Turun yliopisto
University of Turku

Classifying Web Exploits with Topic Modeling

Jukka Ruohonen

TIR 2017, Lyon

Outline

1. Terminology
2. Motivation
3. Data
4. Examples
5. Processing
6. Classification
7. Results
8. Conclusion

Terminology

- Software **vulnerabilities** are security-related bugs
- **Exploits** are implementations targeting such bugs
  - To compromise a system, to cause a denial-of-service, etc.
- Yet, **proof-of-concept** (PoC) exploits are slightly different
  - Neither written nor used for actual attacking
  - Typically used during vulnerability **disclosure**
  - Though, also for money and fame & glory
  - This said, ethical issues are still present also with PoCs

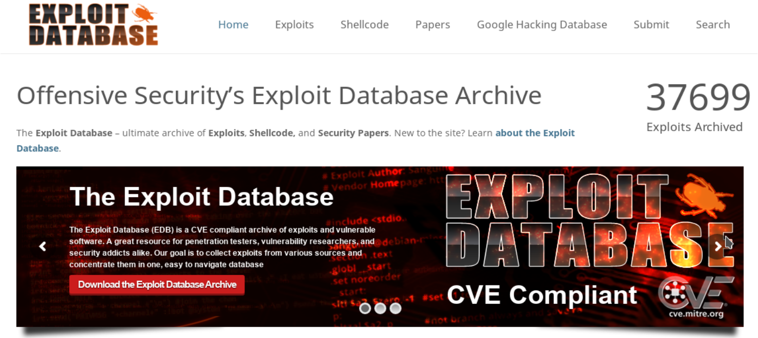
Motivation

- The **demand** for exploits has increased in recent years
  - Penetration testing and offensive security in general, etc.
- **Archiving** of vulnerabilities and exploits requires a lot of work
  - Recent delays in **CVE assignment** via MITRE Corporation
  - **OSVDB** was shutdown due to maintenance problems
- Thus, a basic question is how to **automate** the archiving?
  - Basically, assign a case to a predefined meta-data category
  - Related work in software engineering ("**bug triaging**")

Data

- 36184 raw exploits archived in **Exploit Database** (EDB)
- The exploits archived are in **unstructured text** format
  - PoC code, disclosure events, attribution credits, etc.
  - Gathered from mailing lists, bug trackers, blogs, etc.
- A number of **meta-data categories** are present
  - Based on manual classification done by EDB maintainers
  - In this work, web and PHP categories are used for brevity

Examples (1/3)



Offensive Security's Exploit Database Archive

37699
Exploits Archived

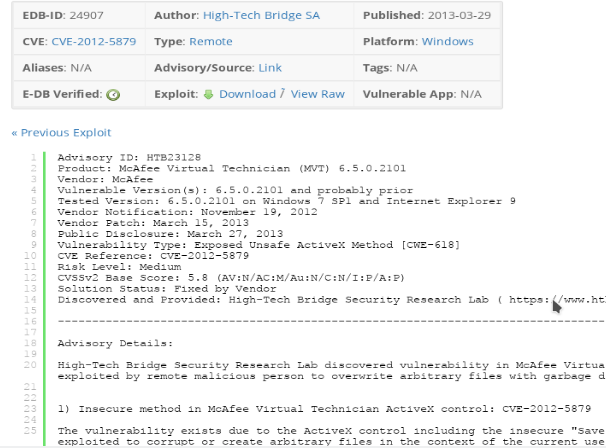
The **Exploit Database** – ultimate archive of **Exploits**, **Shellcode**, and **Security Papers**. New to the site? Learn **about the Exploit Database**.

