A First Look at the Usability of Web3 DApps

***Abstract*—While the usability literature has considered both web 2.0 web services, as well as cypto-currencies like Bitcoin, web3 is more than the sum of these parts. Web3 decentralized applications (DApps) deploy a unique model that creates new usability challenges and risks to user data and funds. This paper uses an expert evaluation method to identify usability challenges across 50 popular Ethereum DApps found 'in the wild,' used alongside the popular wallet MetaMask. Our evaluation uncovers four common usability challenges impacting users' ability to make informed decisions, and points to opportunities for future user studies and areas for improvement.**

I. Introduction

When Ethereum introduced smart contracts that can be executed on a blockchain, user interfaces had to be rethought. Ethereum users are no longer merely making payments to a receiving address, they are interacting in arbitrary ways with decentralized applications (DApps) to perform more complex tasks like trading assets, establishing loans, bidding in auctions, and registering domain names. Furthermore, new DApps emerge too frequently for standalone wallet software—the kind that dominates the Bitcoin ecosystem—to stay updated. The idea that users might transact with smart contracts in a 'raw' way, by manually supplying a contract address, function to be run, and parameters to the function, was an obvious non-starter for non-expert users.

The solution is what we now call web3 (see Figure 1). A DApp augments the smart contract with a companion website, accessible over the standard internet, which provides a user interface and mostly abstracts away the technical details of the contract. The connection between the website and the blockchain is convoluted however, hence the need for a usability analysis. Websites typically pass scripts to the user and the website, now running client-side, will interact with third party APIs to fetch up-to-date blockchain data that is general to all users. To customize the content for the specific user, the website does not maintain accounts server-side for its users, as in web 2.0. Instead the user is expected to run a standalone client, called a wallet (as in Bitcoin). After authorization, the wallet will supply the user's addresses to the website, and the website can propose blockchain actions to the wallet to present to the user for signature.

In summary, (i) users interact with a website first, rather than their wallet; (ii) after user authorization, the website and wallet can pass data between them; (iii) the wallet is agnostic about what the DApp is or does; (iv) all authorizations are done by the user in the wallet software; and (v) users will be asked to sign transactions in their wallet that they, and the wallet itself, will not 'understand,' instead relying on the trustworthiness of the website proposing the transaction.

Our work investigates usability challenges in popular DApp and wallet interfaces. Malicious DApps exploit the complex arrangement between users, DApps, and wallets, attempting to deceive users into granting permissions to their data and access to their funds [1], [2]. Web3 wallets inform users about such attempts by including warnings in the prompts that request users to authorize data access and transactions. Despite these warnings, recent attacks [3], [4], [5] show that attackers continue to trick users into approving malicious wallet prompts that result in loss of funds. While existing usable security research on web3 has focused on understanding users' mental models [1], [6] and extracting security-related information about contracts [7], usability of web3 DApp and wallet prompts remains largely unexplored.

Before deploying full-blown user studies, expert review is used as a starting point to identify key areas requiring attention. In this vision paper, we perform this first step, setting a research agenda for more detailed follow-up work. To this end, our contributions include:

1) An expert review of the top 50 DApps using an established methodology called a cognitive walkthrough [8].
2) Explication of four high-priority usability challenges in current interfaces based on our evaluations.
3) Discussions of potential interface improvements to address these usability challenges to limit user errors and better inform users about permissions granted to DApps.

II. Background

In web 2.0, apps are designed with a front-end website associated with a centralized back-end server and maintained by the website owner. Web3 redesigns this structure by replacing centralized services with a blockchain—a network of anonymous validators that store and run programs in exchange for financial rewards (referred to as "gas" fees).[1] Program logic is defined into *contracts* which run on a virtual environment called the Ethereum Virtual Machine (EVM). We discuss related aspects next. Figure 1 illustrates the web3 components involved in common tasks performed by users of DApps.

