CrowdMe Security Architecture

Version 1.0

Last updated: July 2026

Executive Summary

CrowdMe is a hosted disposable browser service built around a simple principle: every browsing session starts inside a newly created virtual machine that is destroyed when the session ends.

Instead of attempting to protect a long-lived browser installation, CrowdMe minimizes retained state by treating every browsing session as temporary. Browser cookies, local storage, cached content and other session data exist only for the lifetime of the virtual machine and disappear when the virtual machine is destroyed.

Each session starts from the same browser template, giving every user an identical software environment. Internet access is routed through Tor, while network segmentation and firewall rules prevent Browser VMs from communicating directly with each other or with internal infrastructure.

The architecture is intentionally simple. Every CrowdMe server is a complete, independent installation capable of operating without central session infrastructure. Only the minimum information required to validate session tokens is synchronized between servers. Browsing sessions, browser state and virtual machines are never shared between servers.

CrowdMe is designed to minimize the amount of information that normally exists rather than attempting to prove that information cannot be collected. Destroying each Browser VM removes the browsing environment itself instead of attempting to clean or reset it.

Like every hosted privacy service, CrowdMe ultimately requires trust in the system operator. The purpose of this document is to explain exactly where that trust boundary exists, how the architecture is designed, which security properties it provides, and where its limitations begin.

1. Design Goals

CrowdMe was designed around a small number of architectural principles. Every technical decision described in this document supports one or more of these goals.

Disposable execution environment

Each browsing session should begin from the same known software state and leave no persistent browser environment behind when the session ends. Rather than attempting to clean an existing browser, CrowdMe creates a new virtual machine for every session and destroys it afterwards.

Minimize retained state

The system should retain as little information as practical. Browser history, cookies, local storage, cached content and downloaded malware should exist only for the lifetime of a disposable virtual machine unless the user deliberately exports data.

Isolation by default

System components should communicate only when explicitly required. Disposable browser virtual machines, application services, Tor gateways and management infrastructure are separated into different network segments. Communication between them is controlled by firewall policy rather than by application behaviour.

Transparency

Users should be able to understand how the system works. Wherever practical, the disposable browser environment should be directly inspectable so technically inclined users can verify the software running inside their session instead of relying solely on documentation.

Simplicity

Security should come primarily from straightforward architecture rather than unnecessary complexity. CrowdMe intentionally uses a small number of well-understood components—Proxmox, pfSense, Ubuntu, Firefox and Tor—combined in a way that minimizes persistent state and limits communication between components.

Honest trust model

CrowdMe does not attempt to hide where trust is required. The system is designed to reduce the amount of information that normally exists, but users must still trust the operator of the physical infrastructure. Making this trust boundary explicit is considered part of the security design.

2. Threat Model

Every security architecture is designed with a particular set of threats in mind. CrowdMe is no exception.

This section describes the types of attacks and tracking mechanisms the architecture is intended to reduce, those it can only partially reduce, and those that remain outside its design goals.

CrowdMe should not be viewed as a complete anonymity system or as a replacement for safe browsing practices. Instead, it aims to reduce the amount of persistent information created during ordinary web browsing by combining disposable virtual machines, network isolation and Tor routing.

Threat Overview

Threat	CrowdMe's Objective	Comments
Cookies and browser storage	**Designed to reduce**	Browser state is destroyed together with the disposable VM.
Browser cache and browsing history	**Designed to reduce**	No browser environment is reused between sessions.
Cross-session browser fingerprinting	**Designed to reduce**	Every session starts from the same browser template and configuration.
Malware persisting inside the browser VM	**Designed to reduce**	Malware executing only inside the disposable VM disappears when the VM is destroyed.
Compromised websites exploiting the browser	**Partially reduces**	Browser exploitation remains possible, but the compromised environment is isolated and temporary.
Browser VM attacking neighbouring browser VMs	**Designed to reduce**	Network segmentation and firewall rules prevent direct VM-to-VM communication.
Observation by local networks (public Wi-Fi, hotels, airports)	**Partially reduces**	Browsing occurs on the remote disposable VM rather than on the user's own device.
Internet Service Provider (ISP) observing browsing destinations	**Designed to reduce**	Browser traffic exits through the Tor network rather than directly from the user's connection.
Websites identifying logged-in users	**Not designed to protect**	Logging into accounts intentionally identifies the user to those services.
Websites identifying users through information they voluntarily provide	**Not designed to protect**	CrowdMe cannot prevent users from identifying themselves.
Observation by the CrowdMe operator	**Not designed to prevent**	The system operator remains part of the trust model.
Physical compromise of the server infrastructure	**Not designed to prevent**	Anyone controlling the physical infrastructure ultimately controls the system.

Design Philosophy

CrowdMe focuses on reducing **persistent state** rather than attempting to make every browsing session anonymous.

Traditional browsers gradually accumulate information such as cookies, cached files, browser history, local storage and installed browser state. That information can contribute to long-term tracking or survive browser vulnerabilities.

CrowdMe takes a different approach. Rather than cleaning an existing browser, it creates a completely new browser environment for every session and destroys that environment afterwards. The goal is to reduce the amount of information that exists in the first place.

This approach does not eliminate all forms of tracking. Websites may still identify users through logins, browser behaviour, information entered into forms or other techniques that operate during an active session. CrowdMe therefore should be understood as reducing specific classes of persistent information rather than providing complete anonymity.

Threats Outside the Design Scope

Certain threats are intentionally outside the scope of the CrowdMe architecture.

These include:

- Users voluntarily identifying themselves by logging into online accounts.
- Users revealing personal information through websites or online forms.
- Malicious or compromised websites collecting information during an active browsing session.
- A malicious or compromised CrowdMe system administrator.
- Physical compromise of the infrastructure hosting CrowdMe.

These threats require either user judgement or trust in the service operator. They cannot be eliminated solely through the disposable virtual machine architecture.

Why this matters

The purpose of CrowdMe is not to claim protection against every possible threat.

Instead, the architecture focuses on reducing the amount of persistent information created during ordinary browsing while making the remaining trust assumptions explicit.

The following chapters describe how the architecture implements these design goals.

3. System Overview

CrowdMe is built as a collection of independent servers. Each server contains a complete CrowdMe installation capable of operating without relying on central infrastructure.

A CrowdMe server is designed to be self-contained. Adding additional servers increases capacity without changing the overall architecture. Users can be distributed between servers using DNS round-robin or another load balancing method. Each server manages its own browser sessions independently.

The only information synchronized between servers is the minimum information required to validate session tokens:

- Token hash
- Remaining session count
- Last update timestamp

No browsing sessions, browser state, cookies, IP addresses or browsing activity are synchronized between servers.

This minimizes the amount of shared information while allowing tokens to be used on any CrowdMe server.

Server Architecture

Each CrowdMe server consists of a single physical computer running Proxmox Virtual Environment (Proxmox VE).

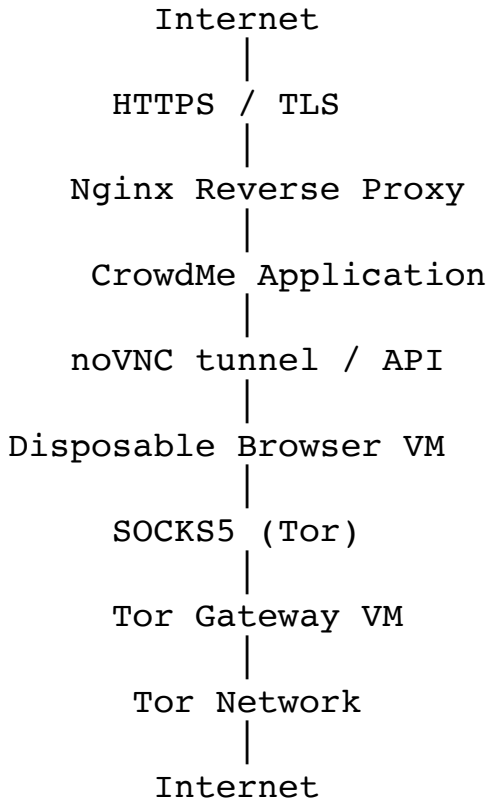
The physical server hosts several virtual machines, each responsible for a single purpose.

