HACKPROVE OFFSIDE LABS

2025 HACKPROVE WORLD

Stop Spoofing My Wallet!

Demystifying Simulation Spoofing Attacks

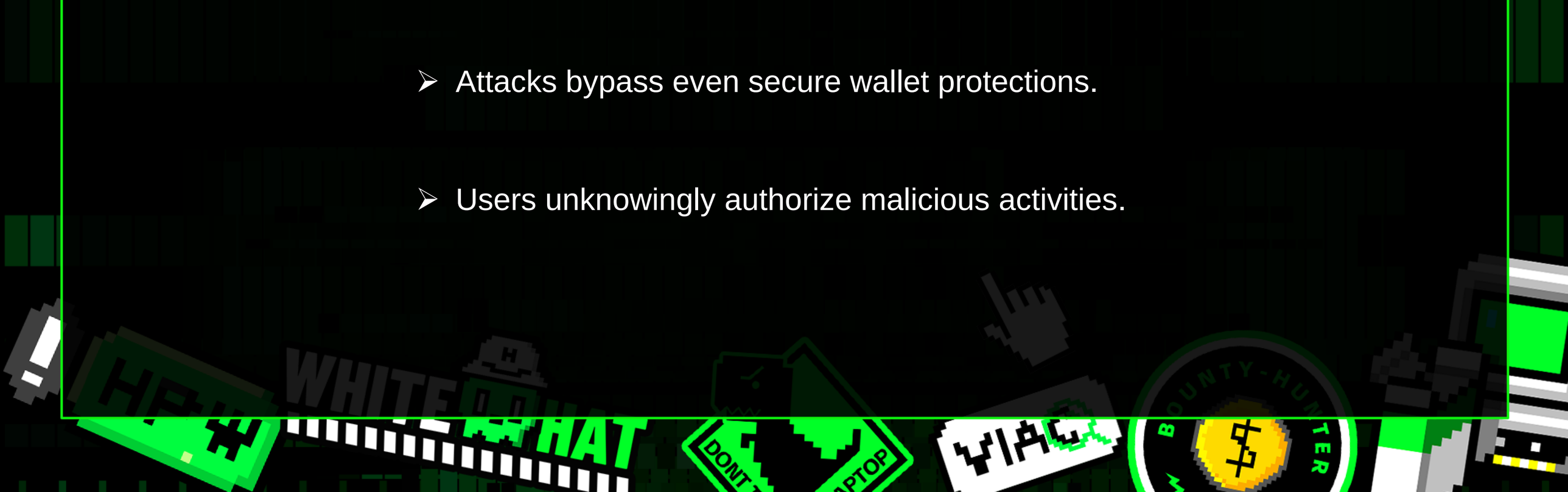
2025/01/11 slipper OFFSIDE LABS

HACKPROVE OFFSIDE LABS

2025 HACKPROVE WORLD

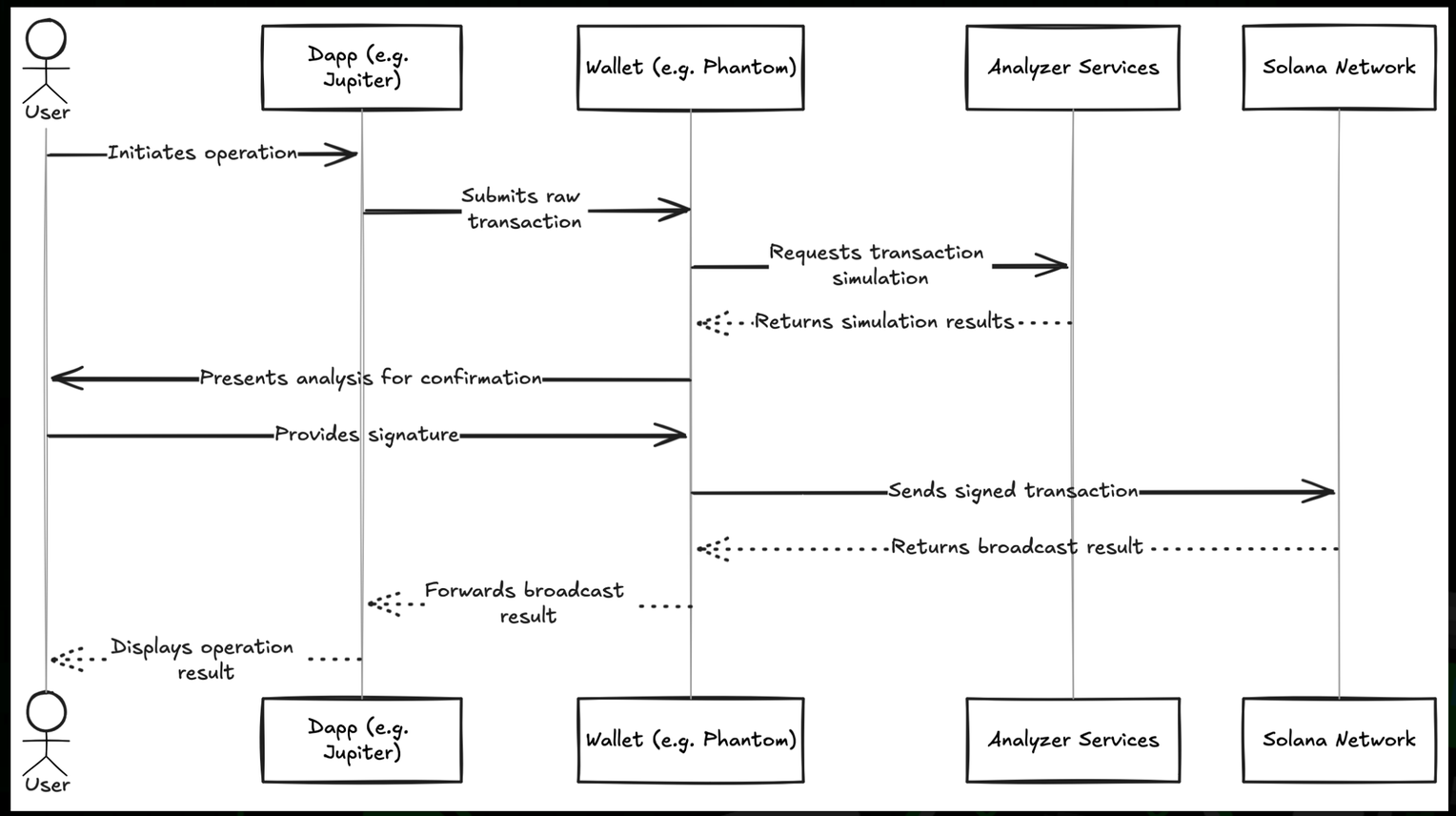
The Growing Threat to Solana Wallets

- Surge in phishing attacks targeting Solana users.
- Attacks bypass even secure wallet protections.
- Users unknowingly authorize malicious activities.



What Happens Behind the Scenes?

- Wallet interactions have hidden complexities.
- A simple "Sign" can authorize dangerous activities.
- Example: Fake Jupiter swap drains user accounts.



HACKPROVE OFFSIDE LABS

2025 HACKPROVE WORLD

Drainer Alert: A Series of Sophisticated Attacks

01 Targets

Solana users using Phantom wallet on Chrome with Windows.

02 Platforms

Jupiter, Raydium, and others.

03 Characteristics

- Not widespread but highly effective.
- Random victims, not targeting whales.
- Triggered by user interactions.

