Aalto University

OPEN ACCESS

This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.

Bui, Thanh; Rao, Siddharth; Antikainen, Markku; Aura, Tuomas

Pitfalls of open architecture

Published in:
Proceedings of the 12th European Workshop on Systems Security, EuroSec 2019

DOI:
10.1145/3301417.3312495

Published: 25/03/2019

Document Version
Early version, also known as pre-print

Please cite the original version:
Bui, T., Rao, S., Antikainen, M., & Aura, T. (2019). Pitfalls of open architecture: How friends can exploit your cryptocurrency wallet. In *Proceedings of the 12th European Workshop on Systems Security, EuroSec 2019* (pp. 6). [3] ACM. https://doi.org/10.1145/3301417.3312495

This material is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of the repository collections is not permitted, except that material may be duplicated by you for your research use or educational purposes in electronic or print form. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone who is not an authorised user.

Pitfalls of open architecture: How friends can exploit your cryptocurrency wallet

Submitted to EuroSec'19

ABSTRACT

Many cryptocurrency wallet applications on desktop provide an open remote procedure call (RPC) interface that other blockchain-based applications can use to access their functionality. This paper studies the security of the RPC interface in several cryptocurrency wallets. We find that, in many cases, a malicious process running on the computer regardless of its privileges can impersonate the communication endpoints of the RPC channel and, effectively, steal the funds in the wallet. The attacks are closely related to server and client impersonation on computer networks but occur inside the computer. The malicious process may be created by another authenticated but unprivileged user on the same computer or even by the guest user. The main contribution of this paper is to raise awareness among wallet developers about the need to protect local RPC channels with the same prudence as network connections. We also hope that it will discourage users to run security-critical applications like cryptocurrency wallets on shared systems or computers with guest account enabled.

CCS CONCEPTS

• **Security and privacy** → *Authentication*; *Software security engineering*.

KEYWORDS

Cryptocurrencies, Remote Procedure Call (RPC), Impersonation

1 INTRODUCTION

Bitcoin [45] and other cryptocurrencies have become remarkably popular in the last decade. In cryptocurrencies, transactions happen between public keys and are communicated as signed messages in an open peer-to-peer (P2P) network. The transactions are collected into blocks, which are appended into a structure, called the blockchain. The global consistency of the blockchain is guaranteed with a consensus mechanism, such as proof-of-work [45], proof-of-stake [35], or a Byzantine fault-tolerant (BFT) variant [52].

Users access a cryptocurrency with a wallet, which manages the private/public key-pairs that are used for transactions and allows the user to store, send and receive the cryptocurrency. There are various types of cryptocurrency wallets: online, mobile, desktop, paper, and hardware wallets, each of which has its own pros and cons [32]. This paper focuses solely on *desktop wallet applications*. For simplicity, in the rest of the paper, we will use *wallet applications* to refer to desktop wallet applications.

Desktop wallet applications often provide a *remote procedure call (RPC) interface*, through which other applications can access the wallet's functionality either locally or remotely. While this open interface enables easy development of other blockchain-based applications, it increases the attack surface of the wallet. In fact, there have been reports on attacks where cryptocurrencies have been stolen from wallets by exploiting the remotely-accessible RPC interface where authentication has not been properly configured [55]. The primary solution for protecting the RPC interface against such attacks has been to block remote access to the interface so that only local processes on the computer can access it. Furthermore, the wallet applications usually require password authentication when accessing the RPC interface. However, while these security mechanisms may help against remote attacks, their effectiveness against local threats has not been considered.

This paper studies the security of the RPC interface of the wallet applications in the presence of a local attacker. Our main contribution is to show vulnerabilities in popular cryptocurrency wallets, which allow *nonprivileged processes that belong to other users on the same computer* to exploit their RPC channel and steal the wallet content. We also show that the currently used authentication mechanisms on the channel are not effective and discuss potential mitigation techniques.

The rest of the paper is organized as follows: Section 2 describes the background information on cryptocurrency wallet applications and our threat model. Section 3 shows our attacks to the RPC interface of various wallet applications, and Section 4 presents mitigation solutions to the attacks. We have reported all the vulnerabilities that we found to the corresponding vendors and received responses from Parity [21] and Bitcoin Core [5] teams. We include these responses in Section 5. Section 6 reviews the related prior research and Section 7 concludes the paper.

