The Net Rat
Rethinking Connected Services for Increased Security

Bernd Prünster[1], Florian Reimair[2] and Andreas Reiter[2]
[1]*Secure Information Technology Center – Austria (A-SIT), Graz, Austria*
[2]*Institute of Applied Information Processing and Communications (IAIK), Graz University of Technology, Graz, Austria*
bernd.pruenster@a-sit.at, {florian.reimair, andreas.reiter}@iaik.tugraz.at

Keywords: Decentralisation, Serverless Services, P2P Networks, Personal Mesh Network, Distributed Services.

Abstract: Traditional desktop computers have been outranked in terms of usage numbers by mobile devices. Still, many popular mobile-first services rely on workflows designed decades ago for a different environment. When relying on cloud-based services, privacy and data protection issues need to be considered. Mostly, however, one can choose between either well-supported legacy applications or innovative niche solutions. In this paper, we introduce the Net Rat, a framework enabling a seamless transition from existing centralised setups to decentralised state-of-the-art services, increasing security while maintaining backwards compatibility to well-established legacy services. We demonstrate the feasibility of our approach with a case study focusing on the decentralisation of the e-mail service—until now, this failed due to missing backward compatibility. A security analysis demonstrates how our approach reaches its goal of protecting user data through decentralisation. The Net Rat is built on a solid foundation as result of a security-first design. The results of this work clearly show the feasibility of decentralising existing services and highlight how well-established services can be improved. Our approach also presents opportunities to develop new services based on a solid foundation.

1 INTRODUCTION

Examining today's device landscape reveals that users are connected to the Internet using multiple devices[1]. In essence, mobile devices became ubiquitous and so did connected, cloud-backed services. On the one hand, the relatively new, mobile-first messenger *WhatsApp* has over one billion active users[2]. On the other hand, traditional services such as e-mail are still highly relevant. Regardless of service and context, users are accustomed to the fact that their data is available on all of their devices. Current solutions often rely on cloud services to meet this demand and users typically have no say in which entities are accessing their data.Service providers might even generate added value out of the users' data by showing targeted advertisements (Google, Inc., 2014). Moreover, business interests of service operators can change over time, possibly leading to changes in the way sensitive user data is handled. External factors such as changes in government policies could also force service providers to share sensitive data with public authorities. This claim is supported by governments having made statements that ways to break encryption are pursued to extract user data from cloud-based services (Blum-Dumontet, 2017).

In this work we propose an innovative solution to put the user back in control of their data using a mesh-like organisation of all the user's devices. We introduce *The Net Rat* as a secure gateway for legacy connected services on the one hand, and, on the other hand, enable the development of new innovative services, targeting the ubiquity of connected devices. The Net Rat takes the characteristics of typical end-user devices into account and delivers improved security properties without sacrificing usability and compatibility. In essence, we propose to transform existing server-centric services into peer-to-peer (P2P) applications running directly on users' devices, with each user operating their own P2P network to connect their devices. However, compatibility to existing (outside) services is imperative. Therefore, the Net Rat was designed to be backwards compatible.

This paper is structured as follows: First, some background and an analysis of the current situation is provided. Next, our system's architecture and its security features are described in detail. Section 4 presents a case study illustrating how we used the Net

[1]https://www.statista.com/statistics/333861/connected-devices-per-person-in-selected-countries/
[2]https://blog.whatsapp.com/616/One-billion

Prünster, B., Reimair, F. and Reiter, A.
The Net Rat - Rethinking Connected Services for Increased Security.
In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Volume 4: SECRYPT*, pages 375-383
ISBN: 978-989-758-259-2

Rat under real-world conditions to provide a decentralised e-mail service and replace existing implementations. Afterwards, we illustrate the security properties of our system by means of a security evaluation comparing the Net Rat to traditional services. Having elaborated on all aspects of our approach, we then discuss related work. Section 7 finally concludes this paper and outlines possible future work.

2 THE PRICE OF MAINTAINING LEGACY WORKFLOWS

Observing some of today's most popular connected services, we have already identified three main issues common to many (usually cloud-based) applications: *data security concerns*, *significant overhead*, and *legacy liabilities*.

At its core, any centralised service processing data in plain introduces some data security risks. Sometimes it is even part of a service operator's business model to have full access to all of its users' data (Dropbox, Inc., 2016). Moreover, everything happening on the back end of a service is typically out of the user's control. Even self-hosting services with limited complexity like e-mail typically involves a central server. Compromising such a central instance is typically enough to affect all devices using a service.

