ROOTCON 13

Pilot Study on Semi-Automated Patch Diffing by Applying Machine-Learning Techniques

Asuka Nakajima (@AsuNa_jp)

NTT Secure Platform Laboratories

#whoami

- **Asuka Nakajima** *(@AsuNa_jp)*

  Security Researcher at NTT Secure Platform Laboratories

  Vulnerability Discovery / Reverse Engineering / IoT Security

- **Founder of “CTF for GIRLS”**

  CTF for GIRLS

  First female infosec community in Japan (est.2014)

- **Black Hat Asia Review Board**

  black hat ASIA 2019

  From 2018-Present

- **Veteran Conference/Event Speaker**

  BlackHatUSA 2019, AsiaCCS 2019, AIS3 2018/2016, PHDays IV, SECCON,etc

Agenda

Background

PART 1 Extracting Security Fix Patterns Using Unsupervised Machine Learning Algorithm*

PART 2 Classifying Security Fixes and Other Fixes*

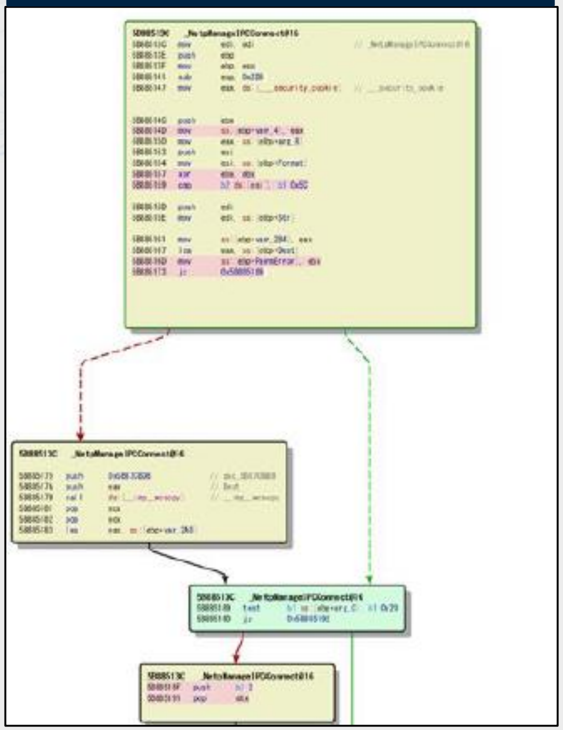
Conclusion

*Original Paper

Asuka Nakajima, Ren Kimura, Yuhei Kawakoya, Makoto Iwamura, Takeo Hariu, "An Investigation of Method to Assist Identification of Patched Part of the Vulnerable Software Based on Patch Diffing" Multimedia, Distributed, Cooperative, and Mobile Symposium, June 2017, Japan

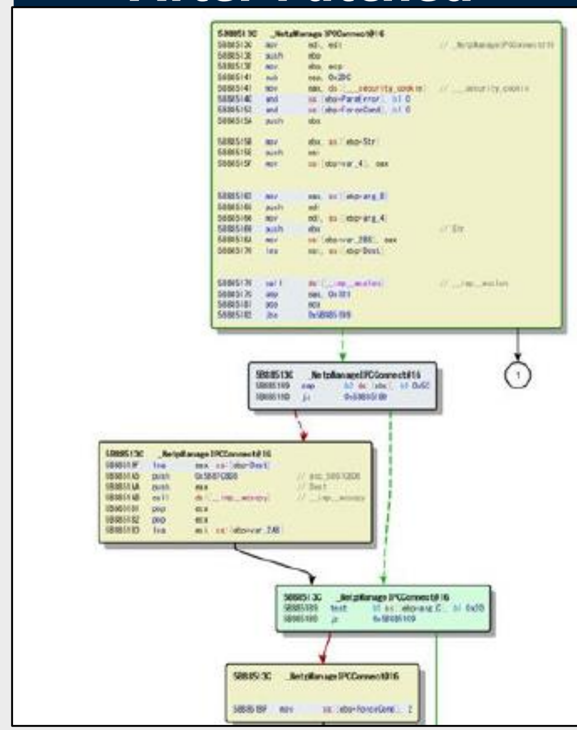
What is Patch Diffing?

Before Patched



Compare

After Patched



Identify vulnerable part & Create 1-day exploit

Example: CVE-2006-4691 (MS06-70)

Windows

Stacked-Based Buffer overflow in NetpManageIPCConnect Function

Before Patched (netapi32.dll)

```
5B88513C    NetpManageIPCConnect(x, x, x, x)
5B88513C    mov     edi, edi                    // _NetpManageIPCConnect@16
5B88513E    push    ebp
5B88513F    mov     ebp, esp
5B885141    sub     esp, 0x2D8
5B885147    mov     eax, ds:[___security_cookie]    // ___security_cookie

5B88514C    push    ebx
5B88514D    mov     ss:[ebp+0xFFFFFFFC], eax
5B885150    mov     eax, ss:[ebp+0x10]
5B885153    push    esi
5B885154    mov     esi, ss:[ebp+0x8]
5B885157    xor     ebx, ebx
5B885159    cmp     b2 ds:[esi], b1 0x5C

5B88515D    push    edi
5B88515E    mov     edi, ss:[ebp+0xC]

5B885161    mov     ss:[ebp+0xFFFFFD4C], eax
5B885167    lea     eax, ss:[ebp+0xFFFFFD54]
5B88516D    mov     ss:[ebp+0xFFFFFD50], ebx
5B885173    jz      0x5B885189
```

Assembly

```
if ( !v5 )
 {
   _wcscpy(&Dest, L"\\\\");
   v6 = (wchar_t *)&v24;
 }
```

Pseudo Code

After Patched (netapi32.dll)

```
5B88513C    NetpManageIPCConnect(x, x, x, x)
5B88513C    mov     edi, edi                    // _NetpManageIPCConnect@16
5B88513E    push    ebp
5B88513F    mov     ebp, esp
5B885141    sub     esp, 0x2DC
5B885147    mov     eax, ds:[___security_cookie]    // ___security_cookie
5B88514C    and     ss:[ebp+0xFFFFFD50], b1 0x0
5B885153    and     ss:[ebp+0xFFFFFD4C], b1 0x0
5B88515A    push    ebx

5B88515B    mov     ebx, ss:[ebp+0x8]
5B88515E    push    esi
5B88515F    mov     ss:[ebp+0xFFFFFFFC], eax

5B885162    mov     eax, ss:[ebp+0x10]
5B885165    push    edi
5B885166    mov     edi, ss:[ebp+0xC]
5B885169    push    ebx                         // Str
5B88516A    mov     ss:[ebp+0xFFFFFD48], eax
5B885170    lea     esi, ss:[ebp+0xFFFFFD54]

5B885176    call    ds:[__imp__wcslen]          // __imp__wcslen
5B88517C    cmp     eax, 0x101
5B885181    pop     ecx
5B885182    jbe     0x5B885199
```

