□ **단국대학교 > 사이버보안 학과 > 인터넷 보안 > 보고서 템플릿**

25년 1학기 인터넷 보안 수업

인터넷 뱅킹 모의해킹

2025년 5월 27일

학번 : 32230324
이름 : 김가람

과정 설명

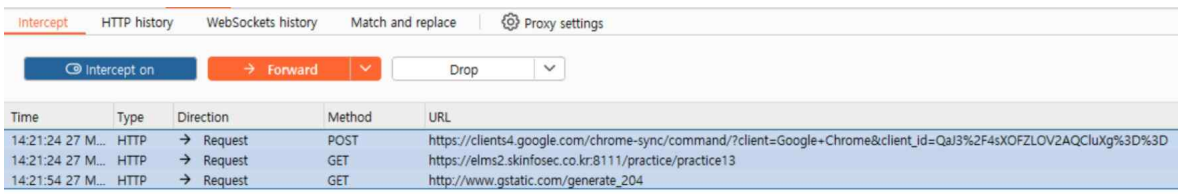
사용자가 거래를 하기 위해 키보드로 정보를 입력하면 이 정보를 서버로 전달하는 전 구간에서 공동인증서 사용에 이용되는 PKI(공개키 기반 구조) 환경을 유지한 상태에서 키보드 보안 솔루션과 연계하여 보호하는 것을 E2E (End to End) 암호화라고 함.
일반 E2E 암호화는 단순히 값만 암호화하는 것이고, 확장 E2E 암호화는 값만이 아니라 파라미터명과 값 전체를 암호화하여 누가 어떤 데이터를 보내는지조차 알 수 없게 함.
금융권에선 이러한 확장 E2E를 사용해 금융 정보를 암호화하고 서버로 전달 후 복호화하는 방식을 사용함. HTML 폼 + JavaScript 조합만이 동적으로 데이터를 암호화하거나 조작할 수 있는데, 데이터 암호화 전에 prompt()를 사용해 파라미터를 변조할 수 있음. 이때, prompt()란 prompt("값", a);와 같이 사용하며 사용자에게 입력을 요청하는 팝업 창을 띄우는 함수임.

<Q13 - 연봉의 200%(최대 1억)까지 대출해주는 상품에서 10억 대출 받기>

Step 1. Intercept & Forward

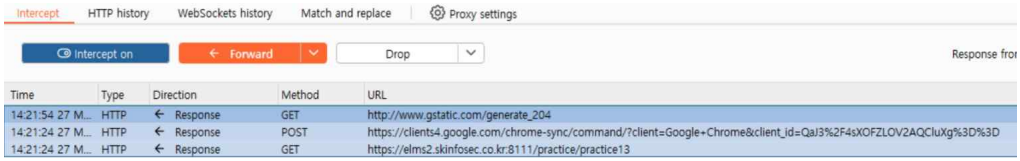
1-1. Intercept

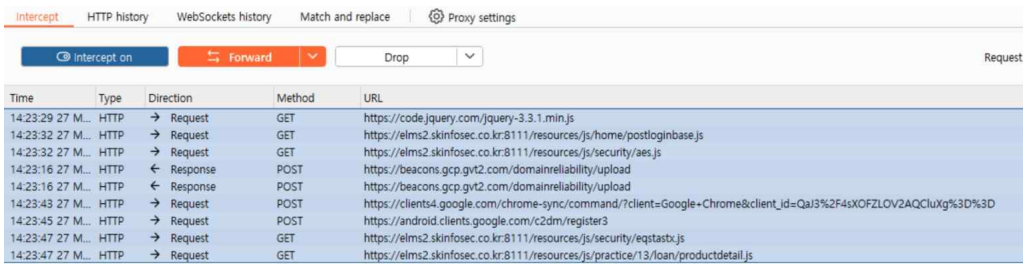
: BurpSuite에서 Intercept를 on한 상태로 상품 설명 페이지의 신청하기 버튼을 누르면 요청을 Intercept 목록에서 확인할 수 있음.



1-2. Forward

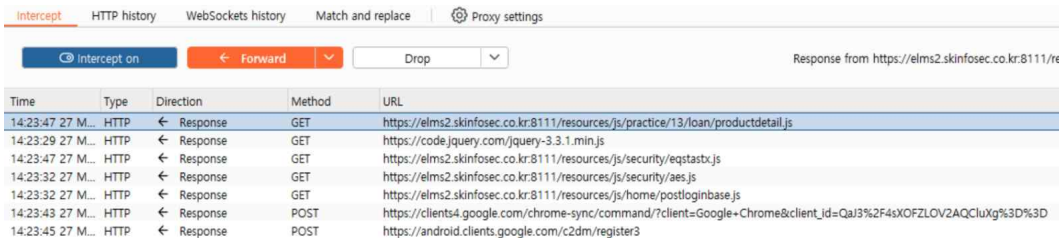
: 목록에 있는 Response나 Request들의 내용에 암복호화 함수가 포함되어 있는지 확인하며 Forward 함.

