Trust as the Elephant in the Room – Security Evaluation of Decentralized Online Social Networks with Mastodon

Lea Laux[1], László Erdődi[2], and Kai Selgrad[3]

[1] OTH Regensburg, Germany
lea.laux@st.oth-regensburg.de
[2] University of Oslo, Norway
laszloe@ifi.uio.no
[3] OTH Regensburg, Germany
kai.selgrad@oth-regensburg.de

Abstract. Federated online social networks are an alternative to centralized and often profit-driven social networks. Instead of providing exactly one main platform, federated and decentralized approaches consist of multiple platforms, nodes or instances, leading to new challenges for guaranteeing confidentiality, integrity and availability. In addition, privacy is taken into close consideration due to the sensitive nature of processed personal data and the purpose of online social networks as well as the user behavior on social media. The recent popularity and broad use of the federated micro-blogging platform Mastodon issues the matter of security and privacy challenges for this type of architecture and the specific platform as well. Mastodon is part of a larger network called Fediverse with several platforms with different purposes. Communication and interoperability between Fediverse platforms is mostly achieved by ActivityPub protocol as standard for decentralized social networking, defined by W3C. We analyze Mastodon as the currently most prominent and largest example of a Fediverse platform. Therefore, we perform tests for typical types of software vulnerabilities as well as evaluate common security challenges built into its design. As a result, we identify trust as security principle as critical issue, leading to multiple weak points such as enabling attackers and malicious actors to spread misleading information as well as network availability impacts. We suggest possible solutions customized to our findings as well as general security recommendations when building a federated online social network such as the Fediverse.

Keywords: Federated Online Social Networks · Security Evaluation · Security Design Review

1 Introduction

In recent years, online social media platforms became a major impact factor on the digital life of every citizen by broad use of those services [3]. Due to the nature and purpose of the platforms, they introduce novel challenges for security

and privacy. They process personal data, initiating the need for proper data protection instead of the need of exposure of sensitive as well as confidential information to others. Centralized architectures lack sufficient mechanisms to protect user data, yet they are processing and selling private information to third parties to earn money out of it, for example by advertisements. In contrast to profit-driven organizations with centralized platforms and the missing awareness for personal data protection, decentralized online social networks claim to return data control to their users, improving privacy as well as security [5,6]. As we discuss in this work, decentralized architectures introduce additional challenges for security as triad of confidentiality, integrity and availability and by processing personal data also privacy. Challenges limited to the architecture are discussed, especially for maintaining availability in the network as well as proper protection of confidential data. The impact and responsibility of node providers is examined further, since they donate a part of their infrastructure to the network. Various architectural approaches for decentralized networks are compared regarding their resource demands and obligation of their users. Examples are given by academic work and popular networks for productive usage as real world implementations. The term of decentralization is hereby used as an umbrella term for all non centralized architectures according to the definition by Narayanan et al. [11] Federated architectures are decentralized by nature and are introduced in detail in Section 2.1 in distinction to peer to peer networks. The main focus of this work specifies federated approaches, while our main assets to protect in terms of security are federated networks as a whole, their single entities as instances as well as users.

The growth factors for decentralized online social networks are evaluated, especially regarding the current issues of popular centralized platforms. An example is given by Twitter, losing a part of its user base after the acquisition of the platform by Elon Musk [16]. This leads directly to the growth of the major federated network of this work: Mastodon as a micro-blogging platform. It serves as the currently most popular Fediverse example, which is introduced in detail in Section 2.2. The results of an evaluation are not only limited to Mastodon, but they are related to the design principles of the Fediverse directly. Therefore, our conclusion includes a general advice for Fediverse platforms and the approach of building decentralized online social networks. We also present a mechanism to build a map of the Fediverse starting with a well known instance with the goal to discover most of the instances by social relations.

Mastodon's security is examined in detail: Its design is taken into close consideration to discover potential entry points for attacks against single nodes as well as the network as a whole. Furthermore, existing vulnerabilities are taken into account to evaluate possible mitigation strategies and robustness of the network against known attacks. Standard penetration testing procedures are performed to evaluate Mastodon's countermeasures against simple attack patterns. The scope of this work includes the software and its design directly, as well as its maintenance by instance providers and several third-party applications, while legal observations are out of our scope. Third-party applications are examined, if

they have an impact on the network, for example by collecting information of the network and publishing it, such as statistical services. They provide a source of information for the network and they are therefore acknowledged and monitored by Mastodon users, potentially leading to misinformation and confusion, if data integrity is impacted. Finally, we present an evaluation of Mastodon's current security state and improvement recommendations based on our work.