Times have changed since relying on central, static instances was often the only way to provide a service, when typical end-users had a single device, and were only online during limited periods. The server-centric, traditional e-mail service is the prime example illustrating how forcing such rigid architectures on today's multi-device users introduces significant overhead: When starting to write a message on one device, it is stored (as a draft) on a server. If a user then chooses to continue composing this message on another device, it first has to be downloaded. Now it is located on three separate devices and has been routed through the Internet at least twice. Once the message is to be sent, it is transmitted to the server, which forwards it to some external server. In addition, it is also stored as a sent message on the device, on the e-mail server, and transferred through this server to all other devices this e-mail account is used on.

Although it would be easy to dismiss such concerns by arguing that legacy compatibility demands certain workflows, this would be missing the point. In fact, many popular Internet-based services have been augmented to meet the needs of today's multi-device users but still operate based on paradigms defined decades ago. For example, instant messengers have been around since the nineties and even today's most popular messenger service, WhatsApp, is still based on a server-centric architecture (Fiadino et al., 2015). Consequently, the security properties of many well-known and critical services have become more complex over the past years. While the problem of directly connecting devices is still not fully solved, protocols like*ICE* (Rosenberg, 2010) have been proven to be effective enough to enable even applications with tight network constraints. The file sharing scene has been dominated by *BitTorrent* (Cohen, 2013) for years[3], which demonstrates that peer-to-peer is a viable way to go. We therefore believe that simply adding new features on top of systems based on concepts tailored to the environment of the 1980s will lead to a dead end.

We show that there is another way to implement even services demanding legacy compatibility by providing a working example of such a system based on the Net Rat. The Net Rat also offers increased security and availability and utilises the full potential of today's connected device landscape by daring to question established paradigms.

3 THE NET RAT

Our approach to harnessing the potential of current devices comes down to creating per-user P2P networks. In essence, a user connects to local proxies, which interface with a decentralised version of an existing service. The decentralised service is deployed on each of the user's devices and is interconnected by a peer-to-peer network. The P2P network synchronises the data among all nodes after translating service-specific operations into service-independent events. On each device connected to the network, these events are transformed back and operations are replicated. Users can therefore still use legacy client applications while regaining control of their data.

As the Net Rat primarily targets multi-device users, it will typically be deployed on few (< 10) devices all controlled by a single user. Bootstrapping of the P2P network is accomplished by simply (manually) registering each instance's identifier at the other instances.

3.1 Scope

The overall goal of the Net Rat is to offer a platform to *transparently* decentralise existing services for increased security. Most importantly, the applications

[3]https://torrentfreak.com/bittorrent-still-dominates-internets-upstream-traffic-151208/

themselves and service operations do not need to be altered in any way. Regardless of whether an application uses external infrastructure, the data a user operates on needs to be present at the user's devices. The Net Rat simply eliminates the need for any external services, thus eliminating all associated risks. By keeping data close to the user, the Net Rat can also offer increased availability at no cost. At the same time, by only ever storing encrypted user data, it becomes harder for attackers to extract data.

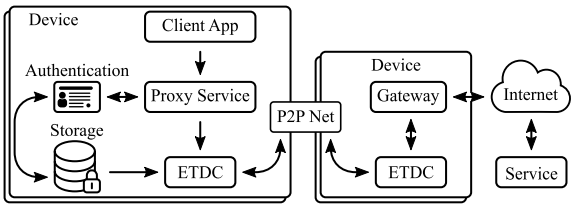

Figure 1: The Net Rat's architecture.

Other aspects of decentralised, redundant storage such as space-optimised data persistence or data recovery from lost devices are not the focus of the Net Rat, as these issues can often be solved on the application level. The same also holds true for remotely wiping stolen or lost devices.

3.2 Architecture

The Net Rat combines concepts of a number of different approaches in order to provide a scalable and flexible solution to the challenges discussed initially. An architectural overview on the Net Rat is depicted in Figure 1. On an end-user device, the following components are typically present: The *client app*, an unmodified existing application, makes a service available to the user as usual. However, instead of using a traditional remote *service*, the client app connects to a local *proxy service*.