Assembly

```
if ( _wcslen(Str) > 0x101 ){
  NetpLogPrintHelper("NetpManageIPCConnect:
                     server name %ws too long
                     - error out\n", (char)Str);
  return 87;
 }
 if ( *Str != 92 ){
   _wcscpy(&Dest, L"\\\\");
   v4 = (wchar_t *)&v24;
 }
```

Security Check

Pseudo Code

Tools for Patch Diffing

NTT

Acquired by Google

- **Bindiff** **(Zynamics)**
  - https://www.zynamics.com/bindiff.html
- **Turbodiff** **(Core SECURITY)**
  - https://www.coresecurity.com/corelabs-research/open-source-tools/turbodiff
- **Diaphora** **(Joxean Koret)**
  - http://diaphora.re/

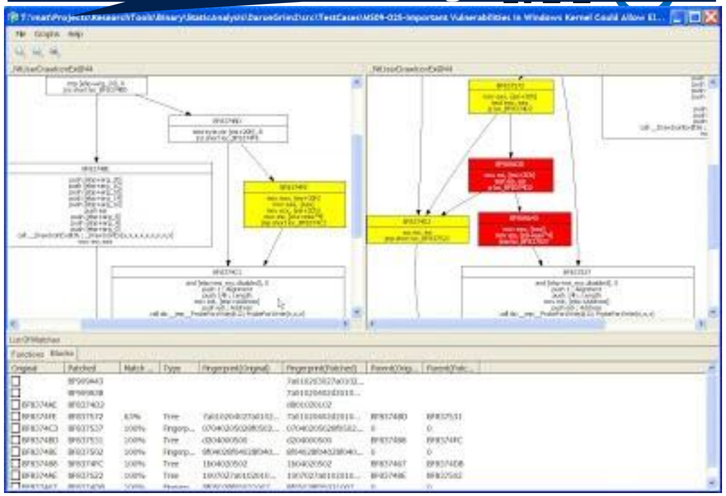
However, patch diffing is still a difficult task because it requires deep knowledge and experience

Semi-automated patch diffing

Previous Work

- **DarunGrim**
  - Shows the candidate functions that security fixes might have been applied
- **Approach**
  - Use heuristics pattern-matching rules to identify the candidate functions
    - **These patterns are manually defined by the developer**

DarunGrim (Jeongwook Oh)



Pattern	Type	Score
cmp	Opcode	+1
test	Opcode	+1
0xFFFFFFF	Immediate Value	+3
wcslen	Function Name	+2
strlen	Function Name	+2
StringCchCopyW	Function Name	+2
ULongLongToUlong	Function Name	+2

Machine learning techniques could be applied

PART 1

Extracting Security Fix Patterns Using Unsupervised Machine Learning Algorithm

Hypothesis

NTT

Similar Types of Vulnerabilities will be Fixed in a Similar Manner

Null Pointer Dereference

- ✓ Occurs when a program attempts to read or write to memory with a NULL pointer
- ✓ Check weather the pointer is NULL or not

CVE-2014-0198

```
@@ -672,10 +675,6 @@ static int do_ssl3_write(SSL *s,

+    if (wb->buf == NULL)
+    if (!ssl3_setup_write_buffer(s))
+    return -1;
+
     if (len == 0 && !create_empty_fragment)
     return 0;
```

CVE-2015-0288

```
@@ -92,8 +92,6 @@ X509_REQ *X509_to_X509_REQ(X509 *x,

     pktmp = X509_get_pubkey(x);
+    if (pktmp == NULL)
+        goto err;
     i = X509_REQ_set_pubkey(ret, pktmp);
    EVP_PKEY_free(pktmp);
```

Extract Fix Patterns Using Unsupervised Machine Learning Algorithm (Cluster Analysis)

Challenges

Challenge 1 : Optimization

1. Basic Block Reordering
2. Instruction Reordering
3. Operand Changes
4. Inline Expansion / Loop Unrolling

Challenge 2 : Other Fixes May Have Been Applied

Challenge 1 Basic Block Reordering

CVE-ID	Program1 (Before Patched)	Program2 (After Patched)
CVE-2015-1788	call 0x81031d0 <BN_copy> test eax,eax setnebl jmp 0x820337a <BN_GF2m_mod_inv+970> mov eax,DWORD PTR [esp+0x38] mov DWORD PTR [esp+0x4],edi mov DWORD PTR [esp],eax call 0x8102f90 <bn_expand2> jmp 0x8203149 <BN_GF2m_mod_inv+409> lea eax,[esp+0x58] mov edx,eax jmp 0x820339a <BN_GF2m_mod_inv+1002> shl ecx,0x5 mov DWORD PTR [esp+0x20],ecx jmp 0x8203307 <BN_GF2m_mod_inv+855> mov eax,DWORD PTR [esp+0x30] mov DWORD PTR [esp+0x4],eax mov eax,DWORD PTR [esp+0x3c] mov DWORD PTR [esp],eax call 0x8102f90 <bn_expand2> jmp 0x82031dd <BN_GF2m_mod_inv+557> mov eax,DWORD PTR [esp+0x30] mov DWORD PTR [esp+0x4],eax mov eax,DWORD PTR [esp+0x34] mov DWORD PTR [esp],eax call 0x8102f90 <bn_expand2> jmp 0x8203190 <BN_GF2m_mod_inv+480>	call 0x81031d0 <BN_copy> test eax,eax setnebl jmp 0x820338a <BN_GF2m_mod_inv+986> mov eax,DWORD PTR [esp+0x30] mov DWORD PTR [esp+0x4],eax mov eax,DWORD PTR [esp+0x3c] mov DWORD PTR [esp],eax call 0x8102f90 <bn_expand2> jmp 0x82031dd <BN_GF2m_mod_inv+557> mov eax,DWORD PTR [esp+0x30] mov DWORD PTR [esp+0x4],eax mov eax,DWORD PTR [esp+0x34] mov DWORD PTR [esp],eax call 0x8102f90 <bn_expand2> jmp 0x8203190 <BN_GF2m_mod_inv+480> mov eax,DWORD PTR [esp+0x38] mov DWORD PTR [esp+0x4],edi mov DWORD PTR [esp],eax call 0x8102f90 <bn_expand2> jmp 0x8203149 <BN_GF2m_mod_inv+409> lea eax,[esp+0x58] mov edx,eax jmp 0x82033aa <BN_GF2m_mod_inv+1018> shl ecx,0x5 mov DWORD PTR [esp+0x20],ecx jmp 0x8203317 <BN_GF2m_mod_inv+871>