2 Related Work

We provide an overview based on decentralized social network architectures, including research prototypes as well platforms, which are productively used by humans. Challenges for security and data storage, transfer and inter-connectivity within a decentralized architecture are discussed. We present several decentralized platforms, until we introduce Fediverse platforms and Mastodon.

2.1 Network Architectures and Implementations

From a historical point of view introduced by Baran [2], there is a distinction between centralized, decentralized and distributed networks. Centralized networks consist of a main node with connections to several other nodes, henceforth referred to as secondary nodes. Decentralized networks consist of multiple main nodes connected to each other, while secondary nodes are connected to a main node. Distributed networks, also referred to as peer to peer networks, consider all nodes as equal, allowing various connections across nodes. Main nodes offer services such as data storage for secondary nodes to provide a coherent network. The characterized architectures are illustrated in Figure 1. An alternate archi-

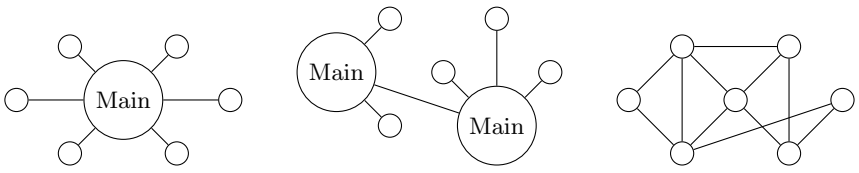

Fig. 1. Various Network Architectures defined by Baran [2] are shown: Centralized – Decentralized – Distributed

tectural distinction is proposed by Narayanan et al. [11]: Decentralization is an umbrella term for all non centralized approaches, while distinguishing between federated and distributed networks. According to Narayanan et al. [11], federated networks share a layer of interoperability, for example by implementing a common protocol. Their federated networks are considered equal to Baran's [2] decentralized networks, while the definition for distributed approaches is similar. As result of their work, we use the term of federated networks and raise awareness for the distinction of architectural types. We consider the variety of features

and add-ons of decentralized networks as reason to introduce a spectrum rather than a binary decision.

Furthermore, there are several academic examples for decentralized online social networks. We present a few of them as an overview of architectural characteristics, possible challenges, arising pitfalls and issues for security. PeerSoN [5] is a distributed network, supporting privacy and data control of users. Buchegger et al. design a two tier architecture with a lookup service and peers with user data. Their lookup service provides information about peers for initiating connections without storing any sensitive information except for an encrypted message storage. The open challenges of Buchegger et al. include long-term durability of data and network availability. They raise attention to DoS attacks and mitigate them by proposing a handshake before exchanging large data amounts. Safebook [6] is motivated by the assurance of integrity and availability and possible threats against privacy. It is a peer to peer network with additional storage nodes for public data. Cutillo et al. identify attack scenarios based on known issues of centralized architectures, including, but not limited to spoofing and impersonation, threats to privacy established by users and social network providers as well as DoS attacks and content censorship by providers. SuperNova [13] has a federated architecture with privileged and resourceful nodes. Sharma and Datta [13] use storage and data replication to ensure availability. In addition to super nodes, friend nodes as direct peers and stranger nodes are able to donate a part of their resources to the network. By distinction between node types, trust to others is introduced as design property. The main goal of SOUP [10] is to achieve a level of availability similar to centralized networks. High availability and network scalability with minimal replication of user data is a priority. Permanent online storage of data by others is excluded, while every node selects a small set of others as data mirror. Possible attack scenarios include malicious nodes abusing the trust within the network as well as flooding attacks.

In addition to the hereby presented approaches of decentralized online social networks, there are several more similar prototypes. As examples with special properties serve Cadros [8] by implementing cloud services for data storage and BCOSN [9] by using blockchain technologies for ensuring data integrity. To summarize the findings and insights of presented networks, categories of Koll et al. [10] and their network SOUP are examined.

- Peer to peer nodes with sufficient resources: All nodes have sufficient resources to ensure a permanent availability. Users of the network have to provide the necessary infrastructure, resulting in a loss of network usability.
- Super nodes: Privileged and resourceful nodes, such as the architecture of SuperNova [13] requires, are introduced. They are permanently online, depending on either paid services or donors. Trust to the responsible maintainers of super nodes is necessary.
- Temporary node cooperation: Nodes cooperate temporarily, which is also the approach of SOUP [10]. Direct peers are preferred for cooperation: There is already a certain level of trust between the involved nodes.

To our observations, the design phase often starts with a peer to peer architecture. Purely peer to peer architectures are not able to ensure a sufficient level of availability, hence, federated elements are introduced. In comparison to centralized networks, users of decentralized online social networks are required to maintain at least a part of the network, persisting in their own node. Depending on the implementation, it could be necessary to trust other entities and to take responsibility for own data and data of others.