HACKPROVE OFFSIDE LABS

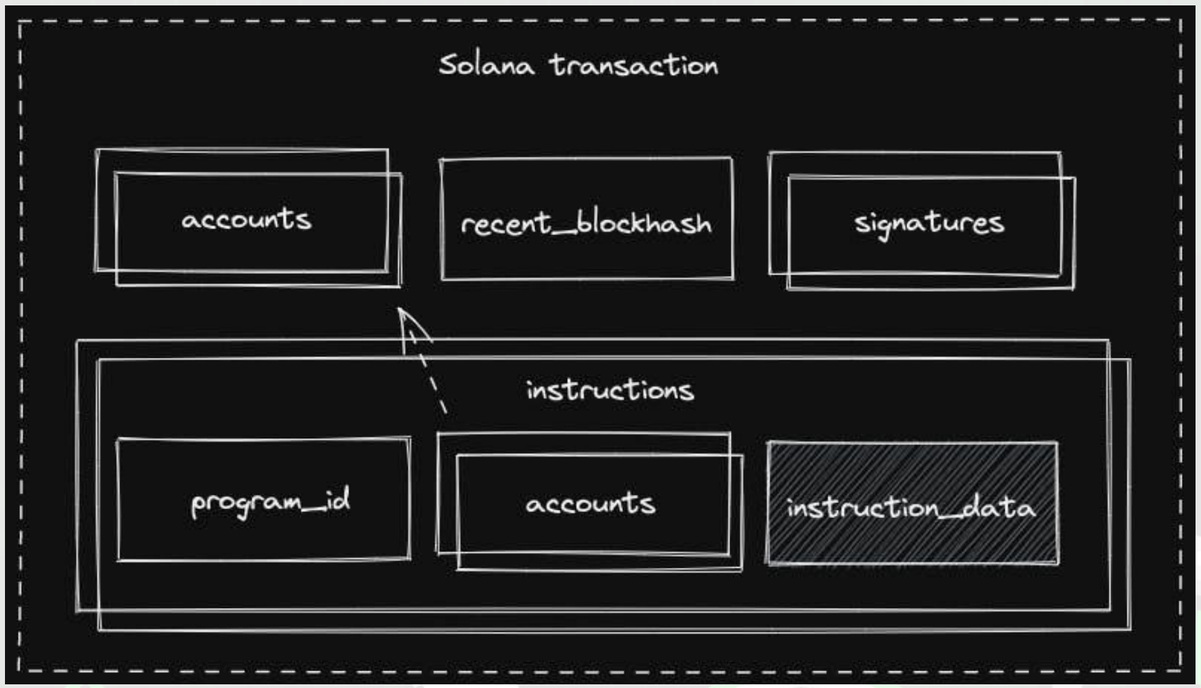
2025 HACKPROVE WORLD

How the Attack Works

Understanding the Complexity of Solana Transactions

Key Components of a Solana Transaction:

1. Signatures: Cryptographically prove authorization for the transaction.
2. Message: Core content of the transaction, containing: Header/Account Addresses/Recent Blockhash/Instructions



Token Swap Process on Jupiter:

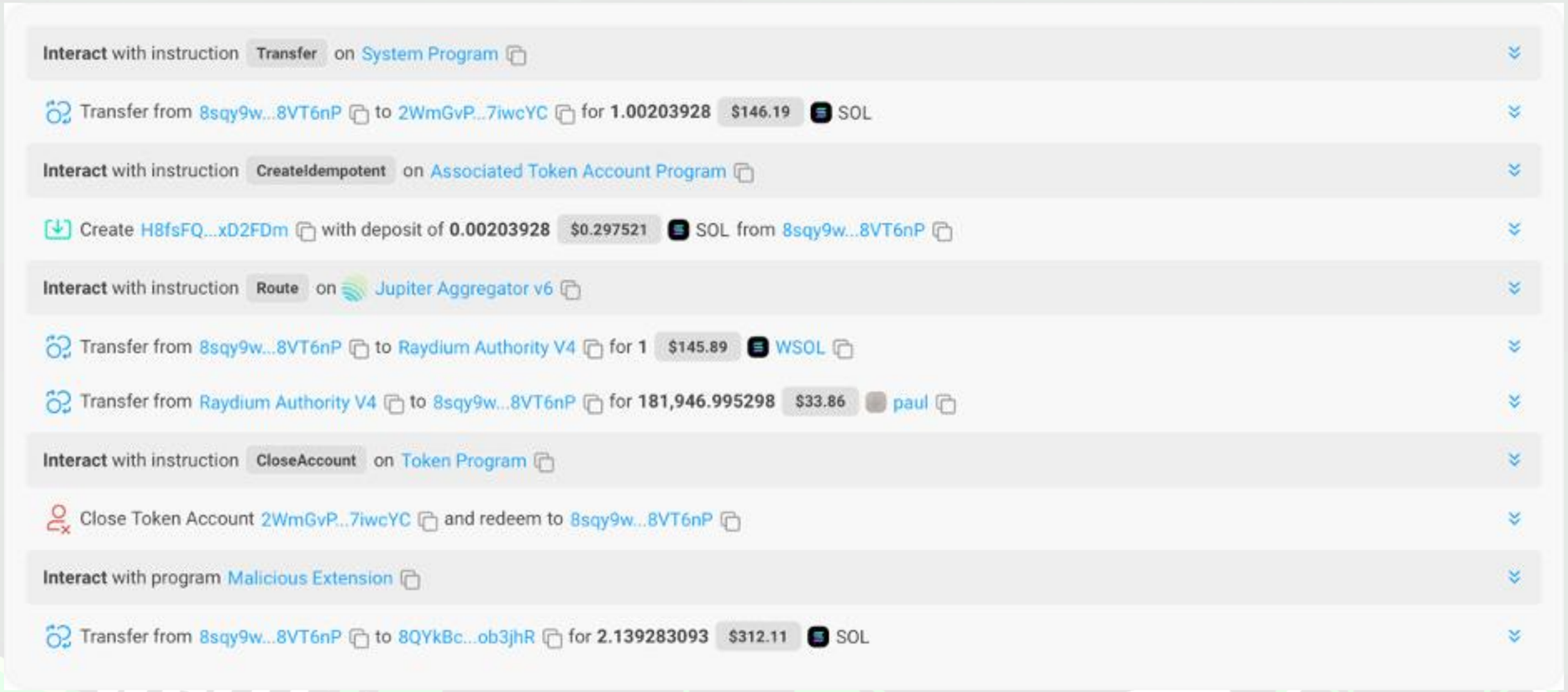
- Fetch token price and determine the **best route**.
- Assemble transaction instructions and send to wallet.
- Wallet simulates, user approves, and transaction is signed & broadcasted.

