UNDERSTANDING AUDITS PROCESS

This is a comprehensive results of what I have learned;

- During my 3 months at nethermind sometimes back.
- From co-auditors on different platforms.
- How I personally approach smart contracts audits.
- Chat GPT.

Feel free to ride on this cumulative knowledge of ours and add yours

Prepared By: Prosperity
Smart contract security engineer

Understanding the Audit Process for Smart Contracts

Step 1: Setting up/ Preparation

- Logistics play a vital role in facilitating the delivery of a thorough audit. This includes determining the number of auditors required and the estimated timeframe needed for completion.

- It is important to note that the commonly used benchmark of 200 SLOC (Source Lines of Code) per hour may not be applicable for larger audits, and a more accurate estimate may be around 100 SLOC per hour.

- Rest and recovery should be a top priority, as a fatigued mind can impair the effectiveness of an audit.

- Conducting general research on the project and related protocols prior to the audit is essential to ensure sufficient time is available for in-depth analysis.

Understanding the Audit Process for Smart Contracts

Step 1: Setting up/ Preparation

- Request for all relevant specifications, RFCs (Request for Comments), or diagrams from the project team.

- Familiarize yourself with past exploits that have targeted similar protocols.

- Study audits of other protocols that are similar to the one you're auditing.

- Collaborating with others can enhance the quality of your audit.
- Determine the communication methods and tools you'll use with your team (HackMd, Notion, Github. etc).

- Choose a code editor that enables you to navigate the project repository with ease.

Understanding the Audit Process for Smart Contracts

Step 1: Setting up/ Preparation

- Prepare a report for identifying and documenting any findings or issues that each auditor discovers during the audit.

- Create a repository for test suites that will be used for testing the contracts.

- Allocate sufficient time to configure any necessary security tools and infrastructure that may be used during the audit.

Understanding the Audit Process for Smart Contracts

Step 2: Beginning Audits

- Typically, it's best to begin by examining the README file.

- Then, look through the .sol files for block comments that explain key designs or potential issues.

- To avoid getting sidetracked, leave comments with an @audit tag when you have ideas, and keep moving through the codebase.

- Later on, you can quickly locate all of your audit-related comments by searching for @audit using the grep command.

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// @audit -- it is understandable that the business logic of of this project requires interaction with aave pool and others,
//           not a good practice to assume that the 3rd party entities as black boxes and assumes their functional correctness.
//           However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition,
//           upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.
//           therefore, approving type(uint256).max could be a vulnerable in the neares future, we can discuss this later on the best approach, we can probably
//           add a function to revoke the approval to 0 by the owner. also still checking cause other way is since the contract is ugradeable,
//           it could be a measure.
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Understanding the Audit Process for Smart Contracts

Step 2: Beginning Audits

- Analyze the test results and coverage to identify areas for further investigation.

- Identify all user tasks provided by the system and trace the code path for each task, building a clear understanding of how the code operates.

- Keep your focus and use @audit tags to annotate any thoughts that come to mind during the review process.

- Collaborate with other auditors to merge your mental models and enhance the audit's effectiveness.

- Utilize tools like Surya (The Sun God: A Solidity Inspector) to create call graphs if necessary.

Understanding the Audit Process for Smart Contracts

Step 3: Chewing the bone

- It's time to revisit the audit tags and begin investigating further.

- Create an HackMd profile for documentation.

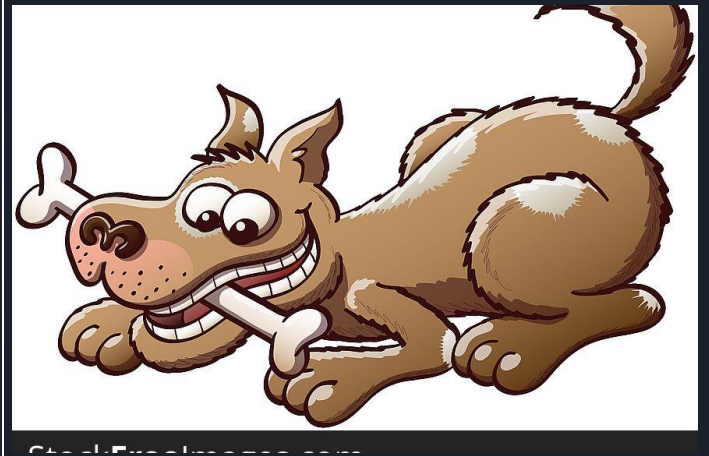
- During this more detailed analysis, be mindful of common attack vectors.

- Simple errors such as typos are often found in this phase, so pay close attention.

- As you follow the thread, make note of your observations by adding more audit tags.

- To confirm your understanding of the system, you may want to PoC an attack as you discover it.

- When you discover valid issues, add them to the findings document.



