NOISESOCKET
For OSSEC
+ PureKit

BY DMITRY DAIN
CoFounder and CTO
Virgil Security, Inc.
@dmitrydain
WHAT IS NOISESOCKET PROTOCOL?

Simple wire protocol, minimal overhead

Built-in padding

Extensible

Ideal for IoT: Sensor Networks, Smart Home, Automotive etc.

Perfect for MicroServices IoT, Distributed Networks

Developed by Alexey Ermishkin and Trevor Perrin
WHY NOISESOCKET > TLS?

X.509 MUST DIE

Make every TCP Connection Secure

You control Trust

Advanced options like 0-RTT, PFS, pre-shared keys

TLS does not support raw public keys. Bringing up CA correctly is not something you’d like to deal with

Key Management is built-in and simple to use
NOISESOCKET: ADVANCING NOISE FRAMEWORK

- Developed by Alexey Ermishkin and Trevor Perrin, a co-author of Signal messenger
- Allows arbitrary data being hashed into handshake state (Prologue)
- Supports mutual authentication & 0-RTT
- Fast
- Long-term static keys
- Ephemeral keys for perfect forward secrecy
- Uses only DH, AEAD (AES-GCM, ChahaPoly), HKDF & Hash (SHA-2, Blake2)
NoiseSocket is a fundamental building block to start the process of transitioning away from the certificate authorities and centralized trust authorities.

Available in OSSEC 3.3.1
NoiseSocket is natural for p2p where NodeID is often a public key. No need for signatures/etc.

We have a Go library for that!
OSSEC 3 REPORT CARD

- You need to reboot every time you add or delete an agent
- Agent keys are symmetric keys + counters (what happens when a counter gets corrupted?)
- These keys are being copy/pasted and moved around by humans which is insecure
- Old Crypto
- No Scalable Key management
- Hard to provision for IoT devices. Scaling is difficult
• Why do you need OSSEC?

• Is there a way to protect key application information?

• Passwords, Tokens, PII etc.

PUREKIT:
Password and User Record Encryption Kit
Facebook’s Struggle to Protect User Passwords

( Yes, I know, for a bit of money they’d just give you whatever you want )
Computationally Protects from Brute Force
Single Key Forever
Limited Protection for Login Tokens
No Protection from Insider Attacks
Can We Do better?

All of the Above PLUS
We want distributed

We want full blinding

We want fluid protection - always changing, never the same
( yes, like in StarTrek! )

We want fast logins

We want to protect *any* user data: Passwords, Tokens, PII
YES, WE CAN.
AND WE DID
WARNING: COMPLEX MATH AHEAD
PUREKIT from Virgil Security
Removes all math, protocol and implementation complexity for developers

Public Key

Backend

Private Key (x)

Enroll?

sNonce, C0, C1, Proof

C0, C1 — точки

C0 = y \cdot (\text{sNonce}, \text{"0"})

C1 = y \cdot (\text{sNonce}, \text{"1"})

Private Key (y)

PHE Service

HCO = x \cdot (\text{cNonce}, \text{password, "0"})

HC1 = x \cdot (\text{cNonce}, \text{password, "1"})

M — random, MC = x \cdot M

T0 = C0 + HCO

T1 = C1 + HC1 + MC

Database:

T0, T1, sNonce, cNonce
Full Brute Force Protection w/constantly changing Keys
Yes, this is actually default state.

You can block brute force even when database was stolen!
Instant database invalidation and instant token update for zero downtime or user interaction.

Complete MITM protection(ZKP)

Optionally Distributed Nodes
Full Blinding (only your users knows password, not even you do)

Simple User Transition

Can be used for Tokens, PII and *any* other data stored in the database per account

Users no longer need to create complex passwords!
THANK YOU!

RESOURCES

- github.com/noisesocket/spec
- VirgilSecurity.com/purekit
- noiseprotocol.com

Special thanks to Rhys Weatherley and David Wong for helpful discussion