SSTC 2007
"Securing Critical Assets with Anti-Tamper Technologies"

Eric Bryant

20 June 2007

Agenda

- Arxan Introduction
- Anti-Tamper Overview
- Attack Vectors
- Anti-Tamper Methods
- Best Practices
- Case Study: Guarding
- Q&A

Arxan At a Glance

- Arxan provides software and hardware Anti-Tamper solutions to protect Critical Program Information (CPI) and Intellectual Property
- Our customers include government agencies, defense contractors, software vendors and content providers
- Our core technology was conceived in 1998 at Purdue University at the Center for Education and Research in Information Assurance and Security (CERIAS).
- Incorporated as Arxan Technologies in 2001
- Arxan continues to innovate through government sponsored research contracts and internal R&D to provide the strongest security
- Arxan is backed by top-tier venture capital firms with a combined total of over 4 billion dollars under management

The Need for Protection

"U.S. Weapons and technologies can be exposed to the risk of compromise when they are exported, stolen, lost during combat, or damaged during routine missions"

– GAO Defense Acquisitions Report

What is AT?

- System engineering activities intended to deter and/or delay exploitation of critical technologies in a U.S. defense system.
  - May include protective software and/or hardware technologies
  - AT is the last line of defense
- AT protects US Critical Information and Technology from exploitation
- AT serves as an enabler for improvements in Coalition Warfighting capabilities by decreasing likelihood of
  - Countermeasure development
  - Unwanted technology transfer
  - FMS systems being modified to increase their capabilities beyond export license limitations

What AT is NOT

- No silver bullet
- Not a substitute for other security practices
- No global solutions that apply to all systems/scenarios
- Not an impenetrable defense
- Not just for FMS - Peacetime/crisis/combat loss may require AT for US system protection

ARXAN

guarding your IP™

The Threat

Attack Vectors

ARXAN

- Reverse-engineering
  - The process of discovering the technological principles of an application through analysis of its structure, function, and operation
- Typically performed using a combination of static and dynamic analysis tools, such as:
  - IDA Pro (advanced disassembler)
  - OllyDbg (user-level debugger)
  - SoftICE (kernel-level debugger)
  - WindView (ethernet-based vxWorks debugging tool)
  - JTAG-based emulation tools

Attack Vectors

ARXAN

- Tampering
  - Unauthorized code modification
  - Typically done to bypass security checks in software
  - May be performed statically (on disk) or dynamically (at runtime)
  - Malware infection can also be classified as a type of tampering

Attack Vectors

ARXAN

- Code Lifting
  - The unauthorized reuse or theft of code from an existing application
  - The thief does not have to reverse-engineer the lifted routine (just has to find it)
  - More common with managed code (e.g., .NET, Java), but any code is susceptible

Attack Vectors

- Fault Attacks
  - Generate a malfunction that causes one or more flip-flops to adopt the wrong state
- Probing Attacks
  - Leverage FIB Workstations, laser cutter microscopes, etc. to alter routines
- Power Attacks
  - SPA and DPA can be used to analyze power traces and glean information about sensitive data
- Memory Resonance Attacks
  - Analyze chips to discover sensitive data after it's been cleared

Know Your Enemy

- Capability of adversaries vary based on skill and funding levels
- IBM Taxonomy of Attackers:

Class I Clever Outsiders	Class II Knowledgeable Insiders	Class III Funded Organizations
• Intelligent individuals • Insufficient knowledge of the target system • Access to moderately sophisticated tools • Take advantage of existing or known weaknesses	• Specialized education and experience • Varying degrees of understanding of the target system and potential access to it • Access to highly sophisticated tools	• Teams of specialists backed by great funding resources • Capable of in-depth analysis of target system and design sophisticated attacks • Access to the most sophisticated tools

The Defense

Software AT

- Anti-Cracking Techniques
  - Techniques targeted at preventing use of common attack tools
- Obfuscation
  - Code transformations designed to defend against reverse-engineering (layout, control, data, and preventative types)
- Encryption
  - White-box cryptography, All-at-once vs. On-demand encryption
- Integrity Verification
  - Protect data values, verify program state, detect tampering
- Application Specific Techniques
  - Customized tactics to thwart specific threats

Hardware AT

- Hardware-Assisted AT
  - Leverage reconfigurable hardware to protect software and hardware components in system
- Volume Protection
  - Common sensors (light, sound, temperature, etc.) that would be appropriate to monitor tamper efforts against an enclosure
- Security Processors
  - Specialized ASICs, TPMs
- Protective Coatings
  - Active and passive coatings designed to defend against specialized hardware attacks

