Securing Skill Based Games

A survey of common hacks and techniques for remediation

Attacking the Server

- dDOS
  - No real value for the attacker (unless perhaps, they're your competition :-)
  - Usually just "kids having fun"
- Penetration and subversion of the server itself
  - Difficult, but real value for the attacker, so it attracts the grownup bad guys
  - Certainly not impossible, as evidenced by the JP Morgan Chase intrusion over the summer, where the attackers had obtained root credentials on ***at least*** 90 of JPMC's internal servers.
- Network packet manipulation
  - Alter the servers state by forging network traffic
  - Usually accomplished from the client side, but technically an attack on the servers state

Attacking the Client (server indirectly)

- Game Data Snooping and/or Input Grooming
  - Aimbots / Triggerbots
  - Radars / ESP
- Game Asset Modification
  - Texture Hacks
- Game Logic Modification
  - Collision Detection Disable
  - Network Traffic Forgery

Aimbots / Triggerbots

Aimbots / Triggerbots

- Aimbot definition
  - Internal or external machine that tracks objects within a game view and automatically aims and/or triggers the players weapon

Aimbots / Triggerbots

- Background
  - Three basic classes of Aimbots
    - Color / Object Tracking Aimbots (COT)
    - Client Hook Aimbots (CH)
    - Graphics Driver Aimbots (GD)
  - General Characteristics of Aimbot classes
    - COT Aimbots
      - Minimally invasive
      - Computationally intensive
    - CH Aimbots
      - Maximally invasive
      - Computationally lightweight
    - GD Aimbots
      - Balance between invasiveness and computational load

Color / Object Tracking Aimbots

- Theory of operation
  - Screen scrape for color / objects
  - Calculate vector
  - Inject input via input drivers
- Detectability / Preventability
  - Practically impossible to detect
  - Effect can be mitigated with intelligent asset design
  - Some hack augmentation such as asset color manipulation that improves effectiveness can be effectively prevented

Client Hook Aimbots

- Theory of operation
  - Hook particular functions within game client
  - Scan game memory for objects
  - Calculate vector
  - Directly invoke firing functions or inject input via drivers or by modification of game client resident buffers
- Detectability / Preventability
  - Generally easy to detect
  - Generally easy to prevent

Graphics Driver Aimbots

- Theory of operation
  - Hook particular functions within the graphics driver DLL (mapped by the game client)
    - Often the hooked graphics function provides direct access to the memory representing object coordinates
  - Calculate vector
  - Directly invoke firing functions or inject input via drivers or by modification of game client resident buffers
- Detectability / Preventability
  - Generally easy to detect
  - Moderately straightforward to prevent

Radars

Radars / ESP

- Radar definition
  - Internal or external machine that tracks objects within world and provides overview of target coordinates (usually a “top down” fixed camera view)

Radars / ESP

- Theory of Operation
  - Scans the local game memory identifying targets
    - Requires knowledge of the game data structure
    - Typically Hackers reverse engineer and publish offsets of data members
    - Theoretically automated processing could be performed to reverse engineer coordinate data by motion vector analysis of random data triples and recording addresses that produce “sensible” vectors
- Detectability / Preventability
  - If done properly, practically impossible to detect
  - Preventable by runtime obfuscation of data

Texture Hacks

Texture Hacks

- Texture hack definition
  - Modification of texture data, usually to obtain transparency or camouflage

Texture Hacks

- Background
  - Two common classes of texture hacks
    - Wall hacks
      - Make walls transparent
      - Alter texture to visually expose enemies
    - Chamming
      - Alter enemy texture to visually highlight them

Texture Hacks

- Theory of operation
  - Alter texture data on disk
  - Alter texture data in memory
- Detectability / Preventability
  - If done properly, difficult to detect, if done poorly, easy to detect
  - Prevented through use of white-box cryptography and anti-tamper

Collision Detection Disable

Collision Detection Disable

- Collision detection disable definition
  - Modification of functions used to perform collision detection

