



# Ensuring Software Security Even in Imperfectly Protected or Hostile Environments

*Dan Stickel, Metaforic CEO*



# The Software Protection Problem



- The software “Garden of Eden” is long gone
- Existing defenses are proving inadequate
- Unreasonable to expect admins and consumers to provide perfectly protected operating environments

# Best Practices from the Data Center



- Firewalls, intrusion prevention systems, whitelisting protocols
- Armed guards, video surveillance, background checks
- Anti-virus, automated software updates, penetration testing
- Two-factor authentication, behavioral profiling, ...

# A Hostile World

“In evaluating computer systems ‘of consequence’ – at government agencies, Congress and in the private sector – I have yet to encounter one computer that has not been compromised by an advanced persistent threat.”

– *Mike McConnell, former director of National Intelligence and now Vice Chairman at Booz Allen Hamilton*

# Best Practices for Devices?

- It's arguable whether best practices for data centers are working well, even with all the armed guards...
- But in any case, how does all this translate to an insulin pump? A car? A mobile phone?



- We know these will be/are under attack as well, but they must exist in a tremendously more exposed environment

# Industrial Infrastructure

- Increasing global reliance on SCADA, M2M and DCS
- IDC projects 12B connected units by 2015 ... and not everything is connected!
- Stuxnet, Flame, Duqu, Nitro, Night Dragon, etc.
- Increasing regulatory and operator security concerns, plus producer liabilities
- Ideally, protections should be able to last for months and years without the need for a constant stream of patches and updates



# Mobile Apps

- BYOD rewards, risks, and responses
- When deploying consumer apps, you cannot manage the device ... or much of anything
- Jailbreaking or rooting removes all protections (code-signing, application sandboxing, etc.)
- xCon and other apps defeat jailbreak detections; targets include:



- NatWest pulled app due to customer-experienced fraud
- Eurograbber credited with stealing \$47M from customers

# Perspectives on Protection



Traditional sysadmin view:  
protect the system from  
risks in the software



But who is protecting the  
software from risks in the  
system (APTs, new hacks,  
swizzles, physical exposure,  
disgruntled employees, etc.)?

# Trust and Uncertainty

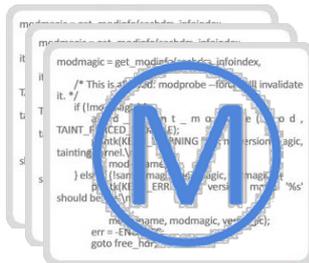
- Is any environment trusted?
- Zero-day exploits, insider betrayal
- TEE versus TEA
- Learning from History



OS Version	Release	Days to Jailbreak
iOS 1.0	6/29/07	11
iOS 2.0	7/11/08	9
iOS 3.0	3/17/09	94
iOS 4.0	6/21/10	2
iOS 5.0	10/12/11	1
iOS 6.0	9/19/12	0

# The Vision

*“Software must  
be built to  
survive in the  
real world”*



# Protecting the Software

- Code signing
- Application wrapping
- Obfuscation
- Vulnerability Scanning
- Hardware Enforcement
- Self-Defending Software



# Code Signing

- Uses
  - iPhone apps, Windows programs (varies), payment terminals, etc.
  - Implies code integrity and definitive identification of source
- Advantages
  - Digital signature strong & simple to implement
- Problems
  - Requires external enforcement mechanism
  - Requires unadulterated deployment environment and process
  - Potential compromise of keys and certificate authorities
  - Typically only checked on install, or startup
  - No defenses against runtime attacks including the insidious patch-execute-unpatch

# Application Wrapping (Containers)

- Encapsulates application within another application, which is responsible for meta-control (run, delete data, etc.)
- Uses
  - Games, Mobile Application Management (MAMs), etc.
- Advantages
  - Simple to deploy
  - Policy management
- Problems
  - Can be issues with performance and size overhead
  - Security is still weak, as unprotected application gets unwrapped/decrypted before use or during use
  - Repackaging not prevented

# Obfuscation

- Various scrambling techniques that make it harder for hackers to understand the workings of a given program
- Uses
  - IP protection, general catch-all
- Advantages
  - Can be simple to deploy
- Problems
  - Professional hackers are barely slowed down, and often do not need to “understand” the code to make desired modifications
  - Can be issues with performance and size overhead
  - Repackaging not prevented
  - Troubleshooting and debugging becomes difficult or impossible

# Vulnerability Scanning

- Automated tools that scan code for weak programming practices that could enable outside users of the program to manipulate it into gaining access to the system
- Very useful and good practice
- But remember, in this discussion we are attempting to defend the correct operation of the software
  - Assume hackers or disgruntled employees have access to it!
  - How do you ensure the software will still retain integrity?



