Trusted Computing

1. Student presentation
2. Questions from reviews:
    a. Used in real world?
    b. Mobile devices?
    c. App-specific OS?
    d. Device drivers?
    e. Revocation – key compromise?
        i. Example – DigiNotar
            1. Cert must be revoked by browsers; all certs it issued must be reissued
        ii. Comodo – registration authority compromised
            1. Couldn't verify businesses asking for certs were the real business
    f. Poor isolation in OS:
        i. Shared resources in kernel (/proc, filesystem)
3. Comment:
    a. Done in 2003; vmware just released in 1999 but not widely used
4. Public key crypto / protocols
    a. General infrastructure: CA hierarchy
    b. How do CA's protect key?
    c. MS example:
        i. locked room off campus
        ii. Armed guard
        iii. Tamper-proof hardware : chip surrounded by mercury that shorts out storage
5. Problem statement
    a. What can applications trust today?
        i. Can they trust the system to keep data private
        ii. Can they games trust the user not to cheat
        iii. Can banks trust client banking software not to disclose secrets?
    b. Fact: today, physical access to a computer guarantees you full access to the information **on that computer**
        i. Question: How?
            1. Debuggers
            2. Virtual Machines
            3. Hook audio / video path
        ii. Any technique only makes it harder, but not impossible

   c. What if you have information so valuable you can't entrust it to the person using the computer?
      i. Goal: Root Secure
      ii. QUESTION: Examples?
         1. Digitally encoded songs / movies
            a. Don't trust user not to give away / sell
         2. Sensitive legal documents
            a. Don't trust user not to leak to competitors or press
         3. Networked applications – multiplayer games
            a. Don't trust users not to cheat
         4. Secure video conferencing devices
   d. Insecure OS allows one app to subvert all others
      i. E.g. WinXP sp2 used in a botnet of 73,000 machines capable of sending 1 billion individual spam messages a day (bot runs anti-virus/anti-spyware to remove other bots)
6. Bad solution: closed platforms
   a. Lock the machine
   b. Obscure the hardware
   c. Make it tamper-proof
      i. Circuits that short-circuit if opened
      ii. Example: Xbox
7. Solution: attestation / digital
   a. Two General frameworks:
      i. Layered code 0 (hardware) → Y (application)
      ii. At run time, code at layer X checks signatures of layer 0 .. X-1 and X (check layers below)
         1. If correct, knows that is running on correct platform
         2. QUESTION: Why can you trust the signature of the layer below?
            a. A: the hardware signatures are trusted
            b. Can follow the chain of trust
      iii. Code at layer X checksums / verifies layer X+1 (check layers above)
         1. Can verify next layer up is not corrupt if needed
         2. Can be used to make sure a computer only boots trusted code
      iv. QUESTION: What is the difference?
         1. Windows/Chrome OS/IOS use option 2 – check layer above
8. Hardware support in TPM

        a. Endorsement key – 2048 bit random private/public key; private key never leaves chip.
            i. Used to sign a random number – show knowledge of endorsement key
        b. Memory curtaining
            i. Memory not available to OS – to store secure software
        c. Sealed storage
            i. Encrypted non-volatile storage that can only be decrypted if software with correct hash is running
            ii. Need to present certified hash to decrypt the data
9. Implementation of Terra – using 1st framework of checking layers below
    a. Problem with second: greatly restricts what you can run. How do you get flexibility and openness?
    b. Vision: virtual applicances
        i. Code can trust it runs in the right place
        ii. Can have a customized OS for each applications
        iii. QUESTION: Is this reasonable?
            1. Yes – change configuration of Linux, use BSD / WinXP Embedded, Flux OS kit
    c. Layers
        i. Hardware
            1. Provides secure storage, secret key
            2. HW must attest to booted system
                a. Checksum & sign TVMM or boot loader
            3. Sealed storage – tamperproof storage controlled by coprocessor, only allows OS with same hash to unseal – for storing private keys
            4. Can buy today!
        ii. Virtual machine
            1. Provides closed & open box environment
                a. Close box cannot be inspected or manipulated by owner
            2. Management VM allows
                a. Starting
                b. Stopping
                c. Installing devices
            3. Relatively small monitor – economy of mechanism, keep it small
        iii. Multiple OSs
            1. Can run any complexity of OS you desire
            2. Communicate with virtual NIC – treat other VMs as external