A. Wallet-DApp interaction

The primary mode for user authorizations for DApps is through wallets typically installed as browser extensions. Wallets such as MetaMask inject the Ethereum Provider API [10] into every website visited by the user, enabling the sites to discover wallets (e.g., through injected fields such as *window.provider.isMetaMask* [11]) and to create wallet prompts for approval to trigger contracts. DApps can also use other Ethereum methods [12] to send requests to the wallet for account addresses (*eth_requestAccounts*) and transaction

[1] In practice, much of web3 traffic relies on services resembling centralization (cf. [9])

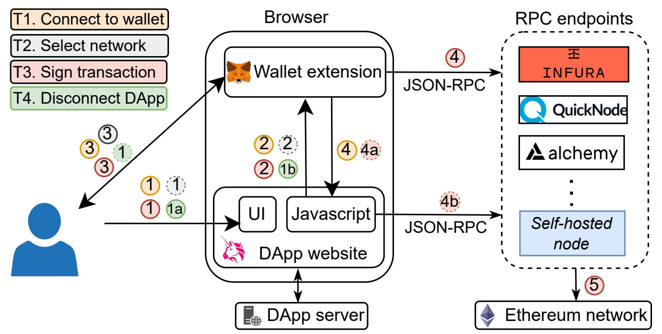


Fig. 1: Overview of web3 architecture and user flow for four common tasks. In T1, the user selects "connect wallet" on the DApp 1 which requests the user's account address 2; the wallet obtains user approval 3 and returns the address 4. In T2, the user selects a blockchain network on the DApp 1 which triggers the wallet 2 to prompt the user to switch network 3; alternatively, the user can directly switch network on the wallet. In T3, the user initiates a transaction on the DApp 1 which sends it to the wallet 2; the wallet prompts the user for approval 3 and if approved, the wallet sends its proof to an Ethereum node via JSON-RPC APIs 4 which is then broadcast to the Ethereum network 5; alternatively, the proofs could be routed through the DApp 4a before reaching the network 4b. In T4, the user selects "disconnect wallet" on the DApp 1 which asks the wallet to revoke its permissions 2; alternatively, the user can directly disconnect on the wallet 1.

requests (*eth_sendTransaction*), which are prompted to the user for approval.

B. EVM blockchains

Another feature of web3 is its scalability enabled by individual blockchains with distinct features (e.g., lower gas fees, and faster transactions).[2] These blockchains, also referred to as *sidechains* and *Layer 2* (L2) chains, run in parallel to the main Ethereum blockchain (L1) but adhere to Ethereum standards, so wallet software can interoperate with only a chain ID (unique identifier for the blockchain) and the Remote Procedure Call (RPC) URL (address of the remote server which acts as the gateway for the DApp to access the blockchain nodes). Many DApps integrate with multiple chains to support transactions across the ecosystem.

C. RPC endpoints

DApps and wallets rely on RPC endpoints[3] to exchange transaction data with EVM networks, using the standard JSON-RPC API [12]. For instance, MetaMask relies on Infura as its default endpoint [13]; note that these separate services are both owned by the same entity. These endpoints relate to different privacy implications, e.g., some providers maintain logs of user data transmitted through their servers while others state that they do not collect any user data. Thus, it is important for users to choose endpoints that align with their privacy preferences.

III. Related Work

The usability literature has explored Bitcoin [14], [15], [16], [17], [18] and generic blockchain technology [19], [20], [21]. In terms of Ethereum, web3, and wallets such as MetaMask, the literature has focused on user perceptions. Panicker et al. [6] found that users do not fully comprehend risks involved in token transfers in MetaMask UIs due to a lack of information about outcomes. Analyzing app reviews of mobile wallets, Voskobojnikov et al. [22] observed misinformed mental models due to an overreliance on understanding of traditional web 2.0 systems. Si et al. [1] explored user interactions and risk perceptions across the Web3 ecosystem, finding user concerns about vulnerable contracts and social engineering attacks involving inadvertent approvals. By contrast, our study learns through direct interaction with a broad set of popular DApps 'in the wild.'

IV. Methodology

A cognitive walkthrough [8] is a usability evaluation methodology often employed first to establish a priority list of usability issues to be considered with other methods. A dual expert (knowledgeable in both usability and in the domain of study—web3, in our case) evaluates if users with a specific persona (*e.g.,* no prior familiarity) can successfully use an application interface and complete a set of core tasks without violating a set of guidelines. For our study, we assume that users are comfortable with web 2.0 websites and browser extensions. We also assume basic familiarity with blockchain user interfaces, e.g., gained from bitcoin wallets that have existed before web3 DApps. Two evaluators independently generated an initial list of tasks users perform on web3 DApps. Then we discussed each task and whether a similar task has been studied in the past (*e.g.,* as part of bitcoin wallet evaluation) and agreed to focus on tasks unique to web3. We excluded one-time tasks (such as setting up an Ethereum wallet) and decided to focus on frequent tasks users need to perform across DApps. Our final list of core tasks included:

- T1. Connect MetaMask wallet to a given DApp website.
- T2. Configure wallet to connect to a desired blockchain network (if it is not already on it). This network has to be supported by the DApp to perform transactions; the supported networks may be different on each DApp.
- T3. Conduct an operation of the DApp site that does require wallet approval, configure and sign the transaction, understand and avoid risks. Covers token balances, gas fees, approvals, signature, confirming transaction, etc.
- T4. Revert, to the extent possible, any past interactions with the DApp including disconnecting the wallet and revoking permissions.