# Hardware Enforcement

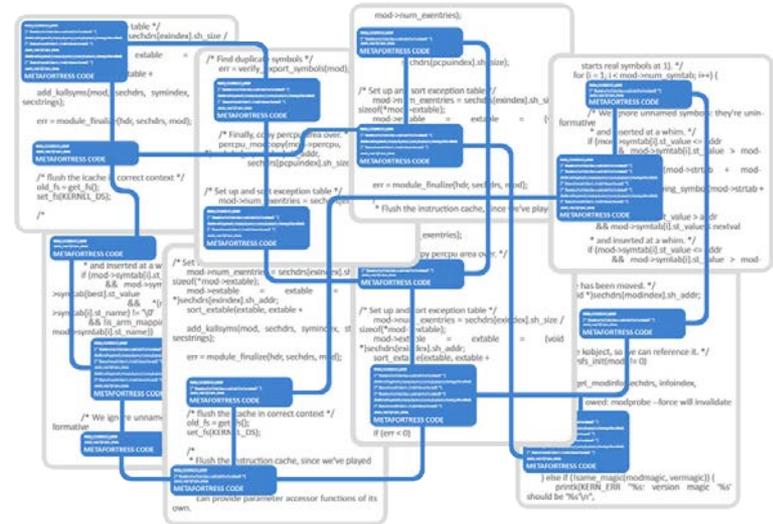
- Hardware only allows “known” code chain to execute
  - Apple iOS devices, Intel TXT, ARM TrustZone, game consoles
- Uses
  - Prevent malware from running, provide credentials to external services, device binding
- Advantages
  - Hardware is extremely difficult to attack via malware
- Problems
  - Only helps on devices with the appropriate hardware
  - Requires whole-hearted software adoption
  - Doesn't prevent malware in practice (e.g., iOS jailbreaking)
  - Very difficult to regain control after the security fails

# Self-Defending Software

- Multi-faceted collection of self-contained internal defenses
  - Active internal integrity checking
  - Secondary features such as anti-debug, h/w breakpoint protection, ...
  - Response mechanisms
- Advantages
  - Strong protection; no known automated method to defeat
  - No reliance on external mechanisms
  - Minimal work required by device/software creator
  - Minimal (<1%) performance impact
- Disadvantages
  - Often requires access to the source code
  - Usually requires some elements of compiled code

# How Does It Work?

- Automatically applied at build time
- Inserts optimized integrity checks throughout copy of code
- Static and run-time analyses automatically determine optimal protection topologies
  - Multi-layered defense centered around linked, variable density, randomized checks
  - Extreme variance: no two checks are the same
  - Checks cover each other in complete self-protecting network
- Very difficult to defeat, even given months or years of access



# Metaforic Product Line

- Metaforic Core
  - The foundational software immune system
  - Available on Windows, Linux, Mac, iOS, Android, Blackberry, etc.
- Metaforic Concealer
  - Adds ability to obscure code or hide small pieces of data (e.g. keys)
- Metaforic Authenticator
  - Helps ensure reliable communications with external components
- Metaforic Enhanced Integrator
  - For high-end complex projects



# Applying an Immune System

Use your favorite IDE or toolchain—  
no developer coding required

## 1. INSTALL

Integrate Metaforic  
into build environment

## 2. ANALYZE

Automatically build program  
in analysis mode and  
exercise to collect  
profile data

