Managing NymBoxes for Identity and Tracking Protection

David Isaac Wolinsky, Daniel Jackowitz, Bryan Ford
Yale University

Abstract

Despite the attempts of well-designed anonymous communication tools to protect users from tracking or identification, flaws in surrounding software (such as web browsers) and mistakes in configuration may leak the user's identity. We introduce Nymix, an anonymity-centric operating system architecture designed "top-to-bottom" to strengthen identity- and tracking-protection. Nymix's core contribution is OS support for *nym-browsing*: independent, parallel, and ephemeral web sessions. Each web session, or pseudonym, runs in a unique virtual machine (VM) instance evolving from a common base state with support for long-lived sessions which can be anonymously stored to the cloud, avoiding de-anonymization despite potential confiscation or theft. Nymix allows a user to safely browse the Web using various different transports simultaneously through a pluggable communication model that supports Tor, Dissent, and a private browsing mode. In evaluations, Nymix consumes 600 MB per nymbox and loads within 15 to 25 seconds.

1 Introduction

Today's Internet users must increasingly assume that by default all of their online activities are tracked and that detailed profiles of their identities and behaviors are being collected by every Web site they visit [69], sold for marketing purposes [20, 56], and ingested into mass surveillance systems [62]. Users may wish to protect their online activities from being tracked or linked with their real identities, by accessing the Internet under several distinct *roles*, *personas*, or *pseudonyms*. Anonymity and pseudonymity are desirable to many types of users, from repressed minorities [70] and dissidents in authoritarian countries [49, 36], to women wishing to hide their pregnancies from advertisers [33, 34], to celebrity authors desiring "feedback under a different name" [77].

Anonymity protocols such as Tor [18], Dissent [80], and Aqua [46] obscure a user's network location, but *client-side* weaknesses can break this anonymity. Web sites may still be able to track the user via third-party plug-ins that circumvent the anonymous channel [58, 25], via browser fingerprints [22], employ software exploits to insert state into, or "stain", the client's browser for long-term tracking [59], or de-anonymize users directly [30, 64]. Anonymity-oriented Linux distributions such as Tails [72] and Whonix [79] mitigate some risks, but leave to the user the error-prone task of managing different online roles or pseudonyms. Users can accidentally de-anonymize themselves by logging in to a sensitive account from the wrong browser window, neglecting to protect anonymity even once [66], or by posting a photo without realizing that the JPEG may contain GPS coordinates [55].

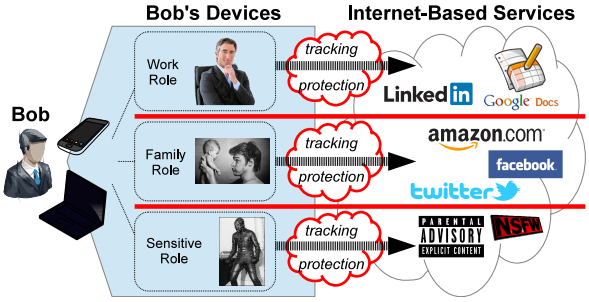

Figure 1: Nymix is a client OS architecture designed to enable users to manage multiple *roles* in their online life, and offer strong protections against their roles being tracked or linked.

To address this need we present Nymix, the first OS architecture designed to help users manage online roles, or *nyms*, and to protect these nyms systematically from accidental or malicious linking. As illustrated in Figure 1, Nymix aims to offer *end-to-end* isolation between nyms, separating all client-side state and browser activity related to each nym into protected virtual machines, or *nymboxes*. Nymix connects these nymboxes to the Internet *only* via separate instances of network tracking protection systems, such as Tor, protecting nyms from being linked by the online services they are used to access.

Nymix enables and encourages users to create ephemeral, "throwaway" nymboxes on demand for activities requiring no long-term state, such as reading news,

reducing the user's vulnerability to long-term tracking or intersection attacks [43]. Users can also create *persistent nymboxes* when needed, which can remember long-lived state such as login credentials for pseudonymous Internet accounts. Unlike common password managers [1], Nymix maintains and structurally enforces an explicit binding between each role a user plays online, the network login credentials related to each role, and all client-side state such as browser history related to each role. By binding client state and credentials to nymboxes, Nymix reduces the user's risk of accidentally entering credentials in the wrong context or browser window – when using the *correct* nymbox the user need not enter those credentials at all.

Like Tails [72], Nymix can boot from a USB drive for easy deployability and avoids leaving any history trail on the host machine, offering deniability in situations where installing anonymity tools may be dangerous or impossible. Nymix encrypts and saves persistent nymbox state to either local media or anonymous cloud storage. By default, Nymix updates nym state *only* at explicit user request (e.g., after the first login), and not after every browsing session, to protect the user from staining attacks on the nymbox's state, adding further deniability and history protection if the nym is ever compromised.

Nymix has been under development for the past two years, predating public knowledge that governments use malware to track [59] and de-anonymize users [30, 64], two key attack vectors that Nymix's design addresses. Throughout its development, the prototype has undergone regular design review and adversarial testing by an independent red-team. The prototype is based on Ubuntu 14.04, and uses two QEMU/KVM virtual machines for each nymbox: one to run communication tools such as Tor, and the other for the Web browser and associated plugins. Nymix supports multiple pluggable anonymous communication systems including Tor, Dissent, and a lightweight *incognito mode* that imposes minimal overhead but does not protect against network-level tracking.

Our experimental results show that Nymix offers similar performance to running the software natively or in similar software distributions, such as Tails. Nymix unsurprisingly requires significantly more memory, however, as each nymbox runs two virtual machines with all file system writes stored in RAM.

This paper's key contributions are: (1) an operating system architecture designed to help users keep local and online state related to different roles isolated and protected from tracking or linking; (2) a composable framework supporting pluggable anonymity tools, (4) anonymous quasi-persistent nym storage either locally or in the cloud, (3) user-directed sanitization to control information leaks across nyms, and (5) the ability to launch the user's installed OS as a nym for deniability and history protection.

Section 2 identifies key challenges for anonymity and pseudonymity. Section 3 presents Nymix's trust model and architecture. Section 4 describes our prototype implementation and our experiences with it. Section 5 evaluates how well our architecture and prototype handle challenges mentioned earlier. Section 6 discusses related work. Section 7 reflects on other challenges and future work, and Section 8 concludes.

2 Background and Motivation

This section motivates Nymix, and outlines the key challenges it attempts to address, via two fictional scenarios.

In the People's Republic of Tyrannistan, the state-controlled ISP monitors all users' traffic to censor and suppress dissent. Bob, a Tyrannistani dissident well aware of these dangers, uses Tor [18] at night to organize protests via his pseudonymous Twitter account [49, 36]. Bob spends his days at the state-run newspaper, using his laptop to grind out grandiloquent paeans to Glorious Leader Tyrannistanus Rex IV. Bob may face imprisonment if the censorware his job requires detects evidence of unapproved activities on his laptop's hard disk. Bob therefore runs Tor only from a Tails USB drive [72].

Bob could still be de-anonymized in many ways unless he is unerringly cautious. Since Tails (deliberately) forgets all state after each session, Bob habitually logs into his Twitter account anew each night – but if by force of habit he even once accidentally enters his Twitter credentials while *not* running Tails, he may be caught [66]. Bob's computer and his Web activity [22, 26] produce unique fingerprints. Using an intersection attack [43], Tyrannistani police can link Bob to his pseudonymous Twitter account. Bob might take a photo with his smartphone at a protest and post it to his Twitter feed via Tor, not realizing that the EXIF metadata in the photo contains GPS coordinates and his smartphone's serial number [55]. Even if Bob makes *no* such mistakes, the Tyrannistani police might obtain a zero-day exploit, and use it against Bob's browser to inject malware onto his running system, which reports his true IP and MAC address to the authorities [30, 64].