Collision Detection Disable

- Theory of operation
  - Alter functions that check for character / object collision
  - Typically all that is required is to disable the code (patch a return)
- Detectability / Preventability
  - Easily detected
  - Easily prevented with code hardening

Network Packet Manipulation

Network Packet Manipulation

- Network packet manipulation definition
  - Modification or temporal disordering of data packets destined for either the server or the client

Network Packet Manipulation

- Background
  - Network packet manipulation can be used to accomplish many types of hacks
    - Artificial lag
      - Software based “lag-switch” (slow down rate at which all packets are tx’d/rx’d)
    - Look-ahead
      - Software induced latency (see what other user action is, then send your action with a prior timestamp)
    - Hack report sinking
      - Identify hack reports going to server and disable or “undo” them
  - General Characteristics of network packet manipulation
    - Although in theory packet manipulation is possible outside of process space most client/server games implement encryption which (if properly done) renders this impractical

Network Packet Manipulation

- Theory of operation
  - Hook the functions that encrypt/decrypt packets within the game client process
  - Because the hooked is in the code, pre/post encryption, encryption offers no protection
- Detectability / Preventability
  - Easily detected
  - Easily prevented with code hardening

Questions?

How Can Arxan Help?

Arxan Technology

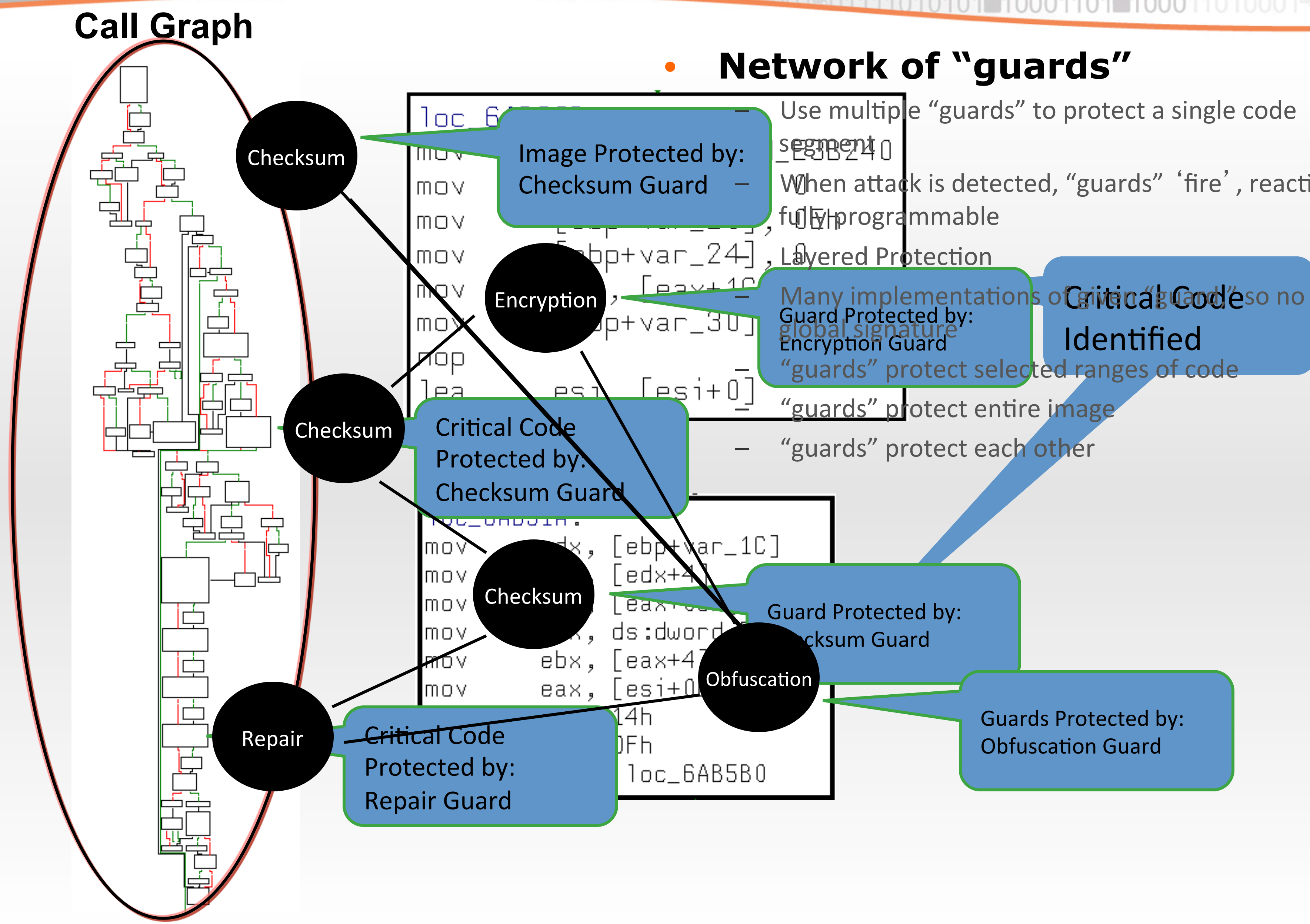
- Anti-Reverse Engineering
  - Prevent the attacker from understanding the code
    - Obfuscation (at the machine code level)
    - Encryption of .text (forces attacker to memory dump)
  - Effective
    - Immediately raises the barrier
- Software Anti-tamper
  - Software version of the epoxy encapsulation for hardware
  - Active guards that are injected into your games client binary at the machine code level
    - If attacker attempts to “pull the code apart” the code will “self-destruct”
    - Code can cloak itself and only reveal itself once it is committed to completing its function (e.g. hack report function)