Fig. 1 shows the workflow of these tasks. We perform walkthroughs by simulating step-by-step user actions in each core task and asking ourselves whether the interface satisfied

[2] As of Sept 2024, 1,061 blockchains as per https://chainlist.org/
[3] https://rpc.info/

a list of usability guidelines. We draw our guidelines directly from the established usable security literature—specifically past studies of Bitcoin [14], [23]. We repeat verbatim:

G1. Users should be aware of the steps they have to perform to complete a core task.
G2. Users should be able to determine how to perform these steps.
G3. Users should know when they have successfully completed a core task.
G4. Users should be able to recognize, diagnose, and recover from non-critical errors.
G5. Users should not make dangerous errors from which they cannot recover.
G6. Users should be comfortable with the terminology used in any interface dialogues or documentation.
G7. Users should be sufficiently comfortable with the interface to continue using it.
G8. Users should be aware of the application's status at all times.

A walkthrough allows for an evaluation across a large number of DApps, in contrast to time-limited and costly user studies. During initial analysis, the two evaluators performed walkthroughs on different DApps to identify problematic areas and inform further walkthroughs of DApps in our full dataset. After refinement based on this analysis, the main researcher evaluated each of the top 50 DApps listed on *DAppRadar*,[4] ranked by the total value of assets held under the DApp's contract over a 30-day period (Nov 2024). We used the MetaMask wallet for these tasks.

A. A Sample Walkthrough

For space considerations, we do not present the full details of each walkthrough of 4 tasks over 50 DApps. Instead, we present one representative sample walkthrough to illustrate what the methodology looks like, and then we present our overall results, based on all walkthroughs, in the next section. The sample is the (T3) transaction approval task in the DApp Uniswap [24], a decentralized exchange service on Ethereum allowing users to exchange one type of (ERC-20) token for another. We assume the persona of a typical web3 user who owns one or more tokens, uses a common wallet such as MetaMask, and has basic understanding about Ethereum tokens and transactions. For this task, we also assume the user has already connected their wallet to Uniswap (T1) and successfully switched to a target network (T2).

The swap page on Uniswap site prompts the user to select the tokens and the exchange amounts. After selecting a token (ETH) to sell, the prompt shows the available wallet balance and an option to use the maximum amount. We input an arbitrary amount to sell, and select to buy USDT using the token search option. Searching "usdt" displayed three entries of which one was marked with an warning symbol. When selected, a prompt cautioned that the token was not among frequently traded tokens. While only one entry was marked with this symbol, two entries triggered the same warning prompt. Although these warnings aim to alert users about potential scams, conflicting cues likely cause confusion (violating G6-G7) and reduce the effectiveness of warnings.

Uniswap triggers a wallet prompt with details about the transaction. At the top of the prompt, a small icon beside the contract address displays a generic message prompting users to verify before trusting the contract. This violates G2 as there is no guidance on how to verify. The prompt also includes a section with predictions about the resulting balance changes, however this is not guaranteed as indicated within an icon. We discuss this issue further in Sec. V-C. The prompt also includes estimated gas fees and an option to set a limit on the maximum fee. This option (in a secondary prompt) lists four choices with different limits and processing times (options with lower fees tradeoff on longer wait times).

After selecting "low" gas and submitting the transaction, the wallet notified that the transaction had failed without any description. An external link from the wallet showed the error output directly from the contract, however this message is ambiguous with no recovery instructions. This violates G4. Uniswap's troubleshooting guide [25] lists potential reasons for transaction failures including possible issues caused by low gas limits, although users cannot be expected to find such resources for each transaction failure.[5] The DApp/wallet interface can address these issues by explaining transaction failures and providing instructions on possible recovery options.

B. Limitations

Our findings cover 50 popular DApps with MetaMask which may not be representative of other DApps and wallets. We also do not cover mobile wallets and DApps accessed through mobile browsers. We evaluated a set of four core tasks which does not cover other tasks users might perform on DApps, such as connecting with other users on social DApps. Finally, our expert evaluations may not accurately represent users' behavior in practice. Our evaluations still uncover important usability issues that carry significant security and privacy implications for users.