To improve his browsing experience, Bob begins experimenting with Tails' persistent state that stores passwords, account settings, and applications on the same USB device as Tails. Unfortunately for Bob, this opens him up to new types of staining [59], where malware might change his browser's user agent string, or fingerprinting, such as

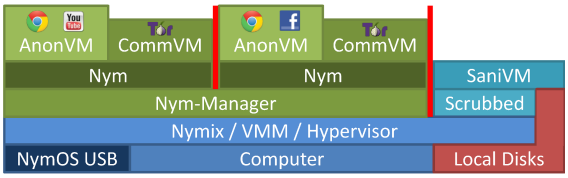

Figure 2: Block diagram of Nymix architecture. The hypervisor hosts one or more nyms. Each nym has an AnonVM for browsing Web content by communicating through the CommVM, which hosts the anonymizer. The SaniVM sanitizes access to local disks by sharing content scrubbed of personal information to AnonVMs.

the evercookie [41] that sticks around even if you disable cookies. In addition, the USB device now becomes evidence of Bob's *misbehavior*. Tyrannistan police can confiscate the device, coerce Bob to decrypt its contents, and then de-anonymize him.

Life is easier for Alice in Freetopia: she does not feel in any imminent danger, and is doing nothing she thinks the Freetopian Fuzz care about. She has made some personal choices that she is not ashamed of, and likes to discuss online in appropriate forums, but she imagines her boss and work colleagues might not understand [70]. She is also concerned that the web sites she visits, and the ads they present to her, seem to know more about her than her own family does. She worries that these web sites might "out" her unannounced pregnancy by sending a stream of diaper ads while her family is around [33]. She finds that the only way to keep such personal information secret is using cloak-and-dagger methods that might themselves raise suspicions of criminal activity [34]. Thus, although Alice does not think anyone is "after her," she would prefer to enforce a strong and inviolate barrier between her online activities related to her work, her family and social life, and her unannounced preparations for motherhood.

3 Nymix Architecture

This section outlines the Nymix architecture, how it binds pseudonyms to client-side and network state and protects nyms from being linked, and how Nymix addresses the challenges discussed earlier in Section 2.

3.1 Architecture Overview

Nymix is designed from the ground up to offer users strong identity and tracking protection by giving them explicit, first-class control over *pseudonyms* representing the multiple roles or personas they may use online. In contrast with the large body of existing work attempting to improve security or isolation between distinct users, or between applications run by a single user, Nymix is the first client OS, we are aware of, to establish strong *separation of roles* through pseudonyms as a primary OS design objective. Nymix places supervisory control over VM creation, longevity, and destruction under the user's control, binding all client-side application state to a particular nym via a nymbox. To protect the ownership and relationships between different nymboxes, Nymix employs *anonymizers* such as Tor and Dissent to protect against network-level linkage, and isolates network path to the anonymizer for each nym, deliberately making it difficult for the user to link nyms accidentally by posting the wrong file or cut-and-paste between the wrong windows.

With Nymix, for example, Alice may instantiate a nym to browse and check her e-mail, optionally loading the encrypted nym's state anonymously from a cloud storage system to avoid leaving a "footprint" on her local machine. During this process, she wants to check the latest news on Twitter and instantiates another nym, which Nymix works to keep unlinkable to the first. Finally, she wishes to post some content to her pseudonymous blog, doing that via yet another nym. She might discard the state of some nyms after each session, to protect her more sensitive activities from long-term tracking and intersection attacks [43], while preserving the state of other nyms locally or in cloud storage.

As shown in Figure 2, Nymix's most crucial component is its *Nym Manager*, which manages nyms and separates all client-side browsing and other activities into separate virtual machines or nymboxes for each nym. Each nymbox in fact represents two virtual machines. All normal client-side activity – such as running web browsers, their plug-ins, and other network-connected applications such as mail clients – is confined to the appropriate nym's *AnonVM*. Nymix treats the guest OS and processes in each AnonVM as untrusted and potentially compromised [59, 30, 64], and for this reason permits *no* interaction between the AnonVM and the "outside world" except via the nym's corresponding *CommVM*. In this CommVM reside anonymity and circumvention tools, or *anonymizers* such as Tor, which ensure that interaction between the AnonVM and the Internet is further protected from network-based tracking.

By isolating each nym's AnonVM from its CommVM, Nymix ensures that software exploits against the AnonVM cannot compromise anonymity or link nyms without also compromising the VMM. Launching a separate CommVM for each nym with an independent instance of the anonymizer, in turn, ensures, assuming the lack of a hypervisor exploit, that even an anonymizer compromise in one nym will not compromise other nyms. Separating CommVMs also ensures that anonymizer state that is commonly shared or reused for efficiency, such as

Tor circuits, cannot accidentally reveal the links between different nyms.

NymBoxes have no access to local storage, i.e., the local hard disk and USB devices. To safely access personal data, Nymix employs a SaniVM to isolate the user's data to a per-nym non-networked environment. The SaniVM sanitizes user data by either automatic or manual scrubbing of personally identifiable material from this data before making it available to an AnonVM.

3.2 Threat Model

Nymix's threat model assumes that an adversary may be able to compromise software within a particular AnonVM or CommVM, install software and inject software exploits, and even gain root access within the VMs. However, Nymix assumes the adversary cannot access the hypervisor of a compromised VM nor the host's file systems. While state-level attackers most likely *can* compromise common VMMs including those Nymix uses [65], we do not set the unrealistic expectation of making compromise *impossible*, but rather attempt to raise the barrier and cost to attackers significantly. We hope eventually Nymix could be implemented in a certified kernel/VMM framework [44, 31], further increasing this barrier.

As the CommVM hosts only anonymizing software such as Tor, a CommVM can only be compromised by an adversary directly attacking the anonymizer. Through a compromised CommVM, the adversary may learn Nymix's public IP address. While the AnonVM may be compromised through remote exploits, the CommVM prevents anything in the AnonVM from learning the user's IP address or other network or physical location information, so long as the anonymizer in the CommVM is not also compromised. A compromised AnonVM or CommVM cannot trivially be linked to other AnonVMs or CommVMs on the same host; however, attacks may be performed using timing attacks and side channels [83, 84].

Nymix cannot protect users who obtain compromised copies of Nymix itself, leaving the important problem of secure software distribution out of scope. We assume users obtain copies of Nymix through trustworthy sources or verify their authenticity prior to use. Nymix also does not improve the anonymity provided by the anonymizer, nor can Nymix prevent a user from explicitly de-anonymizing themselves by typing their real name into an AnonVM for example.

3.3 Anonymizers and CommVMs

Anonymizers such as Tor typically act as client-side Web proxies that redirect TCP connections through relays to hide their source. If the Web browser connected to this client-side proxy is misconfigured or vulnerable, however, an adversary can exploit that vulnerability to bypass the anonymizer: for example by invoking a plug-in that fails to respect the browser's proxy settings [58, 25], or by using malware to directly read and report the user's IP address [30, 64]. Like Whonix [79], Nymix separates anonymizers from the user's potentially vulnerable Web browsing environment via two separate virtual machines – an AnonVM and a CommVM, respectively. Unlike Whonix, which provides only a static user-managed pair of VM images, Nymix's Nym Manager dynamically launches and manages AnonVMs and CommVMs and manages their state as part of a user-controlled nymbox.

The user operates a nym primarily via the AnonVM, in which the web browser and other applications such as E-mail clients run. Each AnonVM has a single virtual network link that connects directly, and *only* to, a CommVM, which runs an instance of the anonymizer for this nym. The CommVM redirects all AnonVM traffic to the anonymizer, which in turns transmits traffic through the anonymity network via the CommVM's NAT-based Internet connection. No software in the AnonVM ever gets access to the physical host machine's IP address, MAC address, or other physical devices or their trackable device identifiers.

Alternative Anonymizers: Nymix treats the anonymizer as a pluggable module, and offers the user a choice of several alternative anonymizers preconfigured to address different security/performance tradeoffs. A lightweight *incognito mode* uses simple VPN relaying to provide low-cost anonymization with weak security. For more sensitive activities the user can employ Tor, which offers excellent scalability and good security against moderate adversaries. Finally, Nymix experimentally supports anonymous browsing via Dissent [80], an anonymizer based on DC-nets [13] that in principle offers formally provable traffic analysis resistance and systematic protection against intersection attacks [81], but is less mature and currently less scalable than Tor. In principle, anonymizers can be combined by connecting CommVMs in serial, or within the same CommVM: we have built experimental Nymix configurations combining Tor and Dissent to achieve "best of both worlds" anonymity, for example.

