캡차(CAPTCHA), 가상키보드, 미디어 플레이어

정보접근성 기술이슈 분석보고서

2017

NIA 한국정보화진흥원

목 차

용이 불편하거나 접근이 불가능하다면 좀 더 나은 대안이 없을지를 끊임없이 고민하고 연구해야 할 것이다. 기술은 인간을 위해 존재해야 하기 때문이다.

가상 키보드(보안 키보드)

가상 키보드란?

가상 키보드는 말 그대로 물리적인 키보드가 아니라 화면에 가상으로 나타나는 키보드이다. 이러한 가상 키보드를 사용하는 이유는 개인의 '정보를 보호하기 위해서'이다.

전화번호나 주민등록번호, 은행 계좌의 비밀번호 등과 같이 개인정보를 물리적인 키보드로 입력할 경우 해킹프로그램이 사용자가 키보드로 입력하는 키 값을 탈취하여 개인정보를 빼낼 수 있는 위험을 미연에 방지하고자 하는 목적을 가지고 있다.

다음은 국내 및 해외 은행의 가상 키보드 화면이다.

[모바일 뱅킹 가상 키보드]

위의 가상 키보드 자판을 살펴보면 QWERTY 유형과 ABC 유형이 있다. QWERTY 혹은 ABC 자판 배열은 일반적인 키보드와 동일하기 때문에 사용자가 쉽게 적응하여 빠르게 입력할 수 있다. 그러나 입력 값의 좌표와 사용하고 있는 자판의 배열 또한 가로채기 쉬워 사용자가 입력한 정보가 손쉽게 유출될 수 있다는 단점이 있다.

입력 정보 유출 문제를 해결하기 위해서 무작위 키 배열을 사용하는 방식도 등장했다. 그러나 무작위 키 배열의 경우 사용자가 사용하는데 매우 불편하다는 치명적인 단점 때문에 최근에는 이를 응용하여, 무작위로 여백을 삽입하는 방식을 사용하고 있다.

이 방식은 기존 QWERTY 자판에 임의의 무작위 여백을 삽입함으로써 전체적인 배열은 유지하되 입력 좌표를 통한 입력 키에 대한 유추를 어렵게 만든 방식이다.

[QWERTY 기반 무작위 여백 방식]

가상 키보드와 보안

일반적으로 키보드에 키를 입력하면 키보드에 연결된 마이크로 프로세서가 키보드를 컴퓨터에 연결하는 케이블을 통해 신호를 보낸다.

이 때 키보드를 통해 입력된 데이터가 운영체제에 도달하면 키보드 장치 드라이버에서 문자, 숫자 및 기호에 대한 키보드 "스캔 코드"의 변환을 처리하게 되는데 이를 키로거(Keylogger)가 가로챈다.

즉, 해커가 사용자 계정 정보를 가로채기 위해 사용자의 입력을 기록하는 악성 코드인 키로거를 사용하여, 키 입력이 이루어지게 되면 운영체제를 통해 정보를 빼내갈 수 있다.

개인의 '정보를 보호하기 위해서' 가상 키보드가 등장하였으나, 일부 해킹 시도는 무력화 시킬 수 있지만 모든 해킹 시도에서 안전하다고 보장할 수는 없다.