    iv. Applications

d. Attestation – provide a signature chain showing what code is running
    i. 2 issues
        1. What code is running? Don't care what code, but want to get verifiable checksum of code
        2. Is the right code running? Can compare against known certificates
    ii. Each layer is checksummed & signed by lower level
        1. Hardware checksums firmware
        2. Firmware checksums loader
        3. Loader checksums TVMM
        4. TVMM checksums VMs / OS
        5. NOTE: you can buy the hardware today + MS includes code that uses it (IBM Laptops have it)
    iii. Checksum includes:
        1. Hash of all persistent state
        2. E.g. bios, executable code, constant data
        3. Does not include removable disks
    iv. For attestation: a layer can call layer below for an attestation
        1. Layer X provides public key to layer X-1
        2. Layer X-1 signs checksum of Layer X state and key
        3. QUESTION: Why include the public key?
            a. A: so can sign for the layer above – makes a chain
        4. QUESTION: What does this tell you?
            a. Which code is running, not whether code is the right code or not
    v. **Example**: Quicken
        1. On secure connection to server, pass list of attestations for all layers of software
            a. HW signs bios, bios signs loader, loaders signs TVMM,
                i. TVMM signs for VM – including OS plus quicken (a packaged appliance)
            b. Send keys for each layer – public key for hw, bios, boot loader, TVMM, OS, etc.
            c. Send certificate for HW from vendor
            d. Verify

    i. Check certificate for HW against vendor (e.g. intel)
    ii. Check hash of next level software & signature using public key provided
    iii. Verify VM hash (OS+quicken) against certificate from vendor (obtained somewhere, could be the VM itself, embedded)

vi. How to use:
  1. SW calls lower levels for attestation, pass certificate chain to other side of a channel
  2. Other side (server) has database of correct hashes, verifies that it is correct
  3. Must do within an authenticated channel to avoid man-in-the-middle attacks

vii. Two certificate chains:
  1. One external: certify the software is known to be good (guest VM, TVMM, its bios)
     a. CA certifies quicken, Quicken certifies a certain VM is good,
     b. CA certifies vmware, vmware certifies TVMM
  2. One internal: certify what code is actually running (based on CPU key on up)
     a. CA certifies HW – Intel, Intel certifies key in CPU (tamper resistant), HW certifies bootlaoder/TVMM, TVMM certifies guest VM

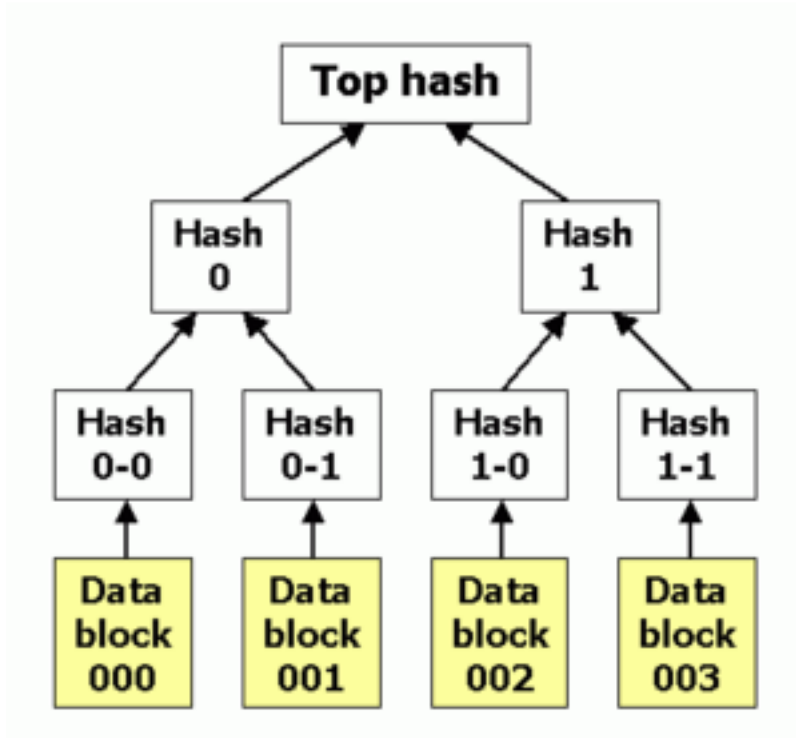
viii. QUESTION: How do you know whether to trust the code that is running?
  1. Server keeps certificates of known good code
  2. Certificates come from vendors:
     a. Intel for processor/ firmware
     b. MS for OS
     c. Quicken for application

ix. QUESTION: Who needs to trust CA? If CA broken, who has to update? How do you securely update?
  1. A: Update servers that validate clients (or everybody in peer-to-peer)
  2. Problem: no way to bootstrap trust to new CA. Need to roll over to another, unbroken CA instead.

