Lua on NetBSD

Scripting Operating Systems with Lua

BSDCon Brazil
October/2015

Lourival Vieira Neto <lneto@NetBSD.org>

"*Any sufficiently complicated C or Fortran program contains an ad hoc, informally-specified, bug-ridden, slow implementation of half of Common Lisp.*"

Greenspun's tenth rule

"Any sufficiently complicated C or Fortran program contains an ad hoc, informally-specified, bug-ridden, slow implementation of half of ~~Common Lisp~~ a good scripting language."

Ierusalimschy's first ~~Greenspun's tenth~~ rule

Topics

- Introduction
  - Scriptable Operating System
- Example
  - Packet Filter Scripting
- Why Lua?
- Kernel-scripting Environment
  - lua(4)
- Conclusions

Introduction

Scriptable Operating System

The combination of extensible operating systems with extension scripting languages.

Scriptable Operating System

- Motivation
  - Flexibility
    - Meet new user requirements
    - Configuration of kernel subsystems
  - Easy development
    - Allow application developers to customize the kernel
  - Prototyping
    - Add new features

Scriptable Operating System

- Key idea
  - OS **kernel scripting** with Lua
- Halfway between..
  - Kernel parameters and kernel modules
- Halfway between..
  - Domain-specific and system languages

Scriptable Operating System

- Two ways of scripting
  - Extending (a scripting language)
    - kernel as a library
      - Lua calls kernel
  - Embedding (a scripting language)
    - kernel as a framework
      - kernel calls Lua

Use Cases

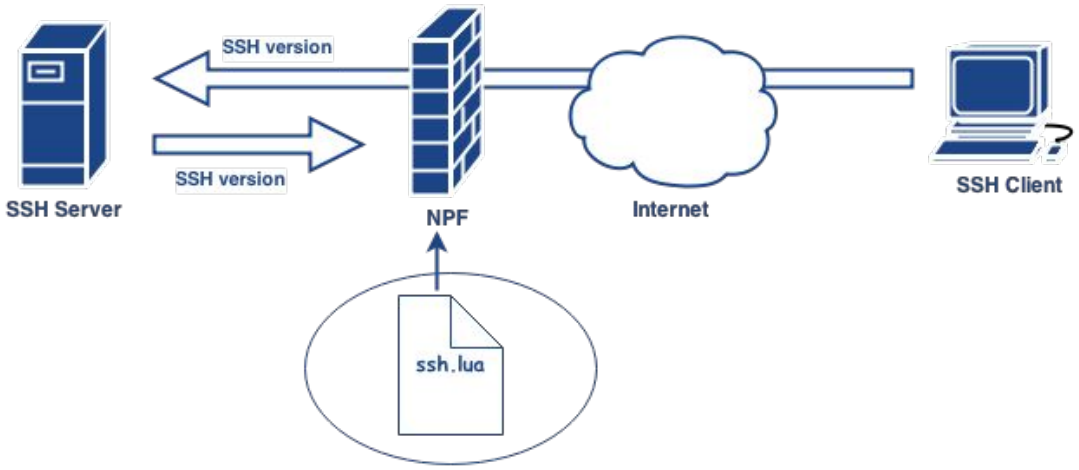
- Embedding
  - Packet filtering
  - Device drivers
  - Process scheduling
- Extending
  - Web servers
  - File systems
  - Network protocols

Example

Packet Filter Scripting

- Motivation
  - Deep packet inspection
    - Traffic shaping
    - Intrusion detection/prevention
  - New features
    - Port knocking
    - Protocols
    - Port stealthing

SSH Version


SSH version
SSH version
SSH Server
NPF
Internet
SSH Client
ssh.lua

SSH Version

```
1.  local data = require'data'
2.
3.  function filter(pkt)
4.    -- convert packet data to string
5.    local str = tostring(pkt)
6.
7.    -- pattern to capture the software version
8.    local pattern = 'SSH%-[^-%G]+%-([^-%G]+)'
9.
10.   -- get the software version
11.   local software_version = str:match(pattern)
12.
13.   if software_version == 'OpenSSH_6.4' then
14.     -- reject the packet
15.     return false
16.   end
17.
18.   -- accept the packet
19.   return true
20. end
```

SSH Version

- No measurable overhead
  - 96 Mbps on both cases (on 100 Mbps virtual NIC)
- Binding
  - 217 lines of C code
- Script (ssh.lua)
  - 22 lines of Lua code

NPF

- The NetBSD Packet Filter
  - Layers 3 and 4
  - Stateful
  - IPv4 and IPv6
  - **Extensible**
    - Rule procedures

NPFLua

- Binds NPF to Lua
  - Kernel module + parser module
  - Rule procedure

```
#npf.conf
procedure "lua_filter" {
  lua: call filter
}

group default {
  pass in all apply "lua_filter"
}
```

  - Script loading

```
luactl load npf ./filter.lua
```

Why Lua?

Why Lua?

- Extensible extension language
  - Embeddable and extensible
  - C library
- Almost freestanding
- Small footprint
  - has 240 KB on -current (amd64)
- Fast
- MIT license

Why Lua?

- Safety features
  - Automatic memory management
  - Protected call
  - Fully isolated states
  - Cap the number of executed instructions

Why not .... ?