Understanding the Audit Process for Smart Contracts

Step 3: Chewing the bone

- Keep track of gas optimizations, although they are not the primary focus.

- Use tools like Slither to create new threads to investigate.

- Protocol-specific bugs and vulnerabilities require creative thinking to identify.

- To stimulate creativity, identify all possible ways an attacker can manipulate the system, such as which storage they can manipulate and functions they can call.

- Instead of just verifying what an invariant holds, use divergent thinking to identify ways that it might break.

Understanding the Audit Process for Smart Contracts

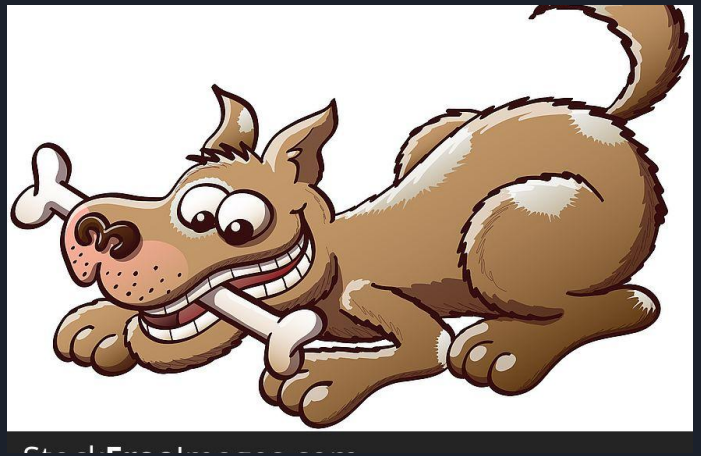
Step 3: Chewing the bone

- Generate more audit tags and possible attack vectors once you have identified the possible knobs (refers to the different attack vectors or ways that an attacker can manipulate the storage or functions of a smart contract to exploit vulnerabilities) attackers can use.

- The auditing process involves examining code paths, adding @audit tags, gaining more knowledge about the code, and finding new attack vectors in an iterative manner.

- If you hit a dead end, go through each file line-by-line, noting any red flags or things you don't understand.

- Collaborate with other auditors to overcome roadblocks and to enhance creativity.



Understanding the Audit Process for Smart Contracts

Step 4: Developing tests and proof-of-concepts

- If test coverage is inadequate, create additional tests to fill in gaps. This will give you a deeper understanding of the system and reveal any bugs that may be difficult to detect visually.

- Take the time to create proof-of-concepts for any findings that haven't already been addressed. It's crucial to provide thorough comments explaining the exploit scenario.

- If you encounter unexpected results during an attack, revisit the approach and determine if there are any other knobs that could be manipulated.

Understanding the Audit Process for Smart Contracts

Step 5: Finalizing the Audit

- Complete any unfinished tasks and address all audit tags.
- Verify each finding as a team, with the help of PoCs if needed.
- Draft the report with specific details and line numbers.
- Share the report along with the test suite and PoC links to provide context.

Understanding the Audit Process for Smart Contracts

Step 6: Post-Audit

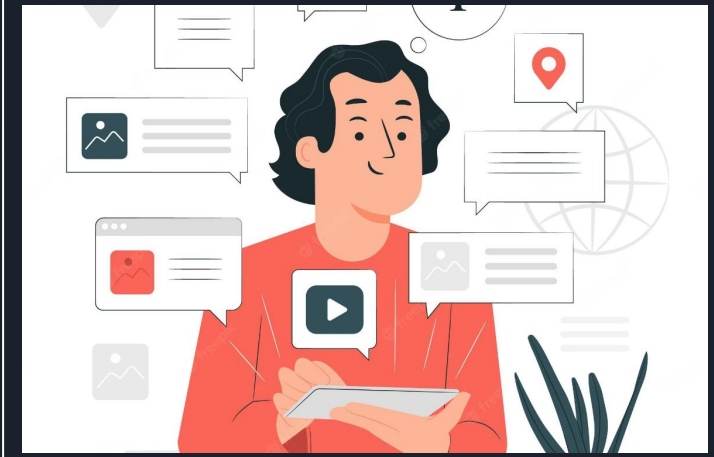
- Be open to answering questions and providing guidance on how to implement recommendations.

- Keep an eye on any changes or updates to the codebase, especially major ones, as they may require another audit.

Understanding the Audit Process for Smart Contracts

Step 7: Additional Information

- You now have a structured method to conduct thorough smart contract audits. Use this process to secure the forefront of blockchain technology!



Understanding the Audit Process for Smart Contracts

Step 8: How to find me

- Portfolio:
https://onahprosperity.github.io/

- Github:
https://github.com/OnahProsperity

- LinkedIn:
https://www.linkedin.com/in/prosper-o-3050791a6/

- Twitter:
https://twitter.com/OnahProsperity