NPF Scripting with Lua

Scripting the NetBSD Packet Filter with Lua

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

Topics

- Introduction
  - Scriptable Operating System
  - Scripting a Packet Filter
- Example
  - SSH version
- Issues
- Why Lua?
- Kernel-scripting Environment
  - Lua(4)
- Luadata
- NPFLua
- Conclusion

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
- Halfway between..
  - Kernel parameters and kernel modules
  - Domain-specific and system languages

Scriptable Operating System

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

Use Cases

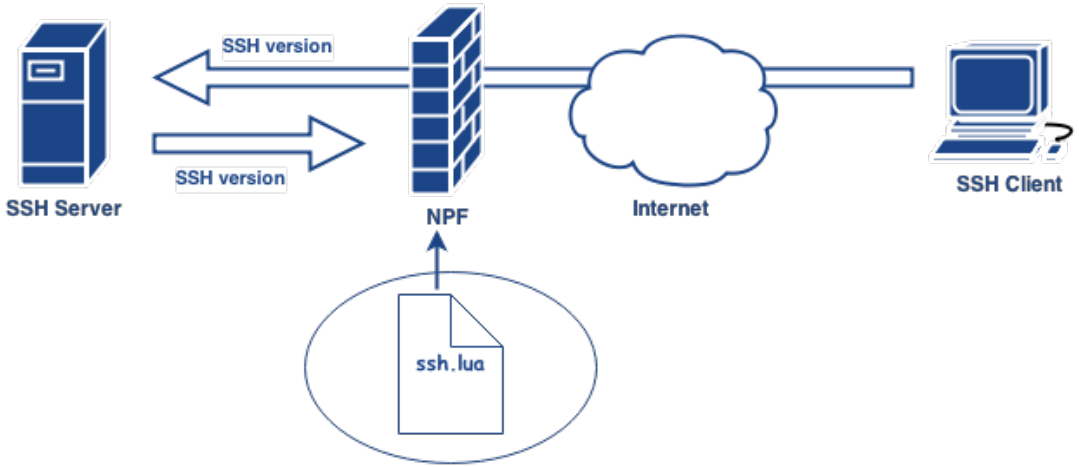
- Embedding
  - Packet filtering
  - Device drivers
  - Process scheduling
- Extending
  - Web servers
  - File systems
  - Network protocols

Packet Filter Scripting

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

Example

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
```

Issues

Issues

- System integrity
  - Correctness
  - Isolation
  - Liveliness
- Ease of development
- Effectiveness and efficiency

System Integrity

- Correctness
  - Sandboxing
  - Automatic memory management
  - “Single” thread
  - Protected call / fail-safe
  - Privileged only
- Isolation
  - Fully isolated execution states
- Liveliness
  - Cap the number of executed instructions

Ease of Development

- High-level language
- Dynamically typed
- Domain-specific API

Effectiveness and Efficiency

- Proper bindings
  - Interface between scripts and kernel
  - Suited for addressing SOS issues
  - Most difficult task

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)

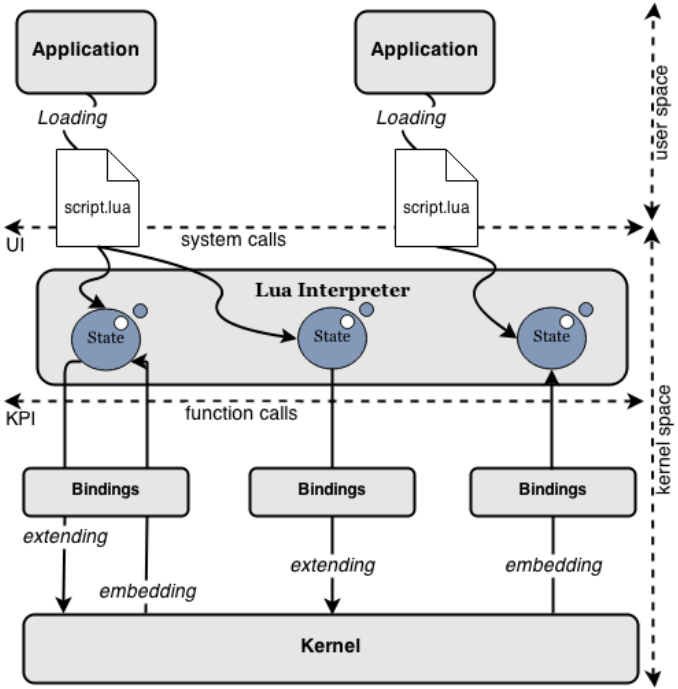
Brief History

- 2008 – Lunatik/Linux
- 2010 – Lunatik/NetBSD
  - Google Summer of Code
  - Kernel-embedded Lua (mainly)
- 2013 – Lua(4)
  - New infrastructure (Marc Balmer)
- 2014 – NPFLua

Lua(4)

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

Lua(4)


Application
Application
Loading
Loading
user space
script.lua
script.lua
UI
system calls
Lua Interpreter
State
State
State
KPI
function calls
kernel space
Bindings
Bindings
Bindings
extending
extending
embedding
embedding
Kernel

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

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
```

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
```

Packet Filter Binding: NPFLua

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
```

Conclusion

Work in Progress

- Actual Lua rules, e.g.:

```
block out final lua-filter "filter.lua"
```

- mbuf handling
  - m_pulldown fail-safe
  - non-contiguous strings using `luaL_Buffer()`
  - packet mangling
- Multiple Lua states
- Automatic script loading
  - filter.lua
- Pre-defined layouts
  - IP, TCP, UDP
- Lua network library
- Rule editing
- Lua user-space configuration
  - /etc/npf.lua

Conclusion

- General-purpose and full-fledged programming language for packet filtering
  - e.g., pattern matching, hash table
- "SSH Version" example
  - no measurable overhead (on 100 Mpbs virtual NIC)
    - 96 Mbps with or without scripting
  - 20 lines of Lua code
- Luadata is a generic binding for memory
  - can be used for other kernel extensions
    - e.g., device drivers, network protocols

References

- 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.
- M. Rasiukevicius. NPF—Progress and Perspective. AsiaBSDCon 2014.
- M. Rasiukevicius. NPF documentation. URL netbsd.org/~rmind/npf/.

- Source code:
  - netbsd.org/~lneto/pending/
  - github.com/lneto/luadata

Questions and Answers

- "Got questions?"

Contact Information

Lourival Vieira Neto
lneto@NetBSD.org