- Python
  - has 2.21 MB on Ubuntu 10.10 (amd64)
- Perl
  - has 1.17 MB on Ubuntu 10.10 (amd64)
- Also..
  - OS-dependent code
  - Hard to embed[1]

1. twistedmatrix.com/users/glyph/rant/extendit.html

Kernel-scripting Environment: lua(4)

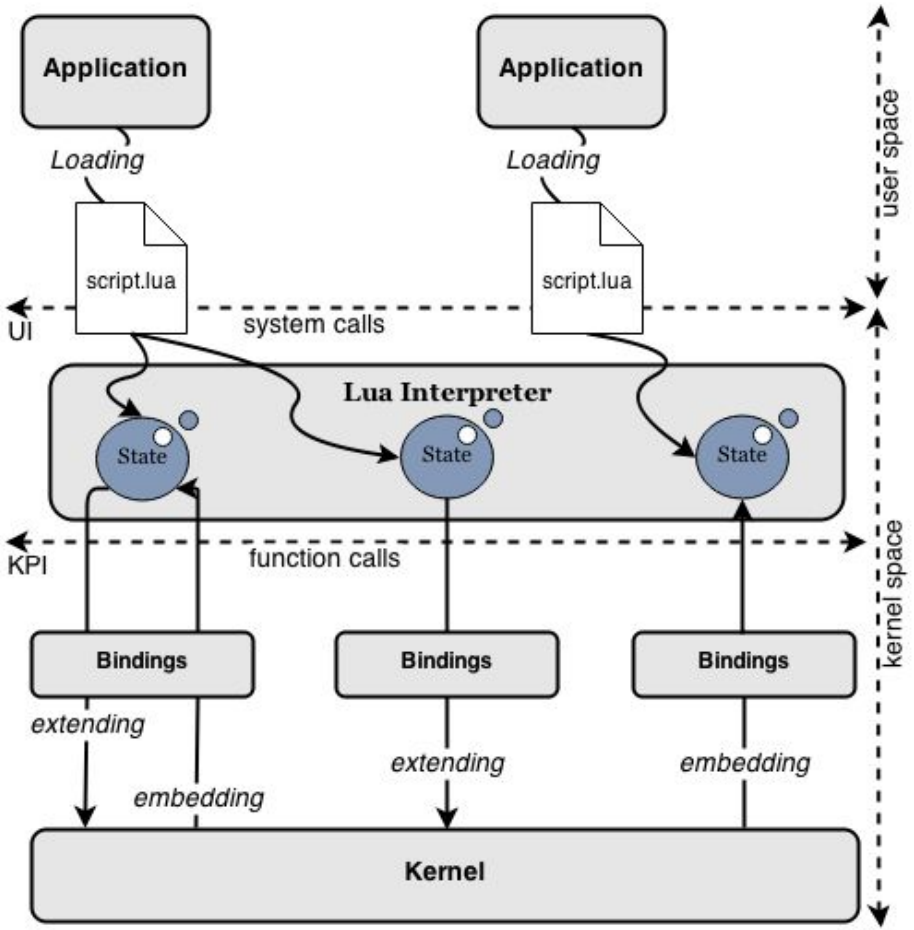
Brief History

- 2008 - Lunatik/Linux
- 2010 - Lunatik/NetBSD
  - Google Summer of Code
  - Kernel-embedded Lua (mainly)
- 2013 - Lua(4)
  - New infrastructure (Marc Balmer)
- 2014 - NPFLua
- 2015 - Ported Lua Test Suite
  - Google Summer of Code (Guilherme Salazar)

Lua(4)

- Kernel-embedded Lua
  - has no floating-point numbers
- User Interface
  - luactl
- Kernel Programming Interface
  - sys/lua.h

Operation Overview



Conclusions

Conclusions

- General-purpose and full-fledged programming language for scripting kernels
  - e.g., pattern matching, hash table
- First to provide scripting both by extending and embedding an interpreter
- Part of the official NetBSD distribution
- Impact
  - A. Graf. PacketScript—a Lua Scripting Engine for in-Kernel Packet. Processing. Master's thesis, Computer Science Department, University of Basel, July 2010.
  - M. Grawinkel, T. Suss, G. Best, I. Popov, and A. Brinkmann. Towards Dynamic Scripted pNFS Layouts. In High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:, pages 13–17. IEEE, 2012.
  - A. Cagney. What happens when a DWARF and a daemon start dancing by the light of the silvery moon? BSDCan 2015 (Talk).
  - A. Koomsin and Y. Shinjo. lua_syscall: Specializing Operating System Kernels by Using the Lua Language. 6th ACM SIGOPS Asia-Pacific Workshop on Systems (APSys 2015) (Poster).
  - K. Rytarowski. Moduły Dynamiczne w Kernelu NetBSD. Programista. 5/2015 (Polish Magazine).
  - A. Koomsin, Y. Shinjo. Running Application Specific Kernel Code by a Just-in-Time Compiler. 8th ACM PLOS 2015.

Questions and Answers

- Questions?

Contact Information

- Lourival Vieira Neto <lneto@NetBSD.org>

More Information

- L. Vieira Neto, R. Ierusalimschy, A. L. de Moura and M. Balmer. Scriptable Operating Systems with Lua. Dynamic Languages Symposium 2014. URL netbsd.org/~lneto/dls14.pdf.

System Memory Binding: Luadata

Luadata

- Regular Lua library
  - Kernel and user space
- Binds system memory
  - Memory block (pointer + size)
  - mbuf
- Safe
  - Boundary verification
- Packed data
  - Declarative layouts

Luadata

- Other features
  - Bit fields
  - String fields and conversion
  - Segments (data decomposition)
  - Endianness conversion

RTP Encoding


```
1.  local rtp = {
2.    version     = {0, 2},
3.    extension   = {3, 1},
4.    csrc_count  = {4, 4},
5.    marker      = {8, 1},
6.    type        = {9, 7}
7.  }
8.
9.  -- apply RTP header layout in the payload
10. pld:layout(rtp)
11.
12. -- if packet is encoded using H.263
13. if pld.type == 34 then
14.   -- reject the packet
15.   return false
16. end
```