The proxy mocks a traditional service and therefore keeps existing client applications functional. In order to provide its service, the proxy has to interface with the *storage*, the *authentication* component and the *Event Transformation and Distribution Core* (ETDC).
Whenever the proxy service fetches data from the storage, an authentication challenge needs to be satisfied—this will typically involve user interaction. However, Section 4 clearly illustrates that this does not affect usability in real-world scenarios. In order to distribute service-specific operations as events, the proxy service has to interface with the ETDC.

3.3 Event Distribution

The *Event Transformation and Distribution Core* forwards incoming events to the storage component. Events can originate from the local device, or from other connected nodes. These are interconnected by a *peer-to-peer network*. Events are the smallest unit of data processed by the Net Rat since any functionality of a service can be described as a series of events.

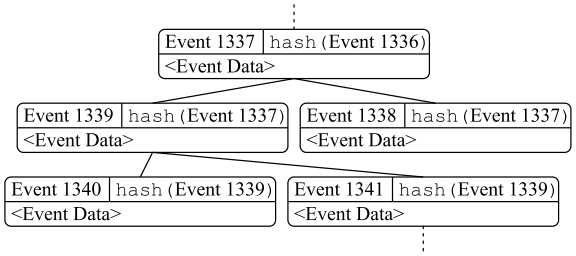

Figure 2: Event Tree.

The ETDC *transforms* service operations into service-independent events, *distributes*, and replicates them in-order. While transformation could also be integrated into proxies, it would then also need to be integrated into gateways. With our approach, purely client-side parts of proxy business logic can be reused for a variety of services using the same client-applications. This does scale, as implementing event transformation for a single protocol (such as HTTP, for example) covers all services relying on this protocol. Consequently, scalability is much less of an issue than it would seem at first glance.

Preconditions need to be met in order for each event to be executable. By chaining events, the exact flow of events can be replicated at any time. A Net Rat event may have multiple successors, leading to a structure similar to an unbalanced hash tree. By relying on a tree-like event log, it is possible to start multiple long-running events in parallel. Part of an example event tree is depicted in Figure 2.

The Net Rat primarily aims for single-user scenarios, where a user will typically only use one device at a time. Therefore, conflict detection is not a priority. Instead, this basic event chaining and replication ensures that events are executed in the correct order. Detecting events lost during transmission is straight forward, since gaps in the event tree can be detected immediately.

With all of the Net Rat's components in place, events can be spread among all devices and users can access event data using their client application of choice. From the user's point of view, it looks and feels just like using a traditional service.

3.4 Connecting to the Outside World

In order to interface with external services and legacy infrastructures, the Net Rat introduces a *gateway*. The setup shown in Figure 1 uses a distinct device with the sole purpose of connecting to the outside world. No data is persisted and the only events produced on the gateway device originate from external services. These events are fed into the P2P network and are therefore available at every client application. Only those events from within the network addressed to external services are forwarded by the gateway.

Contrary to traditional client-server systems, the Net Rat can even provide some functionality during network downtimes, thanks to storing all data at end-user devices.

3.5 Crypto-enhanced Storage

The *storage* is designed to write incoming event data to disk. Incoming data is encrypted using public-key cryptography and directly stored on disk while accessing and decrypting data requires user-input information. This is necessary, since the private key needed for decryption needs to be unwrapped. By relying on key wrapping, no authentication information needs to be stored. Ensuring transport security, on the other hand, is up to the peer-to-peer network implementation.

The crypto-enhanced storage consists of the actual storage and the following supporting building blocks: A bulk encryption engine, a symmetric key encryption engine enabling key wrapping, and a key derivation function providing the secret used for key wrapping.

During the setup of a Net Rat instance, the following actions are performed for each device: The bulk encryption engine creates a fresh key pair $(K^{pub}, K^{priv}, meta)$ (Eq. 1). Next, the user provides a secret s in order to wrap the private key K^{priv} following Eq. 2. The public key K^{pub}, the wrapped key wk, and the key meta data $meta$ are then stored and the system is then ready for use.

$$(K^{pub}, K^{priv}, meta) \leftarrow KeyGen(rand, len) \quad (1)$$

$$wk \leftarrow Enc(K^{priv}, KDF(s)) \quad (2)$$

Whenever data is fed into the storage, it is stored only after being encrypted following Eq. 3. The resulting ciphertext c is stored on disk for future use.