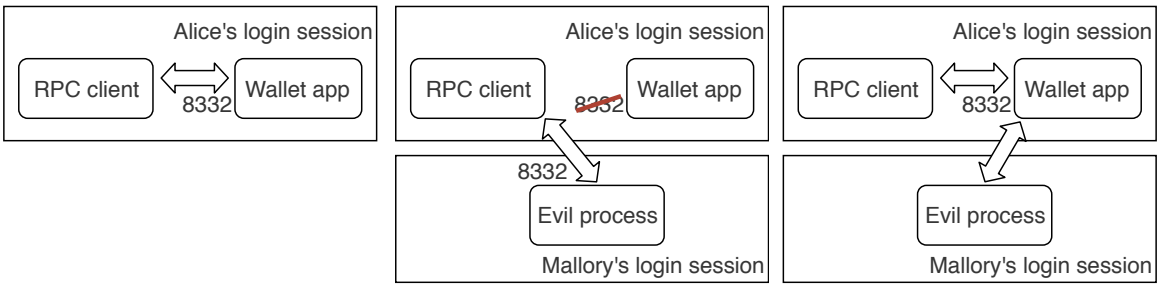
2 BACKGROUND

This section describes how a typical cryptocurrency wallet application works as well as explaining our threat model.

2.1 Cryptocurrency wallets

Most cryptocurrencies have one "official" wallet applications and several recommended third-party applications. A wallet application typically provides a command-line or graphical user interface for the users to manage their cryptocurrency *accounts*. Each account is basically a private/public key-pair that is used for transactions and usually encrypted with a password. (Note that in some places "wallet" is used to refer to such key pair. We chose to use "account" instead to easy differentiation between key pairs and wallet applications.) The user must enter the password to unlock the account before it can be used.

Most wallet applications provide a remote procedure call (RPC) interface with a JSON-RPC over HTTP server [40] that runs on a specific port number on the `localhost`. Other applications can connect to the RPC server as clients to access the wallet's functionality, such as querying account balance or making transactions. Examples of such RPC client applications include web-browser extensions [17], third-party wallet applications that do not want to implement the cryptocurrency protocol by themselves [4], and cryptocurrency



(a) Normal operation (b) Server impersonation (c) Client impersonation

Figure 1: Attacks on the RPC channel of a wallet application by a nonprivileged user

exchange web platforms [3, 22]. While the number of RPC client applications is still fairly low, the availability of open RPC interfaces and client libraries indicate that the wallet developers expect the ecosystem of such add-on software to grow in the future.

2.2 Threat model

In this paper, we consider threats on *multi-user computers* that have processes belonging to two or more users running at the same time. The attacker is a logged-in user who tries to steal cryptocurrency from the wallet of another logged-in user of the same computer. The attacker does this by impersonating the communication endpoints of the wallet's RPC channel, as illustrated in Figure 1. Unlike malware that runs with the victim's privileges or as administrator, the attacker here is *nonprivileged*, and we do not assume that he can perform any kind of privilege escalation.

To exploit the RPC channel, the attacker needs to run a process in the background when the victim is using the computer. On Linux and macOS, the attacker only needs to log in, run the process, and leave it running when he logs out, e.g. with the `nohup` command. On Windows, user processes are killed at the end of the login session, and thus the attacker needs to do *fast user switching* [43] to leave his session in the background. The attacker can also remotely run his malicious process if SSH or remote desktop is enabled on the target computer.

While PC is often considered personal, it is relatively common that several people can access it. For example, in enterprise environments that use centralized access control, users are typically able to sign in to each other's workstations. Shared family computers with multiple user accounts and machines with a guest account enabled are similarly vulnerable.

3 ATTACKS ON WALLET APPLICATIONS

We analyzed the wallet applications of several popular cryptocurrencies. While the applications have taken some measures to protect their RPC interface, we will show through various case studies that the current protections are ineffective against malicious local processes. Table 1 summarizes the vulnerable wallet applications and RPC clients, including blockchain-based applications and RPC client libraries, that we found. We expect that similar vulnerabilities will be discovered over time in other wallet applications with an open RPC interface.