Challenge 1 Instruction Reordering

NTT

CVE-ID	Program1	Program2

CVE-2015-1789	mov [esp+44h+var_4], eax
	push esi
	mov ebp, [esp+4Ch+arg_4]
	push esi
	mov esi, [esp+4Ch+arg_0]
	mov ecx, [esi]
	mov eax, [esi + 8]
	push edi
	mov edi, [esi+4]
	cmp edi, 17h
		mov [esp+44h+var_4], eax
		push esi
		mov ebp, [esp+4Ch+arg_4]
		push esi
		mov esi, [esp+4Ch+arg_0]
		push edi
		mov edi, [esi+4]
		mov ecx, [esi]
		mov eax, [esi + 8]
		cmp edi, 17h

```
text:11071B40        mov     eax, 44h
text:11071B45        call    __alloca_probe
text:11071B4A        mov     eax, ___security_cookie
text:11071B4F        xor     eax, esp
text:11071B51        mov     [esp+44h+var_4], eax
text:11071B55        push    ebp
text:11071B56        mov     ebp, [esp+48h+arg_4]
text:11071B5A        push    esi
text:11071B5B        mov     esi, [esp+4Ch+arg_0]
text:11071B5F        mov     ecx, [esi]
text:11071B61        mov     eax, [esi+8]
text:11071B64        push    edi
text:11071B65        mov     edi, [esi+4]
text:11071B68        cmp     edi, 17h
```

```
.text:1009D790       mov     eax, 44h
.text:1009D795       call    __alloca_probe
.text:1009D79A       mov     eax, ___security_cookie
.text:1009D79F       xor     eax, esp
.text:1009D7A1       mov     [esp+44h+var_4], eax
.text:1009D7A5       push    ebp
.text:1009D7A6       mov     ebp, [esp+48h+arg_4]
.text:1009D7AA       push    esi
.text:1009D7AB       mov     esi, [esp+4Ch+arg_0]
.text:1009D7AF       push    edi
.text:1009D7B0       mov     edi, [esi+4]
.text:1009D7B3       mov     ecx, [esi]
.text:1009D7B5       mov     eax, [esi+8]
.text:1009D7B8       cmp     edi, 17h
```

IDA Pro

Challenge 1 Operand Changes

NTT

<table>
<tr><th>CVE-ID</th><th>Program1</th><th>Program2</th></tr>
<tr><td rowspan="2">CVE-2008-5023</td><td>xor ebx, ebx
add rsp, 38h
mov eax, ebx
pop rbx
pop rbp
pop r12
pop r13
retn</td><td>xor r12d, r12d
add rsp, 38h
mov eax, r12d
pop rbx
pop rbp
pop r12
pop r13
retn</td></tr>
<tr><td colspan="2">Register is different (ebx -> r12d)</td></tr>
</table>

Challenge 1 Inline Expansion/Loop Unrolling

	Source code	Program1 (Before)	Program2 (After)

Inline Expansion	void my_print(int n){
	printf("%d", n);
	}
	int main(){
	int n = 1;
	my_print(n);
	return 0;
	}
		<main>:
		push ebp
		mov ebp,esp
		sub esp,0x4
		mov DWORD PTR [ebp-0x4],0x1
		push DWORD PTR [ebp-0x4]
		call 804840b <my_print>
		add esp,0x4
		mov eax,0x0
		leave
		ret
			<main>:
			push 0x1
			push 0x80484e0
			push 0x1
			call 8048310 <__printf_chk@plt>
			add esp,0xc
			xor eax,eax
			ret

Loop Unrolling	int main(){
	int i;
	for(i = 0; i < 3; i++){
	printf("HelloWorld!");
	}
	return 0;
	}
		<main>:
		push ebp
		mov ebp,esp
		sub esp,0x4
		mov DWORD PTR [ebp-0x4],0x0
		jmp 804842b <main+0x20>
		push 0x80484c0
		call 80482e0 <printf@plt>
		add esp,0x4
		add DWORD PTR [ebp-0x4],0x1
		cmp DWORD PTR [ebp-0x4],0x2
		jle 804841a <main+0xf>
			<main>:
			push 0x80484d0
			push 0x1
			call 8048310 <__printf_chk@plt>
			push 0x80484d0
			push 0x1
			call 8048310 <__printf_chk@plt>
			push 0x80484d0
			push 0x1
			call 8048310 <__printf_chk@plt>

Challenge 2

- **Other fixes may have been applied**
  1. **Bug Fixes**
  2. **Refactoring**
  3. **Feature Updates**

Experiment Overview

NTT

- **Dataset**
  - **Target Software: OpenSSL 1.0.1**
    - Collected **62** Security Fixes
- **Cluster Analysis**
  - Hierarchical Clustering Algorithm

GOAL

Extract security fix patterns which could be used to support the semi-automated patch diffing

Data Collection Method [1/3]

- OpenSSL 1.0.1 (**git** / 4675a56 (openssl 1.0.1 stable)

STEP 1

Analyze Commit Log & Release note*

CVE-ID	Type	Hash value
CVE-2012-2110	Before Patched	d36e0ee460f41d6b64015455c4f5414a319865c3
	After Patched	8d5505d099973a06781b7e0e5b65861859a7d994
CVE-2016-6304	Before Patched	151adf2e5cc23284a059e0f155505006a1c9fad9
	After Patched	2c0d295e26306e15a92eb23a84a1802005c1c137

STEP 2

Diff the source code & Identify the patched part (Function)

```
@@ -260,7 +265,11 @@ int BN_dec2bn(BIGNUM **bn, const char *a)
        a++;
    }

-   for (i = 0; isdigit((unsigned char)a[i]); i++) ;
+    for (i = 0; i <= (INT_MAX/4) && isdigit((unsigned char)a[i]); i++)
+        continue;
+
+    if (i > INT_MAX/4)
```

STEP 3

Compile/Disassemble before & after patched source code

***OpenSSL 1.0.1 Series Release Notes**
https://web.archive.org/web/20170208161711/https://www.openssl.org/news/openssl-1.0.1-notes.html

Data Collection Method [2/3]

NTT

Feature Extraction Method

STEP 1

Before Patched

push
mov
sub
jmp
…

After Patched

push
mov
sub
\+ mov
\+ mov
jmp
…

Increased Instructions

mov
mov
cmp
jne
test
je
push
jmp
lea
pop
…

Extract the increased instructions

STEP 2

Before Normalization	After Normalization
mov	trans
mov	trans
cmp	cmp
jne	jump
test	cmp
je	jump
push	stack
jmp	jump
lea	lea
pop	stack
…	…

Normalize the instructions

STEP 3

Normalized Instructions

trans
trans
cmp
jump
cmp
jump
stack
jump
lea
stack
…

# of occurrences (each instruction)

trans:	2
cmp:	2
jump:	3
stack:	2
lea:	1
…	

Count the number of occurrences of each normalized instruction

Data Collection Method [3/3]

