Advanced Unix System Administration

Lecture 22
December 8, 2008

Steven Luo
<sluo+decal@OCF.Berkeley.EDU>

Digital Cryptography

- What is crypto?
  - Most prominently, set of algorithms, but these algorithms are bundled in a system!
- What can crypto do for you?
  - Confidentiality – ensures that other people can't read your data
  - Authentication – ensures that a message comes from its claimed source
  - Integrity – ensures that data hasn't been modified or corrupted

Digital Cryptography

- What does crypto NOT do for you?
  - Address unrelated weaknesses in your security model
  - Fix weaknesses in your implementation of a cryptosystem
  - Prevent user errors
  - Make users actually use your security system!
- Cryptography is NOT a magic bullet!
  - Crypto is well-studied; it's usually easier to find another way in

Digital Cryptography

- Cryptographic primitives
  - Ciphers
    - Makes data unreadable except by holder(s) of a certain secret (the secret part is important!)
    - Symmetric ciphers share a secret for encryption and decryption, public-key ciphers only use a secret at decryption time
      - Public-key ciphers: RSA, ElGamal
      - Symmetric ciphers: DES, AES, RC4, ...
    - Block ciphers work on chunks of data, stream ciphers work on individual bytes
      - Block ciphers: DES, CAST family, Blowfish/Twofish, AES
      - Stream ciphers: RC4

Digital Cryptography

- Cryptographic primitives con't
  - One-way hash functions
    - Take data and produce some short stream of bytes which somehow “identifies” that data
    - Important properties:
      - Small changes in the input should produce large changes in the output (avalanche effect)
      - Collisions should be difficult to generate
      - Should be difficult to calculate the original datastream from the hash
    - Not as well-studied as ciphers
    - Examples: MD5, RIPEMD-160, SHA family
    - NIST competition for next-gen hash ongoing

Digital Cryptography

- Applications of crypto
  - Encrypting data
    - Apply a cipher to the data
    - Doesn't provide for integrity!
  - Data verification
    - Compare the hash of the data with a known hash
    - Where does one get the hash from?
  - Digital signatures
    - Concept: encrypt something with your private key, so that people can identify it as coming from you

Digital Cryptography

- Applications of crypto con't
  - Hybrid cryptosystem
    - Hash the data, sign the hash, encrypt the data
    - Provides confidentiality, authentication, and integrity verification
  - Communications security
    - In general, you want to verify who you're talking to before you talk to them, so verify their public key
      - Has to be some infrastructure to verify this key!
    - Public-key algorithms are slow, so select a symmetric key via the PK-secured channel and switch to the symmetric cipher

Digital Cryptography

- Applications of crypto con't
  - Secure authentication
    - Lots of possible schemes:
      - Require the user to sign some data of your choosing
      - Use hash functions on a secret
      - Send a secret over an encrypted channel
    - Carelessly designed schemes can be quite insecure!
      - Just accepting the hash of a secret alone leaves you open to replay attacks
      - What happens if one of the endpoints is malicious?

Digital Cryptography

- Attacks on crypto
  - Cipher attacks
    - Data recovery: ability to read a message faster than brute force
    - Key recovery: ability to find the key used from the encrypted messages alone
  - Hash function attacks
    - Collision: generate two reasonably-related things that hash to the same value
    - Reversing the hash: find possible inputs to the hash from the output alone

Digital Cryptography

- Attacks on crypto con't
  - Brute force
    - When your secrets are too small, it's possible to reverse a computation in a “reasonable” amount of time
    - What values provide “enough” security?
      - Symmetric ciphers: 128 bits
      - Public-key ciphers: 2048 bits
      - Hash functions: 256 bits
      - Note that computers and attacks improve with time!
  - Side-channel attacks
    - Lots of creative ways: measure power draw, time to run algorithm, etc.