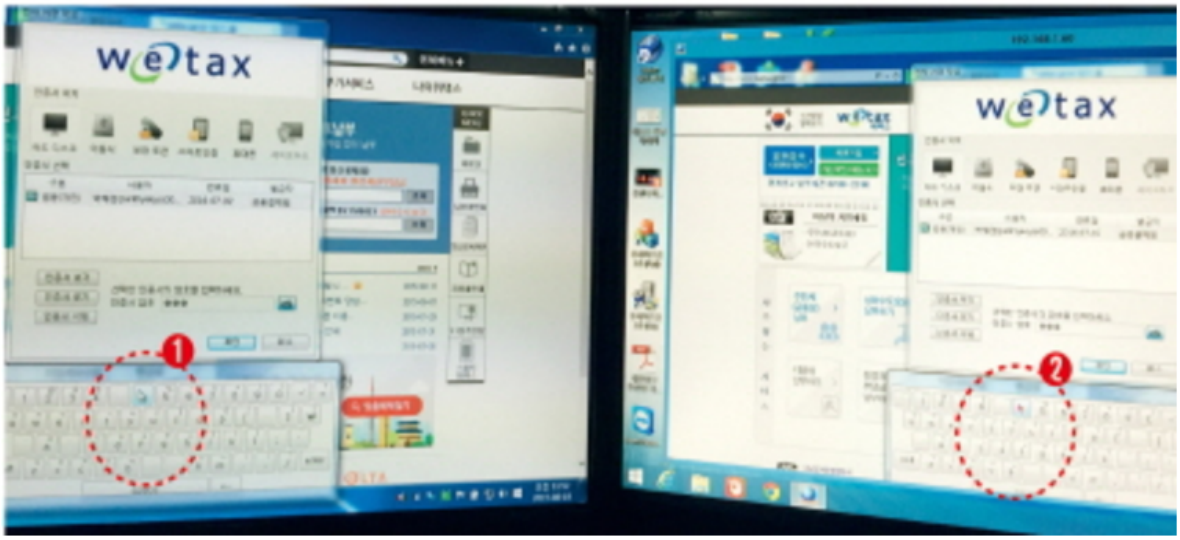
여기서 기억해야 할 가장 중요한 개념은 키로거가 다른 형태의 악성 코드에 불과하다는 것이다. 키로깅(keylogging) 멀웨어(Malware)가 실제로 키 입력을 포함하여 더 많은 정보를 가로챌 수 있다.

마우스를 사용하면 가상 키보드를 통해 사용자가 입력한 키 정보를 감출 수 있지만, 키로거는 모든 마우스 클릭에 대해 클릭 한 위치와 순서를 정확히 보여주는 이미지를 캡쳐할 수 있다. 바로 사용자의 가상 키 입력을 캡쳐한 것이다.

키로깅의 이와 같은 접근 방식은 마우스 인터랙션에 따른 키보드 레이아웃 화면을 획득 함으로 일반적인 보안 기술을 회피할 수 있다. 키로거는 키보드 배치 방식에 관계 없이 보고 있는 내용과 클릭 한 위치를 기록한다.

더 스쿠프(The SCOOP)가 발표한 자료에 따르면 RCS(Remote Control System - 원격조정시스템) 해킹 툴을 이용해 국내 금융회사 35곳(저축은행 제외), 위택스(지방세 인터넷 납부시스템), 공공 I-PIN에 설치된 '가상 키보드'의 보안 능력을 검증한 결과는 참혹했다.

RCS에 의한 해킹 성공률이 무려 90%에 육박했기 때문이다. 해킹 방지를 위해 도입한 가상 키보드가 해커의 공격을 거의 막아내지 못한다는 사실이 밝혀진 것이다.



[RCS에 해킹 된 가상 키보드 시연 모습]

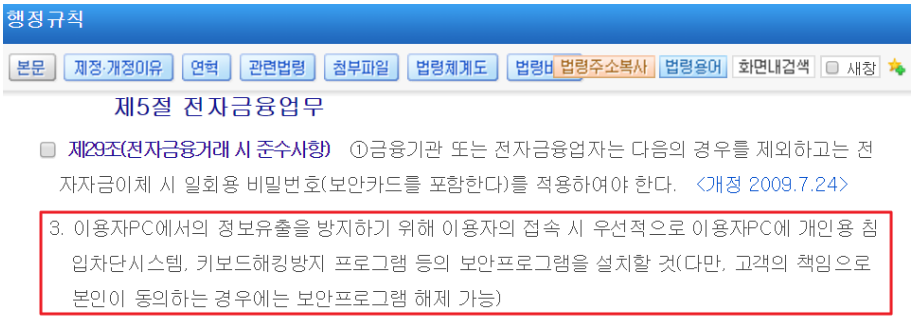
위 그림에서 좌측은 사용자의 PC이고 우측은 해커의 PC이다. 사용자가 자신의 PC에서 가상 키보드에 마우스를 활용해 공인 인증서의 비밀번호를 입력(1)하면, 해커의 PC에 사용자 마우스 커서의 움직임 정보(2)가 고스란히 노출된다.

가상 키보드를 사용하는 사용자의 공인 인증서 비밀번호가 무력하게 해커에게 노출된 셈이다.

가상 키보드 제공의 국내 법적 근거

앞서 살펴본 문제점처럼 '가상 키보드의 보안능력이 약하다'는 지적은 이미 수년 전부터 제기되어 왔다. 그렇다면 보안이 취약한 가상 키보드를 기업에서 아직까지 고집하고 있는 이유는 무엇일까?