Multi-Layered AT

- Neither hardware nor software AT technologies completely address all threats to which a system may be exposed
  - Each have their own set of strengths and weaknesses
- Strong protection can be achieved through combined software and hardware AT layers
  - Each layer actively protects, detects and reacts to unauthorized access attempts
  - Adversary must focus on simultaneous events
- Recommended approach for protecting DoD systems

Best Practices

- Obfuscation should be used whenever possible
- “On-demand” decryption is preferable to “all-at-once” decryption
- White-box cryptography is recommended to hide cryptographic keys
- Anti-Cracking techniques should be continually updated/improved
- Protection code should be designed to blend in w/ surrounding code
- It is effective to split-up components of a protection technique and distribute throughout system
- The much-maligned “security through obscurity” actually works well in the framework of software protection
- Single points of failure should be eliminated by creating interdependencies between protection components

ARXAN

guarding your IP™

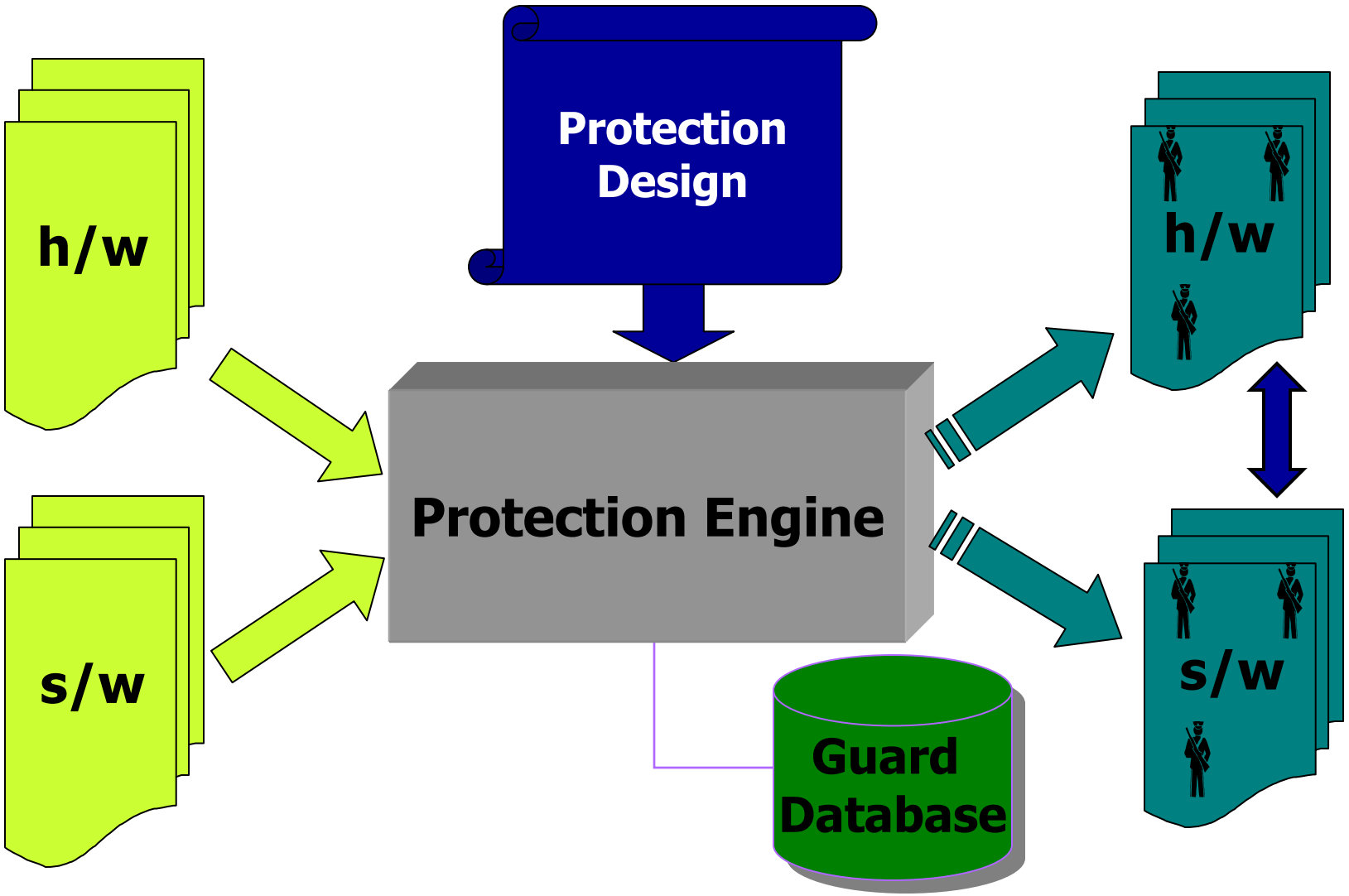
Case Study:
The “Guarding” Approach to AT

What is a Guard?