Arxan Technology

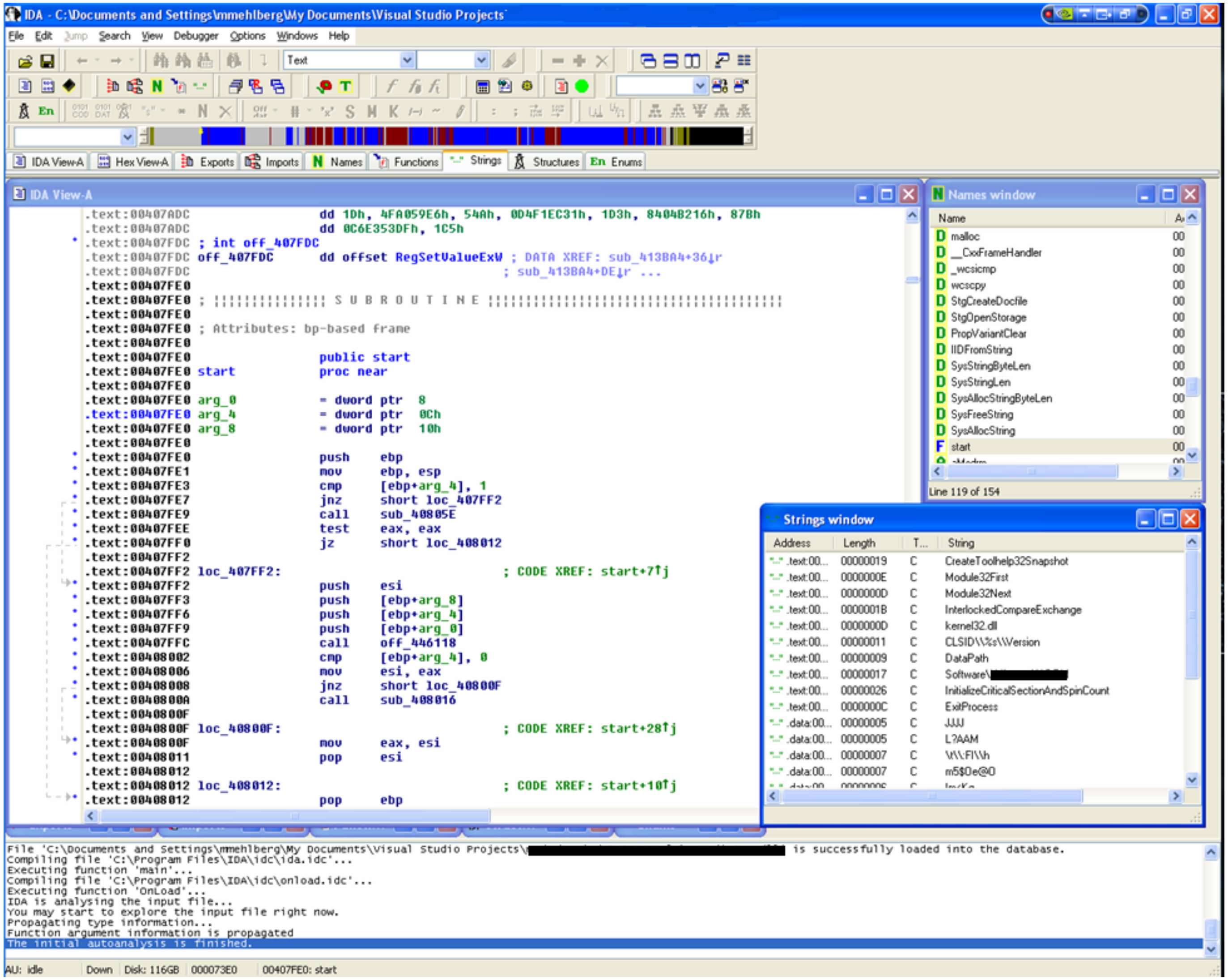
- Software based whitebox cryptography
  - Secures key material
    - Key material remains encrypted at all times, even during cipher operation
    - Key lifting is extremely difficult
  - When combined with code hardening, the code cannot be lifted from game client binary
    - Code hardening becomes the “epoxy” over the crypto chip
    - Difficulty of lifting a key becomes similar in magnitude to lifting a key from a hardware TPM
    - If the white-box is eventually compromised (typically measured in years) breach mitigation is only a software update away

Arxan Code Hardening

- **Network of "guards"**
  - Use multiple "guards" to protect a single code segment
  - When attack is detected, "guards" 'fire', reaction is fully programmable
  - Layered Protection
  - Many implementations of given "guard" so no global signature
  - "guards" protect selected ranges of code
  - "guards" protect entire image
  - "guards" protect each other