3.1 Case 1: Wallets with no authentication

Ethereum [56] has two official command-line wallets: Go-Ethereum (Geth) [14] and CPP [9], out of which only the former supports the RPC interface. If the user enables the RPC interface, Geth runs the RPC server on port 8545 of the `localhost` by default so that only local clients can access it. Geth does not require any authentication mechanism on its RPC interface, and it does not support TLS. If the user wants to access the interface over TLS, he/she must configure a TLS proxy on the same host, which relays requests to the RPC interface. The only security mechanism that Geth provides is allowing the user to configure *cross-origin resource sharing (CORS)* [41] to authorize specific web pages that are open in the user's browser to access the wallet.

Attacks. Since there is no authentication between the RPC client and the RPC server, a malicious nonprivileged process can impersonate the client by simply connecting to the server. CORS is clearly not an effective mitigation solution here because the attacker can set the `Origin` header of its HTTP requests to anything. While an account could be encrypted with a password, Geth allows the user to unlock the account when starting it. This allows the malicious client to access the account without knowing the password and, for example, sends the victim's funds to the attacker's account. It is also possible for the attacker to impersonate the RPC server by taking over the port before Geth is started. However, performing such attack is unnecessary because the attacker already has full access to the wallet's functionality with the client impersonation attack.

Parity [21] is a recommended Ethereum wallet whose behavior is similar to that of Geth, and the above attacks are also feasible on Parity. The main difference between Parity and Geth is that Parity enables the RPC interface by default, while in Geth, the user has to enable it.

3.2 Case 2: Wallets with basic access authentication

Bitcoin Core [5], one of the most popular Bitcoin wallets, operates its RPC server on port 8332. The RPC server authenticates the client with HTTP basic access authentication [47], in which the client sends its credentials (username and password) to the server in a HTTP header. Since the scheme does not protect the confidentiality

Table 1: Discovered vulnerable wallet applications and RPC clients

Authentication	Currency	Wallet application	RPC client
No authentication	Ethereum	Geth 1.8.15 Parity 2.0.3	Metamask 4.9.3 [17] web3.py 3.16.5 [30] web3.js 0.20.6 [29] web3 java v3.5 [28]
Basic access authentication	Bitcoin	Bitcoin Core 0.16.3 Bitcoin Knots 0.16.2	bitcoin-cli [5] Armory 0.96.4 [4] bitcoind-rpc 0.7 [8] python-bitcoinrpc 1.0 [23]
	Qtum	Qtum Core 0.16.1	qtum-cli [24] qtumjs 1.9.1 [25]
	Dash	Dash Core 0.12.3.3	dash-cli [10] dashd-rpc 1.1 [11]
Digest access authentication	Monero	Monero wallet 0.12.3	monero-python 0.4.2 [19] monero-nodejs 4.0 [18]

of the credentials, it should be used in conjunction with TLS. However, Bitcoin Core has stopped supporting RPC over TLS in 2017 to discourage the RPC interface from being used over the Internet [1]. Bitcoin Core allows the user to encrypt each account with a password, and the client must send the password as HTTP POST data to the RPC server to unlock the account before the account can be used. The encryption password does not need to be the same as the password used for client authentication.

Attacks. At first glance, the authentication mechanism appears to prevent local attackers that do not know the credentials from performing client impersonation to the RPC server. However, only the server authenticates the client. Thus, nothing prevents impersonation of the server to benign clients. That is, the attacker can run a malicious server on port 8332 before the benign server starts and can accept any clients regardless of the credentials they present. Luckily for the attacker, when the server port has been taken, Bitcoin Core fails to start the RPC server and does not notify the user about the failure. As a result, the attacker can capture both the plaintext authentication credentials and the account decryption passwords sent by the benign clients without alerting the victim. Once the attacker has obtained these, he can terminate the malicious RPC server and wait for the benign RPC server to restart. He can then use the captured secrets to connect to the benign RPC server and steal the coins from the wallet.

Bitcoin Knots (a recommended Bitcoin wallet) [7], Qtum Core (the official wallet of Qtum) [24], Dash Core (the official wallet of Dash) [10] also use the basic access authentication mechanism and share the same vulnerabilities as Bitcoin Core. Two other cases in this category are the official Litecoin wallet [16] and official Zcash wallet [31]. While they also use the same authentication mechanism, they do support configuration of TLS. However, if the user does not enable TLS on the RPC server, these wallets are similarly vulnerable.