NTT

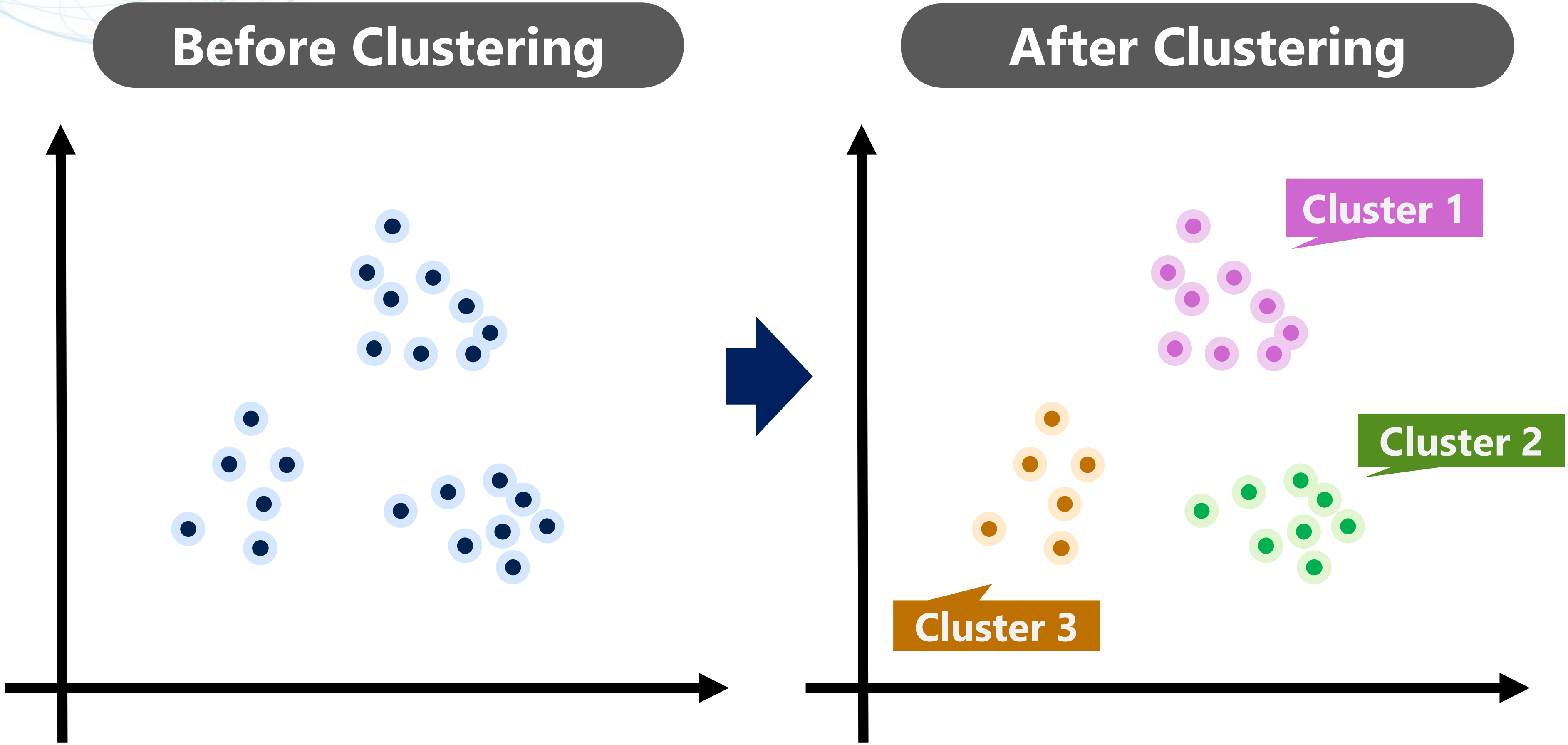
Opcode (Instruction) Normalization

- Summarized and expressed the instructions that fall into similar categories by one (normalized) instruction
  - e.g.) Branch instruction such as jns,jle,jne,jge are normalized as “jump”
- Normalized the instructions which appeared in the security fixes (Function)

Normalized Instruction	Type of Instruction	Target Instuctions
jump	Branch	jns, jle, jne, jge, jae, jmp, js, jl, je, jg, ja, jb, jbe
trans	Data Transfer	movxz, mov, movsx, xchg, cdq
ctrans	Conditional Data Transfer	cmovge, cmovae, cmovs, cmovns, cmove, cmovne
stack	Stack Manipulation	push,pop
logical	Logical Operation	and, xor, or, not
arith	Arithmetic Operation	sub, add, imul, neg, adc
nop	No Operation	nop
bop	Bit/Byte Operation	bt, setne, sete
shift	Shift Operation	shr, shl, sar
func	Function Operation	call, ret
str	String Operation	repz *
cmp	Comparison	test, cmp
lea	Address Computation	lea

Cluster Analysis [1/2]

Divides data into groups that are meaningful/useful



Cluster Analysis [2/2]

Divides data into groups that are meaningful/useful

- **Hierarchical Clustering Algorithm**
  - Produce a classification in which small clusters of very similar data points are nested within larger clusters of less closely-related data points*
    - e.g.) Agglomerative Hierarchical Clustering
- **Non-Hierarchical Clustering Algorithm**
  - Generates a classification by partitioning dataset*
    - e.g.) K-means Clustering

***Hierarchical and non-Hierarchical Clustering**
https://www.daylight.com/meetings/mug96/barnard/E-MUG95.html

Cluster Analysis [2/2]

NTT

Divides data into groups that are meaningful/useful

- **Hierarchical Clustering Algorithm**
  - Produce a classification in which small clusters of very similar data points are nested within larger clusters of less closely-related data points*
    - e.g.) Agglomerative Hierarchical Clustering

*Cluster **A***
*Cluster **B***

Ward's Method

$$d(A,B) = E(A \cup B) - E(P) - E(Q)$$

&

Euclidean Distance

$$\|a-b\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$$

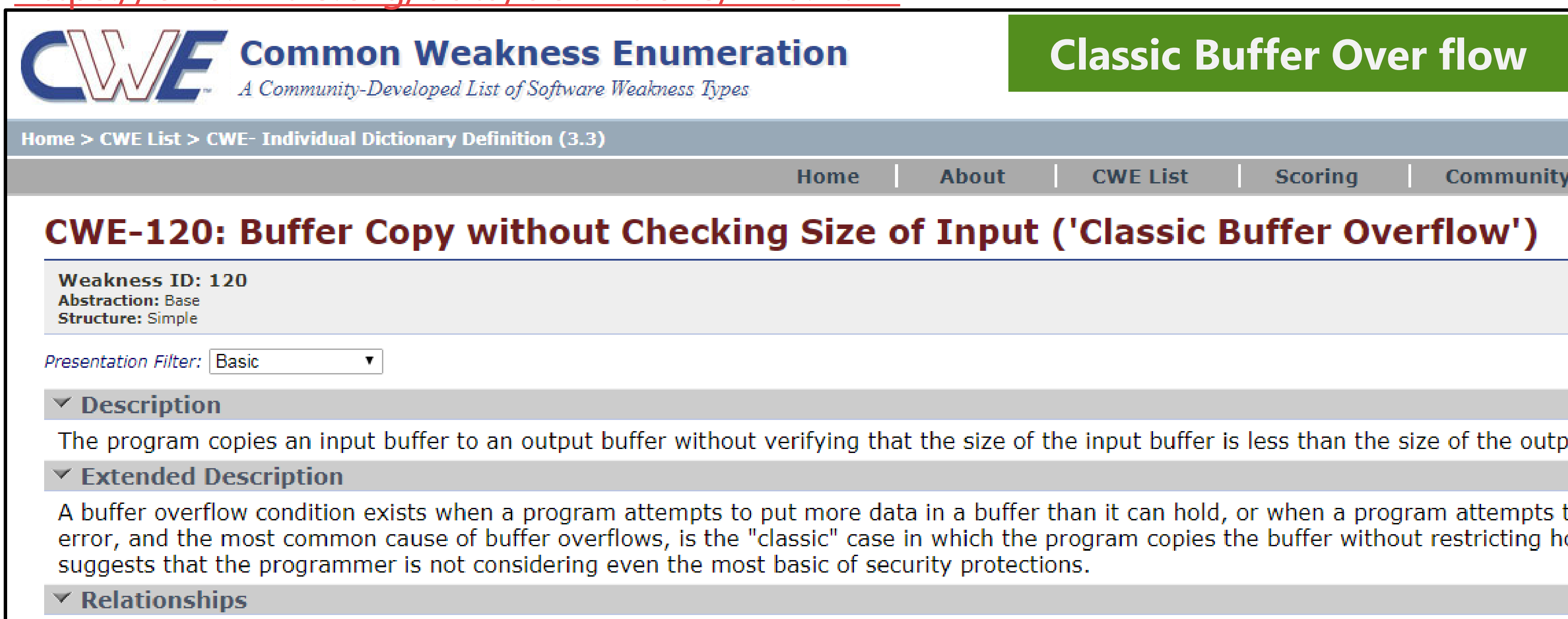
***Hierarchical and non-Hierarchical Clustering**
https://www.daylight.com/meetings/mug96/barnard/E-MUG95.html