Component	Purpose
Proxmox VE	Virtualization platform hosting all virtual machines.
pfSense	Central firewall and router controlling all communication between internal network segments.
CrowdMe Application	Creates, manages and destroys disposable browser virtual machines, validates tokens and proxies user connections.
Disposable Browser VMs	Temporary Ubuntu desktop environments used for browsing sessions.
Tor Gateway VM	Provides the only permitted path from browser virtual machines to the Internet.
Nginx Reverse Proxy	Terminates public HTTPS connections and forwards approved requests to the CrowdMe application.

Each component performs one well-defined function. No component performs multiple unrelated tasks.

This separation reduces complexity and allows firewall rules to explicitly define which components may communicate.

High-Level Architecture



pfSense sits between every internal network segment and enforces all communication. No virtual machine can communicate directly with another network unless an explicit firewall rule permits it.

Independent Server Design

Each CrowdMe server operates independently.

There is:

- no central session manager,
- no central browser infrastructure,
- no shared browser storage,
- no shared virtual machines.

A server can continue operating even if communication with other CrowdMe servers is interrupted. Token synchronization is asynchronous and limited to token state; active browsing sessions remain entirely local to the server handling them.

This architecture limits both complexity and the amount of information shared between installations.

Why This Architecture?

The architecture is designed around three principles:

Independence

Each server should be capable of operating on its own. This simplifies deployment, improves resilience, and reduces dependencies on central services.

Data Minimization

Only the minimum information required for token validation is shared between servers. Browser sessions remain local to the server that created them and are never replicated elsewhere.

Separation of Responsibility

Each component has a clearly defined role:

- Proxmox isolates virtual machines.
- pfSense enforces network policy.
- The CrowdMe application manages sessions.
- Disposable browser VMs execute user workloads.
- Tor provides Internet connectivity.
- Nginx exposes the public web service.

This separation makes the architecture easier to understand, audit and maintain.

Looking Ahead

The following chapters describe each part of the architecture in more detail. They explain how browsing sessions are created, how virtual machines are isolated, how network communication is controlled, and how browser sessions are removed when they end.

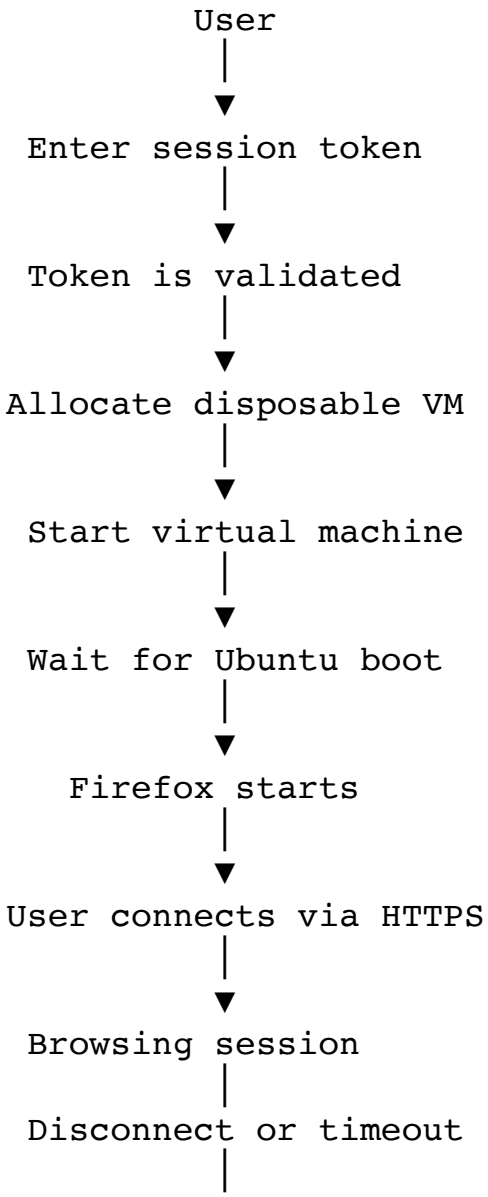
4. Session Lifecycle

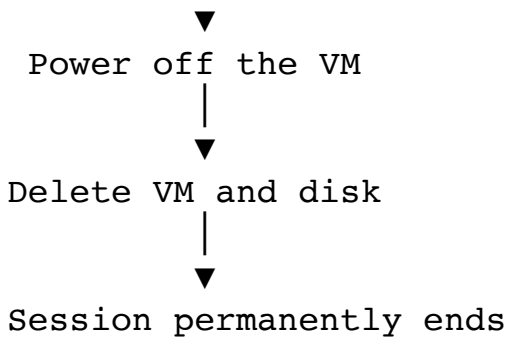
Every CrowdMe browsing session is executed inside a newly created disposable virtual machine (VM). The virtual machine exists only for the lifetime of a single browsing session and is permanently deleted afterwards.

Unlike traditional desktop environments or hosted virtual desktops, CrowdMe does not assign users a persistent machine. Every session starts from the same clean template, and no browser VM is ever reused by another user.

This lifecycle is the foundation of the CrowdMe architecture and is responsible for many of its security properties.

Session Overview





Step 1 – Session Request

A browsing session begins when a user enters a valid CrowdMe session token.

The CrowdMe application validates the token locally. If multiple CrowdMe servers are deployed, token state has already been synchronized between servers. No browser state or user information is synchronized—only the token hash, remaining session count and update timestamp.

Once the token has been validated, one session is reserved and a disposable browser VM is allocated.

Step 2 – Virtual Machine Creation

CrowdMe creates a new browser VM by cloning a read-only template stored on the local Proxmox server.

The template contains:

- Ubuntu Desktop
- Firefox
- noVNC server
- CrowdMe configuration
- Browser hardening and standard settings

The template itself is never modified during a browsing session.

Each browsing session receives its own independent copy.

No previously used browser VM is ever assigned to another user.

Step 3 – System Startup

The newly created VM is started and boots Ubuntu normally.

During startup:

- Ubuntu initializes the desktop environment.
- Firefox starts automatically.

- The noVNC service becomes available.
- The VM receives network connectivity according to the firewall policy.

The CrowdMe application waits until the browser environment is fully operational before allowing the user to connect.

This ensures that every user begins with a fully initialized browser session rather than observing the operating system boot process.

Step 4 – Active Browsing Session

Once the browser VM is ready, the user connects through CrowdMe using HTTPS.

The browser window displayed in the user's web browser is a live view of Firefox executing inside the disposable VM.

Keyboard and mouse events are forwarded to the VM through the CrowdMe application.

During the session:

- Browser cookies are stored only inside the disposable VM.
- Browser cache exists only inside the disposable VM.
- Local browser storage exists only inside the disposable VM.
- Downloads remain inside the disposable VM unless explicitly transferred elsewhere by the user.

The disposable VM has no knowledge of previous sessions and will never be reused after the current session ends.

Step 5 – Session Termination

A session ends when:

- the user closes the session,
- the user disconnects,
- the maximum session time expires, or
- the system terminates the session for operational reasons.

Ending a session immediately begins the cleanup process.

No browser VM remains available for later reuse.

Step 6 – Virtual Machine Destruction

After the session ends, CrowdMe automatically destroys the disposable VM.

The cleanup process includes:

1. Power off the virtual machine.

2. Remove the virtual machine configuration.
3. Delete the associated virtual disk.
4. Release allocated resources.

Once this process completes, the browser environment no longer exists.

The next user receives a newly created VM cloned from the original template rather than the VM that has just been destroyed.

Why Destroy the Entire Virtual Machine?

Destroying the entire virtual machine is a deliberate architectural decision.

Many browser privacy features attempt to remove cookies, clear browser history or reset browser state after use.

CrowdMe instead removes the entire execution environment.

This approach eliminates:

- browser cookies,
- browser history,
- cached web content,
- local browser storage,
- temporary files,
- browser extensions installed during the session,
- malware executing only within the disposable VM,
- configuration changes made during the session.

Rather than cleaning a browser, CrowdMe discards the computer itself.

