MOSREF: Cryptography and Injectable Virtual Machines

Wes Brown
Ephemeral Security
Introduction

Ephemeral Security

We conduct research, development and engineering of very interesting solutions and products. Many of them are available openly with the option to purchase an unencumbered license for integration into commercial products.

Wes Brown
Mosquito is a virtual machine environment with a lightweight framework to deploy and run code remotely and securely, in the context of penetration tests.

It makes a best effort to ensure that communications are secure.

Deployed code is not stored outside of process space.

It protects the confidentiality and trade secrets of code that is deployed and run on the target. This could be an exploit, or a methodology.
Why?

- Often it is desirable to leverage ‘0-day’ code, but doing so in an uncontrolled fashion can have repercussions.
- Many practices have trade secrets and methodologies distilled in the form of audit or exploit code that they would like to keep out of other hands.
- It is a means to ensure that communications between the target and the console is secure.
- Provides a dynamic remote execution environment, allowing ‘in-flight’ modifications.
Others?

Shellcode
- Static, inflexible, low level, and targeted to one environment.
- Can break with patches or environment changes.

Syscall Proxies
- More flexible than shellcode, higher level.
- Driving logic is on attacker’s side. Can be fragile.
Others? (cont’d)

- DLL Injection
  - Can implement higher level features easily.
  - Logic can be placed at target side.
  - Still static, and Windows-only.

- Exploit Compilers
  - Very nice abstraction of lower level code, very flexible.
Lightweight Application VMs

- Can be very small - MOSVM is 128K binary size on Linux.
- Can be even smaller with executable compression techniques.
- Write once, run anywhere - code written on a Linux VM will run on a Windows VM unaltered.
- Can use languages designed for the task - Mosquito Lisp
  - Provides very nice orthogonal development environment.
- Dynamic – code can be altered and updated even on a remote VM – ‘in flight missile reprogramming’
Mosquito Components

- Core - Virtual Machine
- Language - Mosquito Lisp environment and Libraries
- Console - Provides user with interface to manage and deploy drones.
- Drone - Provides a remote process that contacts its matched Console and executes bytecode and statements on its behalf.
Virtual Machine (MOSVM)

- Production ready and stable (beta 3)
- Lightweight and optimized for network tasks
- Easily extensible
  - regex was implemented in a few hours
- Pure ANSI-C, portable (OpenBSD, Darwin, Linux, win32)
  - Even runs on embedded devices (ARM, MIPS, nios2)
  - Wireless routers, anyone?
Virtual Machine (cont’d)

- Virtual machine ‘stubs’ to attach Mosquito bytecode to.
- Programs and applications can be compiled and attached to stubs for different architectures and OS’es.
- Allows standalone executables with no external dependencies, on all supported MOSVM platforms.
- Dependencies are automatically resolved — only the library functions needed are attached to the virtual machine.
- Integrated ECDH, AES encryption with very good entropy generation.
Language (Mosquito Lisp)

- Network-oriented and compact Lisp with strong influences from Scheme.
- Designed for network applications, highly concurrent and provides simple and efficient network and process APIs.
- Rich environment, with over 300 primitive functions, and 200 library functions in the standard library, not including additional libraries specific to MOSREF.
- Well-documented, with a complete reference manual available online and for download.
Language (cont’d)

Self-hosted – the Mosquito compiler is written in Mosquito Lisp. The compiler compiles itself as part of the build process.

Goodies in the standard library – can be available on Drone!

- regex support
- clue (in-memory queryable database)
- XML support
- http
Channels (Overview)

- Language feature, allowing for abstracted communications.
- A cryptographic channel is provided, for easy encryption.
- Transparent negotiation implemented on top of channels.
- Provides a layer of abstraction from the actual communications mechanism in use.
- Programs do not care how communications are handled.
- Processes and sockets have read and write channels that can be mapped to other channels.
Drone

- Virtual Machine + Crypto + Drone Functionality
- Highly optimized to reduce size
- Does not include Mosquito Lisp bytecode compiler
- Drone stores and executes bytecode programs sent by Console.
- Can pull additional libraries from Console over channels rather than embed in Drone stub. Including the compiler!
- Bytecode sent by Console is only stored in process memory.
Console

- Virtual Machine + Crypto + Console Functionality
- Provides a local process to control deployed Drones.
- Provides full Mosquito Environment.
- Includes compiler to compile Mosquito Lisp statements and programs for the Drone on the fly.
- Interface for interacting with Drones in real time.
- Creates Drones when requested using stub functionality.
Refactor exploits into Mosquito Lisp for secure deployment on target.

Network and host reconnaissance code management and results over a secure channel.

Simplify deployment of auditing tools to hosts; all dependencies are included with the Drone and managed by the Console.
Demonstration

- Demonstration of Mosquito Environment
- Demonstration of Ease of Development
- Demonstration of MOSREF
  - Two factor puddle-hop
In the Works (v1.1+)

- Syscall/FFI interface
- DLL and shared library injection
  - Pulling additional binary-level libraries over the wire
- Executable compression techniques
  - Target size of 64K payload
- Additional Transports for Channels (UDP, ICMP, HTTP, DNS)
- Core language and environment enhancements
In the Works (Proboscis)

Proboscis – pull a higher level environment like Mosquito

- Forth Virtual Machine
- FFI/syscall interface
- Abstracts out low level assembler
- Allows multi-architecture shellcode.
- Targeted size of 2K binary, allowing inlining with exploits
- Hand-crafted assembler with relocatable code
In the Works (IPAF)

IPAF - framework for network applications to generate, collect, and analyze network packets.

Sophisticated library to classify packets, and manipulating packet fields without forcing developer to resort to complicated structures and pointer arithmetic.

Extends MOSREF’s Console and Drone with packet sniffer and generation functionality.
Contact!

Wes Brown (wbrown@ephemeralsecurity.com)
Founder

Questions?

- Live question and answer
- Mailing list available for discussion and questions.
- Code is available via LGPL from http://www.ephemeralsecurity.com/