Puffin Secure Browser

Technical White Paper

Part I: Distributed Chromium Architecture for Remote Browser Isolation

CloudMosa, Inc.

February, 2026

Executive Summary

Puffin Secure Browser redefines Remote Browser Isolation (RBI) by transforming Chromium's single-host multi-process architecture into a distributed system. Rather than treating isolation as a remote display problem, CloudMosa extends Chromium at its IPC boundary—preserving its sandbox model and process semantics while relocating untrusted web execution to the cloud.

At the core of this design is **RemoteMojo**, a network-extended implementation of Chromium's Mojo IPC framework. RemoteMojo bridges browser components across machines while maintaining native inter-process contracts. The renderer, network service, and storage execute in the cloud; compositing and GPU rendering remain on the client. No raw DOM or active web code reaches the endpoint.

This approach avoids the compromises of pixel streaming and DOM reconstruction. It preserves native Chromium compatibility, minimizes fork surface, and enables rapid upstream upgrade adoption.

RemoteMojo is not experimental infrastructure. CloudMosa has focused on remote-browser core technology since 2009. Variants of this architecture have powered products across desktop PCs, smartphones, and feature phones, serving more than 200 million cumulative users. Since inception, Puffin Secure Browser has not experienced a publicly documented zero-day exploit resulting in endpoint compromise.

Puffin represents not merely an RBI product, but a mature distributed-browser capability built on deep Chromium expertise—engineered for durability, upgrade velocity, and strategic defensibility.

The RBI Problem: Security Without Usability Collapse

Modern browsers are the primary enterprise attack surface. Remote Browser Isolation mitigates this risk by executing web content remotely. However, most RBI implementations introduce unacceptable tradeoffs.

Pixel Streaming Limitations

Pixel-streaming solutions render remotely and transmit video frames to endpoints. This model:

- Adds encoding/decoding latency (often 80–200ms+ total path)
- Consumes significant bandwidth (multi-Mbps per session)
- Degrades visual clarity under compression
- Struggles with interactive SaaS workflows

DOM Reconstruction Limitations

DOM-rewrite approaches attempt to sanitize and rebuild content client-side. This introduces:

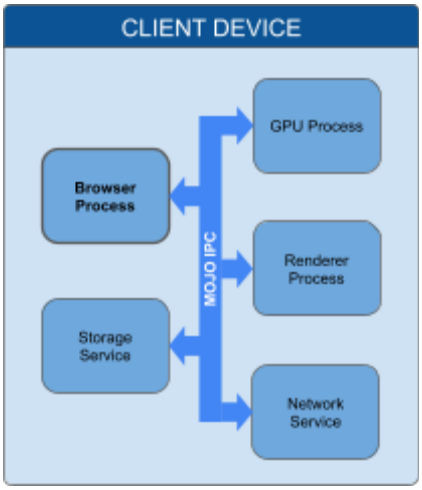
- Compatibility gaps with complex JavaScript applications
- Inconsistent rendering behavior
- Incomplete support for evolving web APIs
- Increased security surface from imperfect sanitization

Both approaches force a tradeoff between security, compatibility, and performance. Puffin eliminates this tradeoff by operating at the architectural level of Chromium itself.

Architectural Innovation:

Extending Chromium Across the Network

Chromium's Native Model



Chromium employs a multi-process security architecture:

- **Browser Process**: policy authority, UI orchestration
- **Renderer Process**: HTML parsing, JavaScript execution, layout
- **GPU Process**: compositing and rasterization
- **Network / Storage Services**: resource handling

These processes communicate through Mojo IPC under strict sandbox boundaries—assumed to operate on a single host with shared memory semantics.

RemoteMojo: Network-Extended IPC

CloudMosa's key innovation is extending Mojo across machines.