기업에서는 가상 키보드를 제공하고 있는 이유로 법률에서 요구하는 요건을 채우기 위한 불가피 한 조치라고 설명한다. 그렇다면 그 근거는 무엇일까?



[가상 키보드 제공의 근거라고 주장하는 행정규칙]

기업은 전자금융감독규정 시행세칙의 행정규칙 중 전자금융업무의 전자금융거래 시 준수사항 항목을 그 근거로 삼고 있다.

해당 행정규칙에 따르면 이용자 PC에서의 정보유출을 방지하기 위해 이용자의 접속 시 우선적으로 이용자 PC에 개인용 침입차단시스템, 키보드해킹방지 프로그램 등의 보안프로그램을 설치할 것(다만, 고객의 책임으로 본인이 동의하는 경우에는 보안프로그램 해제 가능) 이라고 나와있다.

그러나 해당 규정은 2010년 8월 12일의 시행세칙이고, 2012년 5월 29일 시행세칙이 개정되면서 관련 내용은 폐지되었다. 결국 기업은 이미 폐지된 내용을 설치 근거로 내세우고 있는 것이다.

위의 내용대로라면 전자금융감독규정 시행세칙의 개정을 시작으로 금융감독원에서 자율 보안을 선언하였기 때문에 가상키보드를 꼭 써야 할 이유는 없다. 다만, 금융보안원의 보안 취약점 진단에서 보안 가상 키보드 프로그램을 설치하도록 제공하지 않으면 취약점으로 진단을 하기 때문에 시행세칙에서 제외 되었다 하더라도 현실적으로는 설치를 하도록 유도할 수 밖에 없다는 후문도 있다.

가상 키보드 제공 사례 분석

가상 키보드는 사용자 입력 값을 받아야 하는 입력서식 중에서도 보안이 필요한 콘텐츠 유형에 사용되고 있다.

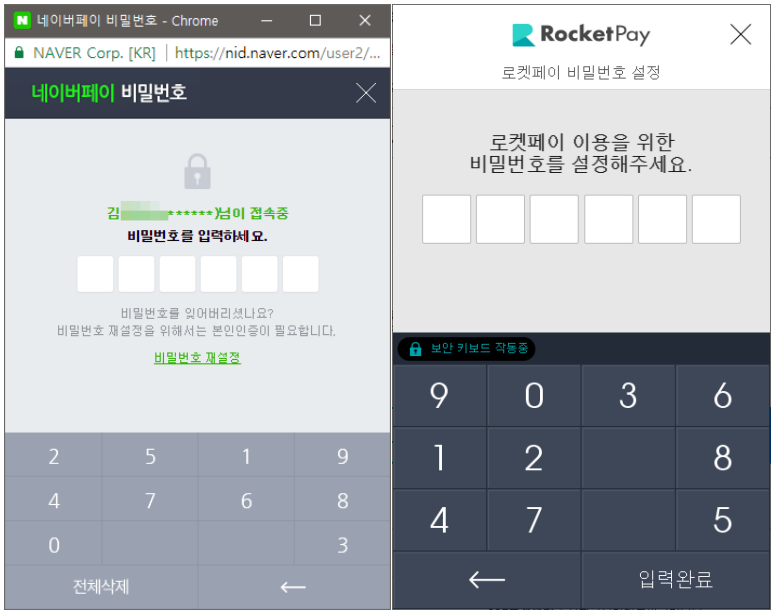
이러한 입력양식은 특별한 분야에서 제한적으로 사용되는 것이 아니라 금융, 결제 등의 분야에서 널리 사용되고 있고, 그 유형으로는 사용자 로그인, 카드번호 입력, 각종 비밀번호 입력 등을 들 수 있다. 웹을 활용하는 사용자라면 누구나 쉽게 접할 수 있는 유형이다.