When the proxy service demands access to the data, it has to be decrypted. First, the user secret s is used to unwrap the wrapped key wk to recover the private decryption key K^{priv} (Eq. 4). Then, the decryption can be performed following Eq. 5.

$$c \leftarrow Enc(d, K^{pub}) \quad (3)$$

$$K^{priv} \leftarrow Dec(wk, KDF(s)) \quad (4)$$

$$d \leftarrow Dec(c, K^{priv}) \quad (5)$$

Similar to traditional schemes, user authentication is necessary prior to retrieving information. At the same time, it is possible to notify the user about incoming data since encryption only happens immediately before data is written to disk. Furthermore, no secrets need to be stored and protected on disk. Receiving data does not require user interaction. Yet, data is enciphered before it is stored on disk. Therefore, an attacker cannot gain access to any data even through physical access to a device and its storage.

3.6 Summary

The Net Rat takes traditional client-server services and provides them using a more secure, more robust, and more flexible network of personal devices. By using P2P networks, the Net Rat can operate in difficult network situations while drastically reducing the attack surface and therefore bringing more security and privacy to the user. By distributing data to all of a user's multiple devices, users experience increased availability. In addition, the Net Rat's foundation can be used to develop new services, fully utilising the connected nature of end-user devices.

4 CASE STUDY: E-MAIL

We evaluated the utility of our approach under real-world conditions by building a decentralised e-mail service on top of the Net Rat. From a technical point of view, we adapted and split traditional *Mail Transfer Agents* (MTAs) and *Mail Delivery Agents* (MDAs) to work as gateways and proxy services. Users can thus continue to use the *Mail User Agents* (MUAs) they are familiar with and benefit from the Net Rat's features.

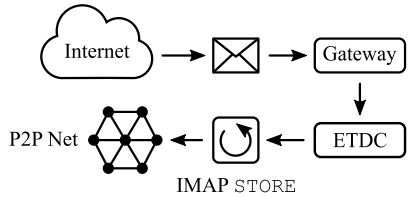

Figure 3: Incoming Mail Processing at the Gateway.

Internally, during system setup, our prototype creates an EC key pair for use with the *Elliptic*

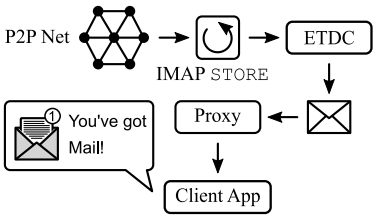

Figure 4: Mail Replication at the Client.

Curve Integrated Encryption Scheme (ECIES). We use the *Password-Based Key Derivation Function 2* (PBKDF2) as a KDF for the key-wrapping key. For key wrapping itself we rely on AES/GCM. Furthermore, we defined IMAP and SMTP commands for sending, fetching, and managing mail as ETDC events.

In essence, users only connect to local services (the proxy services) and the Net Rat abstracts away distribution, synchronisation and message delivery just as an ordinary e-mail server would. The major difference here is that no central instances are involved. Whenever a message addressed to a local recipient arrives at the gateway-enabled device, the message registers as an event and is distributed through the P2P network as shown in Figure 3. Whenever such an event is received by another device it is replicated and transformed back into the original message as shown in Figure 4. Only fetching mails requires for the user to authenticate using an IMAP password, just as it is the case with most traditional IMAP servers.

5 SECURITY EVALUATION

The Net Rat is built on a solid foundation following a security-first approach. Its primary goal is to provide increased data security through decentralisation. To show that this was achieved, we performed a security analysis loosely based on the *Common Criteria for Information Technology Security Evaluation* (International Organization for Standardization, 2014). Due to the flexibility of the Net Rat and the different configurations it can be deployed in, we picked two distinct scenarios for evaluation.

5.1 Threat Model and Assets

We based our threat model on traditional client-server applications typically serving multi-device users. On the one hand, successfully attacking end-user devices can lead to extracting a huge amount of very personal single-user data. Gaining access to central databases of the service back-ends, on the other hand, usually reveals data of many users at once. Therefore, we differentiate between low-volume targeted attacks and high-volume bulk data extraction attacks. The Net Rat is not concerned with protecting users from targeted attacks, since well-crafted scams cannot be countered by technology alone.

The primary asset to be protected is the data processed by services (**A1**) running on top of the Net Rat. All service-related user data stored on disk is considered sensitive and must therefore be protected. The second significant asset which must also be kept secret is the authentication information provided by the user (**A2**). Internally, different services handle data in a multitude of ways, possibly requiring different levels of protection. However, the conservative assumption that all user data and all authentication information needs to be kept secret covers all cases.