While many modern browsers offer incognito or private browsing modes that promise to erase cookies, history, and other state after a session, a single state management bug or security vulnerability in the browser can nevertheless render the user trackable [4]. Even in Whonix, such a state management bug – or malware-based stain attack [59] – renders the statically administered browser VM permanently trackable, and hence vulnerable to long-

term intersection attacks, unless the user manually reinstalls Whonix or resets it to pristine VM images. By isolating both the browser and any such stains in a dynamically managed AnonVM as part of an ephemeral-by-default nym, Nymix ensures that trackable stains disappear immediately when the nym does.

3.4 Creating and Configuring NymBoxes

One of Nymix's goals is to be small enough for users to download conveniently and run from a typical USB drive, like Tails, to support users who wish to leave no trace of their sensitive Internet access activities on their computers. A key practical challenge Nymix's VM-centric design presents, however, is that we effectively need to fit the equivalent of at least three different VM images, typically a gigabyte each, on the same USB drive: one containing the host OS atop which Nymix is built (currently Ubuntu Linux), the second containing an initial disk image for AnonVMs to use (containing the web browser and other applications), and the third containing an initial disk image for CommVMs to use (containing Tor or other anonymizers). Supporting multiple alternative anonymizers as discussed above might further increase the number of VM images that Nymix would need to "ship with."

To address this challenge, Nymix uses the OS image installed on the Nymix USB as the host OS from which the hypervisor/VMM boots, as well as the basic VM image for all AnonVMs and CommVMs. To differentiate these OS images to serve their distinct roles at runtime, Nymix employs union file systems, which logically stack multiple file systems together while merging their contents. The union file system responds to file read accesses with the contents of that file as it exists in the top most stack. The file system stores writes into the top most read-write layer, shielding lower layers from write access using copy-on-write.

Live-bootable operating systems such as Tails often use union file systems with the top layer consisting of a temporary file system that resides in RAM. Nymix inserts between the base image and the temporary file system an additional, intermediary *configuration file system* containing the configuration necessary to start the particular VM – e.g., one configuration file system for its standard AnonVM configuration, and a separate configuration file system for the CommVM representing each alternative anonymizer Nymix supports. The changes this configuration file system may include the network configuration files, the local startup script (/etc/rc.local), and the window manager startup script.

A nymbox's temporary file systems store all writes to the file system in RAM. As a result, turning off a pseudonym results in *amnesia* – Nymix wipes any traces that the pseudonym ever existed and securely erases the AnonVM's and CommVM's memory immediately on shutting down a pseudonym. The USB device used during a Nymix session remains unchanged, ensuring that even if confiscated and thoroughly analyzed neither the computer nor the USB device harbors evidence of Nymix use.

After terminating a nym, Nymix removes all state of that nym from memory. As designed, Nymix and all other existing production solutions retain traces of that state until reboot; however, because the hypervisor cannot be accessed without live confiscation, such state is likely to be inaccessible by an adversary. Recent work by Dunn [21] explores how much information remains on a host after a virtual machine has shut down, yet the hypervisor remains active, as well as various methods for eliminating it. Nymix could employ these methodologies to address adversaries with physical access; however, many of these features require specialized hardware and additional computational overhead, so for now we assume that erasing AnonVM and CommVM memory after shutdown are sufficient.

One security concern, created by Nymix's reuse of the host OS partition as AnonVM and CommVM images, is that Nymix must ensure that the host OS partition is *always* mounted read-only and never modified for any reason. This implies that any state the user wishes to persist across boots – such as persistent nyms, as described below – must be stored elsewhere, either on different local disks or USB drives or in cloud storage. If Nymix ever permitted the host OS partition to be modified from its standard "distribution" state, those modifications, however minute (even mount-time or access-time updates) would manifest in the initial states of all AnonVMs subsequently created, potentially offering adversaries a way to track the user. While Nymix by construction ensures that its host partition is only ever mounted read-only, it cannot prevent *other* operating systems from mounting the partition read/write and potentially modifying it while the USB drive is plugged in. Although not yet implemented, we intend to address this risk by adding a mechanism to check all disk blocks loaded from the host OS partition into an AnonVM or CommVM against a well-known Merkle tree [29] as they are accessed, and safely shut down rather than risk vulnerability if a modified block is detected.

3.5 Quasi-Persistent Nyms

Although ideal from a tracking resistance perspective, a *pure* amnesiac system that never maintains persistent state across reboots would inhibit usability, effectively requiring users to re-initialize all browser configuration preferences during each session, and to re-enter login credentials for any pseudonymous Internet accounts the user

might wish to access during the session. Worse, client OS amnesia can *reduce* the security of users who regularly connect to pseudonymous accounts such as Bob's dissident Twitter feed in Tyrannistan, in at least two ways. First, because Bob must enter his pseudonymous Twitter username and password during each session, this procedure is likely to become habit – but if he ever *even once* accidentally performs this procedure outside of an anonymity-protected context (e.g., on some other Ubuntu distribution he may sometimes run with a GUI look-and-feel similar to Tails), he may be compromised [66, 48]. Second, state-of-the-art anonymizers like Tor are more secure if they can maintain *some* state across boots – in particular, Tor normally maintains the same *entry relay* for several months – and may increase this period further [17, 23]. Users whose pseudonymous actions are linkable (e.g., via Bob's twitter feed) are inherently vulnerable to long-term intersection attacks, which the attacker can exploit far more rapidly if Tor chooses new entry relays frequently [39]. Thus, if Bob uses a pure amnesiac system to post to his Twitter feed, Tor is forced to choose a new entry relay each time he boots, greatly increasing his vulnerability to intersection attacks.

Both to offer convenience by enabling users to maintain content such as bookmarks, usernames and passwords, and application preferences, and to ensure that related anonymizer state is also preserved for security, Nymix supports *quasi-persistent data*. Quasi-persistent data resides on the machine only when actively in use. When not in use, an encrypted copy of the data is migrated to another storage device – either to another local partition or USB drive, or to the cloud, akin to CleanOS [74]. Nymix allows users to store information and data accumulated during a pseudonym session anonymously into the cloud, thus retaining pseudonym information while leaving no potentially "suspicious" state (even encrypted) on local devices that might be inspected or confiscated.

Nymix supports three different nym usage models: amnesiac/ephemeral, persistent, and pre-configured. The latter two both make use of quasi-persistent data, but with different intent. In *persistent* mode, Nymix updates the nym's stored state after each session, presenting a familiar and convenient state management model but increased risk that the effects of a stain or other exploit attack in one browsing session will persist for the lifetime of the nym. In *pre-configured* mode, a user boots a nym once, configures it with appropriate software, settings, bookmarks, pseudonymous account credentials, and any other useful state, then directs Nymix to *snapshot* the nym. Each subsequent use of this nym then starts from this snapshot, never updating the stored nym state unless the user explicitly requests another snapshot. Thus, a malware infection affecting one browsing session will be scrubbed at the user's next session, and even if the nym's state is eventually compromised, the attacker obtains no record of the user's client-side activities using the nym.

Workflow: In a typical workflow, Nymix on boot presents the user with a Nym Manager, offering options to **start a fresh nym** or **load an existing nym**. On first use, the user selects **start a fresh nym**. Each new nym begins with a writable virtual disk image in a standard, pristine state. When done browsing, if the user opts to store his nym, he returns to the Nym Manager and selects **store nym**. The user enters a name for the nym, a password to encrypt it with, and an indication of a cloud service on which to store the nym. In the background, the nym manager pauses the nym's AnonVM and CommVM, syncs their file systems, compresses and encrypts their temporary file system disk images, resumes the VMs, and uploads the contents to the cloud via the nym's CommVM. The nym manager notifies the user once the nym has been saved, after which the user may close the nym or turn off the computer.

Later the user returns to Nymix and selects **load an existing nym**. The Nym Manager, in response, prompts the user to select the cloud service hosting the nym, Nymix starts an *ephemeral* nym for the purpose of gathering the nym's state anonymously from the selected service. As before, the nym manager directs the user to the cloud service's login page, and then prompts the user for the name of the nym and the decryption password. In the background, the nym manager downloads the nym's state, and terminates the ephemeral nym used for downloading. The nym manager then proceeds to decrypt and decompress the loading nym's CommVM and AnonVM images, and resumes the nym by starting a new set of VMs using these images. The user may then continue using the nym.