CWE [1/2]

- **CWE (Common Weakness Enumeration)**
  - **List of Software Weakness Types**
    - Gives a unique identifier (CWE-ID) to each types
    - e.g, CWE-120:Buffer Copy without Checking Size of Input
    - Latest Version:3.4 / Total 808 Weaknesses.

Used as a label for each data

https://cwe.mitre.org/data/definitions/120.html

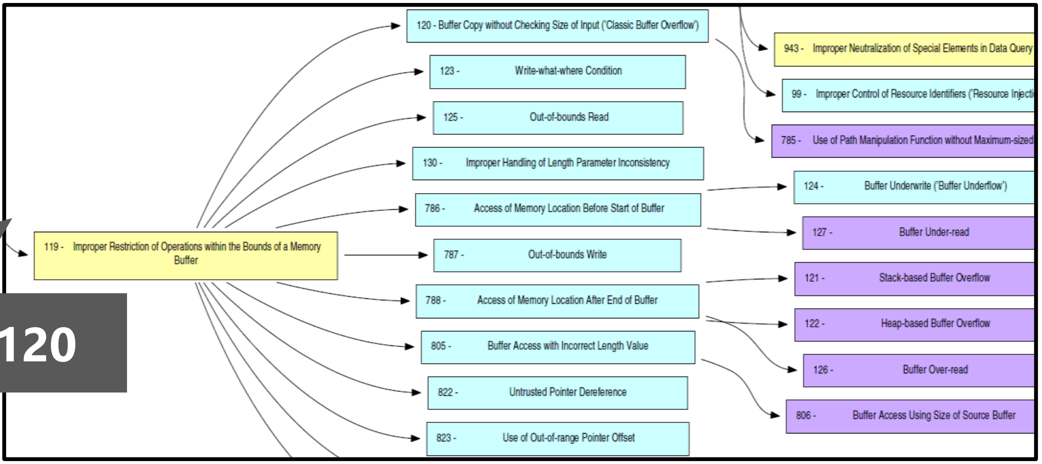


CWE [2/2]

CWE organizes a wide variety of weakness types in a <u>hierarchical structure</u>

- The weakness types at higher levels in the structure gives a more abstract and broader concept*

[Parent] CWE-119 -> [Child] CWE-120



- **Structure Types**
  - Development Concepts
  - Research Concepts
  - Architectural Concepts

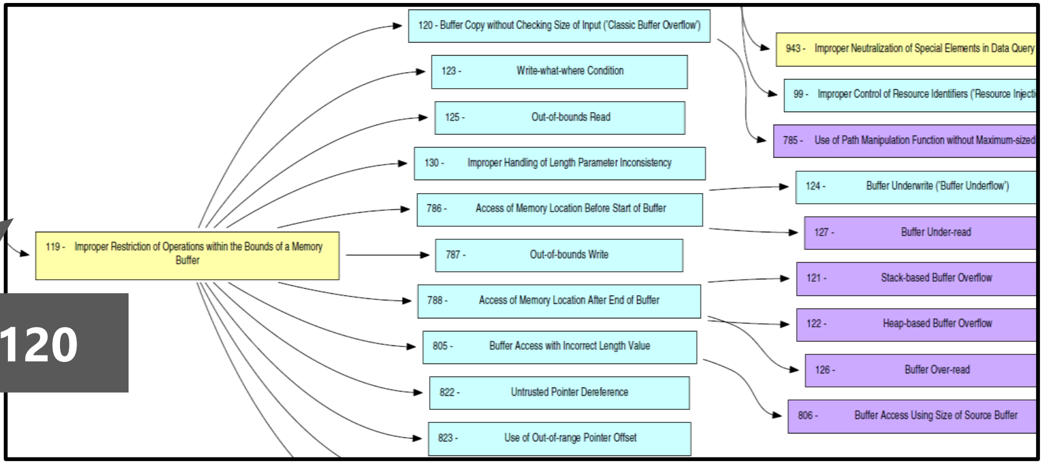
*CWE Overview, IPA
https://www.ipa.go.jp/security/english/vuln/CWE_en.html

CWE [2/2]

NTT

CWE organizes a wide variety of weakness types in a <u>hierarchical structure</u>

➢ The weakness types at higher levels in the structure gives a more abstract and broader concept*



