cloud

ACM Reference Format:

Qiancheng Wu*, Wenhui Zhang*, Gan Fang, Sheng Mao, Biao Gao, David Levitsky, Shawna Murphy Butterworth, Rob Cameron, Roblox, {qwu, wenhuizhang, gfang, smao, bgao, dlevitsky, sbutterworth, rcameron}@roblox.com . 2026. Grimlock: Guarding High-Agency Systems with eBPF and Attested Channels. In *Proceedings*

AgenticOS '26, Pittsburgh, PA, USA
2026. ACM ISBN

of The 1st Workshop on Operating Systems Design for AI Agents (AgenticOS), co-located with ASPLOS 2026 (AgenticOS '26). ACM, New York, NY, USA, 3 pages.

1 Introduction

Agentic software increasingly acts as an orchestrator: it plans, calls tools, invokes other agents, and composes workflows across machines and clouds [2, 3, 7, 9]. This yields a desirable property we call **high agency**: the user specifies intent without micromanaging infrastructure. But high agency often collapses security layering: application code begins to embed identity and authorization logic, duplicates policy checks, and stitches together ad-hoc credentials, reintroducing classic confused-deputy and privilege-management pitfalls [1, 4]. The result is brittle trust, weak auditability, and poor portability.

We argue an Agentic OS should provide both high agency *and* strict trust boundaries by separating concerns: user code handles orchestration, while a higher-privilege sandbox substrate enforces identity, authentication, authorization, and provenance. **Grimlock** realizes this split through eBPF no-bypass mediation and kTLS-bound post-handshake attestation for authenticated agent-to-agent communication.

Why eBPF. High-agency agents are untrusted and highly dynamic: they may spawn processes, open arbitrary sockets, and call tools that live outside the application address space. Purely library- or runtime-level enforcement is therefore easy to bypass. Grimlock uses eBPF because it enables *OS-enforced, application-transparent* mediation at the sandbox boundary. eBPF can interpose on ingress/egress, associate flows with stable sandbox identity, and *force* all traffic through a controlled path such as redirecting sockets to a guard proxy) without modifying agent code.

Why post-handshake attestation. Attested TLS systems can be categorized by when attestation evidence is generated and bound relative to connection establishment [5]. We adopt post-handshake attestation because it preserves interoperability with commodity TLS 1.3 stacks (no handshake modifications) while still binding evidence to the established channel using exporter-based channel bindings and freshness nonces. This design helps mitigate replay, diversion, and relay attacks. [5, 6, 8].

*These authors contributed equally to this work.

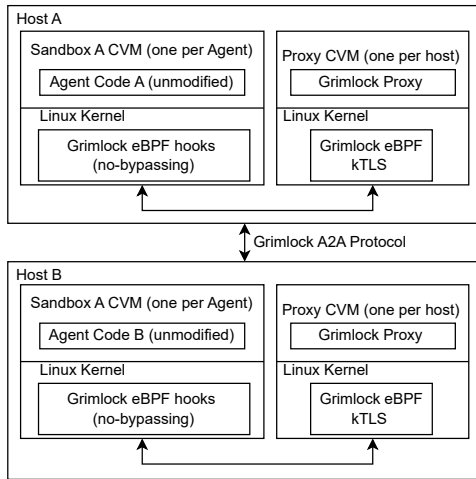


Figure 1. Grimlock deployment. Each host runs a per-agent sandbox Confidential VM (CVM) and a per-host proxy CVM. eBPF enforces no-bypass mediation and steers traffic through the proxy. Proxies communicate via the Grimlock A2A Protocol over TLS 1.3 (kTLS-assisted datapath).

We present **Grimlock**, an Agentic OS guard layer that secures agent communication by enforcing a guarded, no-bypass datapath and channel-bound authorization. Contributions of this paper include:

- **Transparent mediation:** an architecture for transparently intercepting and routing sandbox communication through a guard layer.
- **No-bypass enforcement:** an eBPF-based design for mandatory mediation at the sandbox boundary, preventing agent traffic from evading the guard.
- **kTLS dataplane:** a kTLS-based TLS 1.3 datapath that keeps record processing in kernel space while remaining transparent to agent code.
- **Scope-bound A2A authorization:** a post-handshake, channel-bound attestation and authorization design for secure agent-to-agent communication.

2 Design of Grimlock

Grimlock provides substrate-level security for high-agency agents by enforcing a no-bypass network mediation point at the sandbox boundary and binding authorization to active encrypted channels, shown in Figure. 1.