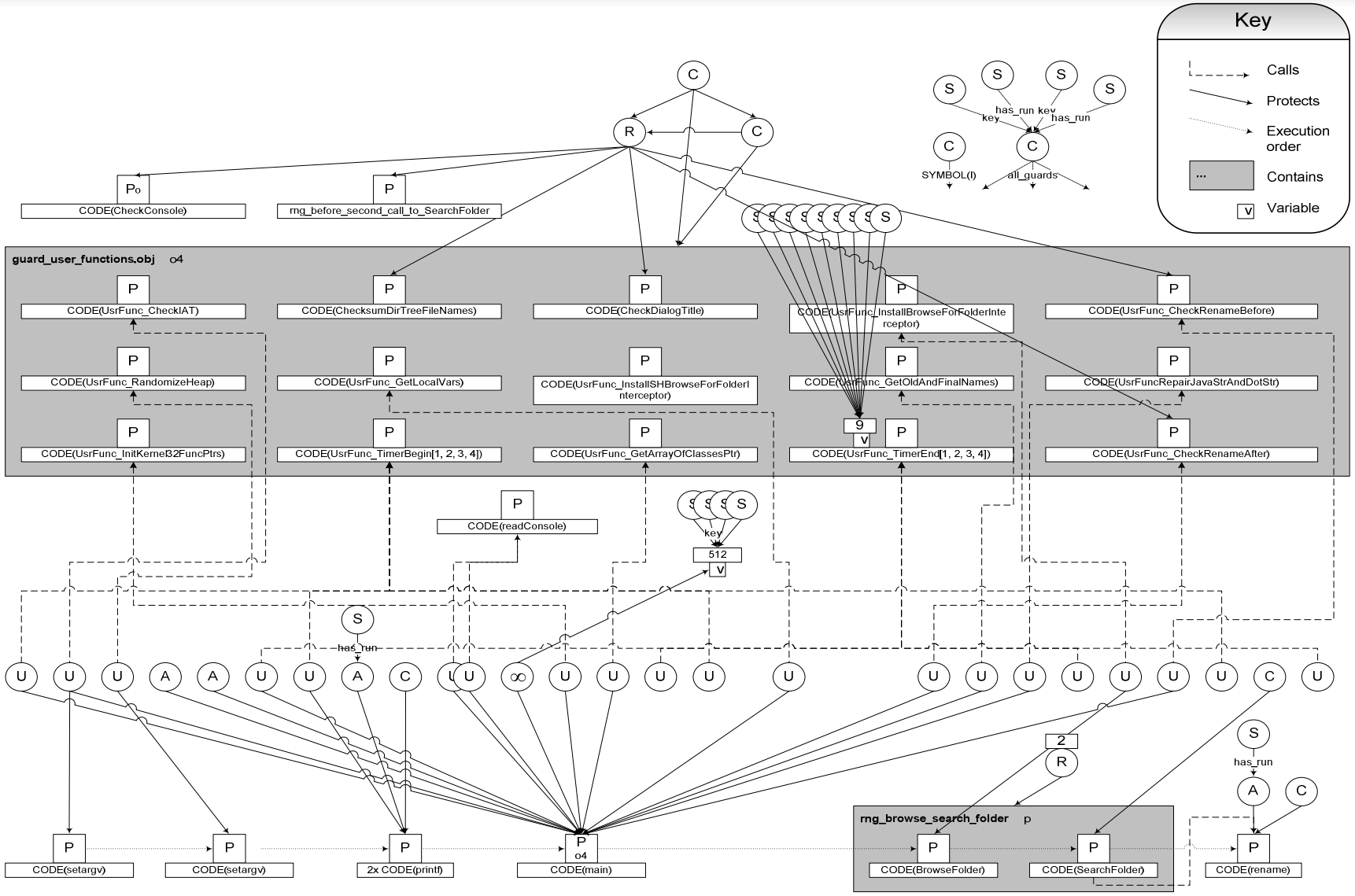
ARXAN

- A small executable object that is injected into software or hardware for the sake of AT protection
- Different types
  - Each offers a specific security function (can be anything)
  - Each can have many instances (variants)
- Injected at the binary-level
  - Executes at a certain location
  - Physically occupies space
- Networks of Guards protect the program and each other
  - Eliminates single point of failure
- May be used in hardware-assisted protection designs

