IEEE SPW: LangSec'17 (San Jose, CA)

Lua Code: Security Overview and Practical Approaches to Static Analysis

Andrei Costin
ancostin@jyu.fi, andrei@firmware.re
University of Jyvaskyla, Finland

Agenda

- **Introduction**
- Contributions
- Implementation, examples, results
- Conclusions
- Acknowledgements and Q&A

Introduction

- **Lua** (Moon in Brazilian/Portuguese)
  - Ierusalimschy et al., Pontifical Catholic University of Rio de Janeiro in Brazil (PUC-Rio) [IER96]
- Interpreted, cross-platform, **embeddable**, performant and low-footprint language
- Supports *"extensible semantics, anonymous functions, full lexical scoping, proper tail calls, and coroutines"* [IER96]
- Many Lua resources: https://github.com/LewisJEllis/awesome-lua

Introduction

- Lua's popularity is on the rise
- TIOBE Index
  - 27th most popular (May 2017)
  - *Par* or above: *T-SQL*, Lisp, Ada, Fortran, Scala, LabVIEW, Prolog, Haskell, Erlang, Bash
- PYPL Index
  - 19th most popular (May 2017)
  - *Par* or above: *Go*, Delphi, Haskell

Introduction

- Lua in numbers
  - PHP is 16x-to-20x more „popular“ (PYPL Index, GitHub repository count by „language:“)
  - Still, around 30k Lua-based GitHub repositories
  - Several millions ESP8266, ready for NodeLua/NodeMCU Lua firmware
  - Huge number of other devices with Lua support/APIs

Introduction

- Lua in notorious use cases
  - Web-facing Projects
    - Wikipedia, GitHub, CloudFlare
  - Tools, Projects
    - Nmap, Wireshark, OpenWRT
  - Conventional Malware
    - Flamer, EvilBunny, ProjectSauron

Introduction

- Lua in notorious use cases
  - IoT-specific Malware
    - LuaBot
  - Incredible amount of other important but less known projects
    - IoT
    - Home Automation
    - SCADA/ICS
    - Automotive
    - Wireless/Mobile Chipsets

Introduction: Motivation

- Zero SAST tools for Lua code
  - Many tools/services for other languages
  - Coverity, VeraCode, AppScan, CodeClimate, RIPS, etc.

- Zero datasets with (intentionally) vulnerable Lua samples for experimentation
  - Many datasets/projects for other languages
  - BugBox, DVWA, WebGoat, SQLol, etc.

- Not much systematic research on Lua security, e.g., [DAR14]

Agenda

- Introduction
- **Contributions**
- Implementation, examples, results
- Conclusions
- Acknowledgements and Q&A

Contributions

- Develop and open-source the first and only static analysis tool for Lua code
- Build and open-source the first public corpus of synthetic Lua code samples
- Create and release the testing setups used in our experiments in form of virtual and reproducible environments

Agenda

- Introduction
- Contributions
- **Implementation, examples, results**
- Conclusions
- Acknowledgements and Q&A

Implementation

- www.lua.re
- ANTLR4-based Python parser [PAR13]
- Lua.g4 from ANTLR's Grammars-V4 repository [SAK13]
- Built-in unit-tests
  - *$MSL/tests/test_msl_defaultconfig.py*
  - *$MSL/tests/test_msl_VariousTests1.py*
  - *$MSL/tests/test_msl_LangSec17.py*
- Own Python-based unsophisticated taint engine
  - *$MSL/taint/*

Implementation

- Flexible configurations and taint rules
  - *$MSL/config/defaultconfig.py*
  - Taint sensitive sinks (e.g., io.write)
  - Taint unsanitizers (e.g., htmlunescape)
  - Taint sanitizers (e.g., htmlentities)
  - Taint propagation/passthru (e.g., strcat and '..' concat operator)
  - Some combinations of above (e.g., see fake_strcat_print_popen)

Examples, Results

- Detects all the simple synthetic TP test-cases and Avoids all the simple synthetic FP test-cases
  - *$MSL/tests/test_msl_VariousTests1.py*
  - *$MSL/tests/test_msl_LangSec17.py*

- Works on simple real-world code
  - CVE-2014-4329: *„Cross-site scripting (XSS) vulnerability in lua/host_details.lua in ntopng 1.1 allows remote attackers to inject arbitrary web script or HTML via the **host** parameter.“*

Examples, Results

- CVE-2014-4329 with our tool: „... *via the **host** **and page** parameters.*“

```
VULN type(s): set(['xss'])
VULN line: 417
VULN column: 1
VULN code: print(..host_ip..)
```

```
VULN type(s): set(['xss'])
VULN line: 688
VULN column: 3
VULN code: print(page)
```

```
 9 dirs = ntop.getDirs()
10 package.path = dirs.installdir .. "/scripts/lua/modules/?.lua;" .. package.path
11
12 require "lua_utils"
13 require "graph_utils"
14
15 page    = _GET["page"]
16 host_ip = _GET["host"]
17
18 active_page = "hosts"
19
20 if(host_ip == nil) then
21    sendHTTPHeader('text/html')
```

Agenda

- Introduction
- Contributions
- Implementation, examples, results
- **Conclusions**
- Acknowledgements and Q&A

Conclusions

- Lua is a powerful and performant dynamic language
- Lua's popularity is on the rise within the embedded/IoT applications
- Obvious lack of both static analysis tools for Lua code and corpora of vulnerable Lua code samples
- We bridge the gap by open-sourcing: Lua SAST tool, vulnerable code samples

Conclusions and Future Work

- Dramatically improve performance
- Improve the parser/lexer (e.g., fails on some real-world code snippets)
- Add missing features (e.g., dofile() and includes)
- Improve taint engine and rules
  - Generic configurable taint engine?
  - Interface with Joern engine [JOER]

Agenda

- Introduction
- Contributions
- Implementation, examples, results
- Conclusions
- **Acknowledgements and Q&A**

Acknowledgements

- NLnet.nl Foundation and Binary Analysis Tools (BAT) Project
  - This project was supported by the NLnet.nl grant: **2014-09-017e**
- Michiel Leenaars from NLnet foundation
- Armijn Hemel from Tjaldur Software Governance Solutions
- LangSec'17 reviewers, shepherds and organizers!

Q&A

- Questions, suggestions, ideas?

www.lua.re

ancostin@jyu.fi

andrei@firmware.re

Twitter: @costinandrei

References

- [IER96] R. Ierusalimschy, L. H. De Figueiredo, and W. Celes Filho, “Lua – an extensible extension language”, 1996
- [PAR13] T. Parr, "The definitive ANTLR 4 reference". Pragmatic Bookshelf, 2013
- [SAK13] K. Sakamoto, A. Alexeev, https://github.com/antlr/grammars-v4/blob/master/lua/Lua.g4
- [JOER] F. Yamaguchi, "An Intelligent and Robust Code Analysis Platform for C/C++"
- [DAR14] F. Daragon, „Lua Web Application Security Vulnerabilities“

IEEE SPW: LangSec'17 (San Jose, CA)

Lua Code: Security Overview and Practical Approaches to Static Analysis

Andrei Costin
ancostin@jyu.fi, andrei@firmware.re
University of Jyvaskyla, Finland