2.2 Fediverse

An ecosystem of federated online social networks is the so called Fediverse. Its name is a portmanteau term of the two words federation and universe. Interaction between various platforms is established by common protocols [1]. For a large and hereby considered most relevant part of the Fediverse, the protocol is ActivityPub, recognized as decentralized social networking standard by the W3C [14]. It provides an API for clients and servers, manifesting the federated architecture of Fediverse platforms. Clients communicate with a server, while servers communicate with each other. Communication between clients is achieved by communication through servers. Every service implementing ActivityPub can participate in the Fediverse, lowering the barrier to entry.

Applying the protocol to services results in an ecosystem of multiple platforms for various purposes, such as Pixelfed for sharing photos or PeerTube for sharing videos. Additional projects as third-party services are built around the Fediverse, such as statistical resources, for example The Federation[4], It supplies a statistics hub to track the state of the Fediverse with its platforms, protocols and user activity.

ActivityPub takes security considerations into account by addressing various security-related topics and possible attack scenarios [14]. A requirement is the implementation of authorization and authentication measures. OAuth 2.0 is recommended as authentication mechanisms for client to server communication and HTTP Signatures for server to server communication. Extensions with Linked Data Signatures are recommended for objects likely to be shared and federated. Requests to localhost could result in unauthorized local machine access. ActivityPub handles with URIs, requiring sanitization and validation before processing. Limiting recursion depth for object references is suggested as prevention of DoS and spam attacks. Developers should implement filters for incoming content, tailored to the level of trust. Extensive submission mechanisms should be avoided to not overload other servers accidentally. Sanitanization of content applies in general, especially when displayed to a user.

2.3 Mastodon

Mastodon is a micro-blogging service and part of the Fediverse. According to its user number, it is currently the largest Fediverse platform. The number of

[4] https://the-federation.info

users as well as nodes can be found in Figure 2 to estimate the growth of the network. An important growth factor in 2022 was Elon Musk's acquisition of

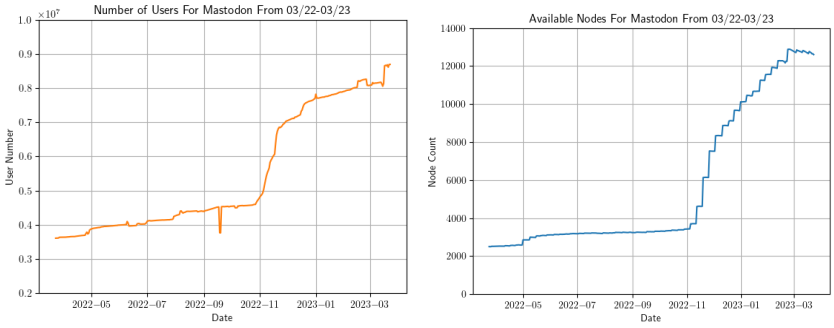

Fig. 2. Two graphs display the number of Mastodon users and the number of available nodes between March 2022 and March 2023 based on the statistics of https://the-federation.info/.

the micro-blogging service Twitter, examined by Zia et al. [16]. User number and activity on Mastodon increased significantly, while most of the users of the top 30 Mastodon instances joined after 26th October 2022. Furthermore, several communities such as the scholarly community, show migration trends towards Mastodon [4]. As a result, we consider Mastodon as currently most important Fediverse platform and as research example for our security evaluation. Nevertheless, we raise attention to the fact that the Fediverse and Mastodon as social media are not critical infrastructure, but a digital place for social relations.

The entry barrier to become an instance provider is low: Everyone with the required computational resources and knowledge to understand the installation manual is able to maintain an instance. Resources can be their own or supplied by an external entity, such as cloud service providers. Mastodon is mostly run by volunteers without any official qualification, which is an advantage for the openness and usability of the platform. On the other side, the competence, or at least its validation, is missing compared to a system mostly run by professionals or a company as well as compliance checks by official authorities.

3 Reconnaissance in the Fediverse

The Fediverse consists of multiple platforms and various instances. Security flaws can potentially affect them all, while testing for vulnerabilities requires a list of available instances. Discovering Fediverse instances is not trivial due to its federated design. The completeness of a list is limited by graph connectivity: The set of known instances is only consisting of instances directly or indirectly connected to the origin. In addition, an instance could exist, but is offline during

a search. The goal is to at least cover a large part of the Fediverse, which is productively used by humans. Due to the limitation of this work to Mastodon, an active scan is performed based on Mastodon instances.

