Designing and Attacking DRM
Nate Lawson | Root Labs | 2008/4/11 | Session Code: HT1-402
If you pay attention, you’ll learn…

• Why software protection matters
• Attack methods
  – Tools & techniques
• Design principles
  – Model: mesh vs. chain
  – Anti-debugging is no panacea
• Update: who’s cracked or not
My background

• Root Labs founder
  – Design and analyze security schemes
  – Emphasis on embedded, kernel, software protection, and crypto

• IBM/ISS
  – Original developer of RealSecure IDS

• Cryptography Research
  – Co-designed Blu-ray disc content protection layer, aka BD+
In the past, DRM was simple

- Copy protection (1980’s)
  - Simple goal: only run from the original floppy media
  - No privacy problems
  - Relatively harmless
    - Some games did format themselves if not properly cracked
  - Legal restrictions only via copyright, not on analyzing schemes or releasing tools
Today, bad DRM is too common

• Goals overly complex
  – Can share among a few devices but not too many
  – Expiration dates, limited number of plays

• Privacy problems
  – Network access
  – Cookies

• Collateral damage when it goes bad

• Dangerous and unclear legal environment

• If there’s going to be DRM, make it simpler!
Definition: software protection

- Software protection is the technical enforcement mechanism
  - DRM is the policy, a different talk
- Composed from:
  - Integrity protection
  - Obfuscation/anti-debugging
  - Encryption/key management
  - Renewability
Why you will care
Laptop disk encryption

- Common myth: software protection only matters for DRM
- Nope!
  - “Cold Boot Attacks on Encryption Keys”, Halderman et al
    - Extract FileVault key from RAM after a reboot
    - Or, freeze RAM with cold spray and move to another computer
- If you use software encryption, you need effective obfuscation
  - Since the key is somewhere in RAM, key hiding vital
  - Exactly same problem as DRM
Exploit protection

• If you want to make exploits difficult, obfuscate
  – Address layout randomization
  – Initial stack configuration
  – Future: compilers will randomize code generation

• Integrity checking
  – Stack canaries
  – Vista PatchGuard (aka Kernel Patch Protection)

• Same measures as DRM
Malware analysis

- Malware hides with same techniques as DRM
  - But worse!
- DRM designers have the harder problem
  - “Do no harm”
    - Debuggers must be allowed for legitimate work
    - No persistent modifications
    - Avoid interfering with Vista PatchGuard

- All these areas require knowledge of the same techniques as designing or analyzing DRM
Designing software protection
Software protection design principles

• Mesh vs. chain
  – Protection measures should be mutually enforcing
    – Example: two threads hashing each other
  – A chain is a long series of links
    – Failure of any one link and it falls apart

• Full integration with application functions
  – Protection must be intertwined with main functionality

• Renewability
  – Provide yourself a way to repair hacks
Integrity protection

• Goal: respond to modifications
  – Explicit: detect and perform some action
  – Implicit: modifications directly cause change
• Many things can perturb the environment
  – Patching
  – INT3 breakpoints
  – Attaching debugger
• Implicit integrity protection better than explicit
  – Check/response are not separable logic
  – Prevent “divide-and-conquer”
Obfuscation/anti-debugging

- Goal: make inspection more difficult
  - Obfuscation: logic becomes harder to understand or patch coherently
  - Anti-debugging: respond to the act of inspecting the program
- Better if mixed with integrity protection
  - Example: hash the results of various anti-debugging checks and use as key to decrypt next function
- Most anti-debugging is poorly-implemented
  - Single point checks with if/then logic
Encryption/key management

- Encryption in software protection is a type of obfuscation
  - Cipher+key in some form present in memory

- Key management
  - Broadcast encryption
    - Encrypt content key with lots of keys
    - Stop doing this for known-hacked keys
  - Software protection can make this more versatile
    - “You must be at least this unharmed to decrypt the video”
Renewability

• Every scheme needs a survival plan
• Online updates
  – Common in a PC environment
  – Caveats: version roll-back, privacy
• Piggy-backing on content
  – Better in embedded/cross-platform environments
  – Caveat: heterogeneity complicates testing
  – Examples
    – DTV/Dish send updates in channel stream ("ECMs")
    – BD+ puts protection logic on each Blu-ray disc
Tools & techniques
Historical reverser’s toolbag

• In the past
  – debug.exe
  – SoftICE with iceext
  – IDA Pro doing static disassembly
  – Hex editor

• Today
  – IDA, Ollydbg
  – Bindiff, binnavi
  – Paimei
  – Custom tools
Tip: go to ring 0

• Even if you’re attacking a user-mode app, kernel access gives a powerful viewpoint
  – Full access to thread state
  – No modification of process memory (i.e., int3 instruction)
  – Access to page protection, MSRs, etc. allow stealth schemes

• More advanced
  – Patch bochs/qemu
  – Write your own hypervisor debug stub
WinDbg is awesome

• Powerful tool for reversing
  – Kernel and usermode access
  – Automatic symbol server
  – x86 32 and 64-bit support
  – Extensions for common tasks (“!peb”)
  – Open plug-in interface
  – Free

• Downsides
  – No third-party CPU extensions (sorry ARM)

• Let’s make it the next SoftICE
Tip: use debug registers creatively

- Intel processors have hardware breakpoints
  - DR0-3: addresses to be monitored
  - DR6: status bits that describe which event occurred
  - DR7: configures the type of event to monitor for each address

- Interesting side effects
  - DR4-5 are aliases for DR6-7 if the CR4.DE bit is clear
  - DR7.GD bit causes reads or writes to any of the debug registers to generate an INT1
    - Legacy behavior from ICE days
Update on recent hacks
Recent hacks this year

• Nintendo Wii
  – Loader compromised (tmbinc; Dec)
  – Software save game hack (bushing, segher, tmbinc; Feb)

• Windows Media DRM
  – drmdbg decrypts keys and is staying current with updates (Jan)

• iTunes
  – requiem decrypter released, removed via C&D (Feb)
  – Tools that rip audio samples from memory still exist (QTFairUse6, myFairTunes7)

• iPhone
  – Bootloader software compromise (iPhone Dev Team; March)
AACS vs. BD+

- Both allow updates to security
  - AACS: new MKBs and player keys
  - BD+: new software on discs
- Effectiveness metrics
  - How long each update survives before being hacked (L)
  - How frequently the updates appear (T)
    - If L < T, you are releasing new discs into an already-hacked environment!
- Goal: increase L, decrease T
Timeline of hacks

v1 (Apr 2006)  v3 (May)  v4 (Oct)  v7 (Apr)  Jul? Oct?

AACS

v1 (Oct)  Mar

BD+

root labs
AACS analysis

• Not doing so hot
  – Very low L (length of time update is unbroken)
    – -2 … +2 weeks if you throw out the initial period
  – Long T (time between updates)
    – < 3 months not allowed by business agreements
    – Up to 18 months if hacked player not identified
    – Appears to be steady now at every 6 months

• Game over or …?
  – Unmasking the Slysoft oracle
  – Sequence keys
BD+ analysis

• Too early to tell
• Flexible L
  – Software protection can be a small or large barrier, depending on effort invested
• Potentially lower T
  – Update schedule in hands of disc authors
    • Potentially less parties to coordinate with for updates
    • Just test and ship a disc
• 2008 will be an exciting year!
Questions?

For more on software protection, check out my blog “rdist” at:

rootlabs.com