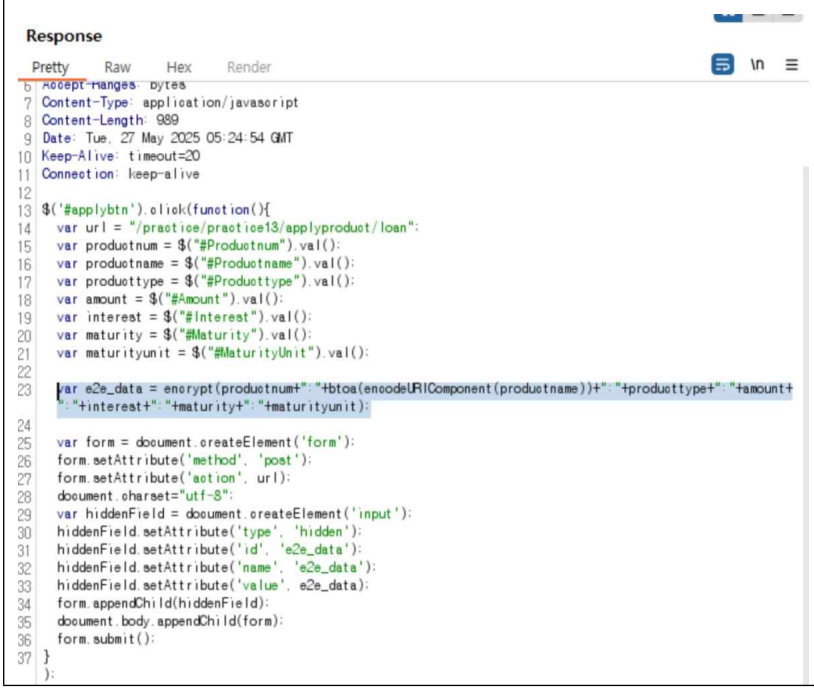




1-3. Response 분석

: Forward를 계속하다보면 encrypt()라는 암호화 함수가 포함된 Javascript 파일인 productdetail.js를 발견할 수 있음. 이때, encrypt() 함수 안에 있는 btoa() 함수는 Base64 인코딩 함수이고, encodeURIComponent() 함수는 URL에 포함될 수 있도록 문자열을 인코딩하는 역할을 함.





Step 2. 파라미터 변조

2-1. prompt() 함수 삽입

: 암호화 함수(encrypt)가 처리되기 전에 파라미터를 변조하기 위해 파라미터 amount, interest, maturity, maturityunit의 값을 입력받는 창을 띄워주는 prompt() 함수를 암호화 함수(encrypt) 전에 삽입 후 Intercept를 off함.

Response

Pretty Raw Hex Render

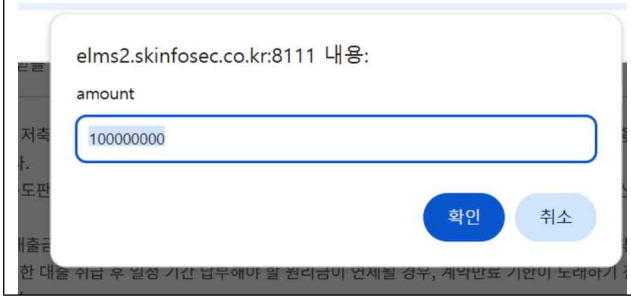
```
Connection: keep-alive

$('#applybtn').click(function(){
  var url = "/practice/practice13/applyproduct/loan";
  var productnum = $("#Productnum").val();
  var productname = $("#Productname").val();
  var producttype = $("#Producttype").val();
  var amount = $("#Amount").val();
  var interest = $("#Interest").val();
  var maturity = $("#Maturity").val();
  var maturityunit = $("#MaturityUnit").val();

  amount = prompt("amount", amount);
  interest = prompt("interest", interest);
  maturity = prompt("maturity", maturity);
  maturityunit = prompt("maturityunit", maturityunit);

  var e2e_data = encrypt(productnum+":"+btoa(encodeURIComponent(productname))+":"+producttype+":"+amount+
  ":"+interest+":"+maturity+":"+maturityunit);

  var form = document.createElement('form');
  form.setAttribute('method', 'post');
  form.setAttribute('action', url);
  document.charset="utf-8";
  var hiddenField = document.createElement('input');
  hiddenField.setAttribute('type', 'hidden');
  hiddenField.setAttribute('id', 'e2e_data');
  hiddenField.setAttribute('name', 'e2e_data');
  hiddenField.setAttribute('value', e2e_data);
  form.appendChild(hiddenField);
  document.body.appendChild(form);
  form.submit();
}
);
```

2-2. 변조

: 신청하기 버튼을 누르면 Step 2-1에서 삽입한 prompt() 함수로 인해 값을 입력할 수 있는 창이 뜸.