[Parent] CWE-119 -> [Child] CWE-120

➢ **<u>Structure Types</u>**

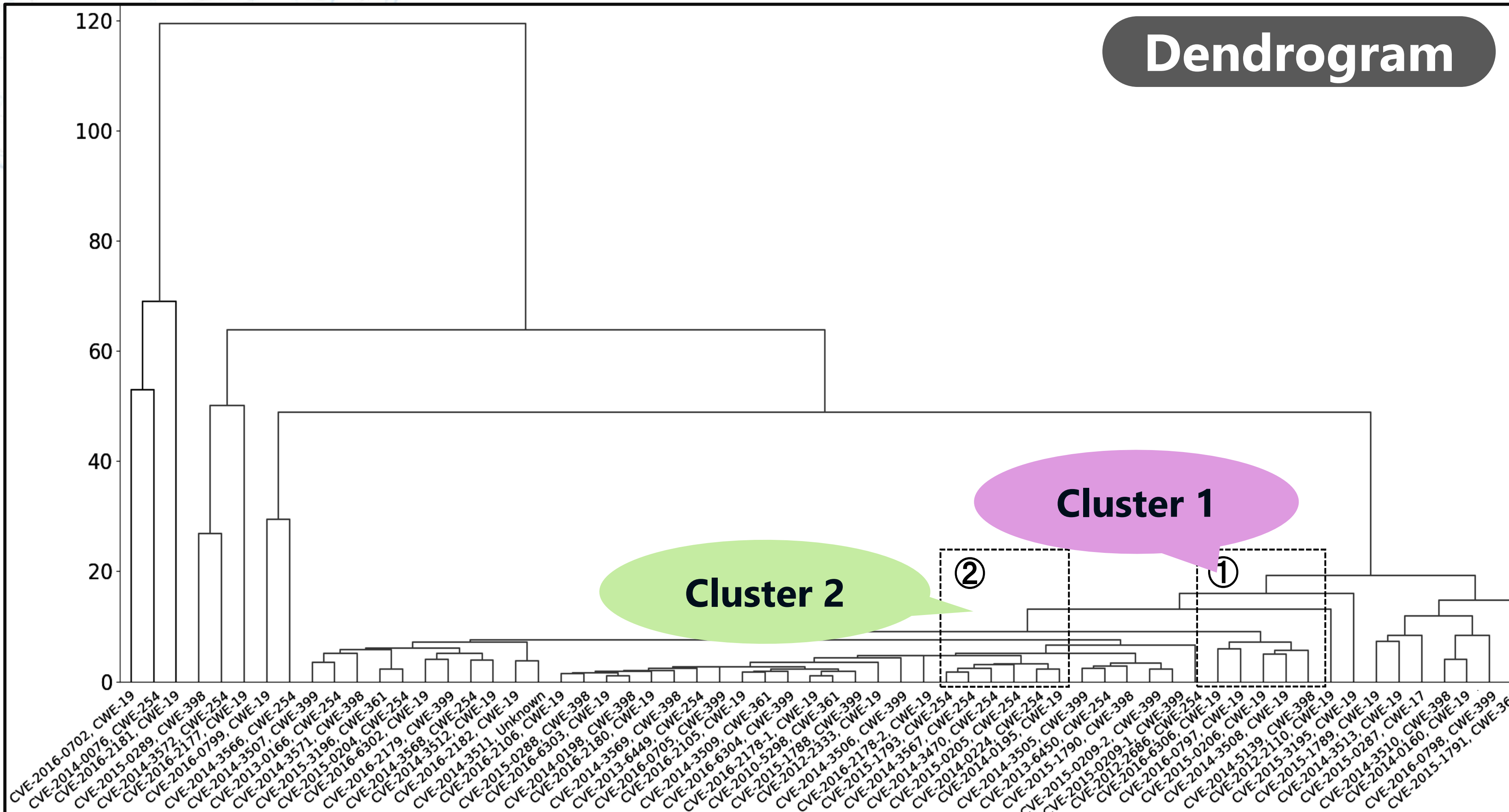
- **<u>Development Concepts</u>**
- Research Concepts
- Architectural Concepts

Used these root CWE-IDs as labels

CWE-ID	Description
CWE-17	Code
CWE-19	Data Processing Error
CWE-254	Security Features
CWE-361	Time and State
CWE-398	Indicator of Poor Code Quality
CWE-399	Resource Management Errors

***CWE Overview, IPA**
https://www.ipa.go.jp/security/english/vuln/CWE_en.html

Result



CVE-ID + Label (CWE-ID)

Details of the Cluster 1

Most of the labels are CWE-19 (Data Processing Error)

CWE-ID	CVE-ID	Feature Vectors (Normalized Instruction : # of Occurrences)
CWE-19	**CVE-2016-6306**	jump: **9**, trans: **7**, cmp: **6**, lea: **4**, arith: **3**, func: **1**
CWE-19	**CVE-2016-0797**	jump: **7**, lea: **4**, cmp: **4**, trans: **2**, arith: **2**
CWE-19	**CVE-2015-0206**	jump: **7**, trans: **5**, func: **4**, stack: **4**, cmp: **4**, arith: **2**, lea: **1**
CWE-19	**CVE-2014-3508**	jump: **9**, trans: **5**, cmp: **5**, stack: **4**, nop: **3**, lea: **2**, arith: **2**, bop: **1**, func: **1**
CWE-398	**CVE-2014-5139**	jump: **7**, stack: **4**, cmp: **3**, logical: **2**, trans: **2**, nop: **2**, func': **1**

Summary

- ✓ Most of the vulnerabilities in this cluster are related to the memory or value manipulation error, which was not initially expected by developers
  - e.g.) Out-of-bounds read, info-leak, integer overflow)
- ✓ A certain number of Comparison/Branch/Arithmetic Operation instructions exist

CVE-2016-0797

Integer Overflow Vulnerability

Patched Part of CVE-2016-0797

```
@@ -190,11 +189,7 @@ int BN_hex2bn(BIGNUM **bn,
    }
+    for (i = 0; i <= (INT_MAX/4) &&
           isxdigit((unsigned char)a[i]); i++)
+          continue;
+
+    if (i > INT_MAX/4)
+          goto err;
 -    for (i = 0; isxdigit((unsigned char)a[i]); i++) ;
```

Will be used in bn_expand(ret, i*4)

Added a check to confirm the integer value is under the expected upper limit

Details of the Cluster 2

Most of the labels are CWE-254 (Security Features)

CWE-ID	CVE-ID	Feature Vectors: (Normalized Instruction : # of Occurrences)
CWE-254	**CVE-2015-1793**	trans: **4**, jump: **3**, ctrans: **1**, nop: **1**, func: **1**
CWE-254	**CVE-2014-3567**	trans: **4**, jump: **2**, func: **1**
CWE-254	**CVE-2014-3470**	trans: **4**, jump: **2**, nop: **2**, lea: **1**, func: **1**, cmp: **1**
CWE-254	**CVE-2015-0205**	trans: **5**, func: **1**
CWE-254	**CVE-2014-0224**	trans: **5**, jump: **3**, logical: **2**, cmp: **1**
CWE-19	**CVE-2014-0195**	trans: **6**, jump: **3**, cmp: **1**

Summary

- ✓ Most of the security fixes for the vulnerabilities in this cluster contain some sort of error handling function
- ✓ A certain number of Data Transfer/Branch/Function related instructions exist

CVE-2014-3470

NTT

NULL Pointer Dereference

Patched Part of CVE-2014-3470

```
@@ -2512,13 +2512,6 @@ int ssl3_send_client_key_exchange
     int field_size = 0;

+     if (s->session->sess_cert == NULL)
+         {
+         ssl3_send_alert(s,SSL3_AL_FATAL,
                     SSL_AD_UNEXPECTED_MESSAGE);
+          SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,
                     SSL_R_UNEXPECTED_MESSAGE);
+          goto err;
+          }
```

Added two error handling function + exit the function

Discussions

- **Why Only Two Clusters?**
  - Some vulnerabilities are found in multiple functions
    - Similar functions contain same vulnerability

We count the number of occurrences of normalized instructions

- **How to Improve**
  - Include other features such as function name?
  - Collect more security fixes
    - Use vulnerability corpus generation tools? (e.g, LAVA)
  - Use other machine learning techniques
- **For Semi-Automated Patch-Diffing**
  - Calculate the similarity between the extracted security fix patterns (instructions) and the difference (increased instructions) found by the patch diffing