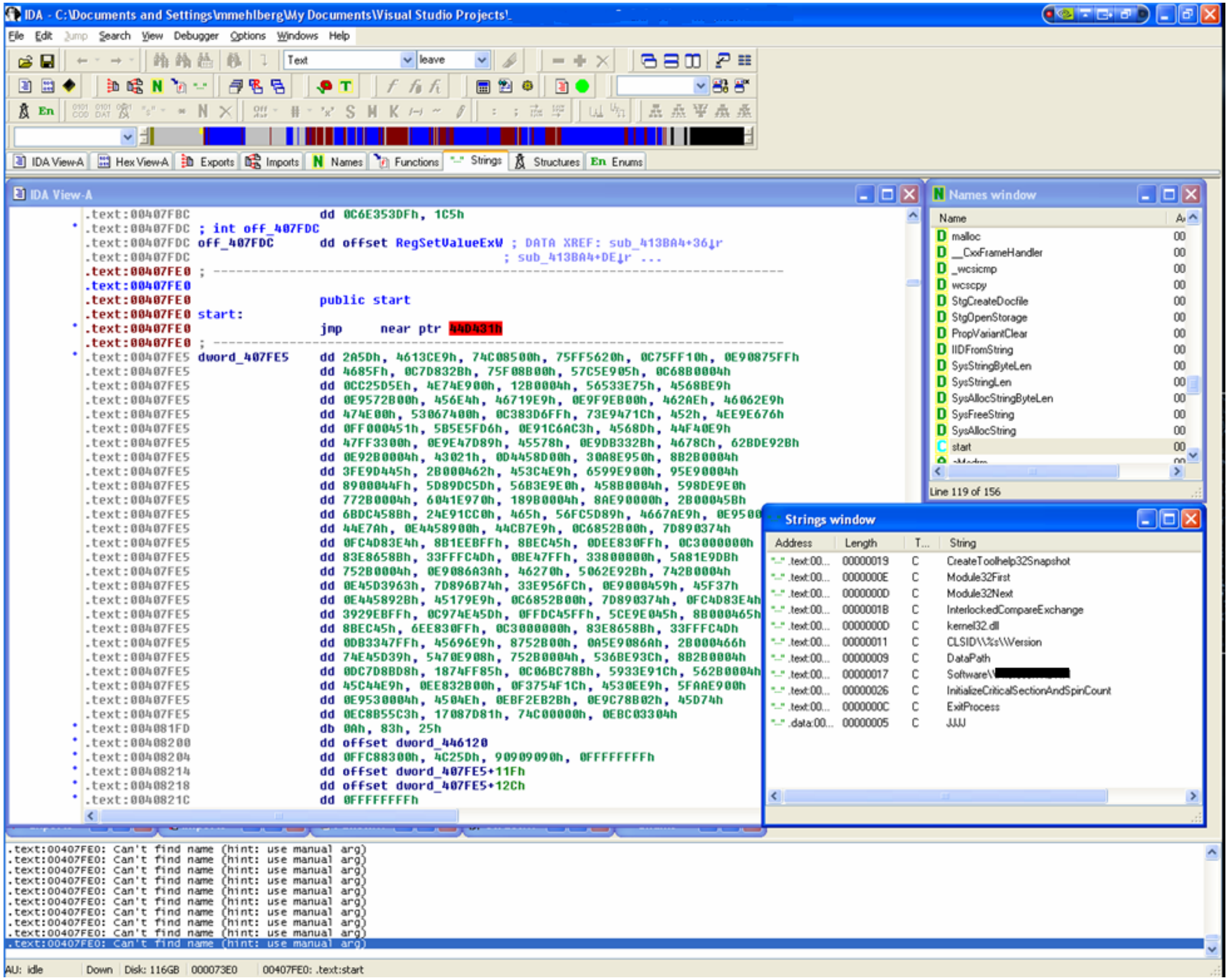
Unprotected Program



Notice:

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

Arxan Protected Program



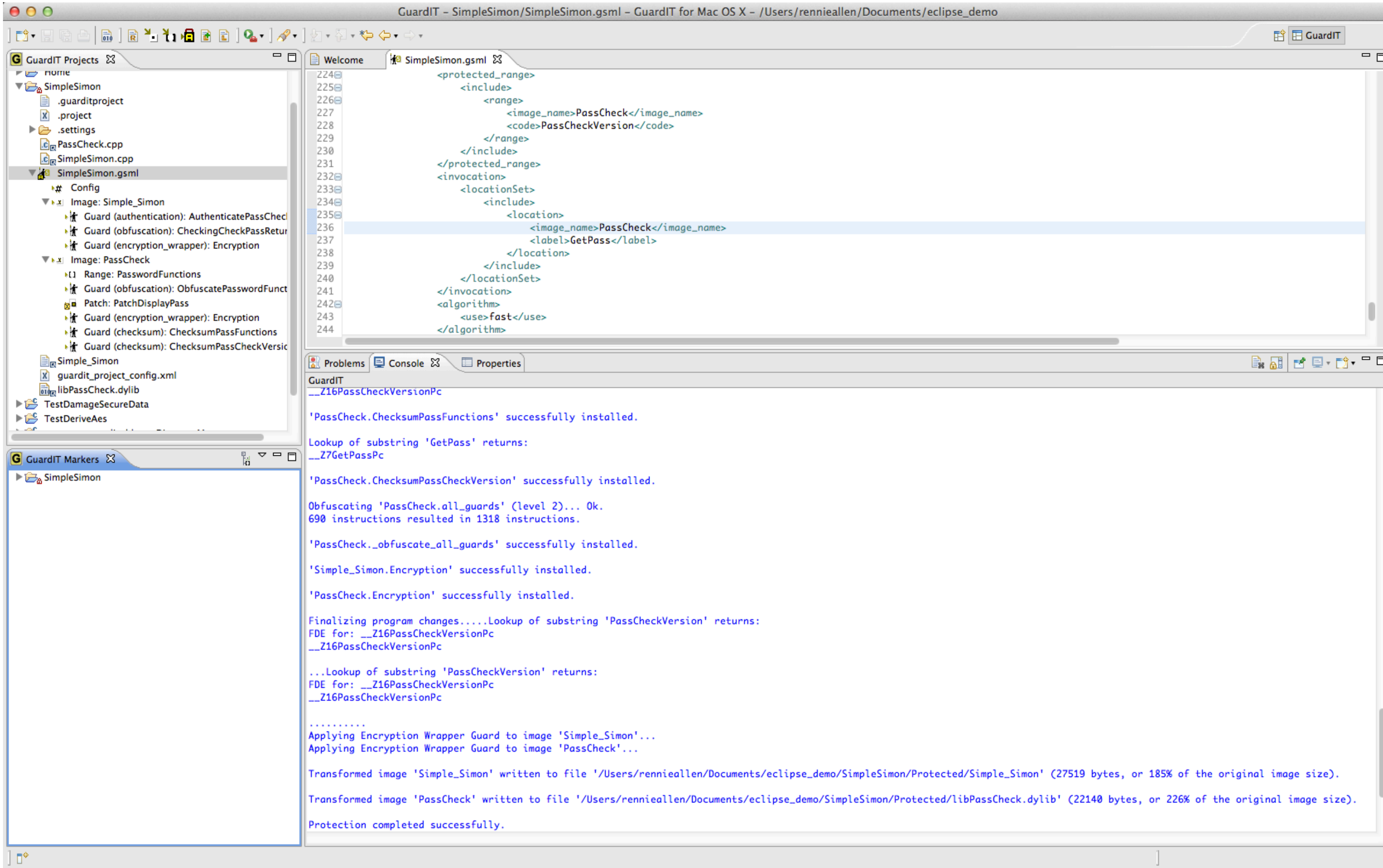
Notice:

- Ida is unable to disassemble Cross references unknown
- Encrypted, damaged, or missing strings
- Forced manual analysis

Specify guards that should be injected

- SimpleSimon
  - .guarditproject
  - .project
  - .settings
  - PassCheck.cpp
  - SimpleSimon.cpp
  - SimpleSimon.gsml
    - Config
    - Image: Simple_Simon
      - Guard (authentication): AuthenticatePassCheck
      - Guard (obfuscation): CheckingCheckPassReturn
      - Guard (encryption_wrapper): Encryption
    - Image: PassCheck
      - Range: PasswordFunctions
      - Guard (obfuscation): ObfuscatePasswordFunctions
      - Patch: PatchDisplayPass
      - Guard (encryption_wrapper): Encryption
      - Guard (checksum): ChecksumPassFunctions
      - Guard (checksum): ChecksumPassCheckVersion
  - Simple_Simon
  - guardit_project_config.xml
  - libPassCheck.dylib

Invoke Engine to Process the Binary



Test the Protection

Without tampering

```
greenheart:Protected rennieallen$ PATH=$PWD:$PATH ./Simple_Simon
Entering EW guard instance
Guard Encryption: invoked.
Integrity algorithm: fast
integrity/integrityvalue:
8b5110e7 /8b5110e7
Guard Encryption ran.
Guard Encryption exited.
Guard AuthenticatePassCheck: invoked.
Guard AuthenticatePassCheck: ran.
Guard AuthenticatePassCheck: exited.

 Simple Simon 2.0

Enter Password: secret
Result is 1
(16231)  Simple Simon met a pieman going to the fair;

(16231)  Said Simple Simon to the pie man, let me taste your ware.

(16231)  Said the pie man to Simple Simon, show me first your penny.

(16231)  Said Simple Simon to the pie man, Sir, I have not any!

The results of the functions is: 20

 begin: 3895
   end: 26588
greenheart:Protected rennieallen$
```