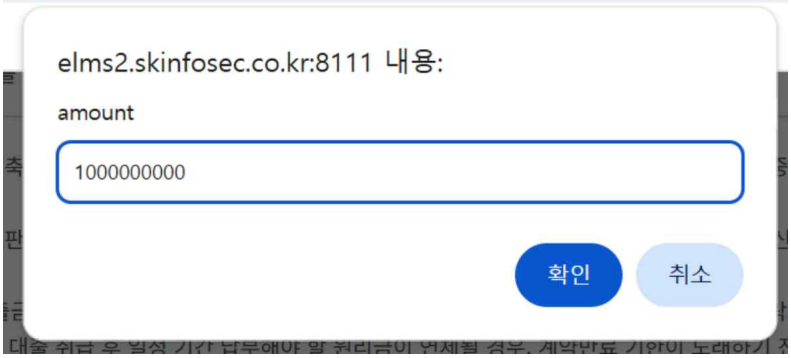
- amount : 1억

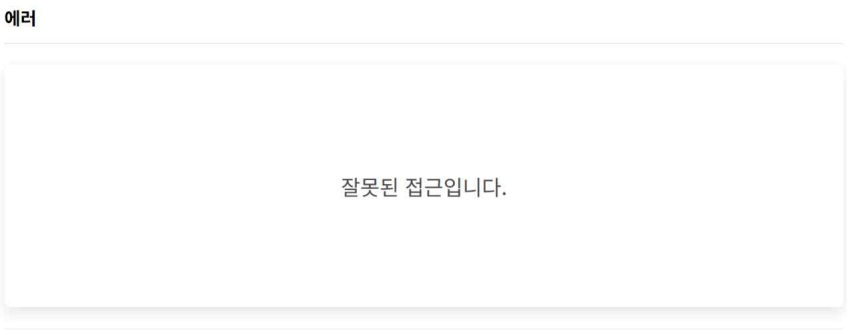
: amount (한도)를 1억으로 설정



- amount : 10억

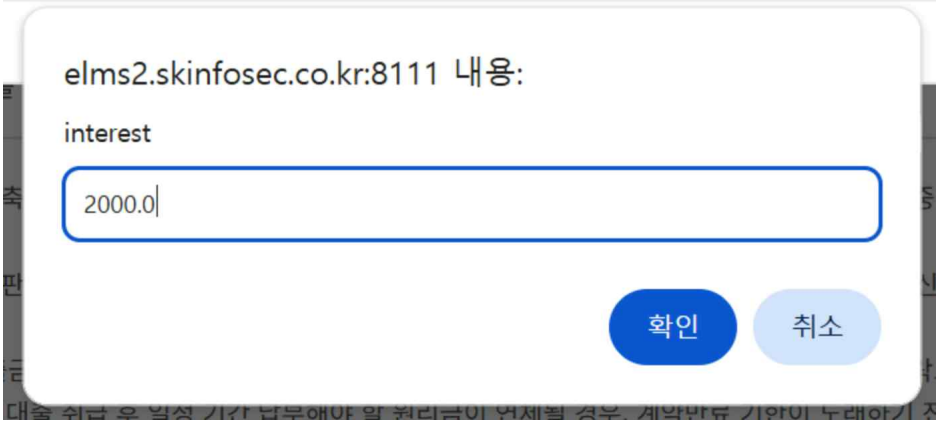
: 이때, 10억을 대출받기 위해 한도를10억으로 변조할 경우 잘못된 접근이라는 페이지로 연결되는데, 이는 amount라는 파라미터가 서버에서 검증받고 있기 때문임.





- interest

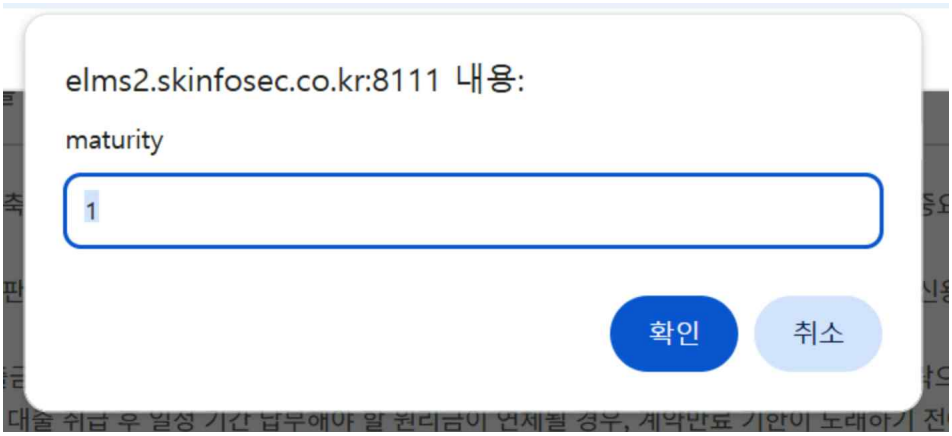
: amount(한도) 파라미터는 서버에서 검증받고 있기에 interest (이자율)을 200%에서 2000%로 변조함. 이때 interest 파라미터는 서버에서 검증받고 있지 않아 변조에 성공함.