## 3. PROTECT

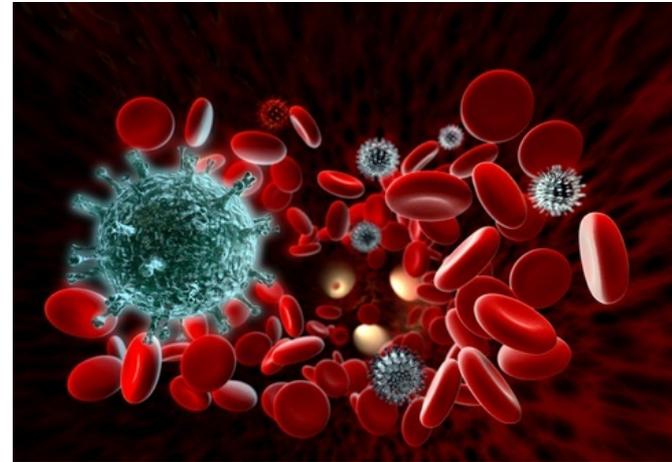
Automatically build  
program in  
protection mode

Strong security, minimal impact—  
protect your application from the inside out

# Learning from Biology



VS.



# Software for the Modern World

- Create program
  - Follow general security best practices
  - Integrate with external security
- Test & address flaws
  - Functionality, performance, usability, etc.
  - Vulnerability scanning (e.g., Veracode)
- Prepare for distribution
  - Install defenses (e.g., Metaforic)
  - Package for installation
- Deploy
- Monitor state