Every instance can publish data about its peers, announcing them publicly as default. This information is accessible by Mastodon's instance API. The peer list consists lifetime peers, hence, it can contain (permanently) offline instances. A current overview of the network can be created by visiting the peers of instances, starting with a large and well-known instance, for example mastodon.social.

```
starter_instance = large_instance
known_instances = []
unvisited_instances = [starter_instance]
while unvisited_instances:
    instance = unvisited_instances[0]
    known_instances.add(instance)
    peers = instance.get_peers()
    unvisited_instances.add_unknown_peers(peers)
    unvisited_instances.remove(instance)
return known_instances
```

Further instance information can be retrieved by its API to determine, if a Mastodon instance is detected and to exclude other ActivityPub service. With this approach, we are able to discover roughly 13.000 federated instances in approximately two hours, depending on computational resources and mainly internet connectivity.

4 Mastodon Design and Consequences

We evaluate Mastodon's design regarding security and privacy properties to find attack surfaces and entry points for exploits. Our attacker profiles include malicious external entities against the Fediverse as a whole as well as against single instances. Furthermore, we analyze the possibility of malicious instance providers. Mastodon as social media platform is a digital social space and not critical infrastructure. Therefore, it is necessary to provide a certain level of security assurance, but the interest for an attacker to breach security is assumed lower than for other web applications, since Mastodon is compared to other social media applications in general a smaller approach. Breaking Mastodon's security by using large amounts of time or computational resources is therefore considered rather unattractive. We introduce the structure of data storage and flow, functionality as well as responsibilities across the network. The findings are consequences of Mastodon's architecture as federated social network or strategies to deal with arising challenges. Therefore, the hereby presented topics do not apply to centralized approaches.

4.1 Trust to Instance Providers

The major Fediverse entity is an instance, maintained by their provider. Instance providers offer a platform for social relations by supplying the infrastructure for

content creation and user interactions. Further functionality include data storage and federation with others. Users have to establish a certain level of trust to their instance provider as they are the responsible entity for functionality and data. Since user data is stored at the instance directly, the provider can control it. Most of the data is unencrypted expect for credentials such as passwords. Instance providers are also able to limit and suspend network traffic to other instances. As a consequence, mature users, understanding Mastodon's design capabilities, are required. The skill to choose an instance with a trustworthy provider and to adapt their usage behavior accordingly to Mastodon's design is recommended from our point of view. Compared to a centralized approach, users are required to evaluate the trustworthiness of the responsible organization. Compliance checks regarding laws and regulations are stricter in comparison to a small Fediverse instance run by volunteers.

An additional consideration is mutual trust across the Fediverse and its instances. Validation and verification mechanisms are not necessarily implemented for all types of features. An example is information spread by instance API, such as (active) user numbers: There is no further validation. Peers are propagated through the network, while instances exchange their peer lists. If a previously not known instance of the peer list responds with correct ActivityPub syntax after probing, it is considered a valid peer now. It is appended to the lifetime list of peers, which are not regularly validated, for example after a defined period of time. The general level of trust applies to Fediverse platforms in general, not only Mastodon instances.

4.2 Availability of Instances and Federation of Data

Permanent availability of instances cannot be guaranteed, hence, Mastodon implements mechanisms to ensure a high level of data availability. A Mastodon instance collects frequently data of its peers, such as user posts, storing it in the instance database. If a peer experiences a downtime, peer data is still accessible for users of the original instance.

The mechanism introduces a challenge for data integrity. If data changes after initial storing, an update process for collected data is necessary. Mastodon's current implementation is not able to track which instance has stored which content with timestamps. Therefore, data currentness is impacted by its federated architecture. An impact on user privacy guaranteed by GDPR [7] could arise, since a user has the right to have their personal data deleted on request. A possibility to mitigate the issue could be the implementation of a data request every time when necessary instead of storing it. Although, this would introduce new challenges for availability by causing significantly more traffic. A request on access mechanism without caching could overload instances with few resources, resulting in DoS by accident.

Another consequence of federated data structures is the missing logging and monitoring functionality. Atypical behavior against the Fediverse as single entity cannot be detected fast. Instances can still detect local anomalies, but the

information is not merged and evaluated centrally compared to a centralized platform.

4.3 Broad Compatibility of ActivityPub Services

In general, the Fediverse is mostly using a common protocol, ActivityPub, allowing interoperability across platforms. As long as a software project supports ActivityPub, it can participate actively in the Fediverse. Malicious services are not excluded, because ActivityPub is simply a protocol, such as SMTP for mail. Nevertheless, a certain awareness is required for incoming traffic and outgoing information, relevant for federated networks in general. A protocol standard can suggest security mechanisms for communication, for example by directly implementing strong authentication and authorization principles as well as content sanitanization.