(다음 페이지에 이어서)

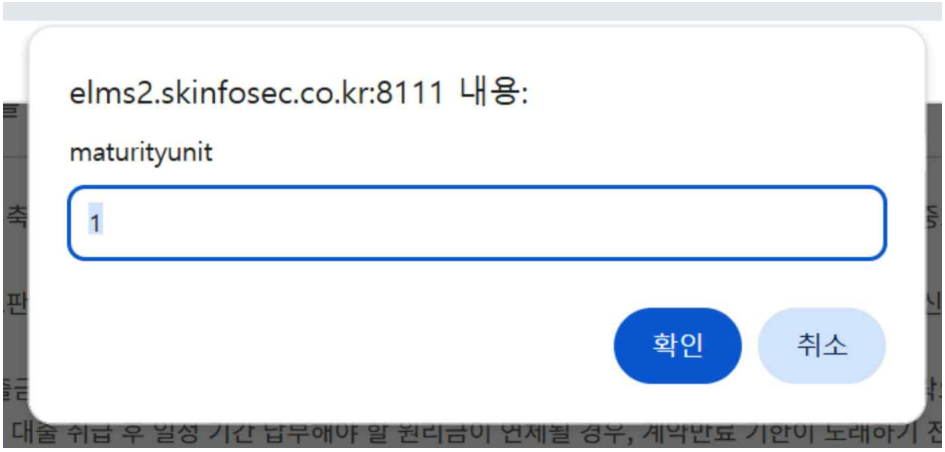
- maturity

: maturity(신청 기간) 값은 변조할 필요가 없으므로 기존값인 1을 그대로 둠.



- maturityunit

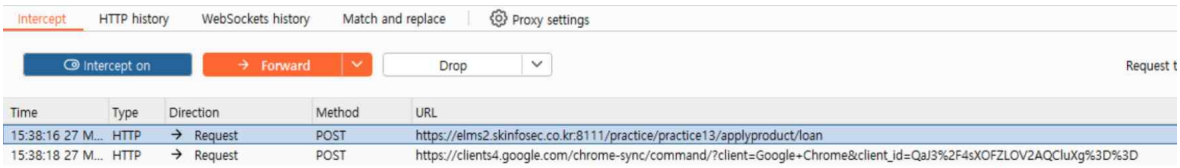
: maturityunit(기간의 단위) 값도 변조할 필요가 없으므로 기존값인 1을 그대로 둠.



Step 3. Intercept & Forward

3-1. Intercept

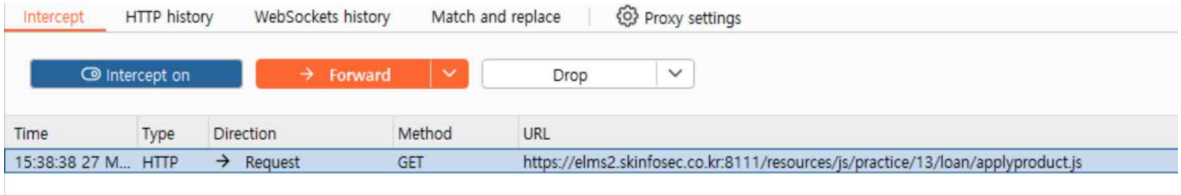
: Step 2-2에서 마지막 입력창(maturityunit)에서 확인 버튼을 누르기 전에 intercept 기능을 켜고 확인을 누르면 목록에서 요청을 확인할 수 있음.



(다음 페이지에 이어서)