Normal Operation

Under normal operation, every browsing session follows the same lifecycle:

- A new VM is created.
- The user browses inside that VM.
- The VM is destroyed when the session ends.

No browsing environment is retained as part of the normal operating procedure.

Retaining a browsing session would require intentionally changing the behaviour of the CrowdMe application or the underlying infrastructure. Such changes are discussed later in this document as part of the trust model.

Design Considerations

The disposable VM lifecycle provides several important security properties:

- Every browsing session begins from the same known software state.
- No browser environment is shared between users.
- Compromise of one browser VM does not affect future sessions.
- Browser state is removed by destroying the VM rather than attempting to clean it.
- Persistent information is minimized by design.

The lifecycle described above is the foundation upon which the remainder of the CrowdMe security architecture is built.

5. Network Architecture

CrowdMe uses network segmentation as a containment mechanism. The purpose is not only to organize traffic, but to ensure that compromise of one component does not automatically provide access to other components.

The architecture is built around a simple rule:

No internal component should be able to communicate with another internal component unless that communication is explicitly required and explicitly allowed.

This is enforced through pfSense, VLAN separation and host-level firewall rules.

Network Segments

Each CrowdMe server uses separate internal network segments for different functions.

Network Segment	Purpose
Public / external network	Receives incoming HTTPS traffic from users.
Reverse proxy network	Handles public web entry and forwards allowed requests inward.
Application network	Runs the CrowdMe application and session orchestration.
Browser VM network	Runs disposable browser virtual machines.
Tor gateway network	Runs the Tor gateway used by browser VMs for Internet access.
Management network	Used for administrative access to Proxmox and infrastructure.

The exact VLAN IDs and IP addresses are implementation details and may change over time. The important security property is that these functions are separated into different network segments and that traffic between them is controlled.

Why pfSense Exists

pfSense acts as the central enforcement point for internal communication.

Rather than allowing virtual machines to communicate freely on a flat network, internal components are placed behind pfSense. Any traffic between the application network, browser VM network, Tor gateway network and management network must pass through pfSense and match an explicit firewall rule.

This provides a clear security boundary:

- browser VMs cannot directly reach the application network,
- browser VMs cannot directly reach the management network,
- browser VMs cannot directly reach each other,
- the Tor gateway cannot freely initiate connections back into browser VMs,

- internal services are not exposed directly to the public Internet.

pfSense is therefore not only a router. It is the policy decision point for the internal network.

Firewall Philosophy

CrowdMe follows a default-deny model.

Traffic is blocked unless it is required for the service to function.

This is intentionally different from a flat internal network where systems can usually reach each other by default. In CrowdMe, each permitted connection should have a clear operational reason.

The firewall policy is based on four principles:

1. **Deny by default**
   Communication is blocked unless explicitly allowed.
2. **Allow only necessary paths**
   A browser VM should only reach what it needs to browse through Tor and maintain basic system operation.
3. **Separate user execution from infrastructure**
   Disposable VMs run user-controlled web content. They should not be able to reach Proxmox, pfSense administration, token storage, application internals or other users' VMs.
4. **Assume browser compromise is possible**
   The browser VM network is designed with the assumption that a website may exploit Firefox or another component inside the VM. Network policy should still limit what the compromised VM can reach.

Browser VM Network

The browser VM network exists solely for disposable browsing sessions.

Browser VMs are treated as untrusted execution environments because they run websites chosen by users. They may load arbitrary JavaScript, media, downloads and browser-exposed content from the Internet.

For that reason, browser VMs are not trusted simply because they are part of the internal infrastructure.

A browser VM should not be able to access:

- other browser VMs,
- the CrowdMe application network,
- Proxmox,
- pfSense administration,
- token databases,
- management services,
- internal host services.

The only normal outbound communication required from a browser VM is access to the Tor gateway and basic time synchronization.

Isolation Between Browser VMs

Disposable browser VMs share the same browser VM network segment, but they are not allowed to communicate with each other.

This is important because a browser VM may become compromised during a session. If VM-to-VM communication were allowed, one compromised browser VM could scan or attack neighbouring browser VMs.

CrowdMe blocks this with host-level firewall policy on Proxmox. The policy drops outbound traffic from browser VMs to the browser VM subnet itself.

Conceptually:

```
Browser VM → Browser VM network: denied
```

This means browser VMs may exist on the same network segment without being able to directly reach one another.

Allowed Browser VM Communication

The intended browser VM communication model is very small.

Source	Destination	Purpose	Allowed
Browser VM	Tor SOCKS proxy	Internet access through Tor	Yes
Browser VM	pfSense NTP service	Time synchronization	Yes
Browser VM	Other browser VMs	Not required	No
Browser VM	CrowdMe application	Not required from inside VM	No
Browser VM	Proxmox / management	Never required	No
Browser VM	Direct Internet	Must use Tor gateway	No

This table is one of the most important parts of the network design. It shows that browser VMs are not general-purpose internal hosts. They are disposable, restricted execution environments.

Application Network

The CrowdMe application runs on a separate application network.

Its role is to:

- validate tokens,
- allocate sessions,
- create and destroy browser VMs through the Proxmox API,
- track session state,
- provide the user-facing web interface,
- tunnel noVNC connections to the assigned browser VM.

The application needs controlled access to infrastructure APIs and selected browser VM services. Browser VMs do not need reciprocal access back into the application network.

This asymmetry is intentional. The application manages browser VMs, but browser VMs should not be able to manage or inspect the application.

Tor Gateway Network

The Tor gateway runs on a separate network segment.

Browser VMs are allowed to reach the Tor gateway's SOCKS proxy. The Tor gateway then connects outward through the Tor network.

The Tor gateway should not be treated as a trusted bridge back into browser VMs. Its role is outbound Internet access, not internal communication.

Conceptually:

```
Browser VM → Tor SOCKS proxy → Tor network → Internet
```

Direct browser VM Internet access is not part of the design.

Reverse Proxy and Public Access

Public users connect to CrowdMe over HTTPS.

Public traffic reaches a reverse proxy first. The reverse proxy forwards allowed web requests to the CrowdMe application.

Disposable browser VMs are not directly exposed to the public Internet. Users do not connect directly to internal VM addresses. Instead, the noVNC session is tunneled through the CrowdMe application.

This keeps browser VM addressing internal and prevents browser VMs from becoming public-facing servers.

Management Network

Administrative access belongs on a separate management network.

This network is used for operating the infrastructure, including Proxmox and pfSense administration.

The browser VM network must not be able to reach the management network. A compromised browser VM should not be able to discover, scan or attack the administrative plane.

Management access remains part of the trust boundary described later in this document.

Summary

The network architecture exists to contain failure.

CrowdMe assumes that browser VMs may execute hostile content. The network is therefore designed so that a compromised browser VM has very little it can reach:

- not other browser VMs,
- not the application network,
- not Proxmox,
- not pfSense administration,
- not direct Internet access,
- only the Tor gateway and required infrastructure services.

This is the core security purpose of the VLAN and firewall design.

6. Browser VM Architecture

The Browser VM is the core execution environment of CrowdMe. Every browsing session executes inside its own disposable virtual machine running a standard Ubuntu desktop and Firefox browser.

Unlike traditional remote desktops, hosted virtual machines or browser profile isolation, Browser VMs are temporary. A Browser VM exists only for the lifetime of a single browsing session and is permanently destroyed afterwards.

The Browser VM is intentionally treated as an untrusted execution environment. It is expected to execute arbitrary websites, JavaScript, downloads and browser content selected by the user. The surrounding infrastructure is designed with the assumption that a Browser VM may become compromised during its lifetime.

Why Virtual Machines?

CrowdMe uses virtual machines because they provide a well-defined isolation boundary between the browsing environment and the underlying infrastructure.

Other approaches, such as browser profiles, incognito mode or browser containers, isolate browser state but continue using the same underlying operating system. Any compromise of that environment may survive between browsing sessions.

CrowdMe instead creates an entirely new operating system for every session.

This means each browsing session begins with:

- a fresh operating system,
- a fresh browser profile,
- a fresh filesystem,
- a new virtual disk,
- a new runtime environment.