Centroid

*LAVA: Large-scale Automated Vulnerability Addition
https://www.andreamambretti.com/files/papers/oakland2016_lava.pdf

PART 2

Classifying Security Fixes and Other Fixes

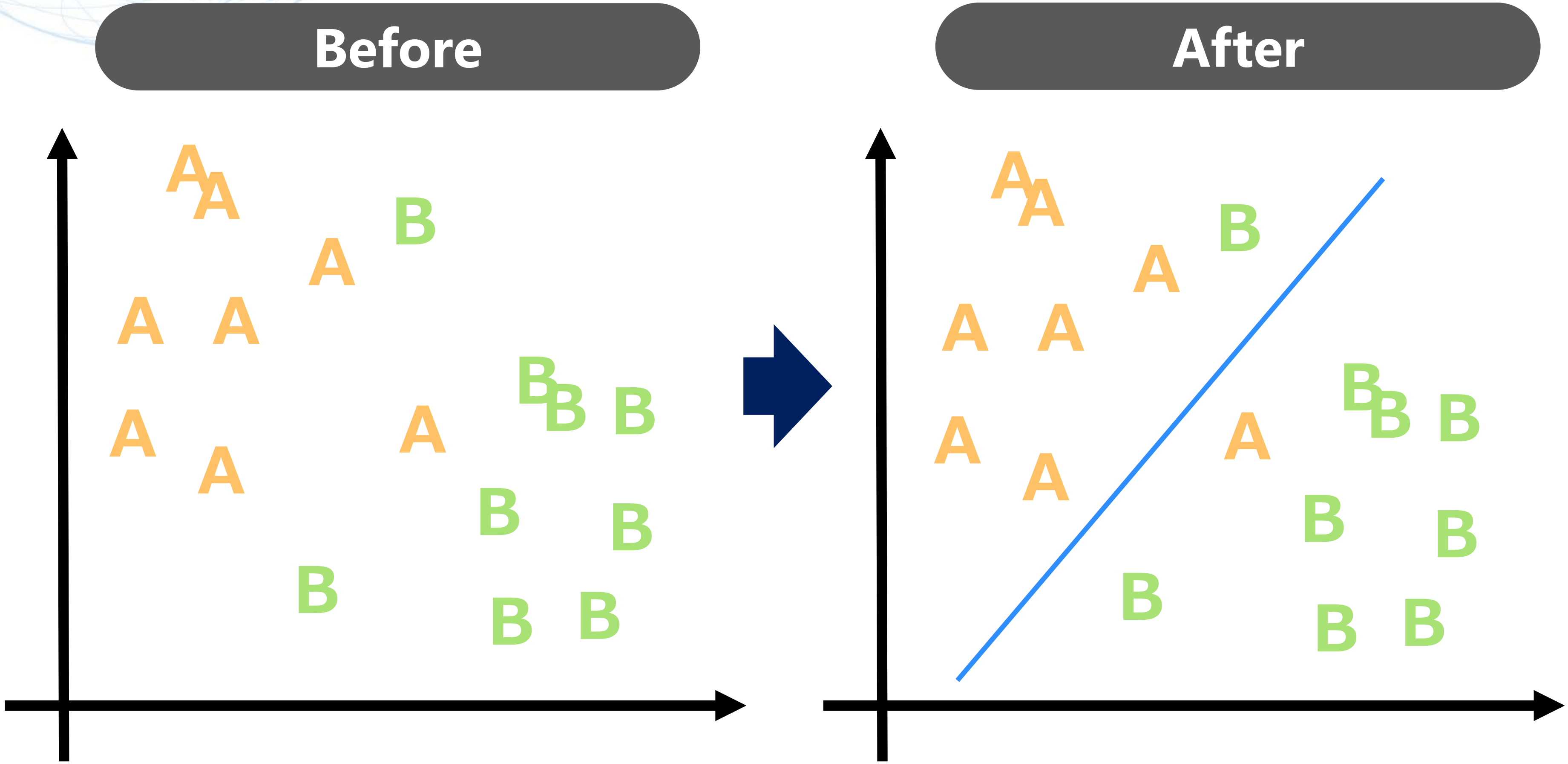
Classifying Security Fixes and Other Fixes

- **Dataset**
  - OpenSSL 1.0.1 (**62** Security Fixes / **377** Other fixes)
- **Classification Method**
  - Supervised Linear Classifier
    - Soft Margin Support Vector Machine (SVM)
    - Kernel: RBF (C=10, $\gamma = 0.001$)
- **Experiment**
  - Used **62** Security Fixes and **62** Other fixes (Random sampling)
  - Conducted 10-fold Cross-Validation **3** times
    - Perform random sampling for each cross-validation
- **Environment**
  - OS: Ubuntu 14.04, Compiler: gcc 5.4.0

Support Vector Machine (SVM)

NTT

Method used for classification (+regression) tasks



Result

	Type of Fix	Dataset 1	Dataset 2	Dataset 3	Average
Accuracy		0.62	0.54	0.54	**0.56**
Precision	**Security Fix**	0.70	0.57	0.57	**0.61**
	Other	0.59	0.54	0.54	**0.55**
Recall	**Security Fix**	0.42	0.49	0.42	**0.41**
	Other	0.82	0.71	0.68	**0.73**
F-Score	**Security Fix**	0.53	0.46	0.44	**0.47**
	Other	0.68	0.61	0.63	**0.64**

Summary of Result

NTT

- **Summary**
  - ❖ **Overall Accuracy : 56% (average)**
  - ❖ **Security Fixes: Precision 61% / Recall 41% (average)**
  - ❖ **Other Fixes: Precision 55% / Recall 73% (average)**

- **Discussions**
  - ❖ **Use other metrics?** (e.g., Cyclomatic complexity)

Glossary

Accuracy	Ratio of the number of correctly labeled fixes to the number of all fixes in the dataset
Precision	Ratio of the number of correctly labeled security fixes to the number of all fixes labeled as "security fix" by the program
Recall	Ratio of the number of correctly labeled security fixes to the number of all security fixes in the dataset
F-Score	Harmonic mean of the Precision and Recall

Summary & Conclusion

- ☑ Patch diffing is still a difficult task because it requires a deep knowledge and experience

- ☑ Extracted security fix patterns which could be used to support the semi-automated patch diffing

- ☑ Conducted an experiment to see if it is possible to distinguish between security fixes and other fixes

Provided insights for future research related to the semi-automated patch diffing

Appendix [1/3]

- **Original Paper**

  Asuka Nakajima, Ren Kimura, Yuhei Kawakoya, Makoto Iwamura, Takeo Hariu, "An Investigation of Method to Assist Identification of Patched Part of the Vulnerable Software Based on Patch Diffing" Multimedia, Distributed, Cooperative, and Mobile Symposium, June 2017, Japan

パッチ差分に基づく脆弱性修正箇所の特定支援技術の検討

中島明日香[1] 木村廉[2] 川古谷祐平[1] 岩村誠[1] 針生剛男[1]