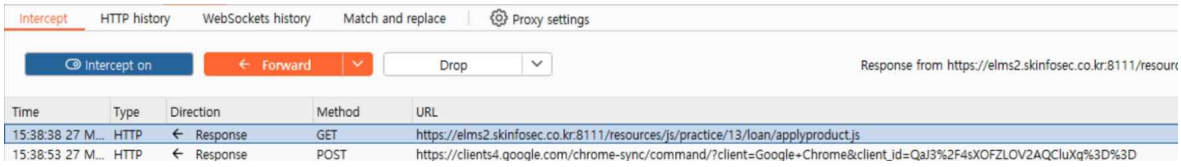
3-2. Forward

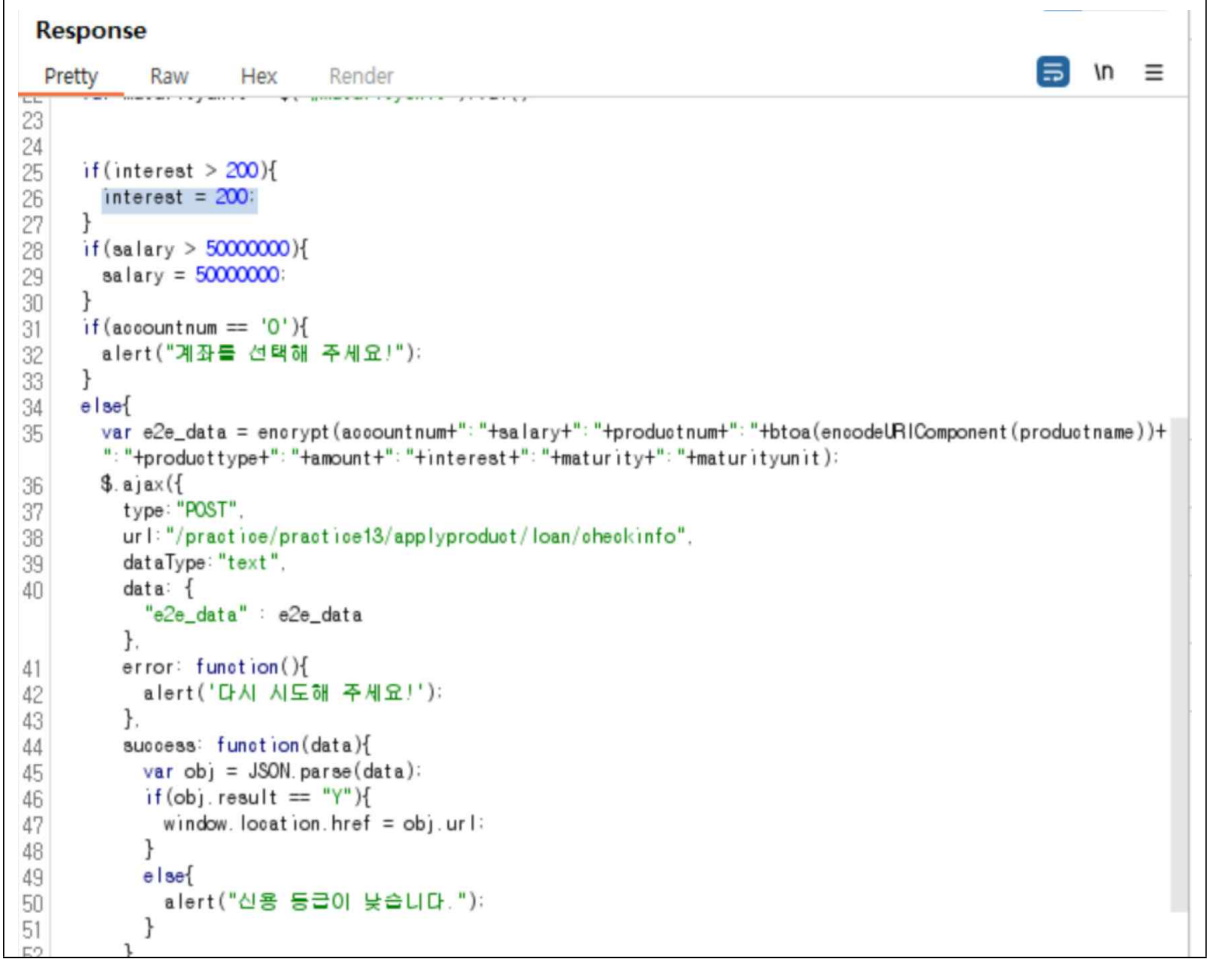
: 목록에 있는 Response나 Request들의 내용을 확인하며 Forward 함.



3-3. Response 분석

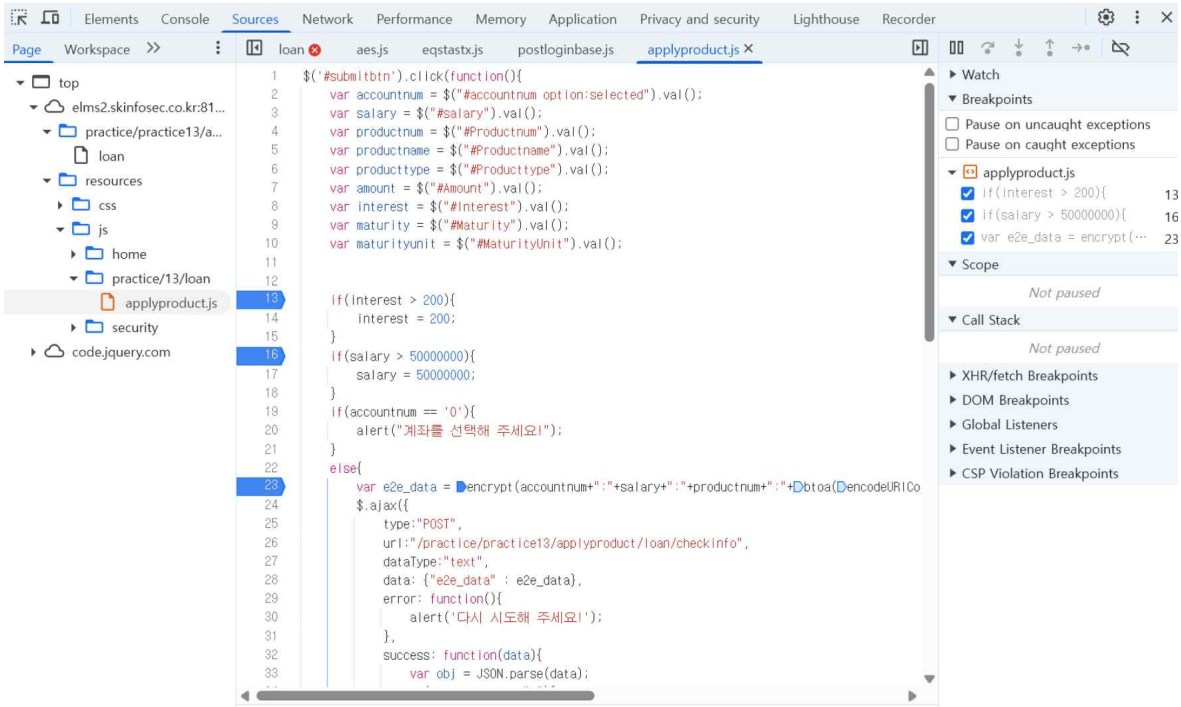
: forward를 계속 하다보면 interest(이자율) 값과 salary(연봉) 값에 대한 조건문이 포함된 Javascript 파일인 applyproduct.js를 발견할 수 있음. 해당 파일을 분석해보면 interest(이자율)이 200 이상일 경우, 200으로 재조정하고, salary(연봉)이 5천만 이상일 경우, 5천만으로 재조정하는 코드가 포함되어 있음. 해당 코드 이후 계좌를 선택하고, 앞서 정해진 파라미터 값들을 암호화 하는 함수가 있음.