V. High Priority Usability Issues

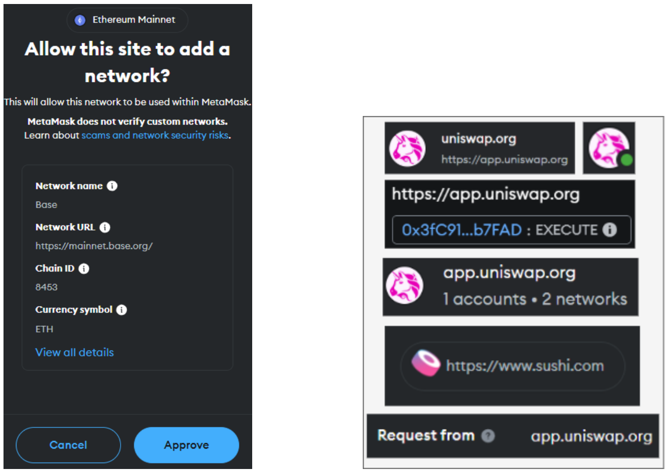
We now present the highest priority issues we found during our walkthroughs of 50 DApps, focusing on issues that may lead users to make dangerous errors (G5). We would direct future research efforts to consider these issues.

A. Issue 1: Weak link between wallet prompts and DApps

Users can have multiple DApp websites open at once, while the wallet is a singular and separate interface. We found MetaMask attempts to associate prompts with its originating DApp by displaying domain names, however it is inconsistent. In 100% of the ($n = 45$) prompts for adding a network (Fig. 2a), we found that the wallet did not link prompts to websites: *e.g.,* 'allow *this site* to add a network?'

[4] https://dappradar.com/

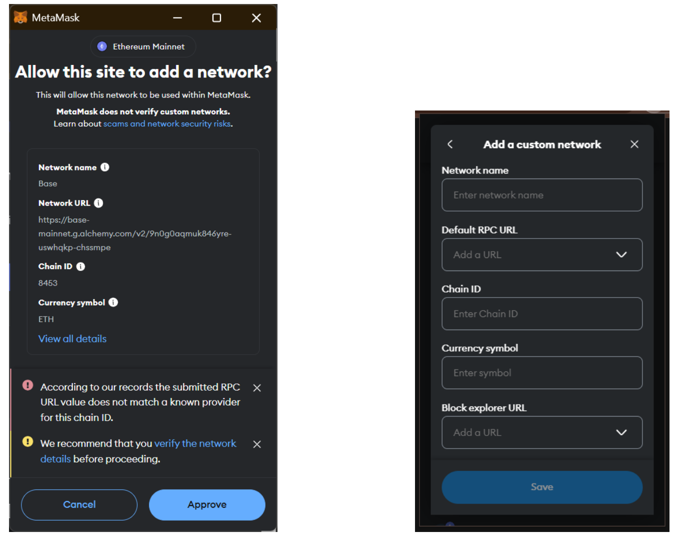
[5] We attempted another transaction submitting with the default ("market") gas fee limit, which was successful.



(a) No DApp indicator

(b) Variations in DApp indicators

Fig. 2: Lack of consistent cues to link prompts to DApps.



(a) Add network prompt prioritizing simplicity over security.

(b) Add network prompt sacrificing usability for security.

Fig. 3: Challenges in adding a new network to the wallet.

Connecting to a malicious network compromises the user's privacy (links user's IP address and wallet addresses), displays information about the state of the blockchain, and can propose fraudulent transactions (timed to follow specific user actions). In other cases, the domain is shown but the visual cue varies (boxed/rounded/unboxed; domain/url/favicon; sizes/colors/locations); overall, we found six variations (shown in Fig. 2b) in how the wallet displays the domains.

Transactions can also queue, say if a user forgets to sign/cancel a transaction in MetaMask or clicks multiple times on the website. MetaMask indicates the number of current requests in the extension icon and a textual indicator in a small font size, but is easy to miss. Users, who have unknowingly queued a transaction and proceed to a new DApp where they take an action that creates a transaction, will be presented with the earlier queued transaction for signature first.

Recommendation: A previous paper [26] introduces TLS-based authentication of contracts and proposes augmenting MetaMask with warnings (similar to web 2.0 browser certificates), though MetaMask has not included such proposals. We recommend that the requesting DApp's URL is in every wallet prompt in a consistent UI component (i.e., location, format). The wallet could also warn users when the requesting DApp is not in the current (in-focus) view, bring the requesting DApp's web page to in-focus view, or disable the approve button until the page is in focus to restrict inadvertent approvals.