The Browser VM is not reset after use—it is destroyed.

Identical Browser Templates

Every Browser VM is cloned from the same immutable template.

The template contains:

- Ubuntu Desktop
- Firefox
- Standard browser configuration
- CrowdMe software
- System configuration required for operation

The template is maintained by the CrowdMe operator and is identical for every newly created Browser VM.

Because every Browser VM starts from the same template, users begin with the same browser version, operating system version, installed software and browser configuration.

This consistency is a deliberate design goal.

Browser Fingerprinting

Modern websites often attempt to distinguish browsers using a combination of software and hardware characteristics. This process is commonly known as browser fingerprinting.

Examples include:

- browser version,
- operating system,
- installed fonts,
- screen resolution,
- graphics capabilities,
- language settings,
- browser configuration,
- available APIs.

CrowdMe does not attempt to randomize these values.

Instead, every Browser VM begins from the same template with the same software configuration. The objective is not to make each user appear unique, but to make every Browser VM appear as similar as practical.

Rather than generating a new fingerprint for every session, CrowdMe attempts to reduce variation by providing a common execution environment.

Fingerprinting remains an active area of browser research, and no system can guarantee that two browser instances are indistinguishable. Websites may still derive identifying characteristics from browser behaviour, timing, user actions or other techniques outside the control of CrowdMe.

The Browser VM architecture should therefore be understood as reducing differences between sessions rather than eliminating browser fingerprinting entirely.

Disposable Browser State

During an active session, all browser data exists only inside the Browser VM.

This includes:

- cookies,
- browser cache,
- browsing history,
- local storage,

- IndexedDB,
- session storage,
- downloaded temporary files,
- browser configuration changes made during the session.

When the Browser VM is destroyed, this browser state is destroyed with it.

The following session receives a newly created Browser VM rather than a cleaned version of the previous one.

Malware and Browser Exploits

The Browser VM executes untrusted content from the Internet. It is therefore possible that a malicious website exploits a browser vulnerability or causes malicious software to execute inside the Browser VM.

CrowdMe does not assume that browser software is free from vulnerabilities.

Instead, the architecture attempts to limit the consequences of such an event.

If malware executes only within the Browser VM, its lifetime is limited to the lifetime of that virtual machine. When the session ends, the Browser VM is destroyed together with its virtual disk and runtime state.

This prevents malware from persisting into future browsing sessions through the normal operation of the system.

It does **not** guarantee that a browser exploit cannot affect the underlying infrastructure. Protection of the host system depends on the security of the virtualization platform, the operating system, firewall rules and other infrastructure components discussed elsewhere in this document.

Isolation Between Sessions

Browser VMs are never reassigned to another user.

Each session receives a newly created Browser VM cloned from the common template.

As a result:

- browser cookies are not inherited,
- browser history is not inherited,
- browser extensions installed during one session do not appear in later sessions,
- malware executing only inside one Browser VM does not normally survive into another session,
- browser configuration changes disappear when the Browser VM is destroyed.

This eliminates an entire class of problems associated with persistent browser environments.

Browser VM as an Untrusted Component

From the perspective of the CrowdMe infrastructure, Browser VMs are intentionally treated as untrusted.

They execute arbitrary Internet content chosen by users and therefore operate under the assumption that they may become compromised.

For this reason Browser VMs:

- execute on their own isolated network segment,
- cannot communicate directly with other Browser VMs,
- cannot access the management network,
- cannot access the CrowdMe application directly,
- cannot bypass the Tor gateway for Internet access.

The Browser VM should therefore be viewed as a contained execution environment rather than a trusted part of the infrastructure.

Design Summary

The Browser VM architecture is based on four principles:

- Every browsing session receives a newly created virtual machine.
- Every Browser VM starts from the same immutable template.
- Browser state exists only for the lifetime of the session.
- Browser VMs are treated as potentially compromised and are isolated from the rest of the infrastructure.

7. Internet Access

CrowdMe separates the user's connection to CrowdMe from the Browser VM's connection to the public Internet.

The user connects to CrowdMe over HTTPS in order to view and control a disposable Browser VM. The Browser VM itself reaches websites through a Tor Gateway. These are two separate network paths with different purposes.

This separation is important:

- the user does not connect directly to the Browser VM,
- the Browser VM is not exposed as a public Internet service,
- websites are reached through Tor rather than directly from the Browser VM network,
- the internal VM address remains private to the CrowdMe infrastructure.

User Access Path

When a user opens a CrowdMe session, the browser window they see is a remote view of Firefox running inside the disposable Browser VM.

The user access path is:

```
User Browser
    ↓ HTTPS
Nginx Reverse Proxy
    ↓
CrowdMe Application
    ↓ noVNC tunnel
Disposable Browser VM
```

The user's local browser does not load websites directly through CrowdMe. It only displays and controls the remote Browser VM session.

The websites visited during the session are loaded by Firefox inside the Browser VM.

Nginx Reverse Proxy

Public HTTPS traffic first reaches a reverse proxy. The current implementation uses Nginx.

The reverse proxy is responsible for public web entry. Its role is to:

- terminate HTTPS connections,
- expose the CrowdMe website,
- forward allowed requests to the CrowdMe Application,
- avoid exposing internal VM addresses directly to users.

Nginx is not the privacy boundary by itself. Its role is controlled public access. The security boundary is created by the combination of the reverse proxy, the CrowdMe Application, pfSense, VLAN separation and VM isolation.

The document uses the term "reverse proxy" for the architectural role, while the current implementation uses Nginx.

noVNC Tunnel

CrowdMe uses noVNC to allow the user to view and control the Browser VM through a web browser.

The noVNC connection is not exposed directly from the Browser VM to the public Internet.

Instead, it is tunneled through the CrowdMe Application.

Conceptually:

```
User → HTTPS → Reverse Proxy → CrowdMe Application → noVNC → Browser VM
```

This means the user connects to CrowdMe.One, not to the internal VM address.

The Browser VM remains on an internal network segment. Its noVNC service is reachable only through the controlled application path.

This avoids making each disposable VM a public-facing host.

Browser VM Internet Path

The Browser VM has a separate outbound path for browsing the public Internet.

That path is:

```
Disposable Browser VM
    ↓ SOCKS5
Tor Gateway
    ↓
Tor Network
    ↓
Public Internet
```

Firefox inside the Browser VM is configured to use the Tor Gateway.

Under normal firewall policy, the Browser VM cannot bypass the Tor Gateway and connect directly to the Internet.

This means websites visited from within the Browser VM see traffic coming from the Tor network rather than directly from the CrowdMe Browser VM network.

Why Tor Is Separate

Tor runs in a separate gateway environment rather than inside each Browser VM.

This separation has several benefits:

- Browser VMs do not need direct Internet access.
- Tor policy can be controlled centrally on the Tor Gateway.
- Browser VM network rules can remain simple.
- The Tor Gateway can be isolated from both the application network and the Browser VM network.
- Browser VMs can be destroyed without affecting the Tor Gateway.

The Tor Gateway is a network component. The Browser VM is an execution component. Keeping them separate makes the architecture easier to reason about.

Why Browser VMs Do Not Accept Internet Connections

Disposable Browser VMs are not designed to be Internet-facing servers.

They are short-lived environments that execute browser sessions. Exposing them directly to the Internet would increase the attack surface and make each VM individually reachable from outside the CrowdMe infrastructure.

CrowdMe avoids this by ensuring that Browser VMs:

- do not have public IP addresses,
- are not exposed through public port forwarding,
- do not accept inbound connections from the Internet,
- are controlled through the CrowdMe Application,
- reach websites only through the Tor Gateway.

This reduces the role of each Browser VM to one purpose: execute a temporary browsing session and then disappear.

Separation of Control and Browsing

There are two different flows in the system:

Flow	Purpose	Path
Control flow	User views and controls the Browser VM	User → Reverse Proxy → CrowdMe Application → Browser VM
Browsing flow	Browser VM accesses websites	Browser VM → Tor Gateway → Tor Network → Internet

These flows should not be confused.

The user's local computer controls the session. The disposable Browser VM performs the browsing.