Security Tradeoffs: The cloud storage solution has the advantage of offering plausible deniability to a user whose devices or USB drives may be inspected or confiscated. By utilizing free-to-use cloud storage options, such as DropBox or Google Drive, a user can create a pseudonymous cloud account for each pseudonym. Because all interactions with the cloud storage are anonymized, the cloud provider learns nothing about the account owner. Similarly, as pseudonyms store only encrypted data, cloud providers learn nothing about the pseudonym therein.

One subtle downside of the cloud approach is that the ephemeral CommVM used to load a nym from the cloud cannot use the nym's "proper" CommVM state – such as Tor entry relays – because that CommVM state has not been retrieved yet. While we do not expect it to be

easy for an attacker to correlate the loading of the nym's state through the ephemeral CommVM with the user's actions using the nym's own stateful CommVM, a sufficiently powerful and all-seeing attacker might in principle do so, making this ephemeral CommVM one remaining point of vulnerability to long-term intersection attacks. One solution is for the user simply to use local storage instead of the cloud. Another possible solution we are exploring is to *seed* critical CommVM state such as entry relay choices using a deterministic hash based on the nym's storage location and password, ensuring that the same seed (and hence same entry relay choices) get used even by the ephemeral CommVM that load the nym.

3.6 Sanitized File Transfers

Users may want to distribute content from non-anonymous sources – e.g., Bob wants to post pictures he took on his digital camera of the day's protests in Tyrannimen Square. Naïvely posting such files are risky, as they may leak the user's identity via GPS coordinates in EXIF metadata for example [10, 12, 55].

To mitigate such risks, Nymix never gives a nymbox direct access to files on the client machine's installed OS. Instead, Nymix delegates this responsibility to a dedicated, non-networked *sanitation VM* or SaniVM. Within this SaniVM, the user can access files on the installed OS and transfer files between nyms, via *scrubbing* tools that assist the user in transferring files safely.

Upon boot, Nymix searches the computer for file systems unrelated to Nymix and mounts them in the SaniVM. Within this SaniVM the user can browse through all files on the computer and select files for transfer into a nymbox. Prior to making any data accessible in the nymbox, however, the SaniVM launches a suite of scrubbing tools that inspect the files to be transferred, attempt to identify potential risks such as hidden metadata or visible faces in photos, present the user a list of these files and potential risks, and offer to apply appropriate scrubbing transformations under control of the user to remove potentially identifying personal information.

Nymix creates a unique SaniVM for each nym on-demand. The SaniVM has a folder that acts as a pipe into the AnonVM. The SaniVM detects when the user moves files into this directory and launches the scrubbing workflow. Once scrubbing completes, the SaniVM finally copies the file into another directory that actually facilitates the pipe-like transmit into the AnonVM.

The SaniVM's scrubbing process builds on the Metadata Anonymization Toolkit (MAT) [75], but Nymix adds atop MAT both additional anonymization methods and a more user-friendly workflow incorporating automated risk analysis and identification, enabling the user to select among alternative transformations, which might be seen as corresponding to different "paranoia levels." With images, for example, the user might choose any combination of: (a) scrub EXIF or other metadata, (b) blur any detectable faces using OpenCV [2], and/or (c) reduce the resolution and add noise in an attempt to disrupt any watermarks the image might contain unbeknownst to the user. With PDF or DOC files, the user can similarly scrub metadata, but also has the option to reconstruct the document completely as a series of bitmaps, effectively scrubbing any nonvisual information that might be concealed (accidentally or intentionally) in document's complex text or vector graphics structures.

While Nymix builds on a wealth of existing techniques to strip files of potentially identifying material [5, 8, 71, 75], clearly no scrubbing suite can be perfect. Developers continuously create new file types, and add extensions to existing file types, which might conceal identifying information. Adversaries can also find improved ways to exploit existing file types. Nevertheless, by designing personal information detection, analysis, and scrubbing into Nymix's *only* cross-nym file transfer path, we hope Nymix's architecture will ensure that users are at least made aware of the risks and offered choices that increase safety in common-case situations.

3.7 Installed OS as a Nym

Even if a user boots Nymix from USB, he likely already has a conventional OS installed on the machine, which he may use for common non-sensitive, non-anonymous activities. This installed OS is likely to have network-related state such as WiFi passwords or VPN software the user regularly employs to access local networks. To reduce Nymix's deployment burden and network configuration effort required on startup, Nymix can boot the machine's installed OS in a (non-anonymous) nymbox, and leverage its existing state to sign onto relevant WiFi LANs, or enable the user to find (or create) files on his installed system that he may wish to transfer to nyms via the SaniVM, using already-familiar applications on the installed OS. Nymix can currently boot several versions of Windows and Linux in this way.

The three keys challenges for installed OS nyms that differ from traditional nyms are booting the OS in a VM, addressing persistency, and specifying network configuration parameters. While Linux usually boots without issue, booting in a VM a Windows instance installed on the "bare metal" can trigger device driver complaints. We found that a standard repair process typically addresses this problem, much like VMWare Fusion [3]..

A user's installed OS may of course be compromised with malware or censorware [45] that may attempt to track

or fingerprint the user. To maximize the safety of booting the installed OS, Nymix treats the machine's hard disk as read-only and boots the installed OS into a copy-on-write virtual disk, so that no changes the installed OS makes while running under Nymix ever persist on the physical disk it was booted from. This design: (a) ensures that the user will not need to run a repair process *again* when switching back to the installed OS on the bare metal; (b) ensures that any other unexpected glitches caused by booting the installed OS in a VM cannot unexpectedly break the installed OS image on the underlying disk; (c) avoids leaving any history indicating Nymix's use on the local disk, offer the user plausible deniability.

If a user needs persistence, he may explicitly allow writes back to the physical disk, or store his copy-on-write COW disk as quasi-persistent data. If he subsequently starts his installed OS outside of Nymix, however, it may need to be repaired again, as in the case of Windows. Further, attempting to use the quasi-persistent COW disk after the underlying disk has changed can lead to inconsistency or corruption. Thus, we consider it safest to treat the installed OS as read-only, and leave exploration of more sophisticated alternatives to future work.

4 Prototype Implementation

Our prototype Nymix systems implements the architecture discussed in Section 3 including support for various anonymizers, circumvention tools, and sanitization techniques. Nymix uses the Ubuntu 14.04 64-bit Linux distribution and QEMU/KVM for running all nymboxes, besides Windows Host OS nyms. We have yet to consolidate nym activity to a single interface; instead we use the graphical user interface of each AnonVM to host the nym's Web browser. The Chromium Web browser was chosen in order to support circumvention software, specifically StegoTorus [78]. We have released Nymix source code, packages that build a fresh Nymix disk image, and Nymix images at `https://github.com/DeDiS/WiNoN`.

4.1 Anonymizers and Circumvention Tools

Nymix has the necessary configuration to support anonymizers, circumvention tools, and other communication tools that use either a SOCKS [47] or virtual network interfaces, such as a tap device. The entire run-time configuration for these tools resides within the CommVM, running completely transparent to the AnonVM. While Tor does not support UDP redirection, it has a built-in DNS server. Dissent, on the other hand, does have support for UDP redirection. For tools that support neither, Nymix would need to convert UDP-based DNS requests to TCP before transmitting them over the communication tool.

Currently, we have tested the following tools within Nymix: Tor [18], Dissent [80], our own implementation of SWEET [35], and an incognito mode. Tor has a built-in DNS server, while both Dissent and SWEET support UDP based proxying. Our incognito mode makes use of Linux' IPTables masquerade mode in order to provide a NAT interface into the Internet.

4.2 Virtual Machine Management

For virtualization, Nymix primarily depends on KVM [60], a virtualization solution built directly into the Linux kernel. KVM builds upon and recently merged with QEMU [7] and takes advantage of hardware virtualization, where available.

The hypervisor configures the network on the AnonVM to talk directly to the CommVM via a UDP port, effectively a host-only network. The UDP sockets bind to localhost in the hypervisor, hence, only applications in the hypervisor can access it. The CommVM connects to the Internet by way of KVM user-mode NAT.

Nymix configures the VM to reduce the ability for an adversary to fingerprint a VM. Each independent set of AnonVMs and CommVMs have the same Ethernet and IP addresses. The resolution within an AnonVM is consistently set to 1024x768, albeit that is configurable to be both smaller and larger. In general, Nymix strives to execute identically on all machines. Each VM has only a single CPU listed in /proc/cpuinfo as a QEMU Virtual CPU. The VM has 256 MB writable storage and 256 MB RAM, both of these consume the host's RAM.