5 Vulnerabilities and Exploits of Mastodon

We present carefully selected vulnerabilities and exploits of Mastodon as a result of our security evaluation. Our selection is not complete due to the broadness of vulnerability types and their relevance. There are several (fixed) vulnerabilities available in related databases with various severeness, such as the National Vulnerability Database provided by NIST[5]. Furthermore, our findings include attack patterns based on a design evaluation that not necessarily lead to vulnerabilities, but to possibilities to abuse responsibilities. By abusing them, it is possible to break confidentiality, integrity and availability for a sub set of users. Our main entities and assets regarding security are the Fediverse network as a whole, single instances as well as users, including their fundamental right of privacy, which is also part of GDPR [7] as privacy regulation within the EEA. Therefore, we describe our choice of reproduced and evaluated vulnerabilities with a current and possible impact on single Mastodon instances or the Fediverse in general.

5.1 Denial of Service by Large Sidekiq Queue Generated by Bot Account Follows

This attack type impacted the Fediverse platform Misskey[6] initially. It is also applicable to Mastodon: The attack relies on producing a large amount of traffic by pull queues for Sidekiq [15,12]. To execute the attack, accounts on a victim instance are created. The accounts follow other accounts on an attack server with a wildcard DNS record to mimic a large number of unique accounts on multiple separate instances. From the viewpoint of a victim instance, it appears to be various other instances instead of only one. The attack server responds with correct ActivityPub syntax to mimic a valid ActivityPub platform. Therefore,

[5] https://nvd.nist.gov/vuln
[6] https://misskey-hub.net/en/

any kind of implementation, such as a static page returning the expected JSON object, is sufficient. The attack server hosts a single user, hence, the accounts on the victim instance can follow the user to achieve federation between the involved instances. Based on federation, the attack server is frequently accessed to index new data. This behavior is a mitigation strategy of Fediverse services related to limited instance availability: They store the data of others to ensure data availability for their own users. If the attack server is not available, the victim instance starts to retry its pull attempts such as indexing new data, resulting in multiple large Sidekiq queries. The behavior is impacting resources and from a long term perspective availability of the victim instance. The attack

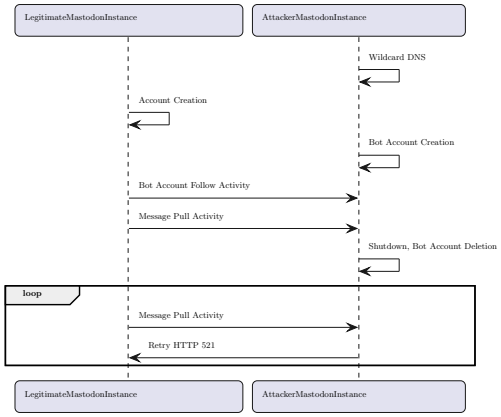

Fig. 3. The bot attack based on an existing vulnerability [15,12] including Wildcard DNS and retry queues is illustrated.

scenario in Figure 3 presumes an attacker's access to an account on the victim instance. This is trivial for an instance with open registrations, which applies to the majority of Mastodon instances.

We do not only reproduce the initial CVE description, we extend it: Fediverse instances start to federate with others, if they become aware of them, for example by exchanging peer lists. Therefore, it is sufficient to be successfully probed by a victim instance at least once as initial peering in contrast to the CVE description refering to follow relations as necessary requirement to execute the attack. During the attack, the attack server is probed by peers of the victim instance, increasing its blast radius. To reproduce and observe the vulnerability properly, we bypass the API limit for follow relations and insert them directly into the database. Due to the insufficient description of the vulnerability, we affect the productive Fediverse, leading to complaints within the network. Our impact is considered low, since we are not able to detect any availability issues such as offline instances. However, it is possible to execute larger attack attempts

with multiple attack servers with wildcard DNS, various victim instances and installing malicious instances over a certain amount of time in the Fediverse.

Overall, there is no official software fix by Mastodon, but they claim to have explicitly mitigated the exploit possibility. The security team of Mastodon was contacted with the purpose to raise awareness to the extended attack vector. As their answer points out, there is a limit for fetching accounts and posts with a single request, implementing a kind of mitigation mechanism. Instance providers can block attacker instances, if they notice a large amount of traffic and resource usage.

