Exploit Writing Using Injectable Virtual Machines

Wes Brown - Scott Dunlop
Ephemeral Security

What?

- Mosquito is 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.
- Special care is taken to ensure that deployed code is not stored outside of process memory 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 the target’s 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.

Technical Overview

- Production-ready Code
- Virtual Machine Environment - MOSVM
- Language - Mosquito Lisp
- AES, and ECDH Encryption
- Extremely Portable (win32, OpenBSD, Darwin, Linux)

Virtual Machine (MOSVM)

- Lightweight and optimized for network tasks
- Easily extensible
- Lisp-family language with Schemish attributes
- Pure ANSI-C, portable (OpenBSD, Darwin, Linux, win32)
- Integrated ECDH, AES encryption with very good entropy generation.

Mosquito Components

- Core - Virtual Machine
- Environment - 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 scripts and statements on its behalf.

Core (MOSVM)

- Virtual Machine, with a Mosquito Lisp language compiler and environment for it.
- Integrates low level bindings such as libtomcrypt and regex.
- ‘Stub’ to append byte-code compiled libraries and programs to.
- Very small and compact.
- Very easy to write code in.
- Allows standalone executables with no dependencies.

Environment (Mosquito)

- Mosquito Lisp functions and libraries.
- Goodies such as XML parsing, regex, HTTP server in library.
- As little or as much as wanted can be attached to the virtual machine stub.
- Libraries can be dynamically pulled across the network.
- Well documented with reference available online.

Drone

- Virtual Machine + Crypto + Drone Functionality
- Highly optimized to reduce size
- Debugging and errors are resolved by the Console, to reduce strings.
- Does not include bytecode compiler; all compilation is handled by the Console.
- Executes and receives bytecode from Console.
- 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.
- Interface for interacting with Drones in real time.
- Creates Drones when requested using stub functionality.

Channels (Overview)

- Language feature, allowing for abstracted communications.
- A cryptographic channel is provided.
- 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.

Uses of Framework

- Refactor exploits into Mosquito Lisp for secure deployment on target.
- Easily extensible.
- 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

- Quick walkthrough of Mosquito Environment.
  - Compiler
  - Examples
  - Reference Manual

Exploit Writing

- Exploit writing in MOSVM is very easy.
- Demonstration
  - Writing Exploit.
  - Demonstrate injecting a Drone using Exploit.
  - Writing second Exploit.
  - Demonstrate injecting a Drone using second Exploit, using the injected Drone from the first exploit.

Who Are We?

- Wes Brown (wbrown@ephemeralsecurity.com)
  - Founder
- Scott Dunlop (swdunlop@ephemeralsecurity.com)
  - Developer
- http://www.ephemeralsecurity.com/ for more information.

Questions?

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