With tampering

```
greenheart:Protected rennieallen$ PATH=$PWD:$PATH ./Simple_Simon
Entering EW guard instance
Guard Encryption: invoked.
Integrity algorithm: fast
integrity/integrityvalue:
2a98d123 /2a98d123
Guard Encryption ran.
Guard Encryption exited.
Guard AuthenticatePassCheck: invoked.
Guard AuthenticatePassCheck: ran.
Guard AuthenticatePassCheck: fired.
Guard AuthenticatePassCheck: exited.

 Simple Simon 2.0

Enter Password: secret
Result is 1
(16231)  Simple Simon met a pieman going to the fair;

(16231)  Said Simple Simon to the pie man, let me taste your ware.

(16231)  Said the pie man to Simple Simon, show me first your penny.

(16231)  Said Simple Simon to the pie man, Sir, I have not any!

The results of the functions is: 20

 begin: 3648
   end: 25305
greenheart:Protected rennieallen$
```

Fired!

Aimbots: GuardIT™ Specific Remediation

- Color/Object Tracking
  - Encrypt all character assets
    - Prevents augmentation for color tracking (i.e. changing asset colors to make characters easily identifiable)
- Client Hook
  - Checksum functions that are used for weapon aiming or character movement
  - Repair functions that are tampered
- Graphics Driver
  - Where the graphics driver DLL (e.g. DirectX) is the attack vector, utilize the hook detection guard (will fire if any standard DLL entry points are hooked)
  - Repair functions that are tampered

Radars/ESP: GuardIT™ Specific Remediation

- Generally not detectable if implemented by pure memory scanning
- Prevention is generally the only viable option
  - Use Data Obfuscation Guards to scramble character position data

Texture Hacks: GuardIT™ Specific Remediation

- Detection of manipulation of texture data on disk can be performed using checksums of asset data
  - Use Data Obfuscation Guard and Checksum Guards to protect the asset checksum (in the game memory) from tampering
- Detection of manipulation of texture data in runtime memory can be manually coded
  - Calculate in-memory checksum of texture data at load time and store this value using Data Obfuscation Guard to protect the checksum value from discovery
- Preventable by using white-box crypto to maintain all assets in encrypted form at runtime
  - By linking environmental checks (e.g. debugger detection) to encrypted routines that damage internal white-box data, texture assets will only be properly constructed in memory if the game client is not being observed or tampered

Coll. Detector Hack: GuardIT™ Specific Remediation

- Detection easily accomplished with GuardIT™ Checksum Guards
  - Typically the coll. detector routines are relatively compact so checksum is fast
- Preventable by utilizing repair guards to repair the tampered code
  - Since the detector routines are relatively compact, the performance impact of prevention is moderate and ***is only paid by the hackers***

Network Packet Hack: GuardIT™ Specific Remediation

- Detection easily accomplished using GuardIT™ Checksum Guards
  - Checksum all network packet encryption functions
  - No need to checksum the downstream functions as the data is already encrypted
- Preventable with use of GuardIT™ Repair Guards and TransformIT™ white-box cryptography
  - Repair guards will restore tampered packet encryption functions
  - White-box crypto will prevent attackers lifting the keys (which would otherwise enable downstream attacks)

ARXAN

Protecting the App Economy™

ARXAN

Protecting the App Economy™

ARXAN

Protecting the App Economy™

ARXAN
Protecting the App Economy™

ARXAN

Protecting the App Economy™

ARXAN

Protecting the App Economy™