Random Escrow

Semester Project

Charles Smutz

Outline

- Goal
- Problem
- Alternative Solutions
- Solution
- Demonstration
- Analysis
- Future Work

Goal

To demonstrate the value of OS level random number escrow by refining a proof-of-concept system which allows a system administrator visibility into cryptographically protected data in a manner that would not otherwise be possible

Problem

- Cryptography hampers forensics/monitoring
    - Examples:
        - File system anlaysis (PGP)
        - IDS (SSL, SSH)
- “Serious” Environments
    - Malware reverse engineers
    - Serious forensic capabilities on hosts
    - Full packet capture on network
    - No expectation of privacy

Alternative Solutions

- Key Escrow
  - Requires Compliance
  - Doesn’t address Forward Secrecy
  - Sometimes requisite key is owned by other party
- Man in the Middle
  - Only works on network
  - Destroys/Complicates Trust Bindings
  - Doesn’t address tunneling
- Data Escrow
  - Requires Compliance
  - Scalability/Efficiency issues
- Brute Force
  - Not practical in most situations

Solution

- OS level Random Value Escrow

Random Values in Crypto

- Use of Random Values pervasive in Crypto
  - Long Term Keys
  - Session Keys
    - Essential to Forward Secrecy
  - Nonces
  - Session IDs, Port #s
- If random values aren’t random, crypto breaks
  - If you know random values used in crypto, often can decrypt or aid in decryption

Canonical Examples

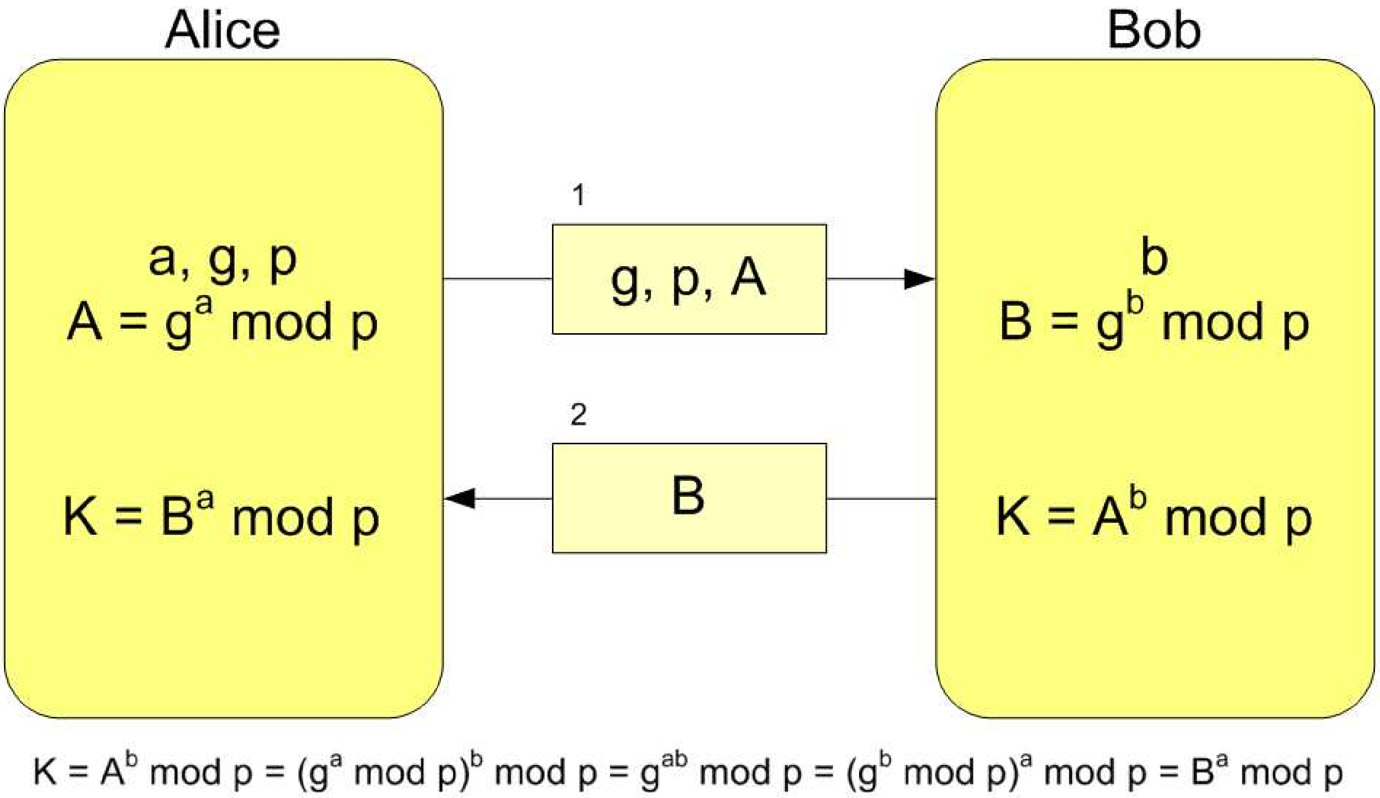
- Secure File
  - Generate Session Key (from random)
  - Encrypt File with Symmetric Cipher, Session Key
  - (hash file)
  - (sign hash)
  - Encrypt Session Key

Canonical Examples

- Secure Stream
  - Negotiated Session Key (from random(s))
    - Forward Secrecy
  - Authentication
  - Encrypt/(Integrity Checks) for rest of data in stream

Forward Secrecy

- (Ephemeral) Diffie Hellman



Taken from: http://upload.wikimedia.org/wikipedia/commons/a/a3/Diffie-Hellman-Schl%C3%BCsselaustausch.png

Real Protocol--SSH

- SSH: Less complicated than others (SSL, IPSEC)
  - In practice:
    - Fewer negotiation options
    - Always provides forward secrecy
      - Uses DHE with world known g,p

Implementation

- Kernel Patch
  - Intercepts calls kernel level functions
    - get_random_bytes()
    - urandom_read() -- /dev/urandom
    - random_read() -- /dev/random
- Escrow values using klog/rsyslog
    - Haven’t addressed security/privacy concerns
    - Filtering, routing through standard mechanisms

Implementation

- Scripted Recovery of Session key
  - Inputs
    - Random Value used to create DH keypair (client)
      - Replay escrowed value through same algorythm
    - Public key from other side (server)
      - Taken from network packet capture

Demonstration

See VM

Analysis

- Shows technique works on real protocol
- Demonstrates ability to correlate random value to key generation
  - Time
  - Size
  - Source
    - Size, Source based on implementation
- Only useful in certain environments
- Still requires key escrow in some situations
- In many situations, not useful to attackers

Future Work

- Full protocol decoder
- More protocols
- Security of escrowed values

Project Homepage

http://mason.gmu.edu/~csmutz/re

Questions?