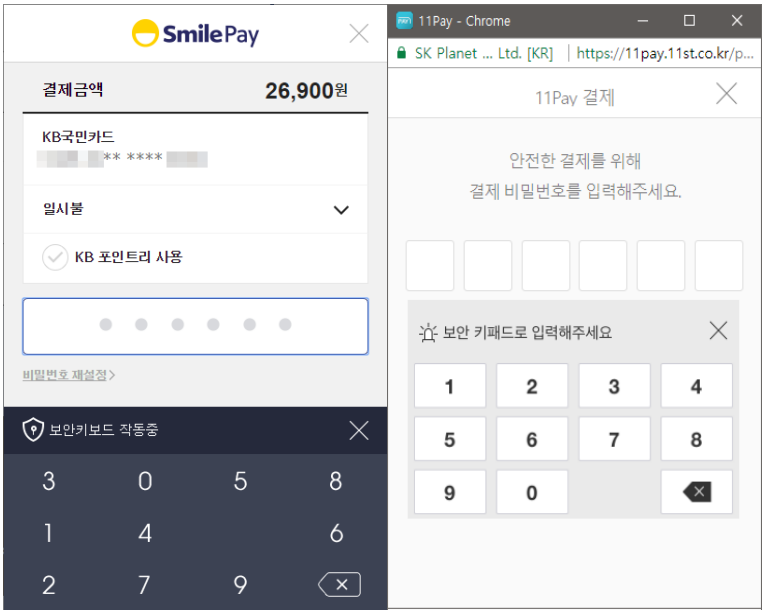
이제부터 다양한 분야에서 가상 키보드를 제공하고 있는 사례를 살펴보기로 하자.

접근할 수 없는 암호입력 키패드

암호입력 키패드의 인식 및 키보드 접근

다음은 간편결제 서비스에서 제공하는 암호입력 화면 사례이다.



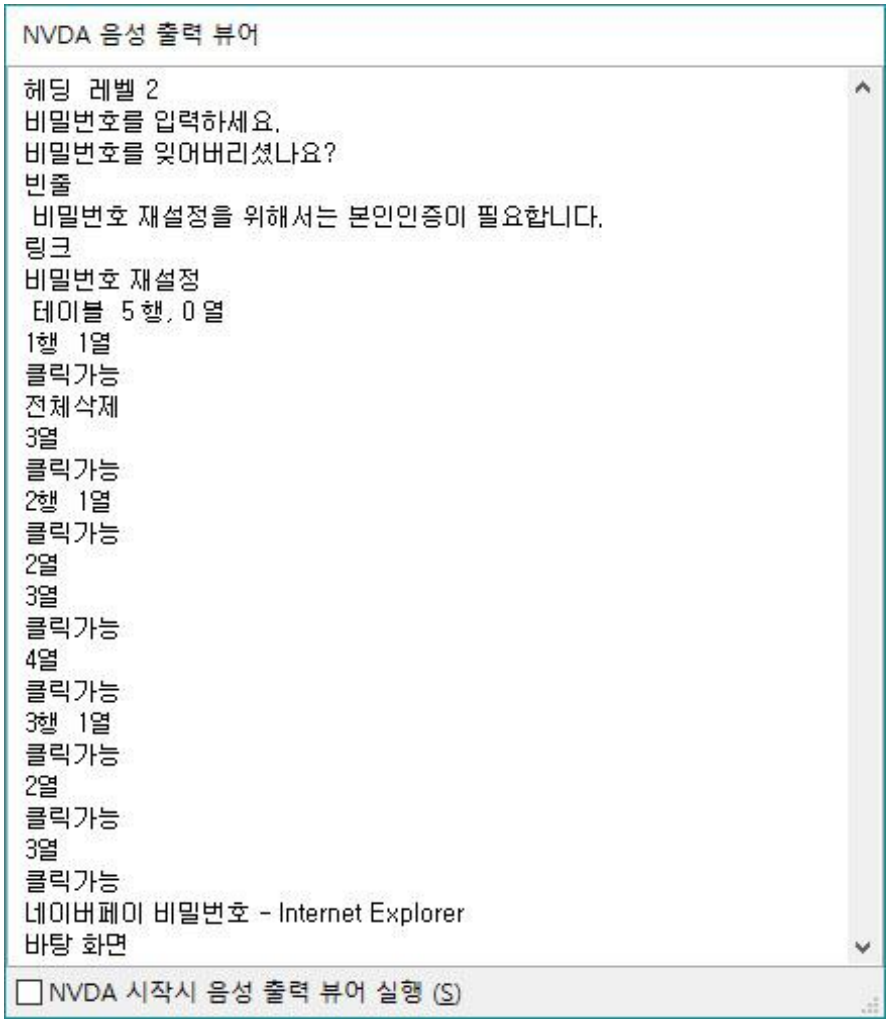


시각장애인이 아닌 사용자의 경우 암호입력 키패드를 사용할 수 있지만, 스크린리더 사용자의 경우, 암호입력 키패드를 인식할 수 없는 등의 문제가 있다.