Nymix enables KSM or kernel same-page merging. KSM is a memory-saving de-duplication feature that scans pages and merges when applicable. Because all Nymix VMs and the hypervisor use the same disk image and hence applications, Nymix can save a bit of RAM through the use of KSM, as we show in our evaluations.

Nymix stacks the file systems together using OverlayFS, a union file system built directly into the Linux kernel. Each VM has three file systems: 1) the base image, 2) a configuration image, 3) and a writable image. The hypervisor and VMs all share a common base image, this is the OS installed on the USB stick. The configuration image masks configuration files on the base image to enable AnonVM, CommVM, or SaniVM functionality. The writable image can either be tossed at the end of a session or stored in the cloud for quasi-persistent data stores.

Many modern virtual machine management tools support loading a real path within the hosts file system onto a guest. KVM makes use of VirtFS [40]. Within Nymix each of the different configuration file systems ex-

ists as paths within the disk image. When starting the pseudonym VMs, Nymix attaches the appropriate path to the VM as a VirtFS.

4.3 Sanitized File Transfers

The SaniVM hosts a multipurpose scrubbing tool that we designed. The scrubbing tool runs in two modes, the first takes advantage of MAT [75], the Metadata Anonymization Toolkit. The second mode converts the document into a series of images, effectively loading the document into a proper viewer, taking one or more screen shots, and then assembling the images together. Both tools strip away metadata; however, our extension does so by requiring only a viewing tool and not a tool that has explicit knowledge about what fields should be stripped. Of course, a malicious entity may embed visible content that neither stripper can remove.

After scrubbing, the SaniVM moves it into a shared folder with the hypervisor. The hypervisor, then in turn, moves it into a shared folder with the paired AnonVM. KVM includes a shared folder technology called VirtFS [40].

5 Evaluation

5.1 Validating the System

We validate the Nymix prototype using KVM and nested virtualization. This process made it easy to verify the state of the system and inspect for potential information leaks. To check for leaks, we connected the Nymix hypervisor to a virtual network interface that tunneled traffic to a NAT running on the host. On the host device, we ran Wireshark and inspected traffic entering and exiting an idle Nymix client. The Nymix hypervisor emitted only traffic for DHCP and anonymizer traffic, while the AnonVM transmitted no traffic.

We also started many pseudonyms simultaneously in order to verify the restricted communication model. We attempted to transmit Ethernet and IP packets from one AnonVM as well as one CommVM to the local network, other AnonVMs and CommVMs, as well as the hypervisor. All attempts failed with a no-response, as if the host did not exist. The AnonVM can only communicate with a functional CommVM and the CommVM could only communicate with the Internet not local intranets.

Beyond internal validation, Nymix has been regularly scrutinized for over 2 years by an independent red-team. The red-team primarily analyzed Nymix for information leaks. Their results confirm the soundness of the basic architecture, while identifying remaining risk areas, particularly user error and compromised Nyms exploiting side-channels as discussed later in Section 7.

5.2 Concurrent Nym Usage

As users explore the new functionality provided by Nymix, there will be a natural increase in pseudonym usage. Each additional pseudonym costs RAM as well as induces network and CPU overhead on other pseudonyms. In this section, we investigate these overheads in a series of experiments using an Intel I7 quad core desktop with hardware virtualization extensions and 16 GB of RAM. The desktop connects to a test Tor deployment running on the DeterLab testbed that eventually reaches the real Internet. The network connection between the DeterLab testbed [16], has a round trip latency of 80ms with a rate limited throughput of 10 Mbit/s through the Linux tool qc. Internally, the DeterLab testbed is connected by low latency, gigabit LANs, on which we did not instrument any additional delays or bandwidth constraints. While we could use the real Tor network, our evaluations focus on the overheads of Nymix and not noise introduced by the dynamic and complex nature of Tor. To analyze overheads, the VMs used two different memory configurations. Our CPU benchmark, Peacekeeper, demanded around 1 GB of RAM, whereas, bandwidth and regular Web access required only 384 MB of RAM. In all tests, we allocated 16 MB disk space and 128 MB RAM to each CommVM and 128 MB disk space to each AnonVM. The host allocates disk and RAM from its own stash of RAM, thus limiting the maximum number of nyms.

To evaluate memory per pseudonym, we launched a series of pseudonyms in succession. Upon loading a pseudonym, we checked the current used memory and kernel samepage merging (KSM) shared pages. We then interacted with a website and again noted the used memory and shared pages. At which point, we loaded another pseudonym and repeated producing 8 different nyms. We accessed the following websites in order: Gmail, Twitter, Youtube, Tor Blog, BBC, Facebook, Slashdot, and ESPN. Where applicable, we signed into Web sites and simulated some typical user behaviors, such as reading the latest news. Our results, Figure 3, show that KVM obtains most of the requested memory for a VM at VM initialization and not during run time. We also see that as more VMs are allocated, KSM manages to reduce overall memory usage resulting in over 5% saving at 8 nyms.

To evaluate CPU overhead, we ran a JavaScript benchmark called Peacekeeper [28] in several pseudonyms, simultaneously. Unfortunately certain experiments with Peacekeeper consume too much memory causing Chrome to crash, therefore we had to increase the RAM allocated to the AnonVM for this evaluation. We ran the evaluation with up to 8 pseudonyms and present the results in Figure 4. In this graph, 0 represents the system running in

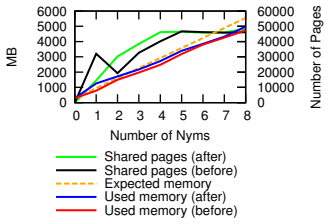

Figure 3: RAM usage and shared pages with varying number of pseudonyms before and after the new pseudonym becomes active. The dash line represents the estimated cost in RAM per-pseudonym.

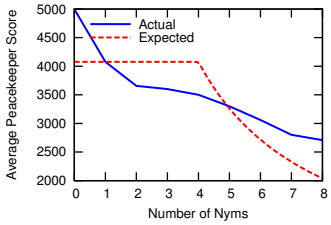

Figure 4: Accumulated values for parallel running instances of Peacekeeper running in independent pseudonyms. 0 represents the evaluation when run directly on the host.

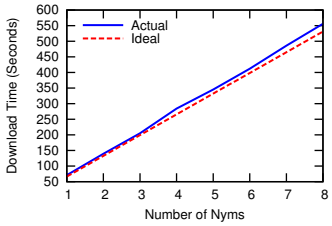

Figure 5: Time to download the Linux kernel with many nyms downloading in parallel. 0 represents the evaluation when run directly on the host.

native mode. Virtualization incurs about a 20% overhead. When running Peacekeeper in parallel, the actual performance outperforms the expected results, based upon the single nyms performance when run multiple times perfectly in parallel with other nyms. These results indicate that CPU performance overheads, while apparent, should not be a significant impediment to Nymix scalability.

Each additional nym uses its own instance of an anonymizer incurring additional bandwidth overhead due to control messages. In this evaluation, we download the current Linux kernel version 3.14.2, from a server running within DeterLab in order to guarantee the 10 Mbit download rate. We varied the number of parallel downloading nyms and present the results in Figure 5. As we scale the number of nyms, the performance remains relatively linear, indicating that Tor, the anonymizer in the CommVM, has a fixed cost, approximately 12% overhead. Performance on the real Tor network may differ significantly.

5.3 Pseudonym Storage

To evaluate the storage requirements for quasi-persistent pseudonyms, we monitored the size on disk of the nym state across ten save/restore cycles over the course of three days. Both the AnonVM and CommVM were equipped with 256 MB disks. We performed the experiment with four different nyms each visiting a different site: Twitter, Facebook, Gmail, or the Tor Blog. We began by launching a new pseudonym, visit the website, sign-in when applicable, and configure the browser to remember login information. We then closed the browser and saved the pseudonym to cloud storage. For all subsequent measurements, we restored the nym from cloud storage and launched the browser, triggering a fetch of any new site updates. After the page finished loading, we closed the browser, and, in the case of persistent nyms, saved the pseudonym back to cloud storage. For each upload we recorded the size on disk of the archived, encrypted pseudonym image.