3.3 Case 3: Wallets with digest access authentication

The Monero wallet [20] provides a separate executable for the user to enable the RPC interface. When the user runs the executable, the RPC server will start on a port number that is specified by the user. Like the wallets mentioned above, the Monero wallet does not support RPC over TLS. The RPC server authenticates the client with the HTTP digest access authentication scheme [38], which is based on a simple challenge-response paradigm. Basically, the client receives a nonce from the server and then replies with a MD5 hash value of the username, the password, the nonce, the HTTP method, and the URI. With the digest access authentication scheme, the client can authenticate itself to the server without revealing the authentication credentials in plaintext.

Attacks. We found that it is possible to perform server impersonation on the Monero wallet by hijacking the port number before the victim starts the RPC server. The digest access authentication mechanism does not help here because it only authenticates the client. However, the RPC executable will fail to start if the port that it uses has already been taken. While this allows the victim to detect the attack, it does not free him from risks. For example, the victim may attach the RPC executable to the operating system's startup to launch it automatically after login for convenience. In that case, since the RPC server process does not have a graphical user interface (GUI) to notify the victim that it has failed, the victim will not notice the failure and thus assume that the RPC server is running. Hence, the attacker's malicious server captures commands from the benign client. An example of such commands is `create_wallet`, which tells the server to create a new wallet account. This allows the attacker to have access to the new account because it is created by the attacker instead of the real wallet application.

Client impersonation, on the other hand, is not practical because the attacker will receive MD5 authentication hash instead of the plaintext credentials. While MD5 is known to be insecure against

collision attacks [42, 53], cracking a given authentication hash (i.e. preimage attack) is still not feasible.

4 DEFENSE MECHANISMS

In this section, we discuss potential defense machanisms for the attacks presented in the previous section. Our recommendations below are inspired by our discussions with cryptocurrency development teams and by the rich literature of network security, where solutions to tackle impersonation and man-in-the-middle attacks in open networks are actively researched. Even though the types of attacks are similar, the communication taking place inside a computer has not been treated with the same level of cautiousness as that of a typical communication over a network. It could be due to the ignorance of desktop application developers towards security in general, or due to the orthodox opinion towards local attackers that physical access to a computer ends any level of security. The latter might be true is some cases, however, we argue that reusing and utilizing existing solutions from network security domain would be highly beneficial as a protection against our attacks.

Limiting the attack surface. Malicious processes are a particular threat in environments where multiple users have access to the computers. The number of possible attackers can be minimized by ensuring that nobody else can access the computer where you keep your cryptocurrency wallets. This includes disabling the guest account. Also, wallet applications should, by default, disable the RPC interface.

Fail-stop failure. One simple change that could help to improve the security of wallet applications with GUI is that when the software fails to start the RPC server, instead of silently ignoring the failure (as e.g. Bitcoin Core does), the application notifies the user about the failure and halts. This would mitigate the server impersonation attack (described in Section 3.2) to some extent because the user would know that the software is not running.

Cryptographic protection. The main cause of the described attacks is the lack of mutual authentication between the RPC server and client: both, basic access authentication and digest access authentication, allow only the server to authenticate the client. One way to achieve mutual authentication is to mandate the use of TLS even when the RPC interface is accesses only from the `localhost`. Strong passwords that are in current implementations used for client authentication could be utilized with TLS as preshared keys. Alternatively, the wallet applications could use TLS with self-signed certificates.

Architectural changes to the wallet applications. Finally, if the RPC interface is only accessed by local processes, the security problems could be eradicated by replacing HTTP with a less open inter-process communication (IPC) mechanism. For example, if the client and server processes were related (e.g. sibling processes) the communication could be done using anonymous pipes or socket pairs. These IPC channels are not vulnerable to nonprivileged local attackers because both endpoints are created at the same time by the same process. Such architecture has been widely used, for example, to provide secure connection between browser extensions and native desktop applications [39].

Solutions that may not be suitable. We now discuss solutions that could be used to protect local communication against local attackers, but may not be suitable for wallet applications.

Since the attacks presented in this paper are performed by another user of the same computer, it might appear that the communicating endpoints (i.e. the RPC server or client) can use access control functionalities of the OS to check whether they are communicating with the correct entity. That is, the endpoints can simply deny any connections by processes whose owner is not the same as theirs. Alternatively, they can restrict access to processes that are owned by administrators or users that belong to a specific group. However, verification of a process's owner is not that straightforward, and it might not work in every case. For instance, users may run the wallet application inside a container (e.g. in a Docker container [12]). In such situation, it is not possible for a process running inside the container to know the owner of the other communication endpoint's process. Similarly, other sophisticated mechanisms that offers protection to local communication by securing the resource access [49–51] are also confined only to a non-containerized environment.