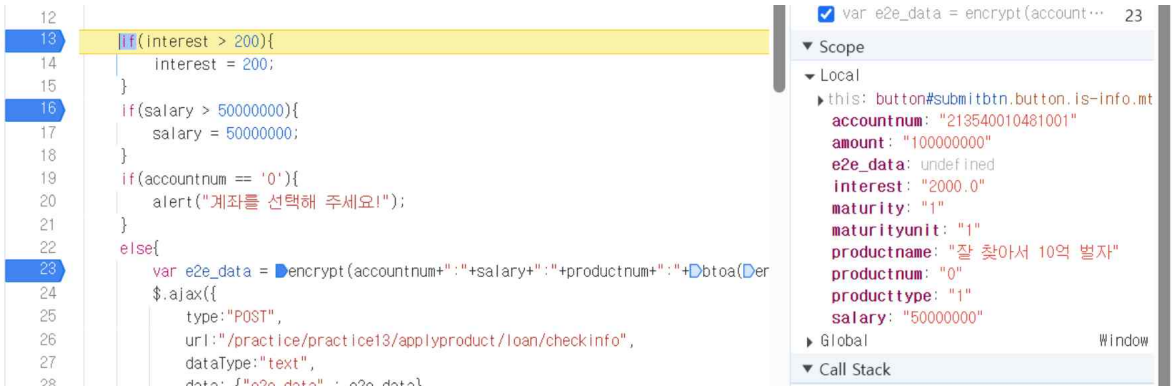
3-3-1. Breakpoint

: 개발자 도구(F12)를 이용해 파라미터 값이 어떻게 변경되는지 확인해보고자 함. interest 조건문, salary 조건문, 암호화 함수 부분 각각에 BreakPoint를 표시함.



3-3-2. 처음 파라미터 값

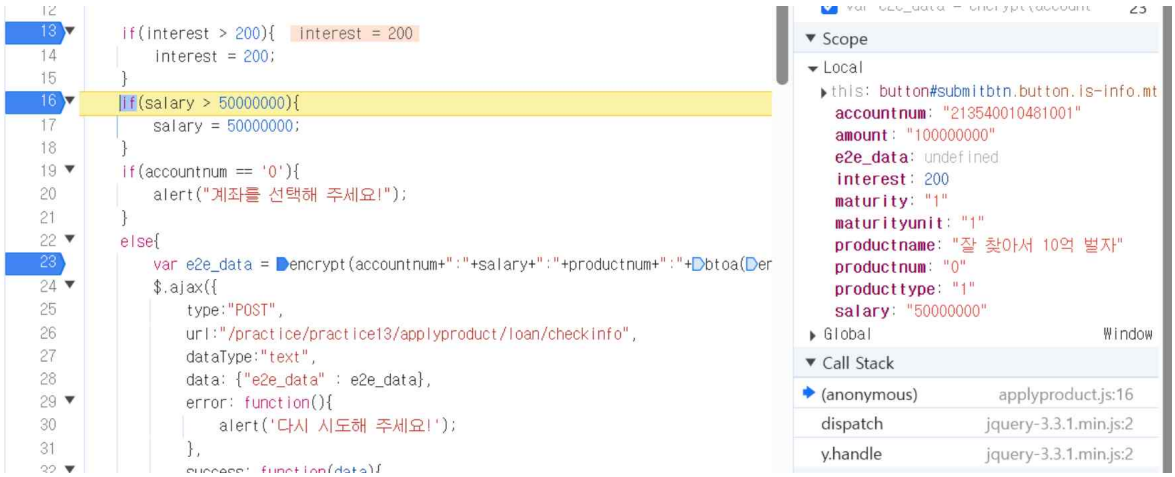
: BreakPoint 표시 후 작동 버튼을 눌러보면 interest 조건문에서 멈춤. 이때, 오른쪽 Scope 부분을 확인해보면 현재 파라미터 값들을 확인할 수 있음. 아직 interest 조건문으로 진입하기 전이므로 값이 여전히 2000임.



(다음 페이지에 이어서)

3-3-3. interest 조건문 실행 후 파라미터 값

: 다시 작동 버튼을 눌러보면 salary 조건문에서 멈춤. interest 조건문을 지나온 상태이므로 값이 200으로 바뀌어 있음을 확인할 수 있음.



3-3-4. salary 조건문 실행 후 파라미터 값

: 다시 작동 버튼을 눌러보면 암호화 함수 부분에서 멈춤. salary 조건문을 지나왔지만 기존 값이 5천만 초과가 아니었기에 변화가 없는 상태임을 알 수 있음.

3-3-5. 실행 결과

: 다시 작동 버튼을 눌러본 결과, 신용 등급이 낮다는 메시지 창이 뜸.