B. Issue 2: Cross-chain switching issues

It is a common task for web3 users to switch their wallet to different blockchain networks when interacting with DApps. When adding a new network, it is the user's responsibility to evaluate its security and privacy as wallets do not verify. However, this tends to be an arbitrary process where users need to rely on ad-hoc web searches to check the network [27], [28]. This manual process is also extensive as it involves verifying the individual details displayed about the network including the network's name, RPC URL, chain ID and its currency.

There are three options for adding a new network on a MetaMask wallet, and each option may involve different usability and privacy and security implications:

1) Users may add a new network from a DApp site (e.g., by initiating a transaction on the relevant blockchain) which would send the wallet network information including the DApp's preferred RPC provider (as shown in Fig. 3a). In this case, the wallet prioritizes simplicity and prompts the user to add the network specified by the DApp. This prompt does not allow the user to select a different (e.g., more privacy-friendly) RPC, or even indicate the possibility of changing providers. Thus, users may consent to using the displayed network though they may not be fully informed or even aware of more privacy-friendly choices.
2) Second, for a small subset of (popular) networks [29], users can add the network automatically from the MetaMask interface using suggested providers (e.g., a related entity—see Sect. II-C). Wallets may be able to inform users about potential security and privacy issues with specific network providers, however, this may be possible only for a limited number of networks.
3) Third, users can add a network on their wallet by manually configuring the network such as their preferred RPC (Fig. 3b). Although this option offers the most control to users, it may not be straightforward to identify legitimate network providers that are trustworthy.

These issues complicate the process of configuring networks by requiring users to perform multiple out-of-band checks (i.e., breaks the user's workflow for the task by involving external resources and ad-hoc web searches). MetaMask prompts include warnings (as in Fig. 3a) when the network information (e.g., name, RPC URL, or currency) provided by the DApp does not match its own records; 62% of the 45 prompts we reviewed included at least one warning about networks. However, users may become desensitized to security warnings especially when they are frequently linked to false alarms [30].

Recommendation: When adding a network, users should be made aware of alternative providers who may offer better security and privacy. To this end, when adding networks via DApps, wallets can simply display an option to change the provider before adding to the wallet. It is important for users to be informed about available choices before they make a decision since adding a network to the wallet immediately exposes personal information such as user's IP address and wallet addresses (which can then be linked to past transactions, compromising privacy). To further support informed user decisions, DApps and wallets could list more than one provider for the user to choose from, and provide information about the benefits of each network in terms of transaction speed, gas fees, security and privacy properties; existing resources such as ChainList[6] already provide information about alterantive RPC servers and their privacy and speed.

C. Issue 3: Unpredictable function behaviour

When prompting for a transaction approval, MetaMask includes predicted changes (Fig. 4) to the user's balances (i.e., estimated amount "sent" and amount "received") for recognized tokens and blockchains [31]. Among 40 transaction prompts, we found 25 prompts (63%) with predictions on received tokens. Such predictions may mislead users into falsely assuming that the transaction would actually result in receiving tokens whereas the wallet does not perform any security checks, i.e., misplacing trust on the contract despite the lack of validation or security checks. Though the wallet warns (in a small icon) that the predictions are not guaranteed, users may not fully comprehend the risks or may miss the warning completely.

Other approaches have been proposed to extract information about contracts. Solidity, the primary language for contracts, provides developers the ability to write special user-facing comments about their contract code [32]. However, most of the contracts in the wild do not include such comments [7]. Automated generation of these comments could help fill this gap [7] but such comments may not provide accurate descriptions of contracts and may lead some users into a false sense of security about contract behavior. Another line of research focuses on security tools (e.g., [33], [34]) to identify vulnerable contracts, however informing users about these issues (e.g., in end-user interfaces) remains an open challenge.

Recommendation: Wallets should use a standard interface design to inform users about contract risks. When interfaces includes predictions about balance changes, it should clearly warn users about the lack of validation about the contract to prevent false trust assumptions. All warnings should be displayed using a consistent and prominent format. Future research can determine the extent to which malicious behavior can be predicted through automated analysis (or DApps could default to untrusted until certified).

[6]https://chainlist.org/?chain=1

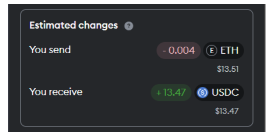


Fig. 4: Transaction predictions by wallet.

