String Hunting for Hardcoded Credentials and Clear Text Data with Control Flow Graph

April 2022

Bryan R Beckman
DISCLAIMER

This information was prepared as an account of work sponsored by an agency of the U.S. Government. Neither the U.S. Government nor any agency thereof, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness, of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the U.S. Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the U.S. Government or any agency thereof.
String Hunting for Hardcoded Credentials and Clear Text Data with Control Flow Graph
Who am I?

Introduction

- Senior Critical Infrastructure Researcher at Idaho National Laboratory for the past 5 years
- BS Electrical Engineering from Brigham Young University
- MS Electrical and Computer Engineering from University of Nevada Las Vegas
- Busy Father of six high-energy children
INL is currently working on 2 major data sets for cyber security of critical infrastructure.

Structured threat is traditionally used for **cyber threat analysis**. We are now using it for modeling implementation specifics for infrastructure enabling codified **attack surfaces**; and using machine learning and graph databases for **translated binary analysis**.
Research History – Critical Infrastructure Resilience

DOE-OE-CEDS or CESER Energy Sector Cybersecurity Funded Projects

- Over 40 Vulnerability Assessments National SCADA Testbed 2003; Training 2005-11; Research since 2010 - Ongoing Firmware, Malware, Structured Threat, Risk Management 2016 – CyTRICS, CyOTE

Department of Homeland Security Funded Projects

- Control System Security Program FY2004-9; Red/Blue Team Training FY05
- Industrial Control System-CERT FY2009-2016 Advanced Analytics Laboratory
- NAVV – Network Analysis Verification and Validation, CSET – Cyber Security Evaluation Tool & RRAP - Regional Resiliency Assessment Program; AVA – Automated Vulnerability Assessment 2016

Other (funding source)

- California Energy System for the 21st Century - Automated Threat Response Evidence Based (CPUC) 2015-19
- DARPA Rapid Attack Detection, Isolation and Characterizations Systems (RADICS) – Plum Island set up FY17-21
- Joint Cyber Training Enterprise (JCTE) More Situational Awareness for ICS (MOSAIC)
- INL LDRD – Binary Analysis RE@Scale FY17-18, Infrastructure eXpression FY20-21. Consequence-Driven Cyber-Informed Engineering 2016/17
- INK Deep Bench on Cyber Security for Control Systems
- Cyber Core; Infrastructure Assurance & Analysis; Critical Infrastructure Protection: Infrastructure Security; Wireless Innovation

>350 Employees
Infrastructure Expression (IX)

Infrastructure Expression (IX)

- Idaho National Laboratory funded internal Laboratory Directed Research and Development (LDRD) FY20-21
- https://github.com/idaholab/IX-DiscoveryTools
  AutoDiscovery, HardwareIX, DataFlo-IX, ForensIX + more utilities and prototyped SBOM from STIX

- Based on open international standard – Structured Threat Information eXpression (STIX) v2.1
- Visualization by Structured Threat Intelligence Graph (STIG) https://github.com/idaholab/STIG

IX – Codified Attack Surface

Distribution

Substation

Hardware IX

Structured Threat Intelligence Graph

IDAHO NATIONAL LABORATORY
IX – Codified Attack Surface Firmware

Distribution
Substation
IX – Codified Attack Surface – Vulnerabilities

Distribution Substation

---

**CVE-2010-5107**

**CVE-2010-4755**

**CVE-2016-0777**

**CVE-2017-15006**

**CVE-2011-5000**

**CVE-2012-0814**

**CVE-2011-4327**

**CVE-2010-4478**

---

**Distribution**

**Substation**

---

**STIG**

**Structured Threat Intelligence Graph**

---

**IDAHO NATIONAL LABORATORY**
Firmware Indicator Translation Products

What is it? Binary (WiiBin) [github.com/idaholab/WiiBin]
- Initial Forensics Triage tool, for unknown binary using ML Methods
- Discover: Endianness; Architecture; Potential OpCodes to aid RE
- Requires Binwalk; Good Results with 30% of the Original File