5.2 Misconfiguration of Object Storage Domain

According to a mail from mastodon.social to their users in March, the object storage domain of the instance was misconfigured with the consequence of world-readability of the related directory. File access by anyone with knowledge of the domain was possible. The provider clarifies that most of the data is publicly available anyway, except for user archive data. User archives include their public profile, their favorites and bookmarks as well as their posts, potentially with media attachments. This does also apply to content with restricted visibility and therefore non public data. The misconfiguration was exploitable during February 2023. They also claim to automatically check other instances now and inform them in case of an issue.

`files.<instance.domain>` and `<instance.domain>/system` are known object storage domains, therefore, it is possible to analyze the previously created list of instances for known security bugs. S3 buckets can be misconfigured to have the 1000 first files readable. Buckets in general can be misconfigured to serve all files with a continuation token, usable with a query parameter. Independently of the underlying storage solution, the `system` directory can be world-readable. According to our findings, only a minor percentage of the overall Mastodon network is affected, while the largest Mastodon instance, mastodon.social, was vulnerable to this misconfiguration. To our knowledge, 54 instances, including mastodon.social, were affected, resulting in potentially 997.000 affected users (approximately) as a worst case estimation.

5.3 Direct Messages of Users – Confidentiality Impact by Instance Provider

An instance provider is able to access all instance data. Content is stored without any kind of encryption, including posts with restricted visibility and direct messages as posts with limited visibility to the conversation's participants. Therefore, an instance provider is able to access private messages of arbitrary users. This is a break of confidentiality, hardly detectable for a user. A recommendation for users is to not exchange private messages with confidential content in Mastodon, but to switch to another service, preferably with encrypted messages. The attacker profile is a malicious instance provider, collecting and analyzing social data, which can be abused for social engineering attempts. Furthermore, instances are often

based on specific topics or provided for certain communities. Confidential information can be misused to harm individuals within a community. Even though an instance provider might not have a malicious intent, any attacker with database access is able to break the confidentiality of private messages. This exploit possibility illustrates the necessity for proper protection mechanisms of private data. The underlying property is the assumed trust to instance providers. Users have to be aware of the circumstance that all their Mastodon data does not meet the requirement of full confidentiality and assume it is potentially publicly available in a worst case.

5.4 Fake Mastodon Statistics

Several services publish statistics about Mastodon, based on data propagated by instances. A relevant metric is the (active) user number to evaluate Mastodon's growth and its activity. Statistics services trust the information an instance publishes, since Mastodon directly does not implement mechanisms to verify data correctness.

Mastodon's mutual trust can be misused to affect statistics, for example by increasing the number of users. We deliver 300.000 users for a research instance, which is accepted by several statistical sources, such as the-federation.info, without any further validation. However, we are not able to influence the official statistics of Mastodon nor other services such as a popular user count bot[7]. Conclusively, there is some kind of validation in place. We assume that multiple instances with a more organic looking user growth might be able to influence additional statistical services. We launch multiple instances comparable to our attack server already used during the reproduction of the DoS attack in Section 5.1. To achieve an impact, we increase the user count of each instance by 500 users per 15 minutes. Again, we do not affect official statistics of Mastodon,

Fig. 4. The user count bot is monitored during the fake statistics attack: https://mastodon.social/@mastodonusercount/110536086578315894 before starting the increase, https://mastodon.social/@mastodonusercount/110539860472044294 during the increase and https://mastodon.social/@mastodonusercount/110602145657111403 after shut down

[7] https://mastodon.social/@mastodonusercount

but the user count bot, displayed in Figure 4. The hourly user increase changes to roughly 3.000 new users per hour and drops to roughly 500 users per hour after swapping to a static user number of 42 users per instance. Since the bot is relatively popular, there are several reactions, trying to explain the fluctuations. Third-party services are considered a trustworthy source by several Fediverse users. Therefore, the reputation of statistical services and Mastodon are affected by this type of attack. If Mastodon appears to be larger than it actually is, it is an impact on data integrity of the project and might mislead to false conclusions as well as overall confusion.

5.5 Fake Likes for Posts

Mastodon as micro-blogging platform provides a like functionality for content. The number of favorites as well as the list of who marked a post as favorite is collected by the instance of the original post. Hence, a complete list of persons liking a post is only available at the original instance. Accessing the post from another instance, the persons liking the post from the same instance of the user are displayed. An instance provider is able to add arbitrary likes to a post, caused

