

코드 리뷰 보고서

리뷰번호	001	작성자	곽채원	작성날짜	2023.05.31
관련규칙	12. Do not use insecure or weak cryptographic algorithms				
중요도	상				
	Before				
대상코드	<pre>private static String signAlgorithm = "SHA256withRSA"; private static String cipherAlgorithm = "DES"; private static String keyAlgorithm = "RSA";</pre>				
	After				
보안약점/ 보안취약점 이유 및 해결법	<pre>private static String signAlgorithm = "SHA256withRSA"; private static String cipherAlgorithm = "AES"; private static String keyAlgorithm = "RSA";</pre>				
	<p>Before 코드에서 서명과 평문을 암호화하기 위한 대칭 암호로 DES 알고리즘을 활용하였다. 그러나 DES는 암호 강도가 낮다고 밝혀져 사용할 시 보안에 취약할 수 있다.</p> <p>이를 해결하기 위해 대칭 암호 알고리즘으로 AES를 활용하도록 코드를 수정해주었다.</p>				

코드 리뷰 보고서

리뷰번호	002	작성자	곽채원	작성날짜	2023.05.31
관련규칙	2. Do not store unencrypted sensitive information on the client side.				
중요도	상				
	Before				
대상코드	<pre style="border: 1px solid black; padding: 10px;"> try (FileInputStream fis = new FileInputStream(ename + "_key.txt")) { byte[] encrypted = fis.readAllBytes(); byte[] decrypted = c.doFinal(encrypted); secretKey = new SecretKeySpec(decrypted, "AES"); } catch (FileNotFoundException e) { e.printStackTrace(); } </pre>				
	After				
보안약점/ 보안취약점 이유 및 해결법	<pre style="border: 1px solid black; padding: 10px;"> try (FileInputStream fis = new FileInputStream(ename + "_key.txt")) { byte[] encrypted = fis.readAllBytes(); byte[] decrypted = c.doFinal(encrypted); secretKey = new SecretKeySpec(decrypted, "AES"); for (int i = 0; i < decrypted.length; i++) { decrypted[i] = 0; } } catch (FileNotFoundException e) { e.printStackTrace(); } </pre>				
	<p>Before 코드에서 decrypted 변수는 복호화한 키에 대한 정보를 그대로 담고 있다. 긴 코드 내에서 계속이 decrypted 변수가 민감한 정보를 담은 채 떠돌게 된다면 공격자가 악용할 여지가 다분하다. 따라서 사용 후에 민감 정보를 지우기 위한 별도의 처리 과정이 필요하다.</p> <p>이를 해결하고자 사용을 완료한 decrypted(바이트 배열) 안의 바이트 값들을 0으로 바꿔주면서 민감 정보에 대한 보안성을 강화하였다.</p>				

코드 리뷰 보고서

리뷰번호	003	작성자	곽채원	작성날짜	2023.05.31
관련규칙	33. Prefer user-defined exceptions over more general exception types.				
중요도	상				
대상코드	Before				
	<pre>\$ dest -vrify rprivate.key sprivate.key 20200941 Exception in thread "main" java.lang.ClassCastException: class sun.se at Project/Project.sign_verify.vrfy(sign verify.java:154) at Project/Project.dest.main(dest.java:105) Signature sig = Signature.getInstance(signAlgorithm); sig.initVerify((PublicKey) kmanager.readKey(Pkey));</pre>				
대상코드	After				
	<pre>public class StrangeObjectException extends Exception { private static final long serialVersionUID = 1L; public StrangeObjectException() { super("Fatal Error: Strange object came in. Use valid key and signature please."); } if (kmanager.readKey(Pkey) instanceof PublicKey) { sig.initVerify((PublicKey) kmanager.readKey(Pkey)); } else { throw new StrangeObjectException(); } if (obj instanceof Key) { Key key = (Key) obj; return key; } else { throw new StrangeObjectException(); } }</pre>				
보안약점/ 보안취약점 이유 및 해결법	<pre>\$ dest -vrify rprivate.key sprivate.key 20200941 Manager.StrangeObjectException: Fatal Error: Strange object came in. Use valid key and signature please. at Project/Project.sign_verify.vrfy(sign verify.java:166) at Project/Project.dest.main(dest.java:105) Verification result : false \$ </pre>				
	<p>만약 검증 과정에서 비밀키가 아닌 공개키를 매개변수로 줬을 경우 그냥 검증이 실패해야 한다. 그러나 공개키가 비밀키로 Class Casting이 되지 않기 때문에 위와 같이 catch로 미처 잡지 못한 ClassCastException이 발생하게 된다. 이런 예외 처리가 뜨면 (1) 공격자가 프로그램 내부 동작을 유추할 수 있으며 (2) 프로그램이 예기치못하게 종료된다는 보안 약점을 갖는다.</p> <p>이를 해결하기 위해 ClassCastException이나 key 객체 외에도 아예 생뚱맞은 객체 정보를 담은 파일이 입력으로 들어왔을 시를 뚫어서 한 번에 처리하는 사용자정의 예외를 만들어 명시적으로 valid한 key를 입력하지 않았기 때문에 발생한 오류라는 것을 보여준다.</p>				