5.2 General Assumptions

The main threats the Net Rat seeks to mitigate are those targeted at back-ends. Even though any service run on top of the Net Rat interfaces with the outside world, we show how eliminating central storage can effectively increase data security. We do not, however, deal with threats posed by attackers gaining root access on end-user devices, as this is out of scope. Due to memory isolation techniques employed by virtually all modern operating systems, the Net Rat is also not concerned with defending against threats involving local attackers being able to read the memory of other processes.

For the Net Rat to effectively provide its services the following assumptions must hold:
AS1 Transport security is upheld by the P2P network implementation.
AS2 Client applications only connect to proxies.
As a consequence of these assumptions, transport security becomes a non-issue, even if client applications are not capable of securing connections.
AS3 The Net Rat runs in a sandboxed environment provided by the isolation mechanisms present on modern operating systems.
AS4 Client applications do not store authentication information
Since users are free to continue using the client applications they are familiar with, evaluating the security properties of such software is out of scope. In order to evaluate the Net Rat's potential in this context, it is assumed that end-user applications do not store authentication information. Due to (un-)wrapping decryption keys using authentication information, no authentication secrets can be obtained by an attacker—Asset A1 does not need to be protected as it is never

stored on disk.
Realistically, many users may choose to store authentication information through their client applications. However, scenarios violating Assumption AS4 are considered out of scope.
AS5 Authenticated encryption schemes are used wherever applicable.

These assumptions lead to some profound ramifications. An immediate consequence is the fact that local attacks become the predominant threats to worry about. However, as long as Assumption AS3 is upheld, local attackers need to either overcome OS-level isolation mechanisms or compromise the client application used to interface with the local proxies—both of which is out of scope.

Assumption AS4 is somewhat relaxed: Storing authentication information in the client application poses a threat to Asset A2, however, this has the same consequences as in a traditional, centralised setup. Therefore, the following corollary can be derived:
Corollary 1 *Using the Net Rat with client software, which stores authentication information, has no more severe consequences than using the same client software with an unmodified service.*
In case Assumption AS4 is upheld, stronger security properties can be observed.

5.3 Scenario 1: Baseline

This baseline scenario does not deal with the deployment details of the Net Rat. It is concerned with aspects independent of the setup:
S1 An attacker gains read/write access to the storage of an end-user device running the Net Rat.
Taking into account the fact that data is only stored encrypted, decrypted only on-demand, never persisted in plain, and no authentication information is stored anywhere, the following corollary can be derived:
Corollary 2 *As long as client applications do not store authentication information, even an attacker gaining full access to the device's storage medium can extract neither user data (Asset A1), nor authentication information (Asset A2).*
Combining these observations with Assumption AS5 leads to an even stronger security guarantee:
Corollary 3 *An attacker having full access to the storage medium used by the Net Rat can at most carry out denial-of-service attacks and delete existing data without learning anything about its contents.*
Due to the Net Rat's flexibility, we have analysed two concrete scenarios reflecting different use cases.

5.4 Scenario 2: Using Gateway-only Devices

This scenario maps to the setup depicted in Figure 1. It features two categories of devices configured differently: *End-user devices* featuring local storage and proxy services and *gateway devices* connecting to the outside world and legacy services.
At first, having a single entity interfacing with the Internet does not seem to differ much from traditional client-server setups. In fact, many of the same threats apply. The consequences of attacks, however, differ significantly. Even successful attacks on the Net Rat can only lead to data consumed and produced by a single user. This scenario is defined as follows:
S2 An attacker compromises the *gateway device* and has complete control of it, including full access to all data passing through.
As no more sensitive data is processed on a gateway device than on a traditional server, another corollary can be derived:
Corollary 4 *An attacker controlling a gateway device can do no more harm than an attacker controlling a traditional server.*
However, due to the fact that no data is persisted at a gateway device existing data cannot be extracted, which leads to the following, stronger corollary:
Corollary 5 *An attacker controlling a gateway device can at the very most access data created after the attack. Previously created data is out of reach.*
The simple fact that data is only stored at end-user devices clearly improves data security even without fully decentralising a service. Of course, using a static gateway device leads to some of the same weaknesses present in traditional client-server setups. However, such limitations are not inherent to the Net Rat, but at most mandated to ensure full backwards compatibility. When using the Net Rat with modern services allowing for dynamic gateways, the system's properties change as described in Scenario 3.