We present the results of our experiment in Figure 6. Persistent nyms do grow over time with the AnonVM content accounting for 85% of the pseudonym size, though much of that is dominated by contents in Chromium cache, which could have been configured to be smaller than the default of 83 MB. Effectively a single save cycle represents usage similar to a pre-configured nym, which tends to be small in the order of megabytes.

5.4 Pseudonym Restoration

To evaluate the overhead incurred by starting a fresh nym and restoring a quasi-persistent nym, we compared startup times for the three different nym usage models: ephemeral, pre-configured, and persistent. For each configuration, we visited the Twitter website and retrieved updates. Upon finishing the page load, we closed the browser and, in the first two models, discarded all changes. For persistent nyms, we saved all changes back to the persistent state. We further divided pseudonym startup time into three phases: AnonVM boot time, Tor startup time, and webpage load time. For quasi-persistent nyms, we include the time it takes to start an ephemeral nym, download the state from the cloud, decrypt it, and prepare a new nym encapsulated as "Ephemeral Nym" We timed five executions from each initial configuration and present the averaged results in Figure 7. Quasi-persistent nyms consistently outperform ephemeral nyms due to stored Tor state; however, they require a one-time use ephemeral nym to download the data as well as user interaction necessary for authentication and nym selection, which was not measured in this evaluation. In practice, we expect user-interaction to play a significant role in nym boot times.

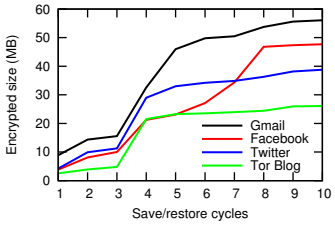

Figure 6: Sizes of quasi-persistent pseudonym data across save/restore cycles.

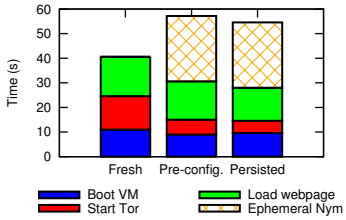

Figure 7: Average startup time by phase for each initial configuration.

5.5 Installed OS as a Nym

Running an installed OS as a nym requires both user interaction and computation resources. User interaction comes in the form of a user executing commands to repair the OS to support the change of hardware. The repair process analyzes the OS state and performs some reconfiguration. While hard to separate the human process from the automated process, this evaluation takes a look at both the time required to perform this process and the resulting impact on memory. Finally, after repairing the OS, the operating system can be booted as a nym. We present the results in Table 1. Memory sizes suggest that the repair process is quite invasive, suggesting that perhaps there may be a more intelligent means to automating this process. Ideally we could automate the repairing of the OS in the background, merging the time a repair takes with the booting process of Nymix and the associated CommVM.

	Repair (S)	Boot (S)	Size (MB)
Vista	133.7	37.7	4.9
7	129.3	34.3	4.5
8	157.0	58.7	14

Table 1: Time and memory costs of using various versions of Windows as a nym in Nymix.

6 Related Work

The Nymix project bears resemblance to other safety-focused bootable media projects. Tails [72] uses external bootable devices which have been preconfigured with Tor, the Tor version of Firefox, and other utilities for privacy protection. Like Nymix, Tails supports persistent state; however, Tails stores that state on the same USB as Tails. Even if the data is encrypted, a sufficiently powerful adversary could likely coerce the key from its owner. Nymix offers deniability by storing persistent state anonymously to the cloud. Because of the large community surrounding Tails, we view Nymix as a development platform for research ideas that eventually will be integrated into Tails, namely, structural protection such as nymboxes and quasi-persistent data.

Like Nymix, Whonix [79] separates the user environment from the communication tool eliminating accidental information leaks due to faulty component configurations. Unlike Nymix, Whonix installs a pair of virtual machines on the users installed operating system. Unlike Nymix, Whonix cannot defend against hardware fingerprinting, confiscation, or correlation attacks.

Similar to how Nymix isolates user data into a single environment, the SaniVM, the US military [42] and NSA [52] have made similar efforts to separate classified and unclassified information. In fact, the US military's approach to separation entitled "secure shared file store options" has a similar construction to Nymix's use of VirtFS without requiring changes to the hypervisor.

Nymix solves an important problem in the domain of scrubbing, negligence in their use [10], by forcing their use as part of an OS primitive Earlier work proposes similar extensions for networks [38]. Nymix builds on MAT [75], but there are many other tools out there and metadata scrubbing may be insufficient. Even intentionally blinded conference papers leak personally identifiable information [5]. Bier [8] shows that removing keywords is insufficient and still requires semantic analysis, in effect, nothing will be perfect. Data may be hidden by steganography and there is not much a scrubber can do to prevent active steganography [12].

As an alternative to selecting files and folders to scrub, Nymix could employ concepts introduced by User-Driven Access Control [63]. In this model, a user could grant access to certain folders and files on the host to a specific nym. Nymix could then delay scrubbing of files until the files have been accessed from within the nym.

The notion of pseudonyms long predates anonymous digital communication. Recent work from Han et al. [32] explores the notion of a pseudonym browsing mode. Their work focuses on a network adversary who links users primarily by IP addresses. To mitigate the effectiveness of this adversary, each pseudonym runs in the same OS but within different browser profiles and IPv6 ad-

dresses. Their approach requires infrastructure changes to deployed IPv6 routers in order to support multiple IPv6 addresses at the same site that could not easily be correlated, effectively creating a one hop proxy or a VPN. They have also implemented a plugin to address adversaries similar to Panopticlick [26]; however, this is an arms race and the plugin will need to be maintained in order to ensure that future fields do not leak information. Nymix's structural approach to homogeneity offers a future proof architecture that remains immune to these types of attacks.

There has been considerable work in using OS sandboxing to enhance isolation and resistance to compromised browsers and applications, such as BOS [15], IBOS [73], and Atlantis [53]. These approaches separate each web page instance into unique processes that have access to a limited API intended for web browsing only. However, this work has yet to address the need for pseudonymity and anonymity. Similar efforts have been made to offer separation on content in general and not just Web interaction [54], a feature Nymix also seamlessly offers. Nymix currently uses virtual machines as sandboxes, but architecturally Nymix could make use of lightweight process-level sandboxes [54], software fault isolation [76, 82], enhanced browser-based sandboxes [73, 53, 37, 15], or hosted instances in the cloud [51].

Nymix could be extended in many directions. A nym only isolates identities, but could benefit from approaches that isolate and protect sensitive data in a browser, such as Configurable Origin Policies [11], TaintDroid [24], and Crypton-Kernel [19]. While Nymix might isolate a key logger, ScreenPass [50] could offer Nymix a means to secure password entry to avoid spoofing attacks by providing a trusted password entry keyboard.

7 Discussion and Future Work

Lack of Perfect Homogeneity: Even while using virtualization and the same set of software, there still exists the possibility for differences between users. An adversary could execute a particularly CPU intensive application, such as a JavaScript application that computes a million digits of PI, and use the timings to produce a fingerprint for that user. Also, all users cannot necessarily be in the same location, and hence if there is a single Tor user in Tyrannistan, the government-owned ISP could easily determine the responsible party for any Tyrannistani traffic related to Tor.

Towards Deterministic Nym Behavior: Nymix currently does not improve upon the underlying system's defense against side-channel attacks [83, 14]. Such attacks could potentially be mitigated by incorporating performance isolation [68] and deterministic execution [6] techniques. Such defenses, however, typically incur significant overhead.

Self-Deanonymizing Users: A reckless user can easily de-anonymize himself in most anonymity systems. While Nymix goes to great lengths to prevent such accidents, there exist some limitations. In Nymix, for example, if a user shares personal information while signing an anonymous petition, by providing a personal e-mail account, Nymix like most other systems would be unable to prevent this type of accident. However, because of nym-browse, a user need not worry about correlation attacks wherein they de-anonymize a single nym instance. Specifically, if a user were to browse their Facebook or personal webmail account in one tab and access sensitive material in another, the adversary would be unable to link this material together. In systems like Tails and Whonix, such behavior might instantly de-anonymize the user.

Long Term Intersection Attacks: Nymix mitigates intersection attacks by reducing a user's fingerprint; however, a fingerprint can be developed even if the user employs amnesia simply by the set of websites he visits or the accounts used on those sites. An adversary performs an intersection attack [61] by tracking the online set of participants and discovering a set of linkable, yet anonymous messages. The adversary constructs an intersection of users that were online at the same time as those linkable messages. With a sufficient number of messages, the adversary will be able to discover the owner of the linkable messages. To enhance Nymix's ability to resist intersection attacks, we plan to integrate Buddies [81]. Buddies offers users anonymity metrics and safe guards a user from falling below a desirable anonymity threshold.