The control flow is inbound to CrowdMe. The browsing flow is outbound from the Browser VM through Tor.

What Websites See

Websites visited during a CrowdMe session interact with Firefox running inside the Browser VM.

They do not interact with the user's local browser directly.

Websites may see:

- the Tor exit node,
- the Browser VM's Firefox configuration,
- the Browser VM's browser fingerprint,
- information typed or submitted by the user during the session.

Websites should not see:

- the user's local browser cookies,
- the user's local browser history,
- the user's local IP address directly through the browsing connection,
- files stored on the user's local computer unless explicitly uploaded through the session.

This does not make website identification impossible. If a user logs in, enters identifying information or behaves in a recognizable way, websites may still identify that user.

Operational Summary

CrowdMe's Internet access model is intentionally narrow.

The public Internet can reach:

```
Reverse Proxy → CrowdMe Application
```

The Browser VM can reach:

```
Tor Gateway → Tor Network → Internet
```

The public Internet cannot directly reach:

```
Disposable Browser VM
```

The Browser VM cannot directly reach:

```
Public Internet
```

This separation reduces attack surface and keeps disposable Browser VMs internal, temporary and controlled.

8. Transparency and Inspectability

CrowdMe exposes more of the user's execution environment than many hosted privacy services.

In many services, the user only sees a web interface and must trust that the underlying servers behave as described. In CrowdMe, the user is given access to a complete Ubuntu desktop running inside the disposable Browser VM.

This means technically inclined users can inspect the environment they are using during a live session.

Inspectability is not a replacement for trust, but it is useful transparency.

What Users Can Inspect

A user can minimize Firefox, open system tools or a terminal inside the Browser VM, and inspect the running environment from within the session.

Examples include:

Area	Examples
Operating system	Ubuntu version, kernel version, installed packages
Browser	Firefox version, settings, extensions, profile state
Processes	Running processes, system services, startup behaviour
Network	IP configuration, routing table, DNS settings, active connections
Certificates	Installed certificate authorities and local certificate store
Filesystem	Home directory, temporary files, downloaded files
System services	systemd services, noVNC/websockify processes, desktop autostart entries
Local logs	Logs available inside the disposable VM during the session

This allows a technical user to verify that the Browser VM looks like a normal Ubuntu environment and that the visible browser configuration matches the documented design.

What Users Cannot Inspect

A user only has access to the disposable Browser VM assigned to their session.

They cannot inspect components outside that VM.

Component	User visibility
Proxmox host	Not visible from inside the Browser VM

pfSense firewall	Not visible from inside the Browser VM
CrowdMe Application	Not visible from inside the Browser VM
Tor Gateway	Not visible except through observed network behaviour
Reverse proxy	Not visible except as the public web entry point
VM template	Not directly visible, only the cloned session VM
Other users' VMs	Not visible and not reachable by design
Token database	Not visible
Physical server	Not visible

These components are outside the Browser VM and remain part of the CrowdMe trust boundary.

Why Inspectability Matters

Inspectability gives users a practical way to evaluate part of the system themselves.

A user can check, for example:

- whether Firefox is configured as expected,
- whether unusual processes are running inside the VM,
- whether the VM has direct routes to the Internet,
- whether the filesystem appears fresh,
- whether unexpected certificates are installed,
- whether browser state exists from previous sessions.

This does not prove that the infrastructure is honest, but it reduces the amount of the system that must be accepted blindly.

A VPN user cannot usually inspect the VPN provider's server while using it. A CrowdMe user can inspect the disposable computer where their browser is actually running.

Limits of Inspectability

Inspectability has clear limits.

A user inspecting the Browser VM cannot prove that:

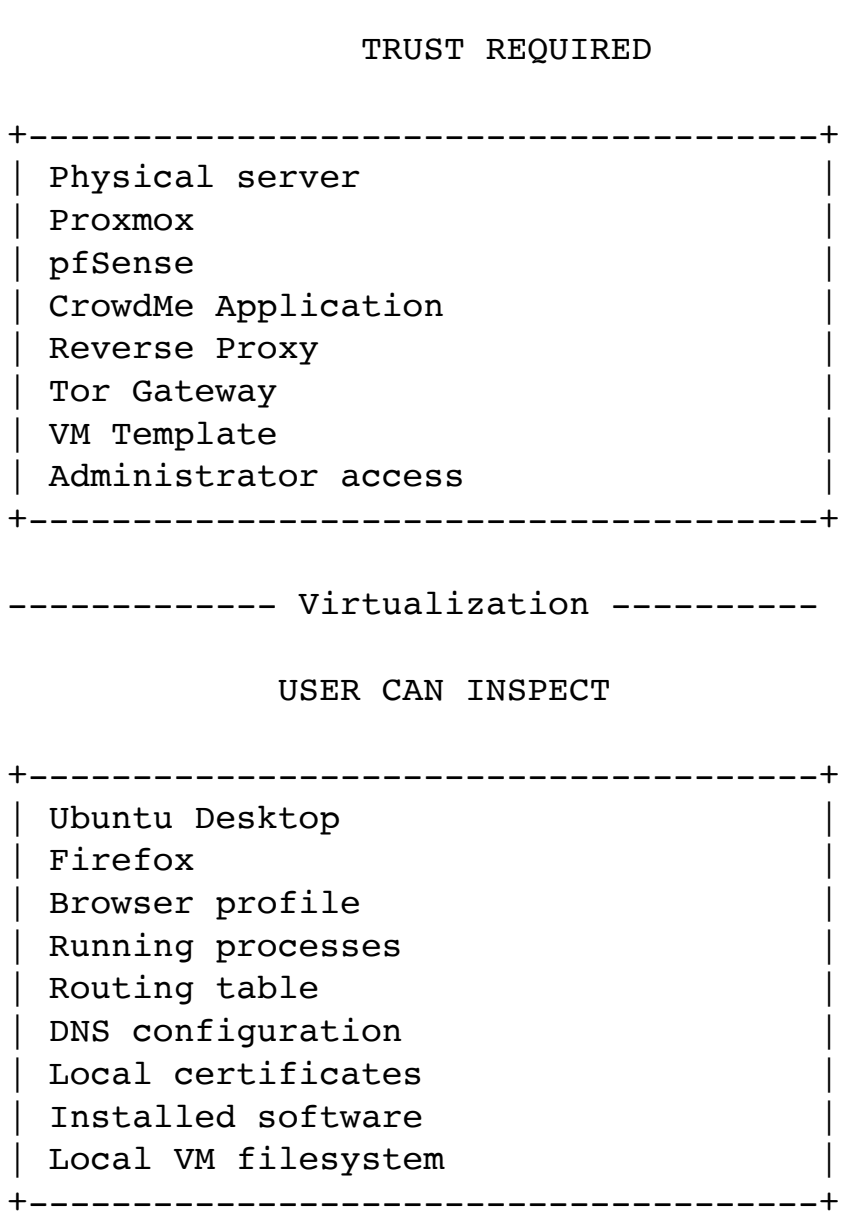
- Proxmox is configured exactly as documented,
- pfSense rules match the published design,
- the VM template has not been modified,
- the operator is not viewing the Proxmox console,
- the operator is not snapshotting or copying VM state,
- the CrowdMe Application will delete the VM after the session,
- no monitoring exists outside the Browser VM.

The user can inspect the disposable VM from the inside. They cannot inspect the infrastructure controlling it from the outside.

This distinction is important.

Trust Boundary

The trust boundary sits between the disposable Browser VM and the infrastructure that creates, connects, controls and destroys it.



The lower part can be inspected by the user during a session.

The upper part must be trusted.

Practical Value

The practical value of inspectability is not that it eliminates trust. It does not.

Its value is that it makes part of the system observable.

A technical user can verify that the visible Browser VM is consistent with the public description of CrowdMe. If the VM does not look as described, that is itself useful information.

This creates a different transparency model from services where the user's execution environment is completely hidden.

Summary

CrowdMe's inspectability model is deliberately limited but still meaningful.

Users can inspect the disposable VM they are using.

They cannot inspect the infrastructure that created or controls it.