One may also think about replacing basic access authentication and digest access authentication with a mutual authentication scheme [33, 46, 57]. Such schemes, however, are not currently supported by HTTP authentication framework [44], making it harder for them to be deployed.

5 RESPONSIBLE DISCLOSURE

We have reported all the vulnerabilities that we discussed in this paper to the corresponding vendors. We described the attacks and their impact as well as providing suggestions on how to mitigate the attacks. Below are the excerpts from two of the vendors who actively engaged with us to discuss the problems as well as potential solutions.

Bitcoin Core development team admitted that the impersonation attacks (see Section 3.2) can be used to to expose user funds on shared systems. From their point of view, using more permissioned IPC mechanisms, such as UNIX sockets, for the RPC channel may help to address the issues. However, they are hesitant to do that because it would greatly change the Bitcoin Core's user experience. Thus, instead they encouraged us to publish our findings as a discouragement for users attempting to use Bitcoin Core on shared systems. The developers stated that they will update Bitcoin Core so that it will notify the user and fail to start if it cannot start the RPC server.

Parity developers also acknowledged the findings as genuine issues that need to be addressed. We initially proposed that the RPC server should accept only connections from processes that are owned by an administrator or an user belonging to a specific group. However, Parity developers noted that this would not work if the wallet application was run inside a container, as discusses in Section 4. Instead, the Parity developers decided that it is better to drop support for the RPC interface altogether instead of trying to fix it.

Others. At the time of writing, we are still anticipating responses from the rest of the vendors. Since most of the mentioned wallet applications are community-driven projects, we assume that

unanimous acknowledgement of the disclosed vulnerabilities and agreement on a suitable solution would take some time. Nonetheless, we will provide more information about how these vendors react to the presented problems in the final version of the paper.

6 RELATED WORK

In this section, we summarize the research literature related to the attacks presented in this paper.

Without appropriate authentication mechanisms in place, IPC channels, for example Windows named pipes, are prone to be abused by attackers who have access to the victim's computer [36, 54]. It has been discussed previously that such ill-secured IPC channels inside the computer allow local attackers to exploit password managers and other security-critical applications [34]. Similar problems found in mobile OSs [37, 48, 58] show that IPC security needs more attention even on environments where the applications are isolated from each other. In this paper, we continue this thread of security research demonstrating that a local attacker with minimal privileges can steal sensitive information from other users. We focus specifically on cryptocurrency wallet applications, which none of the aforementioned works considered.

By exploiting the wallets that expose their RPC interface to the Internet, there are incidents where attackers have stolen large amount of Bitcoins [13] and Ethereum [55]. While most wallets have discouraged the remote usage of the RPC interface [6], accepting requests only from `localhost` does not completely solve the problem. In fact, if those requests are unauthenticated, the RPC interface can still be exploited by malicious websites through the victim's web browsers, either by Cross Site Request Forgery attack (CSRF) [2, 26] or by DNS rebinding attacks [15]. These vulnerabilities can be fixed by enforcing CORS policies. However, an overly permissive policy could still allow attackers to exploit the JSON-RPC daemon misconfiguration like in the case of Ethereum [27]. In comparison, our attacker is an unprivileged process started by another user or guest on the same computer, in which case the CORS policy has no effect.

7 DISCUSSION AND CONCLUSION

In this work, we analyzed the security of the open RPC interface of cryptocurrency wallet applications. We found vulnerabilities that allow any unprivileged process on the same computer to perform impersonation attacks on the RPC channel. The vulnerabilities could be exploited in multi-user computers, where an authorized insider can effectively steal the wallet's funds. We have reported the discovered vulnerabilities to the respective vendors.

Having an open architecture by providing the RPC interface that enables the development of other blockchain applications will indeed helps the cryptocurrency ecosystem to grow in future. However, remotely accessing such RPC interface of wallet applications is often discouraged (or deprecated in some cases) to reduce the attack surface of the wallets. Also, as we discussed in this paper, security of using those RPC interfaces for local communication is overlooked and it can be abused by an attacker having access to the local machine. Unless these interfaces offers secure means of using them, they are indeed the pitfalls of the cryptocurrency wallets.