(다음 페이지에 이어서)

Step 4. 조건문 수정

: interest 값이 2000에서 200으로 재조정되지 않게 하기 위해 interest = 2000으로 조건문을 수정 후 intercept 기능을 off함.

Response

Pretty Raw Hex Render

```
23
24
25  if(interest > 200){
26    interest = 2000;
27  }
28  if(salary > 50000000){
29    salary = 50000000;
30  }
31  if(accountnum == '0'){
32    alert("계좌를 선택해 주세요!");
33  }
34  else{
35    var e2e_data = encrypt(accountnum+":"+salary+":"+productnum+":"+btoa(encodeURIComponent(productname))+
      ":"+producttype+":"+amount+":"+interest+":"+maturity+":"+maturityunit);
36    $.ajax({
37      type:"POST",
38      url:"/practice/practice13/applyproduct/loan/checkinfo",
39      dataType:"text",
40      data: {
        "e2e_data" : e2e_data
      },
41      error: function(){
42        alert('다시 시도해 주세요!');
43      },
44      success: function(data){
45        var obj = JSON.parse(data);
46        if(obj.result == "Y"){
47          window.location.href = obj.url;
48        }
49        else{
50          alert("신용 등급이 낮습니다.");
51        }
```

Step 5. 가입 정보 입력

: Step 4의 결과로 가입 정보 입력 페이지로 연결됨. 대출 금액 부분을 확인해보면 이자율이 2000%로 유지되어 있음을 확인할 수 있음. 연봉에 5천만을 입력 후 다음으로 버튼을 누름.

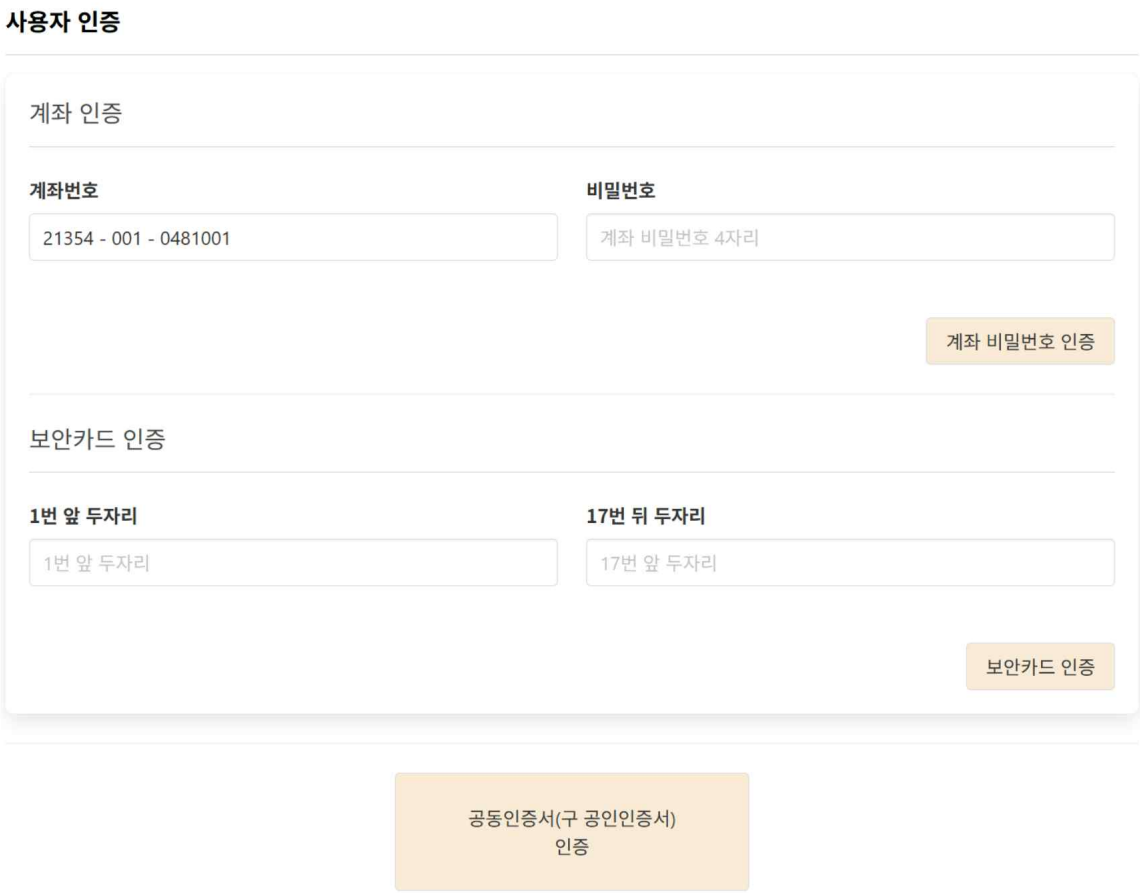
가입 정보 입력

신청 상품	잘 찾아서 10억 벌자
신청 기간	1 년
연봉 정보	50000000 원
대출 금액	연봉의 2000.0% (최대 100,000,000원)
입금 계좌번호	21354 - 001 - 0481001

다음으로

Step 6. 사용자 인증

: Step 5의 결과로 사용자 인증 페이지로 연결됨.