Key Risk: "Unlimited Approval":

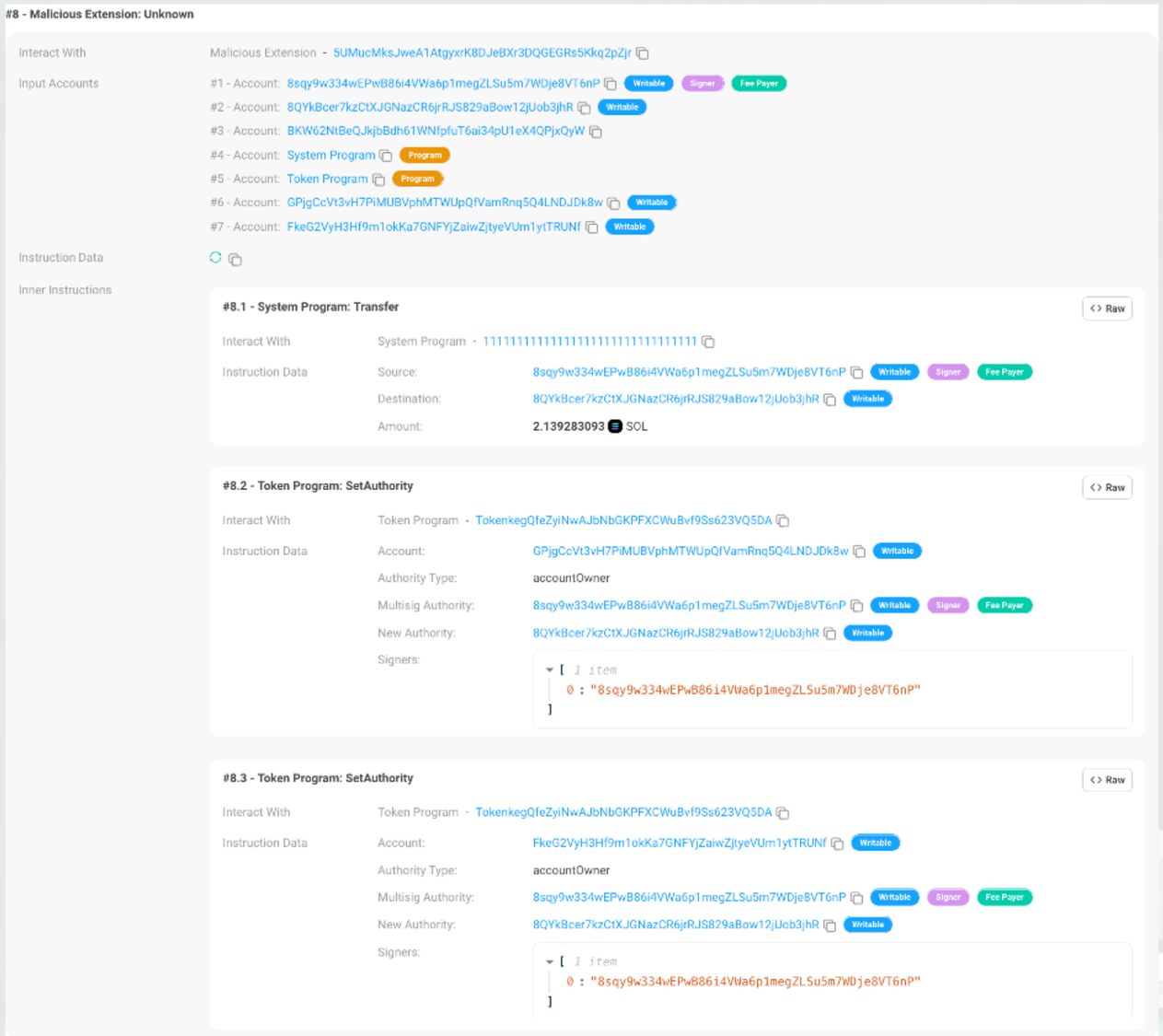
- Signing grants **full authority** over accounts in the transaction.
- Can enable **unexpected actions** (e.g., malicious token transfers).
- Users must review transactions carefully.