x. QUESTION: What if HW key (e.g. Dell, Intel) is broken?
    1. Can no longer trust anything signed by that company; must put new keys/certificates in all hardware securely
xi. QUESTION: How much trust should you put in attestation?
    1. Can trust the right code was loaded
    2. QUESTION: can you tell it is still running?
        a. Not really – could have executable data
        b. Still have bugs in code that could be subverted
    3. Hardware may not be tamper proof
        a. Someone could read secrets from DRAM, modify contents of DRAM externally (e.g. DMA from a device)
xii. QUESTION: Who does the checking?
    1. In Terra, a remote network machine
    2. QUESTION: How do it locally? Have to check at layer X-1
        a. Need to install a key for app, or have a certificate chain for certifying apps
        b. QUESTION: What are implications?
xiii. QUESTION: what about patching?
    1. A: must bundle in service packs
    2. A: reduced size OS means reduced patches
xiv. QUESTION: How limit consumer choice?
    1. Services can refuse to work with clients running non-approved software
        a. E.g. Windows phones, Linux desktops, non-Chrome browsers
        b. Example services: Netflix, Spotify –want to make sure you don't steal video/music when playing

10. Control over devices
    a. TVMM/trustworthy computing separates out platform user from platform admin
        i. Admin – boots TVMM, controls system booting
        ii. User – choose what VMs to load
            1. TVMM just testifies/attests as to what runs, doesn't stop anything
    b. "root secure" – hw is secure from root user or physical access

            i. no boot into single-user mode or enter bios commands to bypass security
    c.
11. Implementation
    a. Device access
        i. QUESTION: do you need to trust devices? How much?
        ii. Storage: 3 levels
            1. Encrypted with per-VM key
            2. Integrity checked: public but not secret data, e.g. binaries
            3. Unchecked – for sharing
            4. Choice depends on performance needs
        iii. Network: rely on higher-level protocols for security
            1. E.g. ssl, ipsec
        iv. Other devices
            1. Must prevent DMA into memory – need HW
                a. IOMMU does remapping to prevent write to arbitrary memory
            2. Where do device drivers run? In TVMM or in VM?
                a. If can partition devices, can run in VM
                    i. No need for sharing; just needs access to MMIO space + interrupt delivery
                b. If can have untrusted drivers, can run in VM
                    i. Q: What do you need?
                        1. Device encrypts, OS decrypts
                    ii. What does this protect?
                        1. Sniffing, but not human eyes
            3.
    b. Performance
        i. Problem: checksum of a large data item is slow
            1. E.g. entire virtual disk
            2. Observation: don't typically read entire disk at once
        ii. Instead: want to checksum individual blocks for fine-grained performance
        iii. Partial checksums of data at layer X
            1. Merkel Trees - http://en.wikipedia.org/wiki/Hash_tree



2.
3. Property: verify log(N) checksums on block access

iv. Ahead-of-time attestation:
1. Hash entire code for next stage of computation before executing
   a. E.g. bootloader

v. Optimistic Attestation:
1. VM Specifies desired checksum in advance
2. Assume checksum correct
3. Check blocks when first accessed, not when loading code
4. Only fail if find a block with bad checksum
5. QUESTION: Why does this work?
   a. A: always fail before bad block is used

12. Hardware needed
   a. Attestation
      i. Can checksum code & sign before running
   b. Sealed storage
      i. Encrypt data tied to hash of OS
         1. Only OS with same hash can decrypt
         2. Used by TVMM to store its private key – can only be retrieved by TVMM with same hash
   c. HW virtualization
   d. Secure IO
      i. Devices that encrypt data

        ii. Simple encrypted interface, full untrusted interface (e.g. 2d graphics for TVMM, 3d for everyone else)
    e. Device isolation
        i. TVMM cannot be modified by a device
13. Political issues
    a. Who designs systems like this?
        i. Stanford – willing to play in commercial space
        ii. Berkeley / MIT – more likely to build systems to defeat this
    b. Who benefits from systems like this?
        i. Music companies, movie companies
            1. Encode data such that needed trusted environment to play data
        ii. Software companies
            1. Require trusted software to prevent piracy
        iii. Large SW companies
            1. Can make file formats that can only be read by their code
            2. Can lock customers in
            3. If have DRM on mail, for legal reasons may need to transfer DRM – hard to move between vendors
            4. Content providers may tend to certify only a small number of large companies – lock out small companies
        iv. Game companies / ethical gamers
            1. Fewer cheaters
        v. Small companies:
            1. Can deploy their own platforms
            2. Less likely to be on existing lists of "trusted softare/firmware/hardware"
        vi. Users?
            1. Not clear what their benefit
            2. Claim: Content providers will provide more content if it can be made secure
            3.