코드 리뷰 보고서

리뷰번호	004	작성자	곽채원	작성날짜	2023.05.31
관련규칙	41. Return an empty array or collection instead of a null value for methods that return an array or collection.				
중요도	상				
	Before				
대상코드	<pre> public byte[] readCipher_sig(String fname, Cipher c) { try (FileInputStream fis = new FileInputStream(fname); CipherInputStream cis = new CipherInputStream(fis,c)) { return cis.readAllBytes(); } catch (FileNotFoundException e) { e.printStackTrace(); } catch (IOException e) { e.printStackTrace(); } return null; } </pre>				
	After				
보안약점/ 보안취약점 이유 및 해결법	<pre> public byte[] readCipher_sig(String fname, Cipher c) { try (FileInputStream fis = new FileInputStream(fname); CipherInputStream cis = new CipherInputStream(fis,c)) { return cis.readAllBytes(); } catch (FileNotFoundException e) { e.printStackTrace(); } catch (IOException e) { e.printStackTrace(); } byte[] empty = {}; return empty; } </pre>				
	<p>Before 코드와 같이 배열을 반환하는 메소드가 null값을 인叭드 오류지표로 사용할 경우 메인에서 null을 제대로 처리하지 않고 배열의 요소를 탐색하거나 배열의 크기를 얻기 위해 .length를 사용하는 순간 널 포인터 오류가 나게 된다.</p> <p>이러한 문제를 해결하기 위해서 빈 바이트 배열을 생성하여 반환해주게끔 코드를 수정하였다.</p>				

코드 리뷰 보고서

리뷰번호	005	작성자	곽채원	작성날짜	2023.06.01
관련규칙	54. Use braces for the body of an if, for, or while statement.				
중요도	致命				
	Before				
대상코드	<pre> if (answer.contains("dest -kp")) { if (tokens.length == 4) sign_verify.generateKeyPair(tokens[2], tokens[3]); else System.out.println("Usage : dest -kp [filename1] [filename2]"); } else { if (tokens.length == 3) sign_verify.generateKey(tokens[2]); else { System.out.println("Usage : dest -k [filename]"); System.out.println("For more details... try \"dest -h\" for help."); } } </pre>				
	After				
보안약점/ 보안취약점 이유 및 해결법	<pre> if (answer.contains("dest -kp")) { if (tokens.length == 4) { sign_verify.generateKeyPair(tokens[2], tokens[3]); } else { System.out.println("Usage : dest -kp [filename1] [filename2]"); System.out.println("For more details... try \"dest -h\" for help."); } } } else { if (tokens.length == 3) { sign_verify.generateKey(tokens[2]); } else { System.out.println("Usage : dest -k [filename]"); System.out.println("For more details... try \"dest -h\" for help."); } } </pre>				
	<p>Before코드의 첫 번째 else문에서 “Usage : dest -kp [filename1] [filename2]”에 대한 출력에 덧붙여 한 줄을 더 출력하고 싶을 시 중괄호 때문에 코드가 예기치 못한 동작을 할 수 있다.</p> <p>이를 해결하고자 모든 if문, else문에 중괄호를 제대로 붙여줌으로써 코드의 가독성 및 유지보수성을 높여주었다.</p>				

코드 리뷰 보고서

리뷰번호	006	작성자	곽채원	작성날짜	2023.06.01
관련규칙	7. Prevent code injection.				
중요도	상				
대상코드	<p style="text-align: center;">Before</p> <pre>public Key readKey(String fname) { try (FileInputStream fis = new FileInputStream(fname)) { try (ObjectInputStream ois = new ObjectInputStream(fis)) { Object obj = ois.readObject(); if (obj instanceof Key) { Key key = (Key) obj; return key; } } } }</pre> <p style="text-align: center;">After</p> <pre>\$ dest -sign plain.txt sprivate.key secret.key rpublic.key --out 0941 =====File Size Error (Maximum is 1GB)===== Generating digital envelope is failed.</pre> <pre>public Key readKey(String fname) throws StrangeFileNotFoundException { KeyManager.isValidFile(fname); try (FileInputStream fis = new FileInputStream(fname)) { try (ObjectInputStream ois = new ObjectInputStream(fis)) { Object obj = ois.readObject(); if (obj instanceof Key) { Key key = (Key) obj; return key; } } } } // 이상한 파일인지 검사하는 메소드 (예외 처리) public static void isValidFile(String fname) throws StrangeFileNotFoundException { // DDos 공격을 막기 위한 파일 길이 제한 File file = new File(fname); long fileSize = file.length(); long maxSize = ONE_GB; // 1GB if (fileSize > maxSize) { System.out.println("=====File Size Error (Maximum is 1GB)====="); throw new StrangeFileNotFoundException(); } // 파일 확장자에 대한 whitelist 작성 String[] allowedEXEs = { "jpg", "png", "key", "bin", "txt", "jpeg", "pdf", "csv", "xlsx", "hwp", "pptx" }; String extension = fname.substring(fname.lastIndexOf(".") + 1); for (String allowedEXE : allowedEXEs) { if (extension.equalsIgnoreCase(allowedEXE)) { return; } } throw new StrangeFileNotFoundException(); }</pre>				
보안약점/ 보안취약점 이유 및 해결법	<p>사용자가 입력한 파일을 열어보는 프로그램이므로 파일에 대한 엄격한 검증이 필요하다. Before 코드는 파일에 대한 아무런 검사 없이 파일 이름으로 FileInputStream을 사용하여 파일에 접근한다. 이는 공격자에 의해 쉽게 공격받을 수 있기 때문에 보안에 취약하다.</p> <p>이를 해결하기 위해서 두 가지 검증을 진행하였다. 하나는 Dos 공격을 막기 위해 파일 크기를 제한하는 것, 다른 하나는 파일의 확장자에 대한 화이트리스팅을 하여 이상한 파일이 입력으로 들어오는 것을 막는 것이다. 이 두 가지 조건을 통과한 파일만을 코드에서 사용하게끔 함수 호출 문을 꼼꼼히 배치하였다. 그리고 공격자가 화이트리스트에 대한 정보를 얻을 수 없게끔 main과 다른 패키지에 해당 함수를 작성하여 민감 정보를 숨겼다.</p>				