다음 재연 절차를 따라 스크린리더가 어떻게 읽어주는지를 살펴보자.

재연 절차(NVDA)

방향키를 이용하여 아래로 계속 탐색



음성으로 들리는 내용 중 가장 심각한 이슈는 가상 키패드의 숫자를 전혀 인식할 수 없는 것이다.

음성만 들은 상태에서는 표 형태로 만들어진 클릭 가능한 콘텐츠들이 나열되어있다는 것만 알 수 있을 뿐, 키패드 숫자에 대한 정보가 없기 때문에 클릭 가능 콘텐츠들이 제공하는 기능을 전혀 알 수 없다.
키패드 영역의 마크업을 살펴보면 다음과 같다.

```
<td>
  <a class="key" onclick="doClick('0')">
    <span class="number key1_1"></span>
  </a>
</td>
```

키패드 영역에서 어떤 정보도 음성으로 들을 수 없었던 이유는 가상 키패드의 버튼이 링크로 구현되었지만 링크에는 기능을 알 수 있는 링크 텍스트가 전혀 제공되지 않은 상태이다.

스크린리더로 들을 수 있는 텍스트 정보가 전혀 없기 때문에 시각장애인은 가상 키패드에 접근하더라도 키패드의 기능을 인식할 수 없다.

유사한 간편결제 서비스의 키패드 마크업도 한 번 살펴보자.

```
<td class="rocketpay-keypad-td">
    <a class="rocketpay-keypad-key" href="#" data-key="E17qphjHIM">
        <span class="rocketpay-keypad-position-2"></span>
    </a>
</td>
```

다른 간편결제 서비스의 암호 입력 키패드 마크업이지만 이전 사례와 크게 다르지 않다. 이 곳에서도 키패드 영역에 키를 구분할 수 있는 텍스트 정보를 전혀 제공하지 않은 상태이다.

스크린리더를 통해 키패드의 내용을 듣고 입력해야 하는 시각장애인은 키패드 정보를 음성으로 인식할 수 없어 암호를 입력하는 것이 불가능하므로 연결된 서비스의 이용이 불가능한 문제를 발생시킨다.

시각장애인을 위한 정보만 누락된 것은 아니다. 키보드만 사용하는 사용자가 키보드로 초점을 이동하여 보안 키패드에 접근하는 것 또한 불가능하다.
키패드 영역의 마크업을 살펴보면 키보드로 접근할 수 없는 이유를 확인할 수 있다.

```
<a onclick="doClick('1');" class="key">
  <span class="number key1_2"></span>
</a>
```

암호를 입력 하는 키패드를 a 요소로 구성하고 href 속성을 정의하지 않은 상태이다. 이로 인해 키보드로 링크에 접근할 수 없어 키패드 입력이 불가능하다.

키보드로 접근할 수 없는 또 다른 간편결제 서비스의 마크업을 살펴보면 다음과 같다.

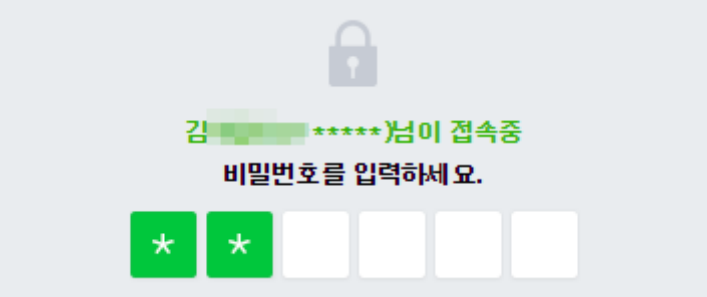
```
<button class="nfilter_keypad_button"
        style="position:absolute; left:79px; top:-1px; width:76px;
               height:45px; display:block;"
        ondragstart="return false;" onmousedown="return false;"
        onmouseup="javascript:nFilterOnKeyClick(this)"
        ondoubleclick="return false;"
        ontouchstart="javascript:nFilterOnTouchstart()"
        ontouchend="javascript:nFilterOnTouchend(this)"
        id="tnn3qbpvfy" alt="Virtual Keyboard">
</button>
.nfilter_keypad_button {
    background: transparent 0 0 repeat scroll
                url(/pages/images/v2/nfilter_keypad_transparent.png);
    border: none;  cursor: pointer;  overflow: hidden;  outline: none;  display: block;
    -webkit-tap-highlight-color: rgba(0,0,0,0);
    -webkit-tap-highlight-color: transparent;
}
```

키패드를 구성하는 버튼에 outline: none 스타일을 사용하고 있어 키보드로 접근하더라도 어느 키패드에 접근했는지 알 수가 없는 상태이다.