How the Attack Works

Case Study – Anatomy of a Malicious Transaction



These malicious instructions were made to steal funds and take control of the victim's token account while pretending to be part of the normal swap process.



How the Attack Works

Default Safeguard: Transaction Simulation

Key Feature：

- **Purpose**: Predict transaction outcomes before execution
- **How it works**:
  - Unsigned transactions sent to **Blowfish** servers for simulation.
  - Identifies threats such as:
    - Unexpected token transfers.
    - Suspicious smart contracts.
    - Rug pulls and abnormal slippage in DEX transactions.

Simulation Limitations：

- **Sophisticated attacks** can bypass checks using clever programming.
- Case studies reveal gaps in detecting advanced malicious behaviors.

Why Simulations Fail

Case Study – Examining a Simulation Log

Key Question: Why didn’t the built-in transaction simulation protect the victims?

--- The simulation failed to reveal the malicious behaviors.

1. In the log, the malicious program did not execute long or consume many compute units:

```
“Program 5UMucMksJweA1AtgyxrK8DJeBXr3DQGEGRs5Kkq2pZjr invoke [1]”,
“Program 5UMucMksJweA1AtgyxrK8DJeBXr3DQGEGRs5Kkq2pZjr consumed 1106 of 102657 compute units”,
“Program 5UMucMksJweA1AtgyxrK8DJeBXr3DQGEGRs5Kkq2pZjr success”
```

2. However, the on-chain log shows different traces:

```
Program  5UMucMksJweA1AtgyxrK8DJeBXr3DQGEGRs5Kkq2pZjr invoke [1]
Program  11111111111111111111111111111111 invoke [2]
Program  11111111111111111111111111111111 success
Program  TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [2]
Program log: Instruction:  SetAuthority
Program  TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 2875 of 97266 compute
units
Program  TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
Program  5UMucMksJweA1AtgyxrK8DJeBXr3DQGEGRs5Kkq2pZjr consumed 8480 of 102668 compute
units
Program  5UMucMksJweA1AtgyxrK8DJeBXr3DQGEGRs5Kkq2pZjr success
```

So, what happened in this program 5UMucMksJweA1AtgyxrK8DJeBXr3DQGEGRs5Kkq2pZjr?