5.5 Scenario 3: Symmetric Setup

This scenario consists only of devices featuring all of the Net Rat's components. No static or central gateway is present. Consequently, each device is interfacing with the outside world and becomes a valid target for attacks. More formally, we identified the following attack scenario which shows the consequences of fully decentralising a service and transforming end-user devices into servers:
S3 An attacker gains control of an end-user device and has full control of the Net Rat instance running on this device.

When considering a traditional client-server setup, Scenario S3 can be reduced to an attacker gaining access to a client application in a client-server scenario. This leads to the following corollary:

Corollary 6 *From an individual user's point of view, an attacker gaining control of a Net Rat instance in the context of a symmetric setup can do no more harm than an attacker controlling a client application interfacing with a traditional server.*

It is therefore reasonable to assume that attacks on the Net Rat are infeasible, when it is also possible to attack (unmodified) client applications. In addition, as each Net Rat deployment only processes data of a single user, another observation can be made, which illustrates how our approach protects users from collateral damage:

Corollary 7 *Compromising a Net Rat deployment only affects the data consumed and produced by a single user.*

As a consequence, other users do not suffer collateral damage, when a service run on top of the Net Rat gets compromised. Compared to centralised services housing the data of potentially millions of users, the impact of an unauthorised access is minimal.

5.6 Summary

Given a realistic environment and assumptions based on real-life observations, the Net Rat matches the security of client-server setups (see Corollaries 1, 4, and 6). When taking into account the encryption mechanisms incorporated by the Net Rat, data is protected better compared to traditional systems. Neither user data nor authentication information can be extracted by an attacker gaining full disk access (see Corollaries 2 and 3). The Net Rat also solves transport security issues between client application and service as users only connect to local proxies. This way, clients not capable of securing data in transit do not prevent using state-of-the-art transport security. Furthermore, when deploying dedicated gateway-only devices, it is impossible for an attacker to extract any information previously processed form the gateway as discussed in Corollary 5. Finally, we showed how using the Net Rat over a traditional service can directly prevent collateral damage.

6 RELATED WORK

The Net Rat uses mesh-like structures to organise users' devices by making heavy use of the P2P paradigm. Network-layer approaches to this problem have already been presented and refined over the past decades. Distributed hash tables (DHTs) like *Chord* (Stoica et al., 2001), and *Kademlia* (Maymounkov and Mazières, 2002), for example, provide P2P connections. However, their primary goal is a distributed key-value storage. Typically, a shared global storage is created without control over which nodes are in charge of storing particular pieces of data. The Net Rat, however, requires fine-grained control over which data is routed where.

Another category of related work consists of designs which primarily aim at providing an application-independent network layer. Anonymity networks such as *Tor* (Dingledine et al., 2004) and the *I2P*[4] are examples of concrete implementations of the concept with a strong focus on keeping users' traffic private.

Concepts and applications more closely related to what the Net Rat accomplishes include *Bitmessage*[5], and *FlowingMail*[6]. While these approaches implement decentralised communication frameworks, compatibility with legacy applications is simply not aimed for. A framework for decentralising legacy applications has been proposed by (Hautakorpi et al., 2009). However, it still tries to establish a closed system without supporting a way of integrating the approach with outside legacy infrastructure. Furthermore, being conceived in 2009, it does not reflect today's multi-device users and the capabilities of current devices.

7 CONCLUSIONS AND OUTLOOK

In this work, we introduced the Net Rat to transform existing client-server setups into decentralised applications for increased security and availability. Our approach achieves backwards compatibility both to external legacy services and towards end-user applications. This way, it is possible for users to continue using familiar applications. Due to the modularity and flexibility of our framework, it is possible to deploy it in a heterogeneous set of devices including somewhat static servers and highly mobile smartphones. In short, virtually any service can be transformed to work on top of the Net Rat as long as its operations can be represented as a series of events. In addition, our framework can be used as a base for new services which can be designed with the Net Rat's features in mind. If, for example, two users deploy a de-

[4]https://geti2p.net/en/
[5]https://bitmessage.org/
[6]http://flowingmail.com/

centralised e-mail service on their devices, it is possible to devise next-generation, Net-Rat-aware gateways without having to rely on legacy infrastructure.