The Exploit Database

The Exploit Database (EDB) is a CVE compliant archive of exploits and vulnerable software. A great resource for penetration testers, vulnerability researchers, and security addicts alike. Our goal is to collect exploits from various sources and concentrate them in one, easy to navigate database

Figure: Source: EDB (`https://www.exploit-db.com/`), August 2017

Examples (2/3)



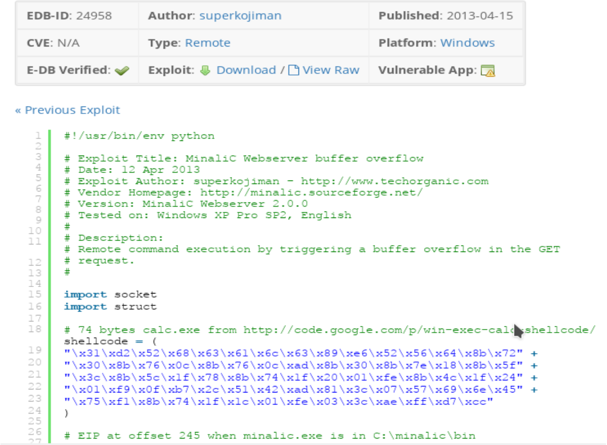
EDB-ID: 24907	Author: High-Tech Bridge SA	Published: 2013-03-29
CVE: CVE-2012-5879	Type: Remote	Platform: Windows
Aliases: N/A	Advisory/Source: Link	Tags: N/A
E-DB Verified: ✓	Exploit: Download / View Raw	Vulnerable App: N/A

« Previous Exploit

```
Advisory ID: HTB23128
Product: McAfee Virtual Technician (MVT) 6.5.0.2101
Vendor: McAfee
Vulnerable Version(s): 6.5.0.2101 and probably prior
Tested Version: 6.5.0.2101 on Windows 7 SP1 and Internet Explorer 9
Vendor Notification: November 19, 2012
Vendor Patch: March 15, 2013
Public Disclosure: March 27, 2013
Vulnerability Type: Exposed Unsafe ActiveX Method [CWE-618]
CVE Reference: CVE-2012-5879
Risk Level: Medium
CVSSv2 Base Score: 5.8 (AV:N/AC:M/Au:N/C:N/I:P/A:P)
Solution Status: Fixed by Vendor
Discovered and Provided: High-Tech Bridge Security Research Lab ( https://www.ht

-------------------------------------------------------------------------------

Advisory Details:

High-Tech Bridge Security Research Lab discovered vulnerability in McAfee Virtua
exploited by remote malicious person to overwrite arbitrary files with garbage d


1) Insecure method in McAfee Virtual Technician ActiveX control: CVE-2012-5879

The vulnerability exists due to the ActiveX control including the insecure "Save
exploited to corrupt or create arbitrary files in the context of the current use
```

Figure: Source: EDB (https://www.exploit-db.com/), August 2017

Examples (3/3)



EDB-ID: 24958	**Author**: superkojiman	**Published**: 2013-04-15
CVE: N/A	**Type**: Remote	**Platform**: Windows
E-DB Verified: ✓	**Exploit**: Download / View Raw	**Vulnerable App**:

« Previous Exploit

```
#!/usr/bin/env python

# Exploit Title: MinaliC Webserver buffer overflow
# Date: 12 Apr 2013
# Exploit Author: superkojiman - http://www.techorganic.com
# Vendor Homepage: http://minalic.sourceforge.net/
# Version: MinaliC Webserver 2.0.0
# Tested on: Windows XP Pro SP2, English
#
# Description:
# Remote command execution by triggering a buffer overflow in the GET
# request.
#

import socket
import struct

# 74 bytes calc.exe from http://code.google.com/p/win-exec-calc-shellcode/
shellcode = (
"\x31\xd2\x52\x68\x63\x61\x6c\x63\x89\xe6\x52\x56\x64\x8b\x72" +
"\x30\x8b\x76\x0c\x8b\x76\x0c\xad\x8b\x30\x8b\x7e\x18\x8b\x5f" +
"\x3c\x8b\x5c\x1f\x78\x8b\x74\x1f\x20\x01\xfe\x8b\x4c\x1f\x24" +
"\x01\xf9\x0f\xb7\x2c\x51\x42\xad\x81\x3c\x07\x57\x69\x6e\x45" +
"\x75\xf1\x8b\x74\x1f\x1c\x01\xfe\x03\x3c\xae\xff\xd7\xcc"
)

# EIP at offset 245 when minalic.exe is in C:\minalic\bin
```

Figure: Source: EDB (https://www.exploit-db.com/), August 2017

Processing

- A **pre-processing** routine with six steps
  - Including tokenization, lemmatization, stop words, etc.
- Separation of English **words** and non-English **terms**
- Word and term frequency matrices are used for LDA
  - That is, the **Latent Dirichlet Allocation** (LDA) method
  - Each exploit is assigned to the **most dominant** (text or word) topic according to the highest membership rate
- **Number of topics** (k) restricted to $k = 5, 10, 20, 30, 40, 50$
  - Default settings and parameters used otherwise (R impl.)