HACKPROVE OFFSIDE LABS

2025 HACKPROVE WORLD

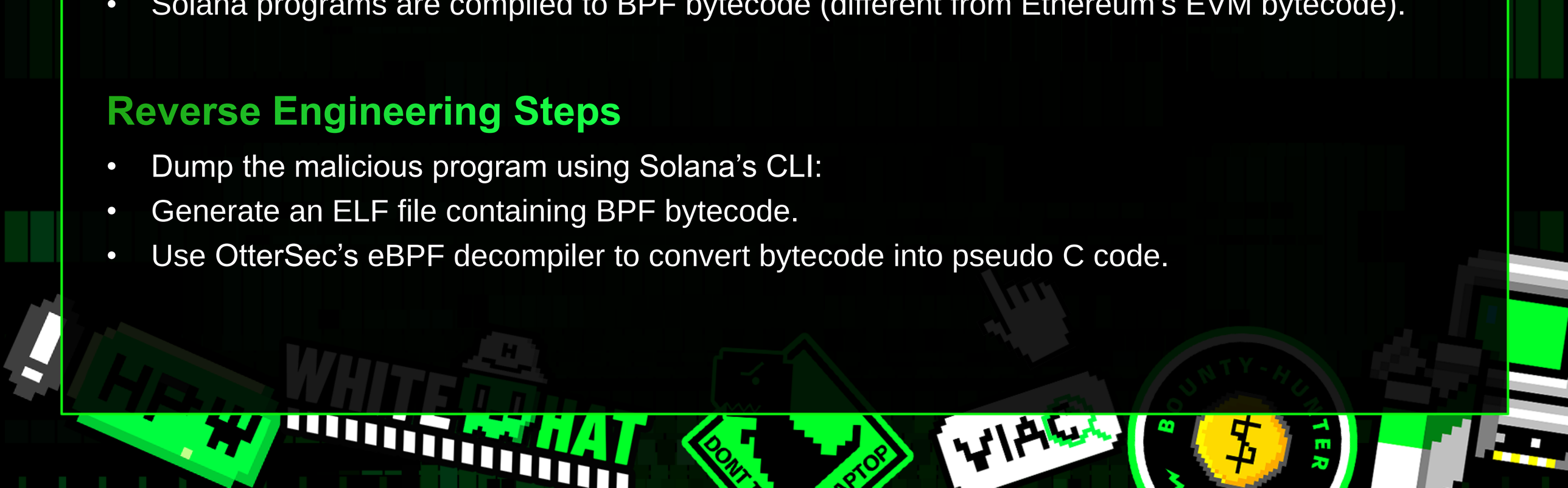
Unmasking the Malware: Reverse Engineering

Executable Program Accounts on Solana

- Smart contracts are deployed as executable program accounts
- Solana programs are compiled to BPF bytecode (different from Ethereum's EVM bytecode).

Reverse Engineering Steps

- Dump the malicious program using Solana’s CLI:
- Generate an ELF file containing BPF bytecode.
- Use OtterSec’s eBPF decompiler to convert bytecode into pseudo C code.



The decompiled pseudo c code is like this:

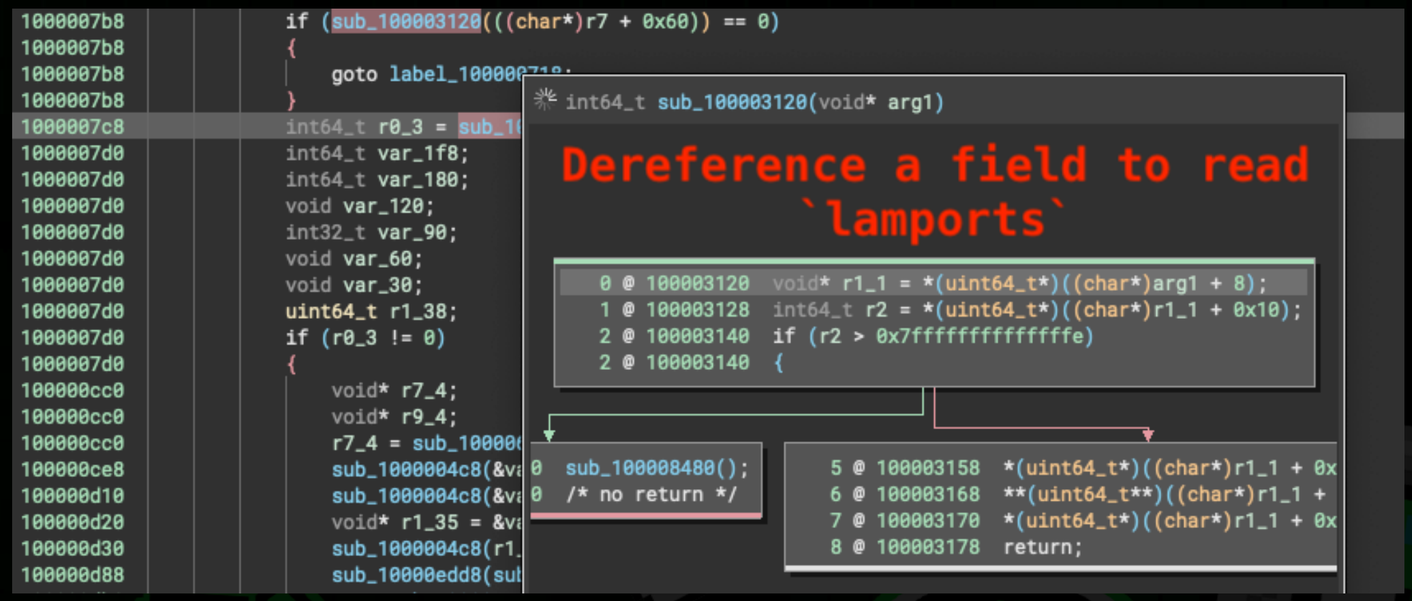
Solana ▾ Linear ▾ Pseudo C ▾

```
int64_t sub_1000005d8(void* arg1, int64_t arg2, void* arg3, int64_t arg4)

1000006c8            *(uint32_t*)r6 = r1_2;
1000006c8        }
100000678        if ((((r8 != 0 && r8 != 1) && r8 != 2) && r8 != 3))
100000678        {
1000006a0            r5_2 = ((char*)r7 + 0xc0);
1000006a8            if (r8 != 4)
1000006a8            {
1000006e0                void* r0_1 = *(uint64_t*)((char*)r7 + 0x30);
100000700                if (*(uint64_t*)r0_1 != 0x4d702b828d0f0a6e)
100000700                {
100000718                label_100000718:
100000718                    r1_2 = 0x1a;
100000720                    goto label_1000006c8;
100000720                }
100000740                if (*(uint64_t*)((char*)r0_1 + 8) != 0x2a8e185ab7b764da)
100000740                {
100000740                    goto label_100000718;
100000740                }
100000760                if (*(uint64_t*)((char*)r0_1 + 0x10) != 0x7752c6d53f4d424b)
100000760                {
100000760                    goto label_100000718;
100000760                }
100000788                if (*(uint64_t*)((char*)r0_1 + 0x18) != 0xef7a7aa1ace0d9)
100000788                {
100000788                    goto label_100000718;
100000788                }
1000007b8                if (sub_100003120(((char*)r7 + 0x60)) == 0)
1000007b8                {
1000007b8                    goto label_100000718;
1000007b8                }
```