Directory Services

- What do directory services do?
  - Classically, a directory service stores information about users and computers on a network
    - Internal address/telephone/email directories were some of the earliest applications for LDAP
  - In a system administration world, you frequently hear about it in the context of providing name and address lookups
    - Directories are optimized for fast read access via the network – hence the suitability

Directory Services

- Network Information Service (NIS/YP)
  - Another Sun RPC based service
  - Simple request-reply protocol, types of data available limited (passwd, hosts, etc.)
    - A limited generic “map” facility is available
  - Data is stored in flat files, with separate flat files needed for each field which can be looked up
  - No over-the-wire security

Directory Services

- NIS+
  - Looks sort of like NIS, but isn't NIS
  - Supports arbitrary key lookups and (weak) over-the-wire security
  - Also allows arbitrary data to be stored
  - Not known for its reliability and support
- Netware Directory Services, Windows NT domains
  - Similar design and scope as NIS

Directory Services

- Lightweight Directory Access Protocol
  - Originally intended as a lightweight, non-OSI-stack based method of accessing X.500 directories
  - Protocol quickly was implemented standalone
  - Can store any type of data, in any organization
    - The data types and the structure are defined in “schemas”
    - With flexibility comes complexity
  - Lots of implementations

Directory Services

- LDAP con't
  - Lifecycle of an LDAP connection
    - The client “binds” to the server, authenticating and negotiating protocol version
    - Stream of requests issued
      - Various search operations and compare operations – LDAP mandates powerful built-in filters
      - Add, modify, delete entries
      - New operations can be defined via the “extended operation” operation
    - “Unbind” is the connection close
  - The use of TCP imposes some overhead

Directory Services

- LDAP con't
  - LDAP security
    - LDAP itself provides no over-the-wire security; connection security is usually managed via SSL/TLS
    - Secure authentication methods are provided by SASL, if both client and server support it
  - Use for Name Service Switch lookups
    - RFC 2307 (and later drafts) defines a standard schema for storage of this data that fits in with the standard schemas for other uses of LDAP
    - Overhead is considerable – use a caching service!

Directory Services

- Active Directory
  - Embrace-extended version of LDAP and Kerberos
    - Standard (if buggy) LDAP and standard Kerberos – but Microsoft uses a proprietary Kerberos-over-LDAP protocol to provide security
  - Can be used to store standard LDAP schema data, as well as M$-proprietary schema data
    - Allows storing client policy and configuration
  - Interoperability with Unix can be difficult, but is possible

Network Storage

- Two forms of network storage:
  - Disk-level access – give each client access to the underlying physical disk
    - Clients do reads from the disk and have to understand the filesystem themselves
    - Server can be simple, but more network I/O
    - SAN implementations (Fiber Channel, iSCSI)
  - File-level access – clients operate on files
    - Clients request files, server fulfills requests
    - Server is more complex, saves on network I/O
    - NFS, SMB/CIFS, AFS/Coda ...

Network Storage

- The Network File System (NFS)
  - First developed at Sun in the 1980s, now controlled by IETF
  - Versions 2 and 3 attempt to provide (mostly) stateless operation, to simplify crash recovery
    - Features requiring state (locking) are implemented in additional protocols
  - Attempts to provide POSIX-like semantics within the possibilities of stateless operation
    - Note that NFS is NOT a POSIX-compliant filesystem!

Network Storage

- The Network File System (NFS) con't
  - Lifecycle of an NFS request
    - Client requests a “file handle” from the NFS mountd
    - Client requests that the nfsd perform operations on this file handle
      - Operations should be simple, atomic, and idempotent
      - In reality, they're not always
    - nfsd returns only when it has finished the operation or the operation has failed

Network Storage

- The Network File System (NFS) con't
  - Problems (in traditional NFS)
    - It's difficult to guarantee that repeating operations will be safe
    - Some features of POSIX filesystems, especially those requiring atomicity, are impossible to implement
    - The default mode can leave clients hanging for a very, very long time on a server crash
    - Security depends on the client saying who it is

Network Storage

- Other network file systems
  - As a rule, all maintain state
    - Allows for more complex commands, more featureful semantics, but makes crash recovery difficult
  - Most implement richer access control
    - The exact model tends to differ – AFS, CIFS, and NFSv4 offer separate, differing access control models
    - Over-the-wire security may also be present
  - Some provide for distributed operation