6-1. 계좌번호, 보안카드 인증

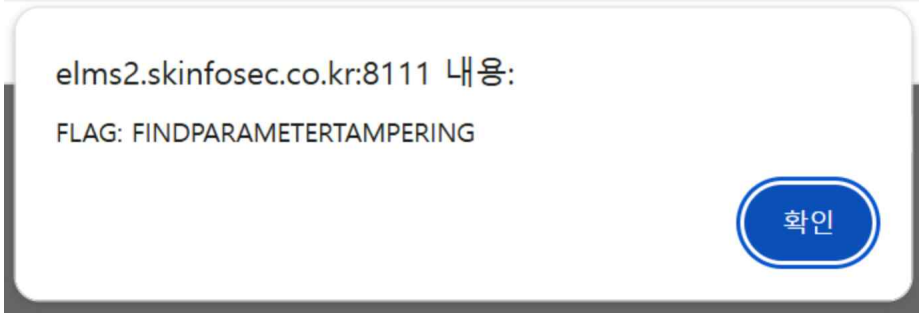
: 계좌번호와 보안카드 인증을 완료하면 인증 되었다는 메시지 창이 뜸.

(다음 페이지에 이어서)

6-2. 공동인증서 인증

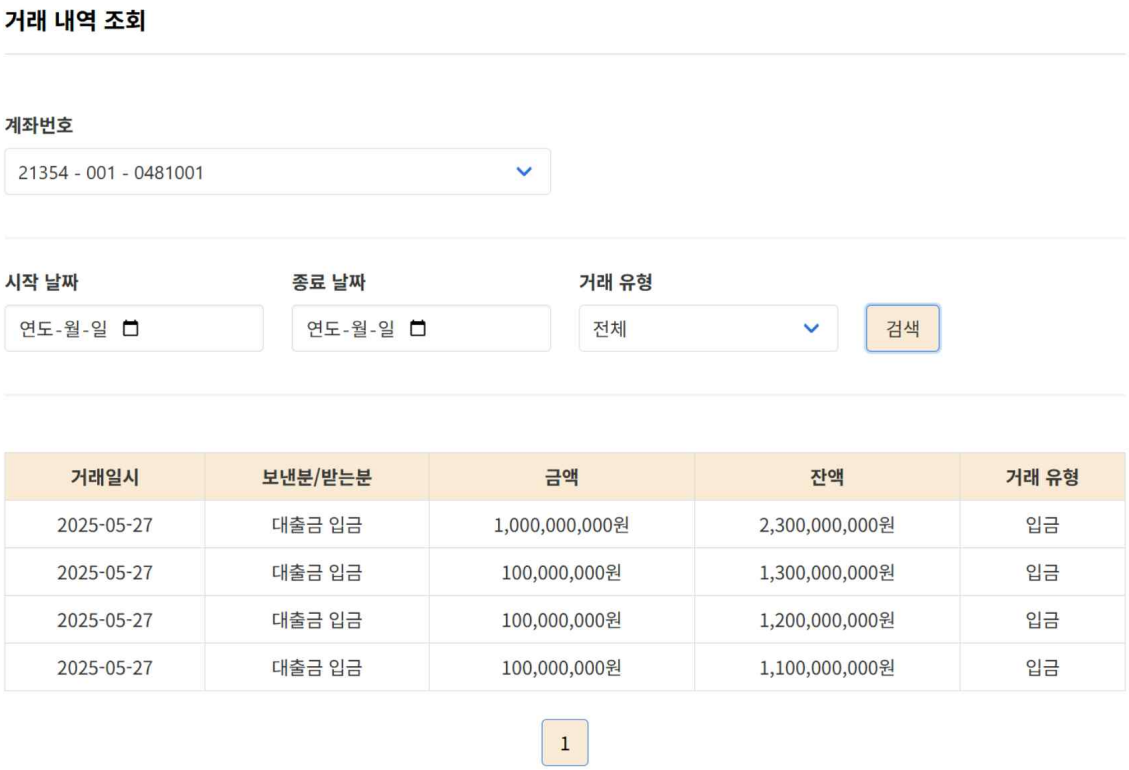
: 공동인증도 완료하면 플래그(FINDPARAMETERTAMPERING)가 있는 메시지 창이 뜸.

- 플래그 획득



Step 7. 거래 내역 확인

: 거래 내역의 가장 위 거래 내역을 확인해보면 10억 대출에 성공했음을 확인할 수 있음.



성명	프로젝트 후 소감
김가람	이번 실습을 하면서 가장 먼저 떠오른 것은 이전에 금융 사이트 버그바운티에 참여하려고 했지만 아무것도 하지 못하고 끝났던 기억이 떠올랐음. 당시엔 취약점을 찾아보고 싶었지만, 막상 들어가 보니 파라미터, 암호화, 구조 등 하나도 이해가 되지 않아 무력감만 느꼈었는데, 실습을 통해 E2E 암호화에 대해 알게 되고, 이것이 금융권에서 어떤 방식으로 적용되는지에 대해 배움으로써 예전의 막막했던 경험이 조금 해소된 느낌이 들었음. 앞으로는 이번 실습에서 다뤘던 것처럼 암호화 처리 방식, 파라미터 동작 방식 등을 더 공부하고 다시 버그바운티에 도전해보고 싶음.