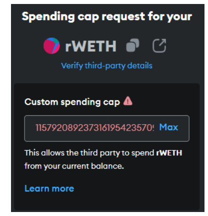


Fig. 5: DApp request for unlimited approval.

D. Issue 4: Concerns with Allowance mechanism

The final issue has been explored in the literature by Wang *et al* [35] but we include the issue for completeness. Tokens deployed on Ethereum follow standard interfaces, such as ERC20, meaning wallet software can interoperate even without knowing anything specific about the token. The ERC20 standard requires an 'approve' function. Instead of Alice sending 10 tokens to Bob, she can approve Bob to take 10 tokens from her at any time until she removes the approval. Many DApps move tokens through approvals rather than transfers.

Prioritizing simplicity and flexibility over security, 18 of 29 DApps (62%) in our study request *unlimited* approvals; Fig. 5 shows an example request. Similarly, Wang *et al* [35] found 60% of DApps requesting unlimited approvals. Unlimited approvals are problematic when a DApp is later compromised (or malicious to begin with) [36], [37]. Many DApps provide no option to adjust the approval amount, while some have been found to display one amount but prompt for unlimited (observed in [35] as an example of a deceptive/dark pattern [38]). While MetaMask does allow approvals to be adjusted, once set, it is up to the user to remember to un-approve.

Recommendation: Attention has been given to allowing users to view, modify, and revoke past approvals, both in the MetaMask dashboard and through third-party tools (e.g., revoke.cash [39]). Because of the ERC20 standard, wallets should recognize an unlimited approval and better warn users, schedule a reminder prompt to revoke for a later time, or allow users to set global limits (with sensible default values) on allowances.

REFERENCES

[1] J. J. Si, T. Sharma, and K. Y. Wang, "Understanding User-Perceived Security Risks and Mitigation Strategies in the Web3 Ecosystem," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2024, pp. 1–22.

[2] C. F. Torres, F. Willi, and S. Shinde, "Is Your Wallet Snitching on You? An Analysis on the Privacy Implications of Web3," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 769–786.

[3] B. Toulas, "LottieFiles hacked in supply chain attack to steal users' crypto," https://www.bleepingcomputer.com/news/security/lottiefiles-hacked-in-supply-chain-attack-to-steal-users-crypto/, October 31, 2024.

[4] V. Vismaya, "Pepe Holder Loses $1.4 Million in Uniswap Permit2 Phishing Attack," https://decrypt.co/286076/pepe-uniswap-permit2-phishing-attack, October 14, 2024.

[5] A. Zmudzinski, "EigenLayer Says Unauthorized Selling Was an 'Isolated Incident', But Critics Still Have Concerns," https://decrypt.co/284903/eigenlayer-says-unauthorized-selling-was-an-isolated-incident-but-critics-still-have-concerns, October 7, 2024.