The architecture therefore provides transparency into the execution environment, while openly acknowledging that the operator and underlying infrastructure remain trusted components.

9. Logging and Session Retention

One of the most common claims made by privacy services is that they "do not log."

CrowdMe approaches this question differently.

Rather than focusing only on log files, the architecture is designed to minimize the amount of information that exists during normal operation. The disposable Browser VM itself is treated as temporary state. When the session ends, the Browser VM and its virtual disk are destroyed instead of being retained for future use.

This distinction is important.

A browsing session can leave information in many places—not only application logs, but also browser storage, temporary files, memory and virtual disks. CrowdMe attempts to minimize these sources by making the execution environment itself temporary.

Normal Operation

Under normal operation, every browsing session follows the same lifecycle:

1. A new Browser VM is created from the standard template.
2. The user browses normally inside that VM.
3. The session ends.
4. The Browser VM is powered off.
5. The VM configuration is removed.
6. The virtual disk is deleted.

The Browser VM is never reused for another session.

As a result, browser state created during the session—such as cookies, browser history, cache and local storage—disappears together with the Browser VM.

Destroying the Browser VM is therefore not a cleanup step. It is one of the primary security properties of the architecture.

What Is Retained?

CrowdMe necessarily retains a small amount of operational information while a session is active.

Examples include:

- session identifier,
- allocated Browser VM,
- remaining session time,
- token validation state.

This information exists only to operate the service.

The architecture intentionally avoids creating persistent user profiles or maintaining browser environments after a session ends.

When multiple CrowdMe servers are deployed, only token information is synchronized between servers:

- token hash,
- remaining session count,
- update timestamp.

No browsing sessions, browser contents or Browser VMs are synchronized.

What Does "No Logging" Mean?

The phrase "no logging" is often interpreted to mean that no information about a user's activity can ever exist.

That is not technically accurate for any hosted service.

While a session is running, the Browser VM obviously contains the user's browser state. Firefox has cookies, memory contains running processes, and temporary files may exist on the Browser VM's virtual disk.

The important question is not whether that information exists while the session is active—it must.

The important question is **whether that information continues to exist after the session has ended.**

Under normal operation, CrowdMe removes the Browser VM rather than retaining it.

Administrator Capabilities

Like every hosted service, CrowdMe ultimately depends on trust in the system operator.

An administrator with full control of the infrastructure could intentionally observe or retain information that is not retained during normal operation.

Examples include:

- opening the Browser VM console through Proxmox while a session is active,
- modifying the Browser VM template before deployment,
- changing the CrowdMe Application to preserve Browser VMs instead of deleting them,
- copying or snapshotting the Browser VM before it is destroyed,
- installing monitoring software inside the Browser VM,
- capturing memory or storage from the virtualization platform.

These actions are technically possible because the administrator controls the underlying infrastructure.

The CrowdMe architecture does **not** claim to make such actions impossible.

Why the Architecture Still Matters

Although an administrator could deliberately retain information, doing so requires changing the normal behaviour of the system or actively observing a running session.

The standard operating model is different.

Under normal operation:

- Browser VMs are created automatically.
- Browser VMs are destroyed automatically.
- Browser state is removed together with the VM.
- Browser environments are never reused.

The architecture therefore minimizes retained information by default rather than relying on periodic cleanup of persistent systems.

The Trust Boundary

CrowdMe cannot eliminate the need to trust the system operator.

Instead, the architecture attempts to make that trust explicit.

Users trust that the operator:

- deploys the documented software,
- operates the published architecture,
- does not intentionally observe running sessions,
- allows Browser VMs to be destroyed as designed,
- does not deliberately modify the Browser VM template to collect information.

These assumptions are no different in principle from trusting the operator of a VPN, hosted email service or cloud platform.

The difference is that CrowdMe minimizes the amount of information that normally exists by making the browsing environment itself disposable.

Why Disposable VMs Are Different

Many privacy services operate on persistent systems.

Those systems may periodically clear browser profiles, rotate logs or delete temporary files.

CrowdMe follows a different model.

Instead of cleaning the browser, CrowdMe destroys the computer that contained the browser.

This significantly reduces the amount of browser state that can persist beyond the lifetime of a session and avoids relying on cleanup mechanisms to return the environment to a known state.

Summary

CrowdMe does not claim that information can never be observed by a trusted administrator.

Instead, it is designed so that retaining browser sessions is **not part of normal operation**.

The Browser VM is created for one session, used for one session and destroyed after one session.

The architecture therefore minimizes retained information by making the execution environment itself temporary, while openly acknowledging that the underlying infrastructure and its administrator remain part of the system's trust model.

10. Trust Model

Every hosted service requires trust.

No technical architecture can completely remove the need to trust the operator of a system that someone else owns and administers. CrowdMe is no exception.

Rather than attempting to hide this fact, CrowdMe explicitly defines where trust is required and where the architecture itself provides technical separation.

Understanding this distinction is essential to understanding the security properties of CrowdMe.

What the Architecture Enforces

Some properties of CrowdMe are enforced by the system architecture itself.

These include:

- Every browsing session executes inside its own Browser VM.
- Browser VMs are isolated from one another.
- Browser VMs cannot directly access the application network.
- Browser VMs cannot directly access the management network.
- Browser VMs use the Tor Gateway for Internet access.
- Browser VMs are destroyed when a session ends as part of the documented operating model.
- Browser state is not intentionally shared between sessions.

These are architectural properties. They are independent of individual users and apply equally to every browsing session.

What Users Must Trust

Some aspects of CrowdMe cannot be verified from inside a Browser VM.

Users must trust that the operator:

- deploys the documented software,
- maintains the published network architecture,
- does not intentionally modify the Browser VM template to collect information,
- does not intentionally observe active browsing sessions,
- allows Browser VMs to be destroyed according to the documented lifecycle,
- protects administrative credentials and infrastructure.

These assumptions are common to every hosted privacy service.

CrowdMe does not attempt to claim otherwise.

Why Administrator Trust Cannot Be Eliminated

The CrowdMe operator controls:

- the physical server,
- the Proxmox hypervisor,
- the network infrastructure,
- the Browser VM template,
- the CrowdMe Application,
- the reverse proxy,
- the Tor Gateway.

An administrator with unrestricted control over these components could intentionally change the behaviour of the system.

For example, they could:

- preserve Browser VMs instead of deleting them,
- install monitoring software,
- inspect the console of a running Browser VM,
- modify the Browser VM template,
- change firewall rules,
- alter the CrowdMe Application.

These actions are technically possible because the administrator controls the infrastructure.

The architecture does not claim to make them impossible.

What the Architecture Does Achieve

Although administrator trust cannot be eliminated, the architecture still provides meaningful security properties.

The default operating model is:

- create a Browser VM,
- use it for one session,
- destroy it afterwards.

This means retaining browser state requires deliberate intervention rather than occurring automatically.

Similarly, communication between Browser VMs, infrastructure services and management systems is restricted through network segmentation and firewall policy rather than relying solely on application behaviour.

The architecture therefore minimizes the amount of information normally available while making deviations from the documented behaviour intentional rather than accidental.

Transparency as Part of the Trust Model

CrowdMe attempts to reduce blind trust through transparency.

The disposable Browser VM is intentionally inspectable.

Users can examine:

- the operating system,
- Firefox,
- installed software,
- running processes,
- routing,
- browser configuration,
- network configuration.

Users cannot inspect:

- Proxmox,
- pfSense,
- the CrowdMe Application,
- the Browser VM template,
- administrative activity.

Transparency therefore has practical limits.

It allows users to inspect the environment they are using, but not the infrastructure operating that environment.

Trust Boundary

The CrowdMe trust boundary can be summarized as follows.

```
ARCHITECTURALLY ENFORCED

• Disposable Browser VMs
• Session isolation
• Network segmentation
• Tor-only Internet access
• Temporary browser state
• Browser VM destruction

_____________________________________

TRUST REQUIRED

• Physical infrastructure
```

- Proxmox
- pfSense
- CrowdMe Application
- Reverse proxy
- Tor Gateway
- Browser VM templates
- System administrator

Everything above the trust boundary can be examined through documentation and, in part, through inspection of the Browser VM.