Concealing Network Identity: Concealing network identity proves difficult even in the Nymix context. Network fingerprinting comes in many forms from operating system interfaces to NIC devices [57], drivers [27], MAC addresses, and even the hardware characteristics of devices [9]. Some of these attacks are avoidable using a common device with a standardized driver, a volatile management framework for the device, and randomized MAC addresses.

For well-equipped adversaries, this approach is insufficient. Brik et al. [9] determined that even devices from the same manufacturer with sequential serial numbers could be fingerprinted due to the errors in the signal. Since we envision that Nymix may be used in moderately well-organized groups, we posit that users could organize WiFi device exchange parties, or WiFi social mixes, akin to

Richard Stallman's "Charlie Card" swapping parties [67] to elude RFID-based fingerprinting. During these parties, each member would place their cards in a box. After collecting all members' cards, each member would randomly select one without knowing which had been taken and which were left. Users might have many such parallel exchanges, so that a user could have several WiFi cards at a time. In addition, individuals could use an active antenna to ambiguate their location as well also to change the physical properties of the device's transmissions.

The Enemy Within: The Nymix model depends on the sterility of both hardware and software. A malicious party could install malware into the hypervisor prior to distribution or in the firmware of WiFi devices prior to a party in order to compromise a user's anonymity. Using trusted platform modules could potentially ensure the running software and firmware; however, all is for naught if the hardware vendor has conspired with the adversary.

8 Conclusion

Nymix offers novel structural solutions for managing online identities or pseudonyms. In contrast to existing system solutions for anonymity and pseudonymity, Nymix provides completely independent state for each of the user's identities. Nymix, however, does not subsume or replace the need for other techniques or hardened systems. We believe Nymix offers a useful platform for researching Web browsing pseudonymity that should eventually be incorporated into Tails and similarly hardened systems.

Acknowledgments

This material is based upon work supported by the Defense Advanced Research Agency (DARPA) and SPAWAR Systems Center Pacific, Contract No. N66001-11-C-4018.

References

[1] KeePass. `http://keepass.info/`.

[2] OpenCV. `http://opencv.org`.

[3] Migrating a Windows PC to run in VMWare Fusion. `https://www.vmware.com/pdf/migrating_physical_pc_to_fusion.pdf`, 2008.

[4] G. Aggrawal, E. Bursztein, C. Jackson, and D. Boneh. An analysis of private browsing modes in modern browsers. In *Usenix Security Symposium*, 2010.

[5] T. Aura, T. A. Kuhn, and M. Roe. Scanning electronic documents for personally identifiable information. In *5th ACM workshop on Privacy in electronic society (WPES)*, pages 41–50, New York, NY, USA, 2006. ACM.

[6] A. Aviram, S. Hu, B. Ford, and R. Gummadi. Determinating timing channels in compute clouds. In *ACM Cloud Computing Security Workshop (CCSW)*, Oct. 2010.

[7] F. Bellard. QEMU, a fast and portable dynamic translator. In *USENIX Annual Technical Conference, FREENIX Track*, pages 41–46, Apr. 2005.

[8] E. Bier, R. Chow, P. Golle, T. King, and J. Staddon. The rules of redaction: Identify, protect, review (and repeat). *Security Privacy, IEEE*, 7(6):46 –53, nov.-dec. 2009.

[9] V. Brik, S. Banerjee, M. Gruteser, and S. Oh. Wireless device identification with radiometric signatures. In *ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 116–127, 2008.

[10] S. Byers. Information leakage caused by hidden data in published documents. *IEEE Security and Privacy*, 2:23–27, 2004.

[11] Y. Cao, V. Rastogi, Z. Li, Y. Chen, and A. Moshchuk. Redefining web browser principals with a configurable origin policy. In *43rd IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, June 2013.

[12] A. Castiglione, A. D. Santis, and C. Soriente. Taking advantages of a disadvantage: Digital forensics and steganography using document metadata. *Journal of Systems and Software*, 80(5):750 – 764, 2007.

[13] D. Chaum. The Dining Cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, pages 65–75, Jan. 1988.

[14] S. Chen, R. Wang, X. Wang, and K. Zhang. Side-channel leaks in web applications: a reality today, a challenge tomorrow. In *IEEE Symposium on Security and Privacy*, May 2010.

[15] R. S. Cox, S. D. Gribble, H. M. Levy, and J. G. Hansen. A safety-oriented platform for web applications. In *IEEE Symposium on Security and Privacy (SP)*, 2006.

[16] DeterLab network security testbed, September 2012. http://isi.deterlab.net/.

[17] R. Dingledine. Improving tor's anonymity by changing guard parameters, 2013. https://blog.torproject.org/blog/improving-tors-anonymity-changing-guard-parameters.

[18] R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *12th USENIX Security Symposium*, Aug. 2004.

[19] X. Dong, Z. Chen, H. Siadati, S. Tople, P. Saxena, and Z. Liang. Protecting sensitive web content from client-side vulnerabilities with CRYPTONs. In *20th ACM conference on Computer and communications security (CCS)*, Nov. 2013.

[20] C. Duhigg. How companies learn your secrets. *The New York Times*, Feb. 2012. http://www.nytimes.com/2012/02/19/magazine/shopping-habits.html.

[21] A. M. Dunn, M. Z. Lee, S. Jana, S. Kim, M. Silberstein, Y. Xu, V. Shmatikov, and E. Witchel. Eternal sunshine of the spotless machine: Protecting privacy with ephemeral channels. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2012.

[22] P. Eckersley. How unique is your web browser? In *Privacy-Enhancing Technologies Symposium*, July 2010.

[23] T. Elahi, K. Bauer, M. AlSabah, R. Dingledine, and I. Goldberg. Changing of the guards: A framework for understanding and improving entry guard selection in tor. In *Workshop on Privacy in the Electronic Society (WPES)*, 2012.

[24] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *USENIX Conference on Operating Systems Design and Implementation (OSDI)*, 2010.

[25] G. Fleischer. Attacking tor at the application layer. http://ww.defcon.org/images/defcon-17/dc-17-presentations/defcon-17-gregory_fleischer-attacking_tor.pdf, July 2009.

[26] E. F. Foundation. https://panopticlick.eff.org/, Oct 2013.

[27] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. Van Randwyk, and D. Sicker. Passive data link layer 802.11 wireless device driver fingerprinting. In *USENIX Security Symposium*, 2006.

[28] Futuremark. Peacekeeper – the universal browser test. http://peacekeeper.futuremark.com, January 2013.

[29] B. Gassend, G. Suh, D. Clarke, M. van Dijk, and S. Devadas. Caches and hash trees for efficient memory integrity verification. In *High-Performance Computer Architecture (HPCA)*, 2003.

[30] D. Goodin. Attackers wield Firefox exploit to uncloak anonymous Tor users. *ars technica*, Aug. 2013.

[31] L. Gu, A. Vaynberg, B. Ford, Z. Shao, and D. Costanzo. CertiKOS: A certified kernel for secure cloud computing. In *2nd ACM SIGOPS Asia-Pacific Workshop on Systems (APSys)*, July 2011.

[32] S. Han, V. Liu, Q. Pu, S. Peter, T. Anderson, A. Krishnamurthy, and D. Wetherall. Expressive privacy control with pseudonyms. In *ACM SIGCOMM*, Aug. 2013.

[33] K. Hill. How target figured out a teen girl was pregnant before her father did. *Forbes*, Feb. 2012. http://www.forbes.com/sites/kashmirhill/2012/02/16/how-target-figured-out-a-teen-girl-was-pregnant-before-her-father-did/.

[34] K. Hill. You can hide your pregnancy online, but you'll feel like a criminal. *Forbes*, Apr. 2014. http://www.forbes.com/sites/kashmirhill/2014/04/29/you-can-hide-your-pregnancy-online-but-youll-feel-like-a-criminal/.

[35] A. Houmansadr, W. Zhou, M. Caesar, and N. Borisov. Sweet: Serving the web by exploiting email tunnels. *CoRR*, abs/1211.3191, 2012.