코드 리뷰 보고서

코드 리뷰 보고서

리뷰번호	008	작성자	곽채원	작성날짜	2023.06.02
관련규칙	39. Use meaningful symbolic constants to represent literal values in program logic.				
중요도	중				
	Before				
대상코드	<pre>// DDoS 공격을 막기 위한 파일 길이 제한 File file = new File(fname); long fileSize = file.length(); long maxSize = 1L * 1024 * 1024 * 1024; // 1GB if (fileSize > maxSize) { System.out.println("=====File Size Error (Maximum is 1GB)====="); return false; }</pre>				
	After				
보안약점/ 보안취약점 이유 및 해결법	<pre>// DDoS 공격을 막기 위한 파일 길이 제한 File file = new File(fname); long fileSize = file.length(); long maxSize = ONE_GB; // 1GB if (fileSize > maxSize) { System.out.println("=====File Size Error (Maximum is 1GB)====="); return false; }</pre> <p style="margin-left: 20px;"><code>private static long ONE_GB = 1024L * 1024 * 1024; // 1GB</code></p>				
	<p>Before 코드에서 파일의 최대 크기를 제한하기 위해 maxSize를 1GB로 설정하고 있다. 1GB를 나타내기 위해 <code>1L * 1024 * 1024 * 1024</code>라는 수식으로 표현했으나 이를 통해 한눈에 1GB라는 것을 알아채기 힘들다. 결과적으로 코드의 가독성 및 유지보수성이 낮아진다.</p> <p>이를 해결하기 위해 심볼릭 상수 <code>ONE_GB</code>를 정의해서 사용했으며 한눈에 파일의 최대 크기가 1GB임을 쉽게 확인할 수 있게 된다.</p>				

코드 리뷰 보고서

리뷰번호	009	작성자	곽채원	작성날짜	2023.06.03
관련규칙	64. Strive for logical completeness.				
중요도	상				
	Before				
	<pre> default: String[] tokens = answer.split(" "); if (answer.contains("dest -k")) { if (answer.contains("dest -kp")) { if (tokens.length == 4) { // 중략 } else { //... 중략 } } else { if (tokens.length == 3) { //중략 } else { // 중략 } } } else if (answer.contains("dest -sign")) { if (tokens.length == 8 && answer.contains("--out")) { // 중략 } else { // 중략 } } else if (answer.contains("dest -vrfy")) { if (tokens.length == 5) { // 중략 } else { // 중략 } } System.out.println(); break; </pre>				
대상코드	After				
	<pre> default: String[] tokens = answer.split(" "); if (answer.contains("dest -k")) { if (answer.contains("dest -kp")) { if (tokens.length == 4) { // 중략 } else { //... 중략 } } else { if (tokens.length == 3) { //중략 } else { // 중략 } } } else if (answer.contains("dest -sign")) { if (tokens.length == 8 && answer.contains("--out")) { // 중략 } else { // 중략 } } else if (answer.contains("dest -vrfy")) { if (tokens.length == 5) { // 중략 } else { // 중략 } } else { System.out.println("=====Command Not Found====="); System.out.println("Check valid commands using \"dest -h\"."); } System.out.println(); break; </pre>				
보안약점/ 보안취약점 이유 및 해결법	<p>Before 코드는 나름대로 사용자의 명령을 엄격하게 검사하려 switch문에 default도 작성하고, 그 안에서 조건문을 통해 촘촘하게 valid한 input을 가려내려 시도했지만, 사용자가 완전히 이상한 명령어를 입력했을 때 처리할 수 있는 코드가 없다. 조건문을 else로 완벽히 마무리하지 않았기 때문에 발생한 논리 구멍이다.</p> <p>이를 해결하기 위해 사용자가 정의된 명령어 밖의 입력을 했을 경우를 처리하는 else문을 추가해주었다. 가독성을 높이기 위해 Command Not Found라는 문자열을 출력한다.</p>				