Network Storage

- iSCSI
  - A translation of the SCSI command set to a network
    - SCSI topology already looks a bit like a network, so this isn't too hard
    - Reliability questions need to be dealt with
  - An “initiator” on the host connects to storage on a “target”
  - Simultaneous access from multiple initiators requires filesystem support (Red Hat GFS, OCFS)

Email

- A brief history of Internet email
  - Mail between users on multi-user machines had existed since the 1960s
  - With the advent of ARPANET (1970s), mail began to be transferred over networks
    - Initially, mail transfer was performed over FTP
  - Sites connected intermittently (usually via dialup serial link) transferred mail using UUCP
    - UUCP allows for transfer of files over serial links
    - ARPANET-UUCP gateways allowed much faster transfers

Email

- History con't
  - Without direct connections between all sites, and no way of determining how best to get a message there, senders specified the routing of their messages
    - The infamous bang-paths: ...foovax!barbox!quuxnet!me, where presumably everyone could find a path to foovax
    - People would often specify paths from multiple hosts: ...{ucbvax,menlo70}!barbox!me
    - Classic example: ...{decvax,philabs}!mcvax!moskvax!kremvax!chernenko

Email

- The modern era: SMTP
  - The protocol is defined in RFC 2821, the message format in RFC 2822
  - Messages are transferred in a simple text format: headers, newline, body
    - Headers: From, To, Subject, Date, ...
    - Content of the body may not be plain text in modern usage – see the MIME RFCs
    - Note that the information in the headers is NOT used in routing the message

Email

- SMTP con't
  - Routing an email message
    - The destination of an email is determined by its envelope recipient, specified to the server on sending
    - The sending mail server looks for an MX record for the destination site to determine who to talk to
      - MX records are tried in order of the priority given
      - If no MX record is present, an MX record pointing to the name itself with priority 0 is assumed
    - If the message doesn't get there, the envelope sender is returned a bounce

Email

- SMTP con't
  - A typical SMTP conversation
    - `HELO hostname` or `EHLO hostname`
    - `MAIL FROM: <envelope-sender@site>`
    - `RCPT TO: <envelope-recipient@othersite>`
    - `DATA`
    - `QUIT`
  - The server replies with the traditional 1xx-5xx codes

Email

- Problems with SMTP
  - No identity verification – anyone can send email claiming to be anyone else
    - In fact, most mailing list software depends on this behavior
    - One of the enabling factors behind spam, blowback bounces, etc.
  - Legacy compatibility causes problems
    - Long timeouts, retries, etc. for unreliable networks lead to messages spending in mail queues before bouncing
    - Bang paths still (theoretically) supported!

Email

- Mail Transfer Agents
  - sendmail
    - The original Internet mailer, an extension by Eric Allman of the original UUCP-era mailers
    - Complex configuration, but can do (literally) anything
    - Inefficient, slow, terrible security record
    - Includes a nice mail filtering architecture in newer versions (milter)
    - Not recommended for new installs unless you have special needs

Email

- MTAs con't
  - qmail
    - Dan Bernstein's attempt at a secure mailer
    - Multi-binary, privilege-separated architecture provides better security and faster mail delivery
    - Introduced maildir mail storage (one file per email); often faster than mbox (one file per mail spool)
    - Now public-domain and “maintained” by a (vocal) community; has some quirky behaviors
    - Not recommended for new installs

Email

- MTAs con't
  - Exim
    - Developed at Cambridge as a fast Internet mailer
    - Single-binary architecture sometimes criticized, but has reasonable security record
    - Extremely fast in environments where most mail is deliverable immediately, but has poor queue management
    - Can embed Perl to process email

Email

- MTAs con't
  - Postfix
    - Wietse Venema's attempt at a secure mailer
    - Multi-binary, privilege-separated architecture
    - Very fast, simple configuration
    - Some processing isn't possible inside Postfix, but external hooks are possible
      - Such filtering setups can be a bit complex
    - Sendmail milters are supported to some degree in newer versions