An important consideration when transforming applications to end-user devices are security aspects. Firstly, users are effectively burdened with the responsibility of being service operators.This raises the second important issue: Maintaining the integrity of every user's devices is imperative. Therefore, ways of detecting Byzantine faults (Lamport et al., 1982) also need to be considered. Approaches such as *PeerReview* (Haeberlen et al., 2007) deal with this issue, however, the problem definition resulting from the Net Rat's architecture is simpler: Most importantly, the impact of a compromised Net Rat instance is comparable to that of a single rogue user in a traditional, centralised setup.

The security evaluation results illustrate how the Net Rat achieves its security properties and how these differ based on the chosen deployment. As our system enforces encrypted storage and never stores authentication information, even attackers with full disk access cannot extract any information. In combination with a robust network layer taking care of transport security, the Net Rat indeed provides increased security compared to traditional services even for worst-case scenarios. Most importantly, innocent users do not suffer collateral damage from infrastructures outside their control being compromised. Furthermore, due to all data being directly distributed among devices, availability can be guaranteed to some degree even during network downtimes.

Device enrolment has not been considered. However, leaving this issue open for future work does not impact the properties of our system, as soon as devices have been connected. In essence, enrolment is a separate issue. When examining single, multi-device-user scenarios, the burden of performing sophisticated enrolment procedures does not outweigh its benefits.
Another area of future work includes supporting user authentication methods which do not rely on a shared secret like client certificates, or the *Qualified Mobile Server Signature* (Orthacker et al., 2010).
The final area of future work concerns a more sophisticated distributed storage. It must be possible to cater to memory heterogeneity within a set of connected devices for the Net Rat to be applicable to a wide range. However, due to the fact that events are the smallest unit of information, this issue is definitely within reach. After all, event replication already lies at the core of the Net Rat's workflow. Therefore, memory heterogeneity can be catered to by managing events in a clever way. By tweaking the ETDC, this issue can be solved separately from the Net Rat's core functionality much like device enrolment.

The results of our case study and the outcome of the security evaluation clearly show the feasibility of decentralising services. This highlights how existing services can be improved through decentralisation and presents opportunities to develop novel services based on the solid foundation provided by the Net Rat.

REFERENCES

Blum-Dumontet, E. (2017). Defeating encryption: the battle of governments against their people. https://www.privacyinternational.org/node/1427, Retrieved: 2017-04-24.

Cohen, B. (2013). The BitTorrent Protocol Specification. http://www.bittorrent.org/beps/bep_0003.html, Retrieved: 2017-04-24.

Dingledine, R., Mathewson, N., and Syverson, P. (2004). Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*.

Dropbox, Inc. (2016). Dropbox Privacy Policy. https://www.dropbox.com/privacy/, Retrieved: 2016-09-27.

Fiadino, P., Schiavone, M., and Casas, P. (2015). Vivisecting WhatsApp in Cellular Networks: Servers, Flows, and Quality of Experience. In *Traffic Monitoring and Analysis: 7th International Workshop*, pages 49–63. Springer International Publishing, Cham.

Google, Inc. (2014). Google Terms of Service. https://www.google.com/intl/en/policies/terms/, Retrieved: 2017-04-24.

Haeberlen, A., Kouznetsov, P., and Druschel, P. (2007). PeerReview: Practical Accountability for Distributed Systems. In *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles*, SOSP '07, pages 175–188, New York, NY, USA. ACM.

Hautakorpi, J., Camarillo, G., and Lopez, D. (2009). Framework for Decentralizing Legacy Applications. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 544–549, Washington, DC, USA. IEEE Computer Society.

International Organization for Standardization (2014). ISO/IEC 15408-1:2008 Information technology — Security techniques — Evaluation criteria for IT security — Part 1: Introduction and general model.

Lamport, L., Shostak, R., and Pease, M. (1982). The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401.

Maymounkov, P. and Mazières, D. (2002). Kademlia: A peer-to-peer information system based on the xor metric. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 53–65, London, UK. Springer.

Orthacker, C., Centner, M., and Kittl, C. (2010). Qualified Mobile Server Signature. In *Security and Privacy –*

Silver Linings in the Cloud, volume 330 of *IFIP Advances in Information and Communication Technology*, pages 103–111. Springer, Berlin, Germany.

Rosenberg, J. (2010). Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245, Internet Engineering Task Force Request for Comments. http://www.rfc-editor.org/rfc/rfc5245.txt, Retrieved: 2017-03-17.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 149–160, New York, NY, USA. ACM.