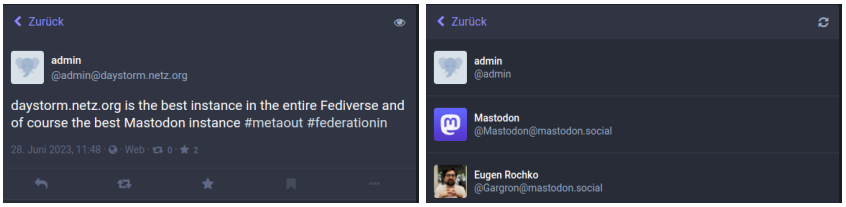

Fig. 5. A fake like is demonstrated by an example post and the list of likers, containing Eugen Rochko as main developer of Mastodon as well as the official Mastodon account, available at https://daystorm.netz.org/@admin/110621211857643127

by missing validation mechanisms for likes. An abuse example is illustrated in Figure 5. This is especially critical for controversial posts, affecting integrity of data. An example of a threat is a scenario of political posts: If controversial politicians and their parties run instances controlled by them, they are able to fake as many likes as they want, potentially confusing people during elections and spreading fake news. This is rather a social issue than a technical one. As a countermeasure, it would be possible to implement a verification mechanism, such as it is available for posts and replies in general. The mitigation strategy might come with a price for availability and general data traffic in the Fediverse: The signature of every like would need to be verify at least two instances.

5.6 Failed Attempts

In addition to our successful attacks and reproductions, we try several standard techniques of penetration testing. To display Mastodon's robustness against

them, we summarize our failed attack attempts. As a social network, Mastodon provides several input fields for user-controlled content. Input fields apply to classical features of social networks in general, such as the one for posting new content. There are several more, for example in settings and moderation tools, taking Mastodon's role concept of administrator, moderator and normal users into account. Tests for SQL injections and XSS in all interfaces are unsuccessful. They are tested with manual tries in the web interface as well as hand-crafted and automated HTTP (POST) requests. In addition, directory scans do not reveal any further information to misuse. Attempts to break API authentication to bypass Mastodon's role concepts and visibility of content is not successful. Furthermore, basic attempts to find and abuse existing session cookies, such as accessing session storage and API endpoints, remain unsuccessful. Our attack to bruteforce registration links for instances with account registration limitation is unsuccessful due to the amount of possibilities with the pattern `/invite/[a-zA-Z0-9]^8`. Further simple exploit possibilities are not discovered by using standard penetration testing techniques. As a consequence, attacks need to be more sophisticated than standard procedures, for example by abusing multiple weaknesses in the Mastodon technology stack, including infrastructure as well as supporting applications and their weaknesses. The task to detect further vulnerabilities in Mastodon requires technical knowledge, penetration testing experience and a deep understanding of either web applications or Fediverse and ActivityPub architecture. In addition, we assume that the task is time consuming in a manner that it would not be beneficial for an attacker regarding the current state of Mastodon as social media network. Furthermore, we asked the security team of Mastodon for permission to start a phishing attack to include a social engineering aspect. Permission was denied, but we mention it here as a potential attack vector.

6 Conclusion

We give an architectural overview of online social networks and focus on federated platforms. Based decentralized architectures, we analyze existing academic approaches and observe their properties, challenges and impact on security. By introducing the Fediverse, we concentrate on federated architectures. Mastodon serves as prominent platform with ActivityPub as protocol. To evaluate Mastodon's general state of security, we identify design challenges, entry points for attacks and possible exploits. Our exploits are either reproduced based on previously published vulnerabilities or results of our design evaluation. Our findings can still impact Mastodon and the Fediverse. Furthermore, we use standard procedures of penetration testing to show Mastodon's robustness against simple attack patterns.

The most critical design issue is the high level of trust established between Fediverse entities. In general, peers of an instance are considered trustworthy, allowing broad communication with low or missing validation and verification mechanisms. An example is the frequent exchange of peer lists: The only val-

idation mechanism is the delivery of correct ActivityPub syntax after initial instance probing. Malicious intents are not anticipated, therefore sanity checks are missing. Information given by an instance, including but not limited to instance statistics, are not validated further. Validation functionality could restrict ActivityPub compliance, requiring to maintain trust as major design decision or to implement well-designed solutions. Examples of trust abuse are given by DoS attacks and faked statistics. The current mitigation strategy is only reactive instead of proactive by blocking malicious instances. Therefore, it would be advisable to balance out the two considerations and to develop components to fit them both: Validation and verification mechanisms for information from others and ActivityPub compliance. Future work can include further protocol testing directly.

A user's point of view shows the necessity of trust to their instance provider. Providers have full control over their instance data and therefore the ability to abuse their power and responsibility. An example is given by breaking confidentiality of private messages, therefore the level of trust to an instance provider needs to be high. Users should have a certain awareness of this circumstance. It would be beneficial for their privacy and security, if they are mature and well-educated, understanding the consequences of Mastodon's design for their individual use. Users uncomfortable with the required trust to instance providers can host their own instance, optionally enabling single user mode. In addition, Mastodon provides a functionality to relocate accounts to another instance. Nevertheless, content with limited visibility is stored unencrypted in the database of related instance peers.