RemoteMojo preserves:

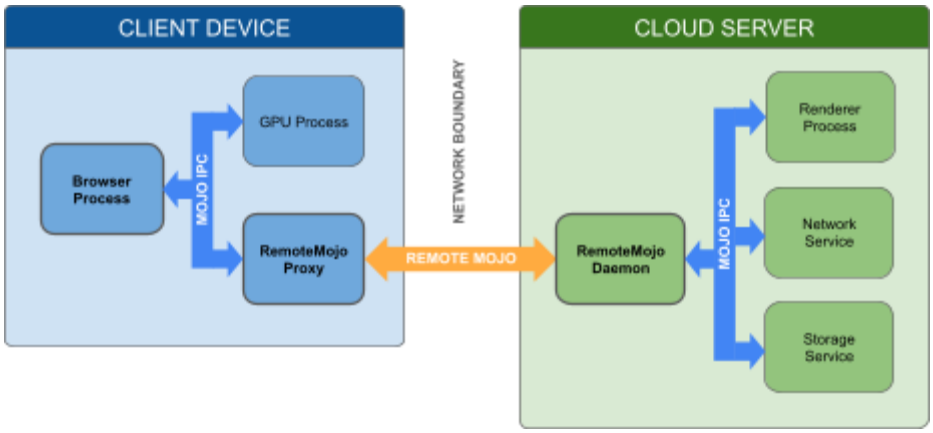
- Message ordering guarantees
- Interface contracts
- Sandbox authority boundaries
- Renderer semantics

It introduces:

- TLS-secured cross-machine transport
- WAN-tolerant message queuing
- Mojo-IPC transparency
- Cross platform interoperability
- Session persistence and reconnection

This mirrors Chromium's native topology while relocating untrusted execution into disposable server-side isolation.

Crucially, Puffin does **not** fork renderer logic, rewrite DOM engines, or reimplement GPU subsystems. The extension occurs at the IPC boundary—minimizing divergence from upstream Chromium.



Client–Server Responsibilities

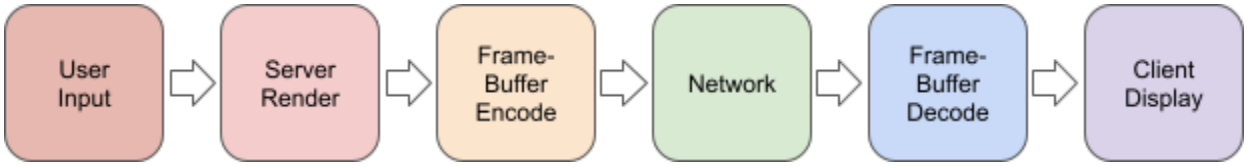
Client-Side (Trusted Rendering – Endpoint)	Server-Side (Untrusted Execution – Cloud)
• Compositing • GPU rasterization • Display presentation • User input capture	• DOM parsing • JavaScript / WebAssembly execution • Layout computation • Network requests • Storage operations

No executable web code or raw DOM is delivered to the endpoint. Only rendering and compositing data necessary to produce pixels locally traverses the network.

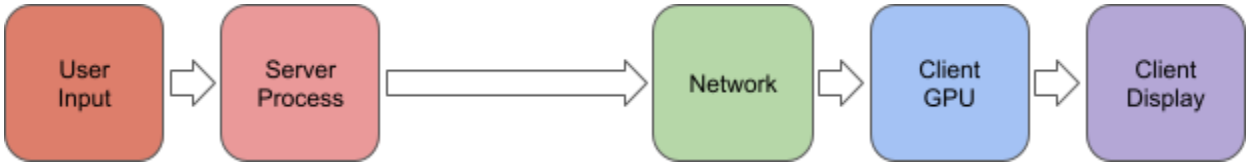
Performance Characteristics

Latency Model

Pixel-streaming pipelines require:



Puffin's model removes encode/decode overhead, transfers less data over network and leverage client-side GPU for rasterization:



Under typical enterprise RTT conditions (20–40ms), latency is largely network-bound rather than codec-bound, enabling a native-feeling interaction loop.