Classification (1/4)

- Separate classifiers for **two categories**
  - Web exploits and exploits targeting PHP
  - Results almost perfectly **balanced** data
- Computation with the **random forest** algorithm
- In total, **40 features** (from which two are LDA-based)
  - Many are well-known metrics (which require manual work)
  - How much performance is gained from the LDA-metrics?

Classification (2/4)

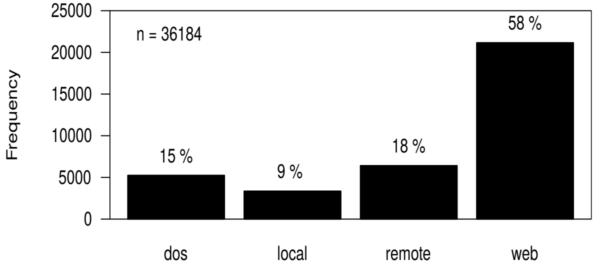


Figure: Response category #1 (“web”)

Classification (3/4)

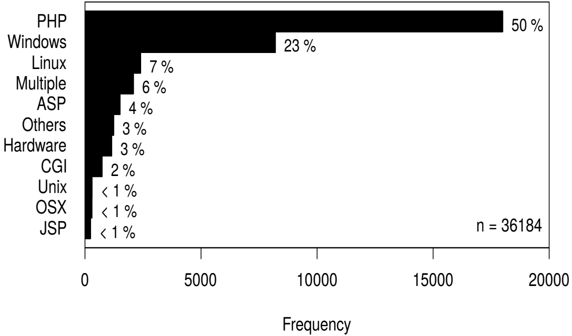


Figure: Response category #2 ("PHP")

Classification (4/4)

#	Description
1.	One for the most dominant **term-based topic** characterizing the exploit.
2.	One for the most dominant **word-based topic** characterizing the exploit.
3.	One if the EDB community has **verified** the exploit.
4.	One if the vulnerable application is available for **download**.
5.	One if a **screenshot** is provided for a demonstration or other purposes.
6.	The number of **OSVDB references** or zero for no such references.
7.	The number of **CVE references** or zero for the absence of CVE references.
8.	The mean of **CVSS base scores** for all CVE references (or zero for no refs.).
9.	The **year** at which the exploit was first published according to EDB.
10.	The **month** at which the exploit was first published according to EDB.
11. – 40.	One if the author of the exploit is among the **“top-30” developers**.

Results

k	Covariates	Accuracy	
		Web [95 % CIs]	PHP [95 % CIs]
0	38	0.788 [0.765, 0.810]	0.742 [0.717, 0.766]
5	40	0.895 [0.877, 0.911]	0.843 [0.821, 0.862]
10	40	0.910 [0.893, 0.925]	0.861 [0.841, 0.880]
20	40	**0.920** [0.904, 0.935]	**0.888** [0.869, 0.905]
30	40	0.912 [0.894, 0.927]	0.881 [0.862, 0.898]
40	40	0.914 [0.897, 0.929]	0.863 [0.843, 0.882]
50	40	0.913 [0.896, 0.928]	0.878 [0.858, 0.895]

Conclusion

- The accuracy range $[0.89, 0.92]$ is **good** in the context
  - But the statistical performance mostly comes from **conventional metrics** that require manual work
  - Should test how well **plain frequency matrices** work
  - **Multi-class** classification required in practice
- How to **separate PoC code** from other content?
  - Not as easy as separating code from code comments
  - Would have practical value in security and threat intelligence

Thank you

Questions?