Annotated Translated Disassembled Code (@DisCo) [github.com/idaholab/AtDisCo] – Additional Export Control
- Reverse Engineer Binaries leveraging RE@Scale methods
- Test Corpora for Supervised ML; finding similarities and difference distinctions: Cluster; Classification; Analysis
- Graph DB – OrientDB; angr with CFGFast for control flow graphs; feature sets: PyVex intermediate language;
- Use Cases: Augmented RE, Supply Chain and SQA

DISCOverFlow
- 3D pin visualization with DB query
- 3D springs visualization
- Control and Execution Flow Graph DB query for investigations

find_string.py aka String Hunter

• Developed as part of the @DisCo toolset by Michael Cutshaw
• Generates control flow graph with an overlay of extracted human readable strings.
• These strings can include:
  – Hard Coded Credentials
  – Sensitive Information
  – Proprietary Details
• The combination of strings and control flow graph can provide additional insights which would be difficult to identify otherwise.
String Hunter Command line Options

- **marking_filter**: All nodes that match the marking_filter are highlighted green (RegEx)
- **funcname_filter**: Only nodes matching the funcname_filter are output (RegEx)
- **string_filter**: Only strings matching the string_filter are output (RegEx)
- **save-pickle**: Saves resulting graph for faster reloading
- **resolve-indirect-jumps-false**: Disables Indirect Jump Resolution in angr
- **functions_only**: Display only function nodes (no block nodes)
- **binary**: Selected binary file to process

Commandline Usage:
```
```

https://www.w3schools.com/python/python_regex.asp
CWE-312 Cleartext Storage of Sensitive Information (Example)

- We can identify CWE-312 through identification of cleartext strings within a binary.
- This test binary was created simply to illustrate the features of String Hunter
CWE-312 Cleartext Storage of Sensitive Information (Example)
CWE-312 Cleartext Storage of Sensitive Information (Example)
CWE-798 Use of Hard-coded Credentials (Test Login)

Source Code of test login binary

```c
#include <stdio.h>
#include <string.h>

struct loginStruct {
    char username[20], password[20];
};
typedef struct loginStruct lStruct;

lStruct prompt1[] = "Please enter a username: ";
char username[20];
char password[20];
printf("%s", prompt1);
scanf("%s", username);
printf("%s", prompt2);
scanf("%s", password);
lStruct ls;
strcpy(ls.username, username);
strcpy(ls.password, password);
return ls;
}

int login(lStruct creds){
    char s[10] = "ADMIN";
    char t[10] = "PASSWORD";
    return strcmp(s, creds.username) && strcmp(t, creds.password);
}

int main() {
    lStruct ls;
    ls = prompt1();
    login(ls);
    return 0;
}
```

Execution of test login binary

bryan@ubuntu:~/string_finder$ ./test_login
Please enter a username: User
Please enter a password: Pass
bryan@ubuntu:~/string_finder$ ./test_login
Please enter a username: ADMIN
Please enter a password: PASSWORD
bryan@ubuntu:~/string_finder$  

CWE-798 Use of Hard-coded Credentials (Zyxel)

- CVE-2020-29583 represents a use of hardcoded credentials within version 4.60 Patch 0 firmware of Zyxel USG devices (firewalls).
- We can identify CWE-798 through identification of cleartext credential strings for a binary within that firmware image.
- Note: This vulnerability was patched on December 18, 2020 via firmware version 4.60 Patch 1.
CWE-798 Use of Hard-coded Credentials (ZyXel)
CWE-798 Use of Hard-coded Credentials (Zyxel)
CWE-798 Use of Hard-coded Credentials (Zyxel Unfiltered)

If filters are not used the size of the resulting graphs can quickly be too large and difficult to used.
Conclusion / Next Steps

Conclusion:

• The overlay of human readable strings on top of a code’s control flow graph enables additional insights to be made which might not be able to be made as easily otherwise

• Identifying that hard coded credentials are present in your binaries can help improve your security posture by motivating one to remove those credentials and close the associated backdoor

Next steps:

• Automate generation of STIX graph from angr control flow graph
• Enhance search and filtering capabilities