Key Findings in Decompiled Code

- **Malicious contract uses**:
  - Early exit conditions to avoid detection during simulation.
  - Account balance checksto trigger hidden “backdoors.”
- **Embedded "magic constants" to target specific accounts.**



Example of Malicious Behavior

- Checks if an account's lamport balance is zero → exits or performs malicious actions.
- Sandwiches malicious instructions between normal transactions to evade detection.

```rust
/// Account information
#[derive(Clone)]
#[repr(C)]
pub struct AccountInfo<'a> {
    /// Public key of the account
    pub key: &'a Pubkey,
    /// The lamports in the account.  Modifiable by programs.
    pub lamports: Rc<RefCell<&'a mut u64>>,
    /// The data held in this account.  Modifiable by programs.
    pub data: Rc<RefCell<&'a mut [u8]>>,
```

Hide failed transaction(s)

Signature	Block	Time	Instructions	By	Value (SOL)	Fee (SOL)
Mi6HxScSZX2PZDPX7P5WaMncuRojTX6YhDjpbToi2...	284324557	10 days ago	transfer 1+	5hiB6cS3ev9LVZYKYcuNo6mbDMC6WPw1PL44ox1ge8gb	0.00101	0.00001
5krgaq2FTZAp9CT1X1ue4dY7JV2rSVYeNLQFgHjWK...	284324557	10 days ago	route 1+	8sqy9w334wEPwB86i4VWa6p1megZLSu5m7WDje8VT6nP	1.002005	0.002005
424fD81zbhrG2pBopeknip4QAYjbz9YMFZA5gnxvd28...	284324557	10 days ago	transfer 1+	5hiB6cS3ev9LVZYKYcuNo6mbDMC6WPw1PL44ox1ge8gb	0.001005	0.000005

Show 10 per page Item 1 to 3

What We Discovered

Root Cause:

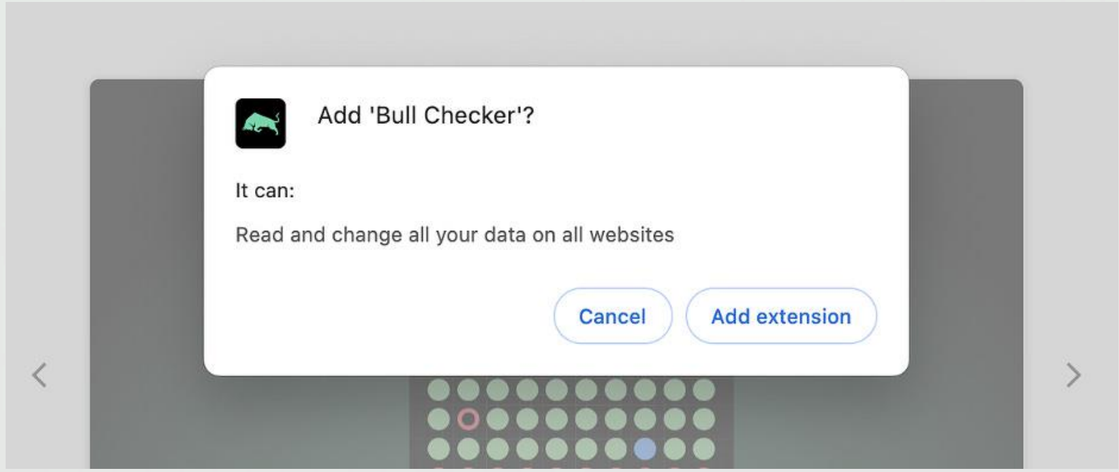
- Malicious browser extension “Bull Checker”.
- Intercepts and modifies wallet adapter functions.

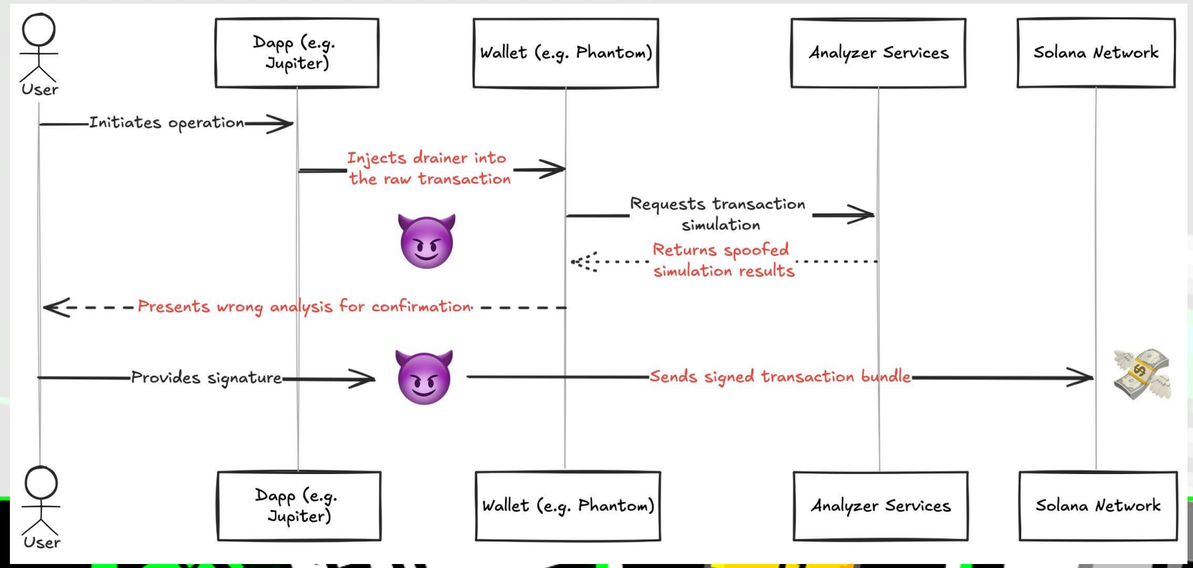
Target Audience:

- Memecoin traders via Reddit and other social channels.

Execution:

- Drainer instructions sandwiched between normal transactions.





How to Stay Safe

Avoid:

- Extensions with excessive permissions (e.g., Bull Checker).
- Unverified tools or recommendations from social media.

Use:

- Wallets with advanced transaction scanning (e.g., Phantom + Blowfish).
- Blowfish's **SafeGuard** feature to prevent simulation spoofing.

Stay Vigilant:

- Always review transactions before signing.
- Verify tools and extensions before use.

HACKPROVE OFFSIDE LABS

2025 HACKPROVE WORLD

THANKS