## Software defenses



Hardware hooks,  
container APIs

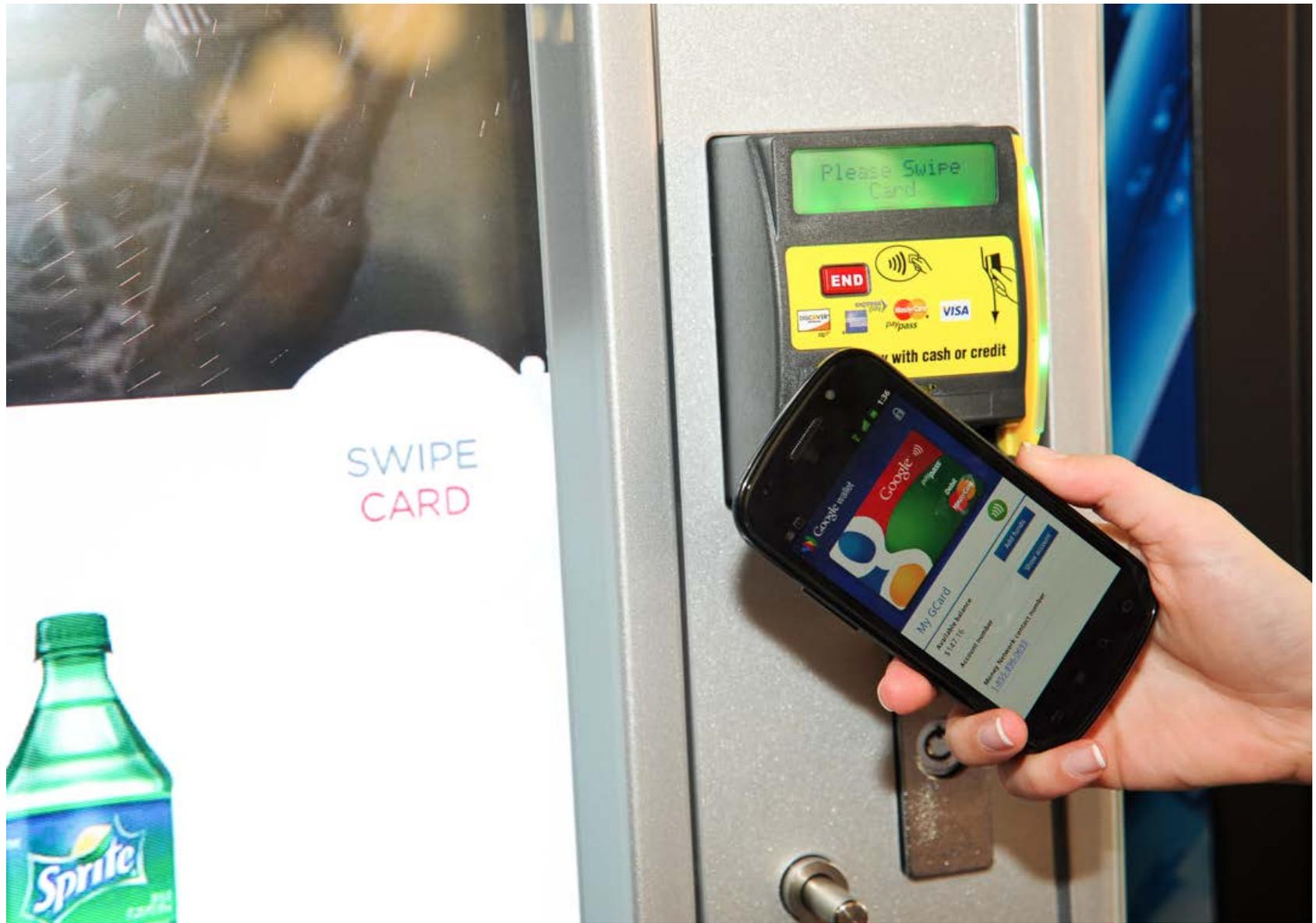


Active defenses,  
obfuscation, wrappers,  
code signing

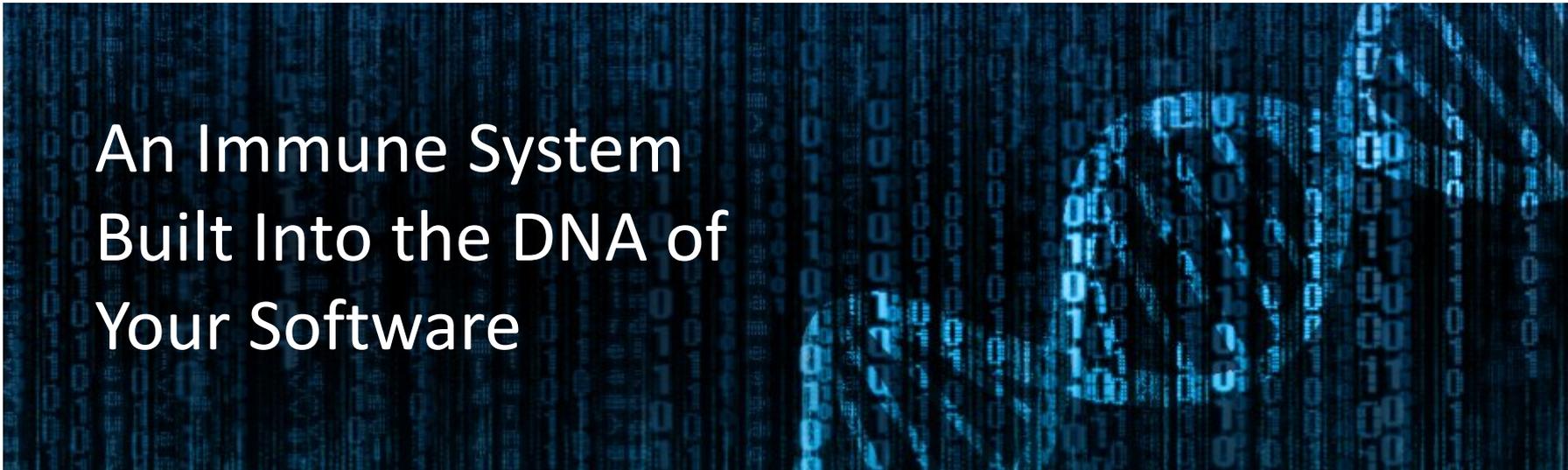
# The Real World

- The real world is filled with people who are motivated to subvert your devices and systems
  - Commercial competitors, other nations, terrorists, criminals, animal rights activists, hackers, anarchists, disgruntled employees, etc.
- However, security isn't the only concern of the manufacturers and operators
  - Strength of security, regulatory compliance, risk
  - Ease of use, ease of deployment, cost, effort
- It behooves all of us to:
  - Create products that can survive real life
  - Insist on buying products that are designed for the real world

# This is Just the Visible...



# Summary



## An Immune System Built Into the DNA of Your Software

- We cannot rely on expectations of a “software Garden of Eden”
- The world of biology has shown us a proven survival mechanism
- Best practice is to build and deploy software and devices that will operate successfully in the real world