[6] Y. Panicker, E. Soremekun, S. Sun, and S. Chattopadhyay, “End-user Comprehension of Transfer Risks in Smart Contracts,” *arXiv preprint arXiv:2407.11440*, 2024.
[7] X. Hu, Z. Gao, X. Xia, D. Lo, and X. Yang, “Automating User Notice Generation for Smart Contract Functions,” in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2021, pp. 5–17.
[8] C. Wharton, J. Rieman, C. Lewis, and P. Polson, “The Cognitive Walkthrough Method: A Practitioner’s Guide,” in *Usability Inspection Methods*. John Wiley & Sons, Inc., 1994.
[9] C. Williams, “Infura Outage Sparks Debate Over Ethereum’s Decentralization,” https://cryptobriefing.com/infura-outage-sparks-debate-over-ethereums-decentralization/, November 11, 2020.
[10] P. Gomes, K. Hemachandra, R. Moore, G. Markou, K. D. Hartog, Glitch, J. Moxey, P. Bertet, D. Yeo, and Y. Sergievsky, “EIP-6963: Multi Injected Provider Discovery,” https://eips.ethereum.org/EIPS/eip-6963, 2023.
[11] MetaMask Documentation, “Ethereum provider API,” https://docs.metamask.io/wallet/reference/provider-api/, Accessed: October 2024.
[12] ——, “JSON-RPC API,” https://docs.metamask.io/wallet/reference/json-rpc-api/, Accessed: October 2024.
[13] MetaMask Support, “What is Infura, and why does MetaMask use it?” https://support.metamask.io/networks-and-sidechains/what-is-infura-and-why-does-metamask-use-it/, Accessed: October 2024.
[14] S. Eskandari, J. Clark, D. Barrera, and E. Stobert, “A First Look at the Usability of Bitcoin Key Management,” *Symposium on Usable Security and Privacy (USEC)*, 2015.
[15] X. Gao, G. D. Clark, and J. Lindqvist, “Of two minds, multiple addresses, and one ledger: Characterizing opinions, knowledge, and perceptions of bitcoin across users and non-users,” in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI ’16, 2016, p. 1656–1668.
[16] K. Krombholz, A. Judmayer, M. Gusenbauer, and E. Weippl, “The Other Side of the Coin: User Experiences with Bitcoin Security and Privacy,” in *Financial Cryptography and Data Security*, J. Grossklags and B. Preneel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, pp. 555–580.
[17] A. Mai, K. Pfeffer, M. Gusenbauer, E. Weippl, and K. Krombholz, “User Mental Models of Cryptocurrency Systems - A Grounded Theory Approach,” in *Sixteenth symposium on usable privacy and security (SOUPS 2020)*, 2020, pp. 341–358.
[18] C. Sas and I. E. Khairuddin, “Design for Trust: An Exploration of the Challenges and Opportunities of Bitcoin Users,” in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ser. CHI ’17, 2017, p. 6499–6510.
[19] M. Fröhlich, F. Waltenberger, L. Trotter, F. Alt, and A. Schmidt, “Blockchain and Cryptocurrency in Human Computer Interaction: A Systematic Literature Review and Research Agenda,” in *Proceedings of the 2022 ACM Designing Interactive Systems Conference*, 2022, pp. 155–177.
[20] H. Jang and S. H. Han, “User Experience Framework for Understanding User Experience in Blockchain Services,” *International Journal of Human-Computer Studies*, vol. 158, p. 102733, 2022.
[21] H. Jang, S. H. Han, and J. H. Kim, “User Perspectives on Blockchain Technology: User-Centered Evaluation and Design Strategies for DApps,” *IEEE Access*, vol. 8, pp. 226 213–226 223, 2020.
[22] A. Voskobojnikov, O. Wiese, M. Mehrabi Koushki, V. Roth, and K. Beznosov, “The U in Crypto Stands for Usable: An Empirical Study of User Experience with Mobile Cryptocurrency Wallets,” in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–14.
[23] M. Moniruzzaman, F. Chowdhury, and M. S. Ferdous, “Examining Usability Issues in Blockchain-Based Cryptocurrency Wallets,” in *Cyber Security and Computer Science*, T. Bhuiyan, M. M. Rahman, and M. A. Ali, Eds. Springer International Publishing, 2020.
[24] Uniswap, “Uniswap,” https://app.uniswap.org/, Accessed: September 2024.
[25] Uniswap Help Center, “Why did my transaction fail?” https://support.uniswap.org/hc/en-us/articles/8643975058829-Why-did-my-transaction-fail, Accessed: Oct 2024.
[26] U. Gallersdörfer, J. Ebel, and F. Matthes, “Augmenting MetaMask to Support TLS-endorsed Smart Contracts,” in *International Workshop on Data Privacy Management*. Springer, 2021, pp. 227–244.
[27] MetaMask Support, “Verifying custom network information,” https://support.metamask.io/networks-and-sidechains/managing-networks/verifying-custom-network-information/, Accessed: September 2024.
[28] Coinbase, “How to add custom networks on Coinbase Wallet,” https://www.coinbase.com/en-ca/learn/wallet/How-to-add-custom-networks-Coinbase-Wallet, Accessed: November 2024.
[29] MetaMask Support, “How to add a custom network RPC,” https://support.metamask.io/networks-and-sidechains/managing-networks/how-to-add-a-custom-network-rpc/, Accessed: November 2024.
[30] K. Krol, M. Moroz, and M. A. Sasse, “Don’t Work. Can’t Work? Why It’s Time to Rethink Security Warnings,” in *2012 7th International Conference on Risks and Security of Internet and Systems (CRiSIS)*, 2012, pp. 1–8.
[31] MetaMask Support, “What are estimated balance changes?” https://support.metamask.io/transactions-and-gas/transactions/simulations/, Accessed: October 2024.
[32] Solidity Documentation, “NatSpec Format,” https://docs.soliditylang.org/en/latest/natspec-format.html, Accessed: October 2024.
[33] S. Kalra, S. Goel, M. Dhawan, and S. Sharma, “Zeus: Analyzing Safety of Smart Contracts,” in *Network and Distributed Systems Security (NDSS) Symposium*, 2018, pp. 1–12.
[34] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Bünzli, and M. Vechev, “Securify: Practical Security Analysis of Smart Contracts,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’18, 2018, p. 67–82.
[35] D. Wang, H. Feng, S. Wu, Y. Zhou, L. Wu, and X. Yuan, “Penny Wise and Pound Foolish: Quantifying the Risk of Unlimited Approval of ERC20 Tokens on Ethereum,” in *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses*, ser. RAID ’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 99–114. [Online]. Available: https://doi.org/10.1145/3545948.3545963
[36] Y. Levi, “Bancor’s response to today’s smart contract vulnerability,” https://medium.com/@yudilevi/bancors-response-to-today-s-smart-contract-vulnerability-dc888c589fe4, June 18, 2020.
[37] Primitive Finance Team, “Postmortem on the Primitive Finance Whitehack of February 21st, 2021,” https://primitivefinance.medium.com/postmortem-on-the-primitive-finance-whitehack-of-february-21st-2021-17446c0f3122, February 23, 2021.
[38] A. Mathur, G. Acar, M. J. Friedman, E. Lucherini, J. Mayer, M. Chetty, and A. Narayanan, “Dark patterns at scale: Findings from a crawl of 11k shopping websites,” *Proc. ACM Hum.-Comput. Interact.*, vol. 3, no. CSCW, Nov. 2019. [Online]. Available: https://doi.org/10.1145/3359183
[39] Revoke, “Revoke.cash,” https://revoke.cash/, Accessed: October 2024.