Bandwidth Efficiency

Because Puffin transmits graphics commands and content deltas rather than continuous video frames, bandwidth usage is often lower than pixel streaming for interactive workloads.

Representative internal testing (1080p viewport, mixed workloads):

Content Type	Pixel Streaming	Distributed IPC
Static page	2–5 Mbps continuous	~50–200 KB initial transfer
Interactive SaaS	3–8 Mbps	Command deltas only
Image-heavy content	5–10 Mbps	Compressed image transfer

Actual results vary by workload and policy, but the architectural advantage is structural: commands instead of pixels.

Enterprise-Grade Security

Puffin's security posture derives from architectural isolation rather than content sanitization.

- Zero local execution of web code
- Disposable server-side session containers
- Centralized security policy enforcement
- No direct internet access from endpoints
- Full Chromium sandbox model preserved

The distributed IPC design maintains logical browser authority boundaries even when physically separated across machines.

Since launch, Puffin Secure Browser has not experienced a publicly documented zero-day exploit resulting in endpoint compromise—reflecting the containment properties of cloud-side execution.

Sustainable Engineering Advantage

Chromium releases major updates every four weeks. Security patches and API changes are continuous. RBI systems that heavily modify Chromium internals face compounding merge complexity and delayed patch adoption.

Puffin's IPC-boundary approach minimizes fork surface:

- Renderer and GPU pipelines remain aligned with upstream
- Most upgrades rebase cleanly onto new Chromium releases
- Security hardening is inherited rather than reimplemented

This enables faster security patch adoption and sustained compatibility with rapidly evolving SaaS ecosystems.

For enterprises, this translates to:

- Reduced vulnerability windows
- Fewer browser compatibility exceptions
- Lower long-term operational risk

Proven Technology at Scale

CloudMosa has focused on remote-browser core technology since 2009. The distributed browser model underlying Puffin has been deployed across:

- Desktop PCs
- Smartphones
- Feature phones

Collectively serving over 200 million users.

This operational history demonstrates:

- Real-world WAN tolerance
- Large-scale session orchestration
- Cross-platform rendering adaptability
- Adversarial environment resilience

The architecture is production-proven, not theoretical.

Why This Is Hard

Extending Chromium across a network boundary is fundamentally different from streaming pixels.

Chromium assumes:

- Co-resident processes
- Shared memory primitives
- Local handle passing
- Low-latency IPC

Reinterpreting this model across machines requires solving:

1. **IPC Semantics Preservation**
   Maintaining ordering, interface contracts, and backpressure handling under WAN latency.
2. **Shared Memory Virtualization**
   Translating buffer and handle semantics into secure, network-safe representations.
3. **Cross Platform Interoperability**
   Interconnecting processes in homogeneous runtime environments via a unified network protocol.
4. **Identity and Access Management**
   Authenticating user/device identity and managing access to cloud server resources.
5. **Trust Boundary Integrity**
   Preserving Chromium's sandbox and authority model despite physical distribution.
6. **Upgrade Sustainability**
   Avoiding deep forks that accumulate merge debt across rapid Chromium release cycles.
7. **Failure Recovery**
   Handling reconnection, state continuity, and partial network partition scenarios.

These challenges demand deep knowledge of Chromium internals, Mojo IPC design, renderer/browser contracts, and GPU/compositor pipelines. The expertise required is accumulated over long-term, production-scale iteration.

The intellectual property lies not in streaming techniques, but in safely extending a complex browser architecture beyond its original host assumptions without breaking its security or upgrade velocity.

Strategic Positioning

Puffin Secure Browser represents a rare combination of:

- Architectural elegance
- Proven global deployment
- Strong isolation guarantees
- Rapid upstream alignment
- High replication barrier

It is not simply an RBI solution layered atop a browser. It is a distributed reinterpretation of Chromium's process model—engineered to preserve semantics while relocating risk.

For organizations evaluating strategic browser isolation capabilities, Puffin offers infrastructure-grade browser IP built over more than a decade of focused development.

CloudMosa, Inc.