Everything below the trust boundary ultimately depends on the integrity of the system operator.

Why This Matters

CrowdMe is not designed around the assumption that users should simply trust statements such as "we do not log."

Instead, the architecture attempts to minimize retained information, clearly separate system components and openly document where trust remains necessary.

This allows users to make informed decisions based on documented architecture rather than marketing claims.

Comparison with Other Hosted Services

The trust required to use CrowdMe is similar in principle to that required when using:

- a VPN provider,
- a hosted email provider,
- a cloud virtual machine,
- a remote desktop service.

In each case, the operator controls infrastructure that the user cannot directly verify.

The primary difference is that CrowdMe minimizes the lifetime of the browsing environment itself. Rather than maintaining a persistent desktop or browser profile, every session begins with a newly created Browser VM and ends with its destruction.

The architecture therefore focuses on minimizing retained information while making the remaining trust assumptions explicit.

Summary

CrowdMe does not claim to eliminate trust.

Instead, it distinguishes between **technical guarantees provided by the architecture** and **trust placed in the system operator**.

The architecture limits communication, minimizes retained state and makes Browser VMs disposable.

The operator remains responsible for honestly operating the documented system.

Making this boundary explicit is an intentional part of the CrowdMe security architecture.

11. Security Properties

The previous chapters describe the individual components of the CrowdMe architecture. This chapter summarizes the security properties that emerge from those design decisions.

These properties should not be interpreted as absolute guarantees. They describe the behaviour of the system when operated according to the architecture documented in this whitepaper.

Disposable Execution Environment

Every browsing session executes inside its own disposable Browser VM.

Rather than attempting to reset or clean a persistent browser environment, CrowdMe creates a new Browser VM for each session and permanently destroys it afterwards.

This minimizes the amount of browser state that can persist across browsing sessions.

Minimal Retained State

CrowdMe is designed to minimize retained information.

Browser cookies, browsing history, cached content, local storage and temporary files exist only for the lifetime of the Browser VM unless the user deliberately exports information.

Likewise, CrowdMe servers synchronize only the minimum information required to validate session tokens. Browser sessions and browser state remain local to the server that created them.

Session Isolation

Each browsing session receives its own Browser VM.

Browser VMs are never reused between users and are never shared simultaneously by multiple users.

This prevents browser state from one session becoming part of another session through normal operation.

Network Containment

Browser VMs execute on an isolated network segment with a default-deny firewall policy.

They cannot:

- communicate directly with other Browser VMs,
- access the management network,

- access Proxmox,
- access the CrowdMe Application directly,
- bypass the Tor Gateway for Internet access.

This limits the impact of a compromised Browser VM and reduces opportunities for lateral movement inside the infrastructure.

Consistent Browser Environment

Every Browser VM begins from the same immutable template.

As a result, users receive the same operating system, browser version and browser configuration.

This consistency reduces variation between Browser VMs and helps minimize browser fingerprint differences that arise from differing software configurations.

CrowdMe does not claim to eliminate browser fingerprinting, but it does attempt to reduce unnecessary differences between sessions.

Transparent Execution Environment

Unlike many hosted privacy services, CrowdMe exposes the Browser VM directly to the user.

Users can inspect:

- the operating system,
- browser configuration,
- installed software,
- running processes,
- routing,
- network configuration,
- local certificates.

This allows technically inclined users to verify that the Browser VM behaves consistently with the published architecture.

Separation of Responsibilities

Each major system component performs a single primary role.

Component	Primary Responsibility
Proxmox	Virtual machine isolation
pfSense	Network policy enforcement
CrowdMe Application	Session orchestration
Browser VM	User browsing environment
Tor Gateway	Internet connectivity

Reverse Proxy	Public HTTPS entry

This separation reduces system complexity and makes security policy easier to understand and audit.

Limited Shared Information

Multiple CrowdMe servers can operate independently.

Only token information required for session validation is synchronized between servers.

Browser sessions, Browser VMs, cookies, browsing history and browser state are never replicated between CrowdMe installations as part of normal operation.

Explicit Trust Boundary

CrowdMe does not attempt to conceal where trust is required.

The architecture clearly separates:

- security properties enforced through technical design,
- operational behaviour,
- trust placed in the system operator.

Making this distinction explicit allows users to understand both the capabilities and the limitations of the system.

Summary

Taken together, the CrowdMe architecture provides the following security properties:

- Disposable browsing environments.
- Minimal retained browser state.
- Isolation between browsing sessions.
- Network containment of Browser VMs.
- Tor-routed Internet access.
- Limited synchronization between servers.
- Transparent and inspectable Browser VMs.
- Clearly defined administrative trust boundary.

These properties arise from the overall system architecture rather than from any single component. No individual technology—virtual machines, Tor, pfSense or the CrowdMe Application—provides these properties alone. They emerge from the way the components are combined and operated.

12. Limitations

No security architecture eliminates every risk.

CrowdMe is designed to reduce specific classes of tracking, persistence and exposure by using disposable Browser VMs, network isolation and minimal retained state. It does not attempt to solve every privacy or security problem.

Understanding these limitations is essential to understanding what CrowdMe is, and what it is not.

CrowdMe Does Not Eliminate Trust

CrowdMe is a hosted service.

Users do not control the physical servers, virtualization platform or supporting infrastructure. The system operator therefore remains part of the trust model.

The architecture minimizes retained information and documents how the system operates, but it cannot prevent a malicious operator from intentionally modifying that behaviour.

Users who do not trust the operator should not use the service.

Logged-In Accounts Identify Users

CrowdMe cannot prevent websites from identifying users who voluntarily log in.

If a user signs in to an online service such as email, social media or banking, that service will naturally associate the browsing session with the user's account.

Disposable Browser VMs do not change the relationship between a user and services they intentionally authenticate to.

Information Voluntarily Shared

CrowdMe cannot prevent users from revealing information about themselves.

Examples include:

- entering personal information into forms,
- uploading personal documents,
- sharing identifying photographs,
- revealing an email address,
- participating in online conversations.

Privacy ultimately depends on both technology and user behaviour.

Browser Vulnerabilities

CrowdMe assumes that browsers are complex software and that security vulnerabilities may exist.

If a malicious website exploits Firefox during an active session, that compromise occurs inside the Browser VM.

The disposable architecture limits the persistence of such compromises by destroying the Browser VM after the session ends.

However, CrowdMe cannot guarantee protection against every browser vulnerability, particularly vulnerabilities capable of escaping the virtual machine or compromising the underlying infrastructure.

Virtualization Is Part of the Trust Model

The security properties described in this document depend on correct operation of the virtualization platform.

If the hypervisor were compromised, the isolation between Browser VMs and the underlying infrastructure could no longer be assumed.

CrowdMe therefore relies on Proxmox and the technologies it is built upon to provide the expected virtual machine isolation.

Tor Has Its Own Limitations

CrowdMe routes Browser VM Internet traffic through the Tor network.

This changes the network path used to access websites, but it does not eliminate all forms of tracking.

For example:

- websites may still identify logged-in users,
- browser behaviour may still be observable,
- websites may correlate user activity across sessions,
- Tor exit nodes are publicly known and may be treated differently by some websites.

CrowdMe uses Tor as one component of the overall architecture rather than presenting it as a complete anonymity solution.

Local Computer Security

CrowdMe does not replace good security practices on the user's own computer.

The local computer remains responsible for:

- displaying the Browser VM,
- sending keyboard input,
- sending mouse input,
- protecting user credentials,
- protecting any files uploaded into the session.

If the user's own computer is compromised by malware or remote access software, CrowdMe cannot protect information captured before it reaches the Browser VM.

Network Availability

CrowdMe depends on Internet connectivity between the user and the CrowdMe server.

If connectivity is interrupted, the browsing session may end.

Because Browser VMs are intentionally temporary, interrupted sessions are normally not recoverable.

No Guarantee Against Future Discoveries

Security evolves continuously.

New browser vulnerabilities, virtualization vulnerabilities, network attacks and fingerprinting techniques may be discovered after this document is published.