Based on our findings, we emphasize that protecting cryptocurrency wallets from local attacks has not received sufficient attention. It has become common wisdom among information security experts to think that if the attacker has access to the computer there is nothing that can be done to protect the user data. This may be true for malware because it usually runs with the privileges of the victim or an administrator. However, there are many other kinds of local attacks. The attacks that we presented in this paper could be performed by any other user on the same computer regardless of their privileges, including guest users. Furthermore, we discussed various ways to prevent attacks by these nonprivileged attackers. We hope that this work will raise awareness about the importance of the security of local communication channels and thus prevent similar vulnerabilities in not only cryptocurrency wallets but also in other software.

REFERENCES

[1] 2015. Enabling SSL on original client daemon. https://en.bitcoin.it/wiki/Enabling_SSL_on_original_client_daemon
[2] 2016. CSRF Vulnerability Allows for Remote Compromise of Monero Wallets. https://labs.mwrinfosecurity.com/advisories/csrf-vulnerability-allows-for-remote-compromise-of-monero-wallets/
[3] 2018. Bisq the P2P exchange network. https://bisq.network/
[4] 2018. Bitcoin Armory. https://btcarmory.com/
[5] 2018. Bitcoin Core. https://bitcoin.org/
[6] 2018. Bitcoin JSON-RPC API. https://en.bitcoin.it/wiki/API_reference_(JSON-RPC)
[7] 2018. Bitcoin Knots. https://bitcoinknots.org/
[8] 2018. Bitcoind-rpc library. https://github.com/bitpay/bitcoind-rpc
[9] 2018. Cpp Ethereum wallet. https://github.com/ethereum/aleth
[10] 2018. Dash Core wallet. https://github.com/dashpay/dash
[11] 2018. Dashd-rpc library. https://github.com/dashevo/dashd-rpc
[12] 2018. Docker Parity documentation. https://wiki.parity.io/Docker
[13] 2018. Electrum Bitcoin Wallets Left Exposed to Hacks for Two Years. https://www.bleepingcomputer.com/news/security/electrum-bitcoin-wallets-left-exposed-to-hacks-for-two-years/
[14] 2018. Go Ethereum wallet. https://geth.ethereum.org/
[15] 2018. How your Ethereum can be stolen through DNS rebinding. https://ret2got.wordpress.com/2018/01/19/how-your-ethereum-can-be-stolen-using-dns-rebinding/
[16] 2018. Litecoin wallet. https://litecoin.org/
[17] 2018. Metamask Ethereum client. https://metamask.io/
[18] 2018. Monero-nodejs library. https://github.com/PsychicCat/monero-nodejs
[19] 2018. Monero-python library. https://github.com/emesik/monero-python
[20] 2018. Monero Wallet. https://getmonero.org/
[21] 2018. Parity Ethereum wallet. https://www.parity.io/
[22] 2018. Peatio: an open-source assets exchange. https://www.peatio.com/
[23] 2018. Python-BitcoinRPC library. https://github.com/jgarzik/python-bitcoinrpc
[24] 2018. Qtum Core wallet. https://github.com/qtumproject/qtum
[25] 2018. Qtumjs library. https://qtumproject.github.io/qtumjs-doc/
[26] 2018. Unauthenticated JSON-RPC API allows takeover of CryptoNote RPC wallets. https://www.ayrx.me/cryptonote-unauthenticated-json-rpc
[27] 2018. Vulnerability Spotlight: Multiple Vulnerabilities in the CPP and Parity Ethereum Client. https://blog.talosintelligence.com/2018/01/vulnerability-spotlight-multiple.html
[28] 2018. Web3 java Ethereum library. https://web3j.io/
[29] 2018. Web3 javascript Ethereum library. https://github.com/ethereum/web3.js
[30] 2018. Web3 python Ethereum library. https://web3py.readthedocs.io/en/stable/index.html
[31] 2018. Zcash Wallet for Linux. https://github.com/zcash/zcash
[32] Jean-Philippe Aumasson. 2018. Attacking and Defending Blockchains: From Horror Stories to Secure Wallets. https://www.blackhat.com/eu-18/briefings/schedule/index.html#attacking-and-defending-blockchains-from-horror-stories-to-secure-wallets-12711.
[33] Steven M Bellovin and Michael Merritt. 1992. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of IEEE Computer Society Symposium on Research in Security and Privacy*. IEEE, 72–84.
[34] Thanh Bui, Siddharth Prakash Rao, Markku Antikainen, Viswanathan Manihatty Bojan, and Tuomas Aura. 2018. Man-in-the-Machine: Exploiting Ill-Secured Communication Inside the Computer. In *USENIX Security 18*. USENIX Association,