Appendix

A. Selection of Core Tasks

We make the following assumptions about the user during our walkthroughs:

- Technical awareness:
  - Comfortable with web 2.0, smartphone apps, browser extensions, browsers
- Blockchain-specific awareness:
  - A wallet contains tokens (or an address that owns tokens)
  - Tokens are digital things of value that can be transacted and wallet tracks the "balance"
  - Transactions are approved by wallet
  - Approval tied to some secret value (key, seed phrases) which is "in" the wallet
  - Where the Bitcoin usability literature leaves off
- Study specific assumptions:
  - Always connect from mainnet Ethereum

Our initial list of tasks included:

- Installing and configuring MetaMask the software
  - Not part of our study, common to all web3 DApps
- Configuring MetaMask to have an address to receive tokens
  - Not part of our study, common to all web3 DApps
  - Assume user has default settings (address on L1 Ethereum with balance 0)
- Funding the wallet with ETH (for gas costs) and other tokens as necessary
  - Not part of our study, common to all web3 DApps
  - Tricky because of different networks: a future user study could explore Ethereum fragmentation into sidechains and L2s
- Visit a Web3 website and confirm site corresponds to intended website:
  - Not part of our study, common to all web3 DApps
  - Phishing studies have been conducted on web 2.0 that are relevant to this.
- Proceed to connect wallet to website with a practical mental model (G1-G3) of what connecting means, why the process is what it is (different web3 apps might use different processes), understanding and avoiding risks (G4-G5), and confirming connection is successful (G3) (via the website and via MetaMask).
  - Core task
- Configure wallet to connect to a desired blockchain network (if it is not already on this network). This network has to be supported by the DApp to perform transactions. The supported networks may be different on each DApp.
  - Core task
  - Might be subsumed by another task
- Conduct an operation on the web3 site that does not require wallet approval
  - Not a core task for our study as it does not involve the wallet by definition
- Conduct an operation of the web3 site that does require wallet approval, configure and sign the transaction, understand and avoid risks. Covers token balances, gas fees, approvals, signature, confirming transaction, etc.
  - Core task
- Revert, to the extent possible, any past interactions with the DApp. Disconnect the wallet, unapprove tokens, etc.
  - Core task

Our list of Core Tasks included the following:

T1. Proceed to connect wallet to website with a practical mental model (G1-G3) of what connecting means, why the process is what it is (different web3 apps might use different processes), understanding and avoiding risks (G4-G5), and confirming connection is successful (G3) (via the website and via MetaMask).

T2. Configure wallet to connect to a desired blockchain network (if it is not already on this network). This network has to be supported by the DApp to perform transactions. The supported networks may be different on each DApp.
- Might be subsumed by T1 or T3.

T3. Conduct an operation of the web3 site that does require wallet approval, configure and sign the transaction, understand and avoid risks. Covers token balances, gas fees, approvals, signature, confirming transaction, etc.

T4. Revert, to the extent possible, any past interactions with the DApp. Disconnect the wallet, unapprove tokens, etc.