The CrowdMe architecture should therefore be viewed as an evolving system rather than a permanent solution to every future threat.

Not a Complete Anonymity System

CrowdMe is designed to reduce retained browser state and isolate browsing sessions.

It is **not** intended to guarantee anonymity.

Whether a user can be identified depends on many factors outside the scope of the Browser VM architecture, including user behaviour, website design, browser vulnerabilities and the surrounding Internet ecosystem.

The architecture should therefore be understood as reducing specific risks rather than eliminating identification.

Summary

CrowdMe intentionally focuses on a specific problem:

Providing a fresh, disposable browsing environment for every session while minimizing retained information and isolating that environment from the surrounding infrastructure.

Many important privacy and security challenges remain outside that scope.

The purpose of this document is not to claim otherwise, but to describe the architecture accurately so that users can make informed decisions about when CrowdMe is, and is not, an appropriate tool.

13. Future Improvements

The CrowdMe architecture is intentionally designed to be simple. However, no security architecture is ever complete, and the system will continue to evolve as operational experience is gained and new technologies become available.

The items below represent areas currently being considered. They should be viewed as architectural direction rather than committed product features.

Multiple Independent Servers

The current architecture is designed to scale horizontally by adding additional independent CrowdMe servers.

Each server is capable of operating independently and requires only minimal synchronization of token information.

Future deployments may increase the number of geographically distributed servers while maintaining the same architectural principles:

- independent operation,
- disposable Browser VMs,
- minimal synchronization,
- local session handling.

Expanding the number of servers improves capacity and resilience without introducing central session infrastructure.

External Security Review

Security benefits from independent review.

Future versions of CrowdMe may undergo external architecture reviews or security assessments by experienced third parties.

Such reviews cannot prove the absence of vulnerabilities, but they can provide valuable feedback on architectural assumptions, implementation details and operational procedures.

Reproducible Browser Templates

One long-term goal is to make Browser VM templates easier to verify.

Possible approaches include:

- publishing template version information,
- publishing cryptographic hashes,

- documenting template build procedures,
- using reproducible build techniques where practical.

These measures would make it easier for external reviewers to understand exactly what software is deployed inside Browser VMs.

Increased Transparency

Transparency is considered an architectural principle rather than a marketing feature.

Future improvements may include:

- additional public technical documentation,
- architecture diagrams,
- configuration examples,
- operational documentation,
- explanation of design decisions,
- versioned security documentation.

The goal is to allow technically interested users to understand how the service operates without exposing information that would unnecessarily increase operational risk.

Security Hardening

As the platform evolves, additional hardening measures may be introduced where they provide meaningful security benefits without significantly increasing complexity.

Examples may include:

- browser hardening,
- operating system hardening,
- virtualization hardening,
- improved monitoring of infrastructure health,
- stronger isolation between system components.

CrowdMe favors straightforward, understandable security mechanisms over unnecessary complexity.

Continuous Improvement

Security is not a fixed state.

New attack techniques, browser vulnerabilities, virtualization technologies and operational experience will continue to influence the CrowdMe architecture.

This document should therefore be regarded as a living description of the current design rather than a permanent specification.

Future versions of the Security Architecture document will document significant architectural changes as they are introduced.

Summary

CrowdMe is designed to evolve carefully rather than rapidly.

Future improvements will continue to follow the principles established throughout this document:

- minimize retained information,
- isolate execution environments,
- reduce unnecessary complexity,
- increase transparency,
- document trust assumptions honestly.

These principles are expected to remain stable even as the underlying implementation changes.

14. Security Assumptions

The security properties described in this document depend on a number of assumptions about the underlying infrastructure, software and operation of the service.

If one or more of these assumptions no longer hold, the security properties described throughout this document may be reduced or no longer apply.

Making these assumptions explicit is an important part of understanding the CrowdMe architecture.

Trusted Infrastructure

CrowdMe assumes that the underlying physical infrastructure is operated according to the documented architecture.

This includes:

- physical server hardware,
- storage devices,
- networking equipment,
- virtualization host,
- supporting infrastructure.

Users do not have direct visibility into these components and therefore must trust that they are operated honestly and securely.

Proxmox Virtualization

CrowdMe relies on Proxmox Virtual Environment (Proxmox VE) to provide isolation between virtual machines.

The architecture assumes that:

- Browser VMs are correctly isolated from one another.
- Browser VMs cannot directly access the host operating system.
- Administrative access to the hypervisor is properly protected.
- Virtual machine lifecycle operations behave as expected.
- Security updates are applied in a timely manner.

If a vulnerability were to allow a Browser VM to escape the hypervisor, the isolation properties described in this document could no longer be assumed.

Like any virtualization platform, Proxmox is trusted to enforce the separation between guest virtual machines and the host.

pfSense Firewall

CrowdMe assumes that pfSense correctly enforces the configured network policy.

The architecture relies on pfSense to:

- separate internal network segments,
- enforce firewall rules,
- prevent unauthorized communication between components,
- route permitted traffic through the intended paths.

Incorrect firewall rules or misconfiguration could weaken the intended isolation between Browser VMs and other infrastructure components.

Tor Network

CrowdMe uses the Tor network to provide outbound Internet connectivity for Browser VMs.

The architecture assumes that:

- Browser traffic is routed through the Tor Gateway,
- the Tor software operates as designed,
- Browser VMs cannot bypass the Tor Gateway under normal operation.

CrowdMe does not assume that Tor provides complete anonymity or protection against every form of network analysis.

Tor is one component of the overall architecture rather than the sole security mechanism.

Browser Software

CrowdMe assumes that Firefox and the underlying operating system provide the expected level of security for normal operation.

No browser can be assumed to be free from vulnerabilities.

The CrowdMe architecture therefore assumes that browser vulnerabilities may occur and attempts to reduce their long-term impact by making Browser VMs disposable rather than persistent.

Browser VM Template

Every Browser VM is cloned from a common template.

The architecture assumes that this template:

- contains the documented software,
- has not been intentionally modified to collect user information,
- is maintained and updated appropriately,
- remains identical for every newly created Browser VM.

Because every session begins from this template, its integrity is fundamental to the architecture.

CrowdMe Application

The CrowdMe Application is responsible for:

- validating session tokens,
- allocating Browser VMs,
- managing session lifecycle,
- destroying Browser VMs after use,
- proxying user connections.

The architecture assumes that the application behaves according to the documented design and has not been intentionally modified to retain session information beyond normal operation.

Administrative Operation

CrowdMe assumes that administrative access is carefully controlled.

At the time of writing, administrative access is intentionally limited to a single trusted operator.

The architecture assumes that administrative credentials are protected and that administrative privileges are not used to intentionally inspect, retain or modify user sessions outside the documented operating model.

This assumption cannot be verified technically by users and therefore remains part of the trust model.

User Behaviour

The architecture also makes assumptions about how the service is used.

For example, CrowdMe assumes that users understand:

- logging into online accounts identifies them to those services,
- information voluntarily submitted to websites remains under the control of those websites,
- CrowdMe does not replace safe browsing practices or good endpoint security.

Technology cannot prevent every form of user identification.

Security Is Layered

No single component described in this document provides CrowdMe's security properties on its own.

Instead, the architecture depends on multiple independent layers working together.

These include:

- virtual machine isolation,
- network segmentation,
- firewall policy,
- disposable Browser VMs,
- Tor routing,
- minimal retained state,
- controlled administrative access.

If one layer is weakened, the remaining layers continue to provide value, but the overall security properties may be reduced.

This layered approach is a deliberate design decision.

Summary

The CrowdMe architecture is based on a number of clearly stated assumptions.

Rather than assuming that every component is perfect, the design combines well-understood technologies into a layered architecture that minimizes retained information and limits communication between system components.

Users should understand that the security properties described in this document depend both on the architecture itself and on these underlying assumptions remaining true.

Final Statement

No security architecture can eliminate trust or guarantee protection against every future vulnerability.

The goal of CrowdMe is more modest and more practical:

To create a disposable browsing environment that minimizes retained information, limits communication between system components, and clearly documents both its capabilities and its assumptions.

Understanding those assumptions is an essential part of understanding the system itself.