Overall, considering the current growth of Mastodon and the Fediverse, further functionality and mechanisms to decrease the necessity of trust are required to prevent its abuse. Reviews of design and of the current implementation can help to increase security by validation and verification of external information. In addition, the power of instance providers is considered particularly high. Lowering their power is advisable, for example by implementing mechanisms to demonstrate users that their data cannot be abused or is less likely to be abused. A possible solution is the introduction of verified builds, reducing the risk of secretly implemented backdoors, as well as message encryption or the recommendation to use external services for messaging. In general, it is open to debate how large federated online social networks can and should grow depending on the current level of trust. With a growing network, there is a higher probability to introduce users that are less mature and less educated about the functionality and design of the network they choose. Accordingly, instances and their providers are in the need of stronger defenses. The maturity and competence of instance providers would need to grow as well as the popularity of the platform. Therefore, an open question is how large can a network reasonably grow and which amount of trust between its various entities is appropriate. A more general question is the user base Mastodon and the Fediverse want to attract and to integrate, including considerations about their maturity and usage behavior.

References

1. Anderlini, J., Milani, C.: Emerging Forms of Sociotechnical Organisation: The Case of the Fediverse, pp. 167–181. University of Westminster Press (11 2022). https://doi.org/10.16997/book54.m
2. Baran, P.: On distributed communications: I. introduction to distributed communications networks. Tech. rep., RAND CORP SANTA MONICA CALIF (1964)
3. Bell, G.: Building Social Web Applications: Establishing Community at the Heart of Your Site. O'Reilly Media (2009)
4. Brembs, B., Lenardic, A., Murray-Rust, P., Chan, L., Irawan, D.E.: Mastodon over mammon: towards publicly owned scholarly knowledge. Royal Society Open Science **10**(7), 230207 (2023). https://doi.org/10.1098/rsos.230207
5. Buchegger, S., Schiöberg, D., Vu, L.H., Datta, A.: Peerson: P2p social networking: Early experiences and insights. In: Proceedings of the Second ACM EuroSys Workshop on Social Network Systems. p. 46–52. Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1578002.1578010
6. Cutillo, L.A., Molva, R., Strufe, T.: Safebook: A privacy-preserving online social network leveraging on real-life trust. IEEE Communications Magazine **47**(12), 94–101 (2009). https://doi.org/10.1109/MCOM.2009.5350374
7. European Parliament, Council of the European Union: Regulation (EU) 2016/679 of the European Parliament and of the Council, https://data.europa.eu/eli/reg/2016/679/oj
8. Fu, S., He, L., Liao, X., Huang, C., Li, K., Chang, C., Gao, B.: Cadros: The cloud-assisted data replication in decentralized online social networks. In: 2014 IEEE International Conference on Services Computing. pp. 43–50 (2014). https://doi.org/10.1109/SCC.2014.15
9. Jiang, L., Zhang, X.: Bcosn: A blockchain-based decentralized online social network. IEEE Transactions on Computational Social Systems **6**(6), 1454–1466 (2019). https://doi.org/10.1109/TCSS.2019.2941650
10. Koll, D., Li, J., Fu, X.: Soup: An online social network by the people, for the people. In: Proceedings of the 15th International Middleware Conference. p. 193–204. Association for Computing Machinery (2014). https://doi.org/10.1145/2663165.2663324
11. Narayanan, A., Toubiana, V., Barocas, S., Nissenbaum, H., Boneh, D.: A critical look at decentralized personal data architectures (2012). https://doi.org/10.48550/ARXIV.1202.4503
12. National Institute of Standards and Technology: Cve-2022-46405 detail, https://nvd.nist.gov/vuln/detail/CVE-2022-46405
13. Sharma, R., Datta, A.: Supernova: Super-peers based architecture for decentralized online social networks. In: 2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012). pp. 1–10 (2012). https://doi.org/10.1109/COMSNETS.2012.6151349
14. Social Web Working Group of WC3: Activitypub, https://www.w3.org/TR/2018/REC-activitypub-20180123/
15. @za6-weO-Qp2hnlXTYhjdqw Tani: Context attack on instances by troll accounts, https://hackmd.io/rD9nsTz1QeuPT-erxqjY-A
16. Zia, H.B., He, J., Raman, A., Castro, I., Sastry, N., Tyson, G.: Flocking to mastodon: Tracking the great twitter migration (2023). https://doi.org/10.48550/arXiv.2302.14294