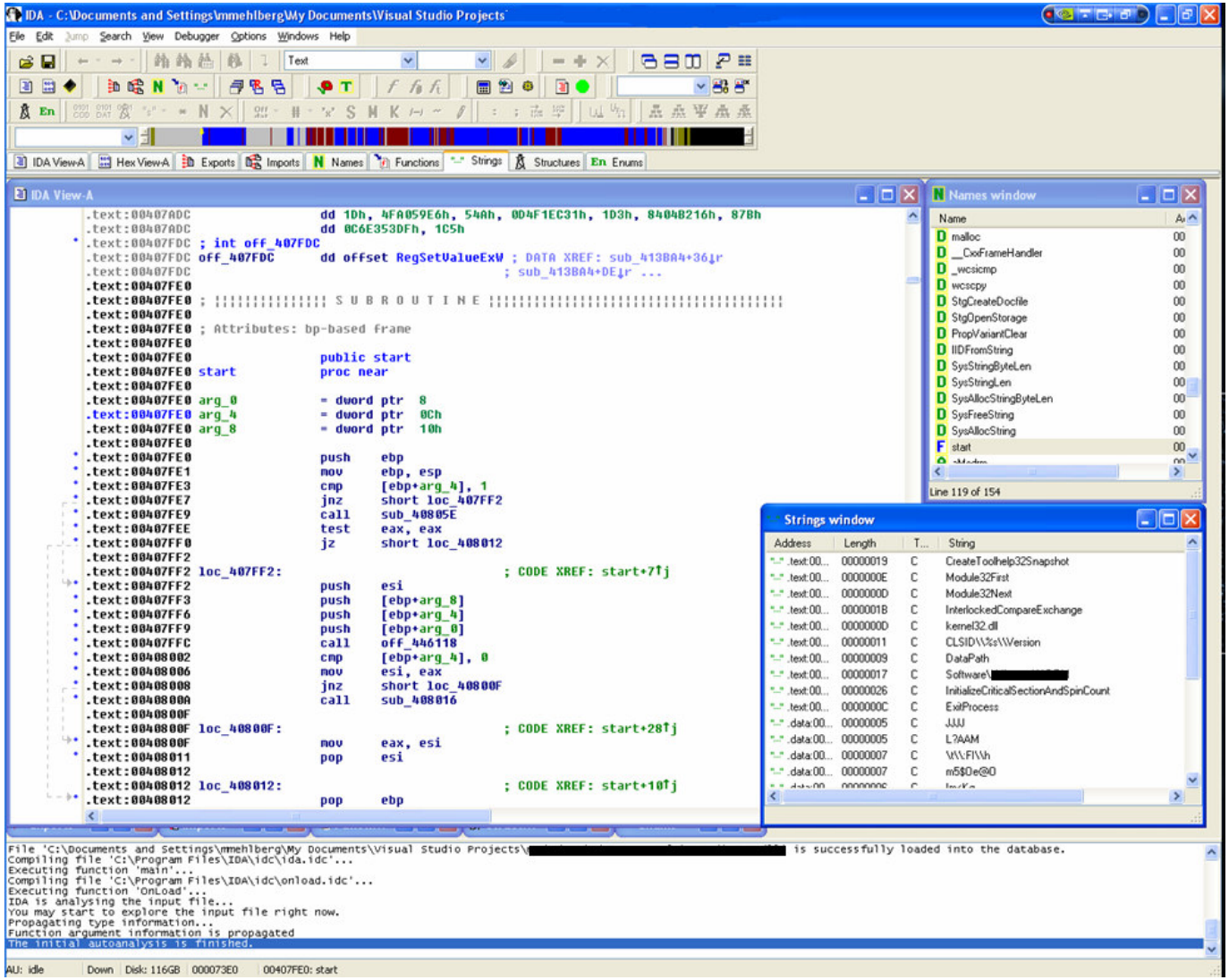
Example Implementation



Example Protection Design



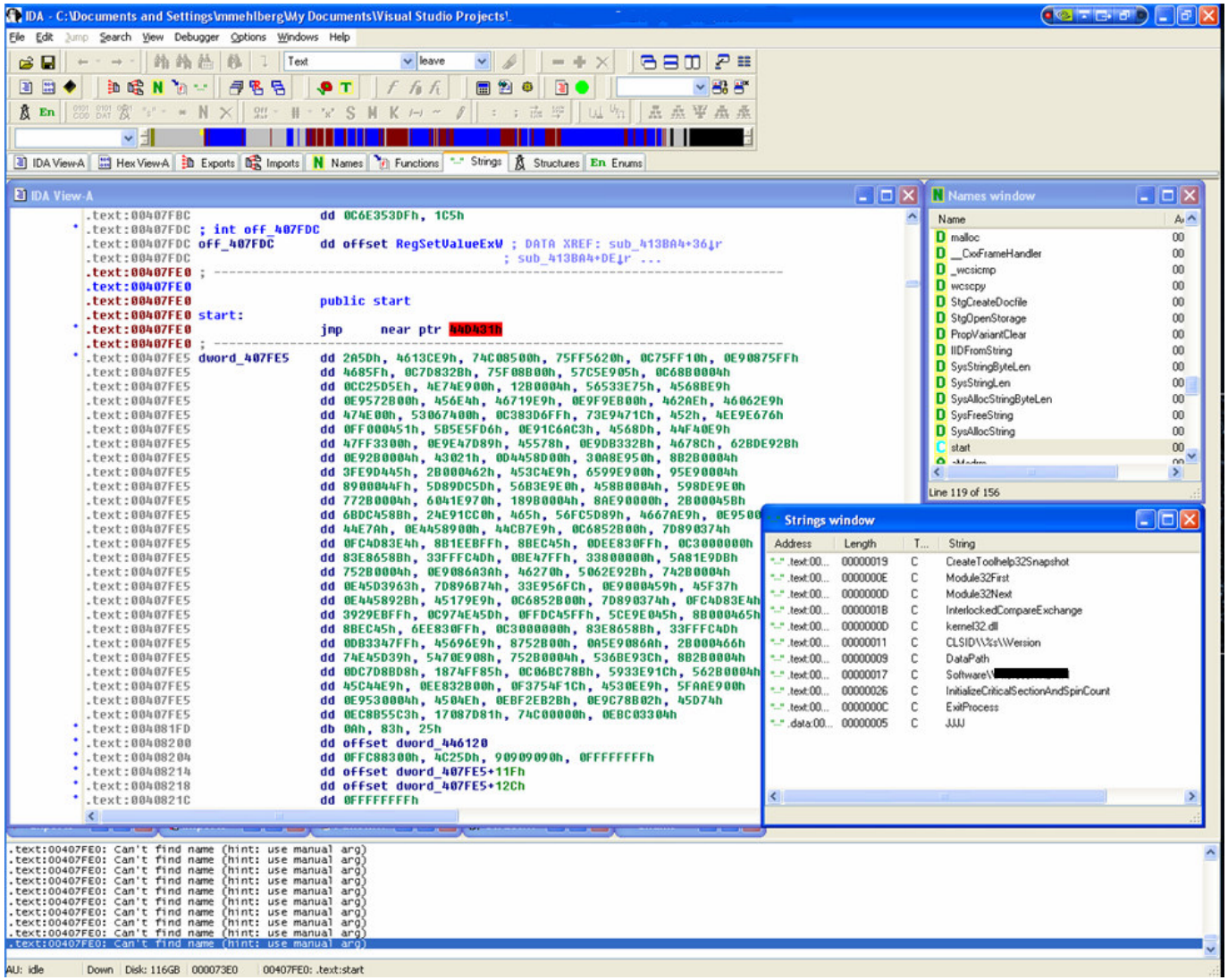
Unprotected Program



Notice:

- Easily disassembled instructions.
- Strong cross references.
- Valid, readable string references.

Protected Program



ARXAN

guarding your IP™

Concluding Remarks

Key Points

- AT protects US Critical Information and Technology from exploitation
- AT is the last line of defense
- AT implementers & integrators need to be familiar with attack countermeasures
- There is no silver bullet AT technique/method
  - Combinations of AT techniques yield best defense
  - Must balance security vs. performance/size constraints
  - The best AT implementation results from an iterative testing process
- Plan early for AT – it will save you time and money!

Questions?

ARXAN

Contact Information:

Eric Bryant
ebryant@arxan.com
www.arxan.com