Baltimore, MD, 1511–1525.
[35] Vitalik Buterin. 2013. What proof of stake is and why it matters. *Bitcoin Magazine* (2013).
[36] Gil Cohen. 2017. Call the plumber - You have a leak in your (named) pipe. In *DEF CON 25*.
[37] Adrienne Porter Felt, Helen J Wang, Alexander Moshchuk, Steve Hanna, and Erika Chin. 2011. Permission Re-Delegation: Attacks and Defenses. In *20th USENIX Security Symposium*.
[38] J. Franks, P. Hallam-Baker, J. Hostetler, P. Leach, A. Luotonen, E. Sink, and L. Stewart. 1997. *An Extension to HTTP: Digest Access Authentication.* RFC 2069. RFC Editor.
[39] Google. 2018. Native messaging. https://developer.chrome.com/apps/nativeMessaging.
[40] JSON-RPC Working Group and others. 2012. JSON-RPC 2.0 specification.
[41] Anne Kesteren. 2018. Cross-Origin Resource Sharing. https://www.w3.org/TR/cors/
[42] Jie Liang and Xue-Jia Lai. 2007. Improved collision attack on hash function MD5. *Journal of Computer Science and Technology* 22, 1 (2007), 79–87.
[43] Microsoft Developers Network. 2018. Fast User Switching. https://msdn.microsoft.com/en-us/library/windows/desktop/bb776893.
[44] Mozzila. 2018. HTTP authentication. https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication.
[45] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).
[46] Yutaka Oiwa, Hajime Watanabe, Hiromitsu Takagi, K Maeda, Tatsuya Hayashi, and Y Ioku. 2017. *Mutual authentication protocol for HTTP.* RFC 8120. https://tools.ietf.org/html/rfc8120
[47] Julian Reschke. 2015. *The 'Basic' HTTP Authentication Scheme.* RFC 7617. https://tools.ietf.org/html/rfc7617
[48] Yuru Shao, Jason Ott, Yunhan Jack Jia, Zhiyun Qian, and Z. Morley Mao. 2016. The Misuse of Android Unix Domain Sockets and Security Implications. In *2016 ACM SIGSAC Conference on Computer and Communications Security, CCS 2016*. ACM, 80–91.
[49] Hayawardh Vijayakumar, Xinyang Ge, Mathias Payer, and Trent Jaeger. 2014. JIGSAW: Protecting Resource Access by Inferring Programmer Expectations. In *23rd USENIX Security Symposium*. 973–988.
[50] Hayawardh Vijayakumar, Joshua Schiffman, and Trent Jaeger. 2012. STING: Finding Name Resolution Vulnerabilities in Programs. In *21th USENIX Security Symposium*. 585–599.
[51] Hayawardh Vijayakumar, Joshua Schiffman, and Trent Jaeger. 2013. Process firewalls: Protecting processes during resource access. In *8th ACM European Conference on Computer Systems, EuroSys'18*. ACM, 57–70.
[52] Marko Vukolić. 2015. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *International Workshop on Open Problems in Network Security*. Springer, 112–125.
[53] Xiaoyun Wang and Hongbo Yu. 2005. How to break MD5 and other hash functions. In *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 19–35.
[54] Blake Watts. 2017. Discovering and Exploiting Named Pipe Security Flaws for Fun and Profit. http://www.blakewatts.com/namedpipepaper.html.
[55] Wang Wei. 2018. Hackers Stole Over $20 Million in Ethereum from Insecurely Configured Clients. https://thehackernews.com/2018/06/ethereum-geth-hacking.html
[56] Gavin Wood. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151 (2014), 1–32.
[57] Thomas D Wu et al. 1998. The Secure Remote Password Protocol. In *NDSS*, Vol. 98. 97–111.
[58] Luyi Xing, Xiaolong Bai, Tongxin Li, XiaoFeng Wang, Kai Chen, Xiaojing Liao, Shi-Min Hu, and Xinhui Han. 2015. Cracking app isolation on Apple: Unauthorized cross-app resource access on macOS. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS 2015*. ACM, 31–43.