Attacking and Defending Blockchain Nodes

Peter Kacherginsky, Blockchain Security Engineer @Coinbase
Hello friend,

Join me on Discord:
#bcv-general-text on **DEFCON**
Peter Kacherginsky

Blockchain Security Engineer @Coinbase - BlockSec
- Blockchain Threat Intelligence newsletter
- Capture the Coin CTF
- Break blockchains and smart contracts
- Secure and monitor blockchain systems

Malware Reverse Engineer @FireEye - FLARE
- FLARE VM, FakeNet-NG, Malware Training
- Lot’s of APT malware reversing

Penetration Tester @Federal Reserve System - NIRT
- Password Analysis and Cracking Kit (PACK)
- Breaking Finance 1.0
Agenda

01. How many coins do you have?
02. Breaking the nodes
03. Node Security Threat Model
How many coins do you have?
How many coins do you have?
Let's ask the network!
What about the nodes?
Node ecosystem is complex and relies on a number of 3rd party components
FriendsCoin Node Components
We trust node software, libraries, and infrastructure to not drop, modify, or craft transactions to steal funds.
FriendsCoin Threat Model
<table>
<thead>
<tr>
<th>Date</th>
<th>Event Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>May, 2014</td>
<td>BGP Hijacked to Eclipse attack Doge and Bitcoin miner pools</td>
</tr>
<tr>
<td>November, 2018</td>
<td>EOS node code execution vulnerability. First of many.</td>
</tr>
<tr>
<td>November 2019</td>
<td>Monero website compromised to serve backdoored client stealing funds.</td>
</tr>
</tbody>
</table>

A few bad nodes
A few bad nodes

March, 2020
Solana testnet node failed to validate transaction signatures. 500M SOL were stolen.

June, 2020
FileCoin inflation bug discovered and exploited on testnet. 9B FIL minted.

July, 2020
Tendermint DoS vulnerability when parsing invalid blocks. Results in a network halt.

July, 2020
Ravencoin backdoored and inflation bug exploited. 300M RVN minted.
Breaking the nodes
**Consensus**
Consensus mechanisms, 51% attacks, reorgs, protocol design issues.

**Key Management**
Handling key material outside of node software.

**Incorrect Usage**
Not understanding protocol quirks such as Ripple's `tfPartialPayment`.

**Smart Contracts**
Layer 2 vulns in smart contracts, DeFi, etc.
IN SCOPE

**Implementation**
Protocol, software flaws, attack resilience.

**Infrastructure**
Underlying OS, network stack.

**Management**
Access management, configuration, source control
TERMINOLOGY

Threat Model
Process to identify potential threats, assess their risk, and prioritize mitigations.

Attack Surface
Sum of different attack vectors with a threat agent can interact with an environment.

Threat Agent
An entity that can manifest a threat.

Source: OWASP Threat Modeling Cheat Sheet
Let's break some nodes

Join me on Discord:
#bcv-general-text on DEFCON
Miners

External Nodes

Network Infra

Node Software

VM Interpreter

Blockchain Storage

Node Configuration

Node Administrator

RPC Client

Software Repository

Software Dependencies
Blockchain specific vulnerabilities and implementation flaws (e.g. block/tx validation, following correct fork, malicious governance upgrades, inflation bugs, etc.)

**Protocol Vulnerabilities**

Severity: **High**  Probability: **Med**
Generic Software Vulns
Severity: High Probability: High

General software vulnerabilities in node software (e.g. buffer overflows when parsing blocks/tx) and supporting software (e.g. Parity web server, EOS databases, etc.)
Backdoored node software from a compromised or untrusted resource results in unwanted behavior (e.g. Monero key stealing, Ravencoin money printing, etc.)
Supply chain attack on a critical component of the node software (e.g., encryption or consensus library) results in unwanted node behavior.
Node Administrator can have an impact on node’s availability and alter its core functionality. (e.g. shut down the node, hard-code evil set of nodes like, disable tx/block verification…)
Network Infra Exploitation
Severity: High Probability: Med

Taking over node’s supporting network infrastructure (e.g. router, switch, DNS, BGP, etc.) to perform DoS, Eclipse, and other attacks.
Node Threat Model
MORE TERMINOLOGY

STRIDE

Likelihood
Possibility of a threat event occurring where a threat actor will exploit a weakness.

Source: OWASP Threat Modeling Cheat Sheet
Generic Node Threats

<table>
<thead>
<tr>
<th>Components</th>
<th>Total Threats</th>
<th>Critical Threats</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>Threat ID</td>
<td>Component</td>
<td>Threat Name</td>
</tr>
<tr>
<td>----------</td>
<td>--------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>N-001</td>
<td>Node Software</td>
<td>Node software generic vulnerabilities</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Denial of Service</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N-002</td>
<td>Node Software</td>
<td>Consensus mechanism</td>
</tr>
<tr>
<td></td>
<td></td>
<td>implementation vulnerabilities</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N-003</td>
<td>Node Host</td>
<td>Malware on the node host</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Prioritizing Mitigations

- Severity
- Probability
- Cost
Generic Node Mitigations

- Total Mitigations: 41
- Critical: 22
- Low Cost Critical: 11
Mitigations by Cost

- Critical
- Severe
- Substantial
- Moderate
Node Security Defense Top 10
1. Secure node software repository handling

- Repository Pinning
- Verify Signatures
2. Build all nodes from source
3. Securely configure nodes

Diversify Node Connections

Lock down RPC Interfaces
4. Restrict node access

- Require consensus for administrative tasks
- Define and enforce admin roles
- Restrict internal traffic (ingress and egress)
5. Monitor and log node activity

- Investigate network anomalies for Eclipse attacks
- Check on block heights and hashes
- Log all access and network events
6. Lock down base OS

- Dockerize
- SE Linux or similar
7. Harden node’s network connections

- TLS
- BGP
- DNS
- DoS Protection

Harden network protocols
8. Consider node/protocol specific threats

- Stellar trusted nodes configuration
- EOS trusted mode
9. Verify node functionality before deployment

Rosetta Validator

https://www.rosetta-api.org/
10. Check node software with basic static analysis tools.

Salus: Guardian of Code Safety and Security

https://github.com/coinbase/salus

GoSec

https://github.com/securego/gosec

Semgrep

https://github.com/returntocorp/semgrep
Node Security Top 10

1. Secure node software repository handling
2. Build all nodes from source
3. Securely configure nodes
4. Restrict node access
5. Monitor and log node activity
6. Lock down base OS
7. Harden node’s network connections
8. Consider node/protocol specific threats
9. Verify node functionality before deployment
10. Check node software with basic static analysis tools.
Thank you!

Does anyone have any questions?