2.1 Threat Model

We assume an active network adversary capable of eavesdropping, replay, relay/diversion, and man-in-the-middle attacks. We also consider a malicious or buggy agent runtime that may attempt to bypass mediation (e.g., direct sockets, alternate stacks) or escalate scope.

We do *not* trust user-layer code to correctly implement identity or authorization, and we treat the host network and surrounding infrastructure as untrusted.

Grimlock targets three properties: (i) **no-bypass**, all sandbox traffic must traverse the guard; (ii) **channel binding**, authorization artifacts are bound to a specific established channel; and (iii) **least privilege**, delegation propagates auditable, scoped permissions.

2.2 Architecture and Data Path

Grimlock comprises per-agent sandboxes (CVMs) and a per-host guard proxy. Each agent executes inside a confidential, attestable CVM. A CVM is trusted only after remote attestation succeeds under operator policy, covering the boot chain and measured guest software stack (guard and kTLS kernel mechanisms).

Grimlock enforces mandatory mediation at the sandbox boundary. eBPF hooks interpose on all sandbox ingress/egress and redirect all traffic to the local guard proxy CVM, preventing agents from bypassing policy. The proxy establishes a standard TLS 1.3 handshake (using kTLS for efficient record processing) and then runs post-handshake attestation bound to the established channel via exporter-derived channel bindings. After successful appraisal by a verifier/issuer, a short-lived, channel-bound *Scope Token* encoding least-privilege delegation is minted. Traffic is then forwarded to the destination host, where the receiving guard re-validates the attestation result, token scope, audience, expiry, and channel binding; terminates TLS; and releases plaintext to the destination sandbox only after policy checks succeed.

2.3 Grimlock A2A Protocol and Scope Token

Agent A initiates A2A communication by issuing a standard socket connect() from within its sandbox. Grimlock enforces *no-bypass* at the sandbox boundary using eBPF-based mediation. The guard maintains per-flow state (flow metadata → source sandbox identity, source and destination ip, destination port, requested-scope hash, expiry) and coordinates authorization over a long-lived host-to-host control channel. This amortizes setup while allowing connection establishment and guard authentication and authorization to proceed concurrently.

Between guards, Grimlock establishes a standard TLS 1.3 channel. The first contact with a peer performs full mutual authentication and may cache a peer authentication context for reuse on subsequent connections. Once the TLS session is established, kTLS is configured on the guard-to-guard channel so that record encryption and decryption occur in the kernel dataplane without requiring any changes to agent code.

Authorization is enforced as a *post-handshake* gate bound to the established channel. Immediately after the TLS handshake completes, and before any application payload is released, the guards compute a channel binding:

`cb = Exporter(TLS, label, ctx)`

where `ctx` includes a fresh nonce, the intended audience, and the requested delegation scope. The responder returns TEE evidence that commits to $H(cb)$. A verifier appraises this evidence under operator policy and mints a short-lived, channel-bound *Scope Token*. The receiving guard validates the token and channel binding before terminating TLS and releasing plaintext into the destination sandbox.

3 Conclusion

High-agency agentic software demands both flexibility and strong trust boundaries. Grimlock delivers this separation by enforcing no-bypass traffic mediation with eBPF and securing agent-to-agent communication over TLS 1.3, with kTLS providing an efficient protected dataplane.

Acknowledgments

We thank the AgenticOS'26 reviewers and organizers for feedback.

References

- [1] Norm Hardy. 1988. The Confused Deputy: (or why capabilities might have been invented). *ACM SIGOPS Operating Systems Review* 22, 4 (1988), 36–38.
- [2] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. *International Conference on Learning Representations, ICLR*.
- [3] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688* (2023).
- [4] Ravi S. Sandhu and Pierangela Samarati. 1994. Access Control: Principles and Practice. *IEEE Communications Magazine* 32 (1994), 40–48.
- [5] Muhammad Usama Sardar. 2026. Pre-, Intra- and Post-handshake Attestation. Internet-Draft, IETF SEAT WG. Work in Progress: draft-usama-seat-intra-vs-post-02.
- [6] Nick Sullivan. 2022. Exported Authenticators in TLS. RFC 9261. doi:10.17487/RFC9261
- [7] Yanlin Wang, Wanjuan Zhong, Yanxian Huang, Ensheng Shi, Min Yang, Jiachi Chen, Hui Li, Yuchi Ma, Qianxiang Wang, and Zibin Zheng. 2025. Agents in software engineering: Survey, landscape, and vision. *Automated Software Engineering* 32, 2 (2025), 1–36.
- [8] Samuel Whited. 2022. Channel Bindings for TLS 1.3. RFC 9266. doi:10.17487/RFC9266
- [9] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. 2024. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*.