[36] P. N. Howard and M. M. Hussain. *Democracy's Fourth Wave? Digital Media and the Arab Spring*. Oxford University Press, Mar. 2013.

[37] L. Ingram and M. Walfish. TreeHouse: JavaScript sandboxes to help Web developers help themselves. In *USENIX Annual Technical Conference (ATC)*, June 2012.

[38] S. Ioannidis, S. Sidiroglou, and A. D. Keromytis. Privacy as an operating system service. In *Proceedings of the 1st USENIX Workshop on Hot Topics in Security*, HOTSEC'06, pages 8–8, Berkeley, CA, USA, 2006. USENIX Association.

[39] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson. Users get routed: Traffic correlation on Tor by realistic adversaries. In *20th ACM Conference on Computer and Communications Security (CCS)*, Nov. 2013.

[40] V. Jujjuri, E. V. Hensbergen, A. Liguori, and B. Pulavarty. Virtfs–a virtualization aware file system pass-through. June 2010.

[41] S. Kamkar. evercookie. `http://http://samy.pl/evercookie/`, oct 2010.

[42] P. A. Karger. Multi-level security requirements for hypervisors. In *Computer Security Applications Conference, 21st Annual*, ACSAC '05, Dec. 2005.

[43] D. Kedogan, D. Agrawal, and S. Penz. Limits of anonymity in open environments. In *Workshop on Information Hiding*, Oct. 2002.

[44] G. Klein et al. seL4: formal verification of an OS kernel. In *22nd ACM Symposium on Operating System Principles (SOSP)*, Oct. 2009.

[45] J. Knockel, J. R. Crandall, and J. Saia. Three researchers, five conjectures: An empirical analysis of TOM-Skype censorship and surveillance. In *USENIX Workshop on Free and Open Communications on the Internet*, Aug. 2011.

[46] S. Le Blond, D. Choffnes, W. Zhou, P. Druschel, H. Ballani, and P. Francis. Towards efficient traffic-analysis resistant anonymity networks. In *ACM SIGCOMM*, August 2013.

[47] M. Leech et al. SOCKS protocol, Mar. 1996. RFC 1928.

[48] D. L. Leger. How fbi brought down cyer-underworld site silk road, 2013. `http://www.usatoday.com/story/news/nation/2013/10/21/fbi-cracks-silk-road/2984921/`.

[49] M. Lim. Clicks, cabs, and coffee houses: Social media and oppositional movements in egypt, 2004 – 2011. *Journal of Communication*, 62:231–248.

[50] D. Liu, E. Cuervo, V. Pistol, R. Scudellari, and L. P. Cox. Screenpass: Secure password entry on touchscreen devices. In *11th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, June 2013.

[51] L. Martignoni, P. Poosankam, M. Zaharia, J. Han, S. McCamant, D. Song, V. Paxson, A. Perrig, S. Shenker, and I. Stoica. In *USENIX Annual Technical Conference (ATC)*, June 2012.

[52] R. Meushaw and D. Simard. NetTop: Commercial technology in high assurance applications. *Tech Trend Notes*, 2000.

[53] J. Mickens and M. Dhawan. Atlantis: Robust, extensible execution environments for web applications. In *23rd ACM Symposium on Operating System Principles (SOSP)*, Oct. 2011.

[54] A. Moshchuk, H. J. Wang, and Y. Liu. Content-based isolation: Rethinking isolation policy in modern client systems. Technical Report MSR-TR-2012-82, Microsoft Research, Aug. 2012.

[55] D. Oakes. Hacking case's body of evidence. *The Age*, Apr. 2012.

[56] OECD. Exploring the economics of personal data, Apr. 2013. OECD Digital Economy Papers No. 220.

[57] J. Pang, B. Greenstein, R. Gummadi, S. Seshan, and D. Wetherall. 802.11 user fingerprinting. In *ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 99–110, 2007.

[58] M. Perry. To toggle, or not to toggle: The end of torbutton. `https://blog.torproject.org/blog/toggle-or-not-toggle-end-torbutton`, May 2011.

[59] W. Post. GCHQ report on 'MULLENIZE' program to 'stain' anonymous electronic traffic. `http://apps.washingtonpost.com/g/page/world/gchq-report-on-mullenize-program-to-stain-anonymous-electronic-traffic/502/`, oct 2013.

[60] Qumranet. Kernel-based virtual machine for linux. `http://kvm.qumranet.com/kvmwiki`, March 2007.

[61] J.-F. Raymond. Traffic analysis: Protocols, attacks, design issues and open problems. In *Workshop on Design Issues in Anonymity and Unobservability*, pages 10–29, 2000.

[62] J. Risen and L. Poitras. NSA report outlined goals for more power. *The New York Times*, Nov. 22, 2013.

[63] F. Roesner, T. Kohno, A. Moshchuk, B. Parno, H. J. Wang, and C. Cowan. User-driven access control: Rethinking permission granting in modern operating systems. In *IEEE Symposium on Security and Privacy*, May 2012.

[64] B. Schneier. Attacking Tor: how the NSA targets users' online anonymity. *The Guardian*, Oct. 2013.

[65] B. Schneier. Nsa surveillance: A guide to staying secure. *The Guardian*, Sept. 2013.

[66] E. Security. Notes on Sabu arrest, 2012. `http://blog.erratasec.com/2012/03/notes-on-sabu-arrest.html`.

[67] J. Sedgwick. The shaggy god. `http://www.bostonmagazine.com/articles/2008/04/the-shaggy-god/`, May 2008.

[68] A. Shieh, S. Kandula, A. Greenberg, and C. Kim. Seawall: Performance isolation for cloud datacenter networks. In *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2010.

[69] A. Soltani, S. Canty, Q. Mayo, L. Thomas, and C. J. Hoofnagle. Flash cookies and privacy, Aug. 2009.

[70] E. Stein. Queers anonymous: Lesbians, gay men, free speech, and cyberspace. *Harvard Civil Rights-Civil Liberties Law Review*, 2003.

[71] L. Sweeney. Replacing personally-identifying information in medical records, the scrub system. *Journal of the American Medical Informatics Association*, 1996.

[72] Tails: The amnesic incognito live system, September 2012. `https://tails.boum.org/`.

[73] S. Tang, H. Mai, and S. T. King. Trust and protection in the illinois browser operating system. In *9th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Oct. 2010.

[74] Y. Tang, P. Ames, S. Bhamidipati, A. Bijlani, R. Geambasu, and N. Sarda. Cleanos: limiting mobile data exposure with idle eviction. In *USENIX Symposium on Operating Systems Design and Implementation*, Oct. 2012.

[75] J. Voisin, C. Guyeux, and J. M. Bahi. The metadata anonymization toolkit. `http://arxiv.org/abs/1212.3648`, may 2013.

[76] R. Wahbe, S. Lucco, T. E. Anderson, and S. L. Graham. Efficient software-based fault isolation. *ACM SIGOPS Operating Systems Review*, 27(5):203–216, Dec. 1993.

[77] R. Watts. JK Rowling unmasked as author of acclaimed detective novel. *The Telegraph*, July 13, 2013.

[78] Z. Weinberg, J. Wang, V. Yegneswaran, L. Briesemeister, S. Cheung, F. Wang, and D. Boneh. StegoTorus: a camouflage proxy for the Tor anonymity system. In *19th ACM conference on Computer and Communications Security (CCS)*, Oct. 2012.

[79] Whonix. `http://sourceforge.net/p/whonix`.

[80] D. I. Wolinsky, H. Corrigan-Gibbs, A. Johnson, and B. Ford. Dissent in numbers: Making strong anonymity scale. In *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Oct. 2012.

[81] D. I. Wolinsky, E. Syta, and B. Ford. Hang with your buddies to resist intersection attacks. In *20th ACM Conference on Computer and Communications Security (CCS)*, November 2013.

[82] B. Yee et al. Native client: A sandbox for portable, untrusted x86 native code. In *IEEE Symposium on Security and Privacy*, May 2009.

[83] Y. Zhang, A. Juels, A. Oprea, and M. Reiter. Homealone: Co-residency detection in the cloud via side-channel analysis. In *IEEE Security and Privacy SP (IEEE SP)*, 2011.

[84] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart. Cross-vm side channels and their use to extract private keys. In *ACM Conference on Computer and Communications Security (CCS)*, 2012.