概要：近年，既知の脆弱性箇所の特徴に基づいた脆弱性発見手法が注目されており，その手法を利用して少なくはない脆弱性が発見されている．しかし既存手法では，利用する脆弱性箇所の特徴を，主にソースコードから収集しており，実行ファイルに含まれる脆弱性箇所は，その解析・収集の難しさ故に対象外であった．そのため必然的に，プロプライエタリなソフトウェアの脆弱性の特徴は利用できず，発見可能な脆弱性も限られていた．脆弱性修正前後の実行ファイルを比較(パッチ差分解析)して，脆弱性箇所の特定を支援する技術は存在するものの，既存手法では特定可能な脆弱性箇所が限られており，かつ実行ファイル中で脆弱性修正以外の修正が行われていた場合，その判別が難しかった．

そこで本論文では，脆弱性箇所収集を最終目的として，脆弱性修正を含めたプログラム修正が存在する実行ファイル中から，網羅的に脆弱性修正箇所の特定を支援する技術の検討を行った．具体的には，網羅的な脆弱性修正箇所発見を目指して，まずはオープンソースソフトウェアから収集した，既知の脆弱性修正箇所をクラスタ分析手法で分類し，共通する特徴の有無を調べた．そして次に，線形分類器を利用して，収集した脆弱性修正箇所とその他の修正が識別可能か否かを検証した．その結果，脆弱性修正箇所の分類では部分的に共通する特徴は見られたが，脆弱性修正とその他の修正の識別は正解率 56%と高くはなく，識別に利用する入力データの改善が課題として浮かびあがった．

An Investigation of Method to Assist Identification of Patched Part of the Vulnerable Software Based on Patch Diffing

ASUKA NAKAJIMA[1] REN KIMURA[2] YUHEI KAWAKOYA[1]
MAKOTO IWAMURA[1] TAKEO HARIU[1]

1. 研究の背景と動機

サイバー攻撃の根本的な原因の一つとしてソフトウェアの脆弱性がある．脆弱性とは，ソフトウェアのバグの一種で，第三者から悪用可能なバグのことを指し，マルウェア感染をはじめとした各種サイバー攻撃に用いられる．この脆弱性に起因するサイバー攻撃の被害を防ぐためにも，開発段階で脆弱性を作りこまない努力が各ソフトウェアベンダによってなされているが，それも完璧では無く，脆弱性が残存したまま公開されてしまう場合が多い．そのため，それらの脆弱性をいち早く見つけ修正するためにも，ソフトウェア中から脆弱性を発見するための技術が大事になってくる．

ソフトウェアから脆弱性を発見する技術には，ファジング[1]やシンボリック実行[2]などが挙げられるが，近年はその中でも，コードクローンの脆弱性を対象とした脆弱性発見技術が着目されている[3][4][5][6]．このコードクローンの脆弱性とは，ソフトウェア中に存在した脆弱性部分がソースコードの流用などが原因で，他所にも複製されてしま

このコードクローンの脆弱性発見技術は，既知の脆弱性箇所を基に，検査対象のソフトウェア中に同様の脆弱性が存在するか否かを検査するものである．そのため，事前に既知の脆弱性箇所を把握することが必須ではあるが，修正された脆弱性箇所の情報は脆弱性情報データベース[7]やベンダー公式サイトに掲載されてない場合があり，知り得たい場合には各自で調査する他なかった．

脆弱性箇所の特定方法としては，ソフトウェアのソースコードが入手可能な場合は，修正前後のソースコードの差分(パッチ差分)に着目すれば，ある程度容易に修正箇所を特定できる．しかし，ソースコードが公開されていない実行ファイルのみが入手可能なソフトウェアの場合，脆弱性箇所を特定するのは，逆アセンブルされたプログラムコードを読む能力に加え，コンパイラやリンカの最適化がアセンブリに与える影響などの知見が必要になり，容易ではない．その上，開発者が脆弱性修正以外の修正も加えていた場合，脆弱性修正とそれ以外の部分を区別する必要もあり，それには手間と時間を要した．

Download URL
https://ipsj.ixsq.nii.ac.jp/ej/?action=repository_uri&item_id=190132&file_id=1&file_no=1

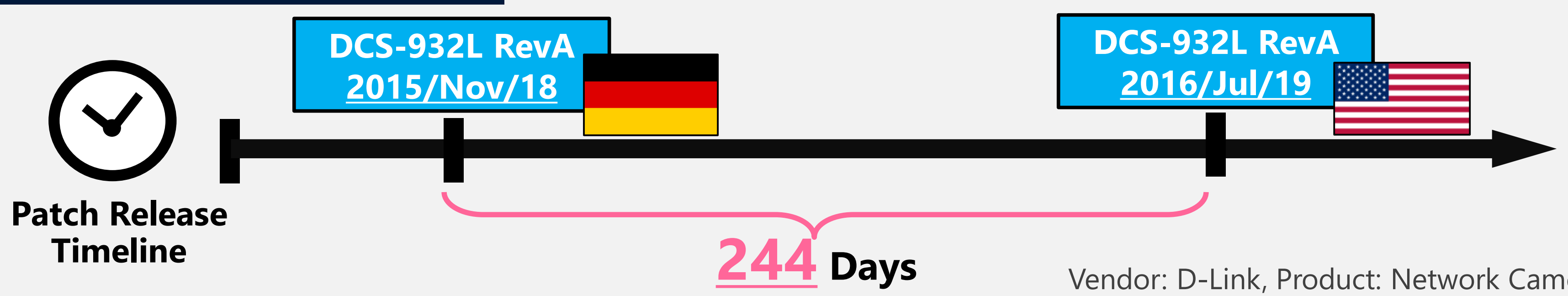
Appendix [2/3]

● Other Research (1)

- ***Asuka Nakajima***, Takuya Watanabe, Eitaro Shioji, Mitsuaki Akiyama, and Maverick Woo **"A Pilot Study on Consumer IoT Device Vulnerability Disclosure and Patch Release in Japan and the United States"** Proceedings of the 14th ACM ASIA Conference on Information, Computer and Communications Security **ASIA CCS 2019**.
  - [PDF] https://www.cylab.cmu.edu/_files/pdfs/tech_reports/CMUCyLab19001.pdf

1-Day Risk: Unsynchronized Patch Release (Geographical Arbitrage)

Example: CVE-2017-7852



Vendor: D-Link, Product: Network Camera

Revealed Significant 1-Day Risk Related to IoT

Appendix [3/3]

- **Other Activities (Female InfoSec Community)**
  - ***CTF for GIRLS: http://girls.seccon.jp (Twitter:@ctf4g)***
  - ***Asuka Nakajima***, Suhee Kang, Hazel Yen, “Women in Security: Building a Female InfoSec Community in Korea, Japan, and Taiwan”, BlackHatUSA 2019

Women-Only CTF Workshop

Talk about Asian Female InfoSec Community

Questions?

Asuka Nakajima
E-mail: asuka.nakajima.db@hco.ntt.co.jp
Twitter: @AsuNa_jp