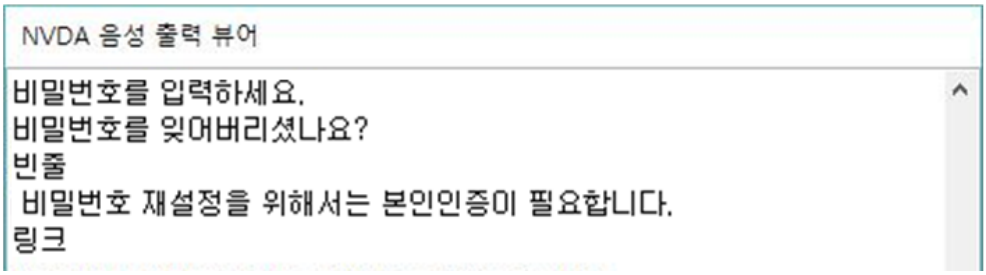
Tab키에 의한 키보드 초점 이동이 키패드 영역에 도달하지 못하도록 막고 있기 때문에 키보드 사용자 또한 키패드 사용이 불가능하다.

간편결제 콘텐츠에서 키패드 영역에서만 이슈가 존재하는 것일까? 그렇지 않다.

다음과 같이 키패드를 입력할 때 현재 몇 글자가 입력되었는지 확인하는 것은 매우 중요한 과정이며 반드시 필요한 정보이다.



이 사례에서는 "비밀번호를 입력하세요"와 "비밀번호를 잊어버리셨나요?" 사이에 입력해야 할 개수와 현재 입력된 개수를 알 수 있는 박스 영역이 존재한다. 이 부분을 스크린리더로 들어보고 어떤 정보가 인식되고 있는지 다시 한 번 확인해보자.



암호글자수에 대한 정보가 표시되는 영역에서 음성으로는 아무런 정보도 인지되지 않는다. 사용자가 입력해야 하는 입력 양식이 있다는 것을 알 수 없고, 총 몇 글자를 입력해야 하고 현재 몇 글자까지 입력되었는지 구분할 수 있는 정보가 전혀 없다.

왜 그럴까? 이 부분의 마크업을 살펴보면 다음과 같다.

```
<h2 class="h2 center">비밀번호를 입력하세요</h2>
<div class="password center">
    <span class="charactor on" id="key_1"></span>
    <span class="charactor" id="key_2"></span>
    <span class="charactor" id="key_3"></span>
    <span class="charactor" id="key_4"></span>
    <span class="charactor" id="key_5"></span>
    <span class="charactor" id="key_6"></span>
```

```
</div>
```

마크업을 보면 글자수를 확인할 수 있는 텍스트 정보를 제공하고 있지 않다.

IR(Image Replacement)기법을 사용하여 글자수를 구분하는 배경 이미지를 제공하면서 그와 동등한 수준의 대체텍스트를 제공하지 않아 스크린리더로는 어떤 정보도 인식할 수 없는 것이다.

다른 간편결제 서비스에서는 이 영역을 어떻게 제공하고 있는지 마크업을 살펴보자.

```
<button type="button" class="bt_scu active" id="scu1" name="keyPadPassword"
style="">
  <span id="b0" class="spr active">*</span>
  <span id="b1" class="spr active">*</span>
  <span id="b2" class="spr">*</span>
  <span id="b3" class="spr">*</span>
  <span id="b4" class="spr">*</span>
  <span id="b5" class="spr">*</span>
</button>
```

해당 글자수 영역에 *를 제공하여 글자수가 몇 개인지 확인은 가능하지만 현재 입력된 글자수와 입력되지 않은 글자수를 구분할 수는 없다. 서로 다른 class를 사용하여 시각적으로 입력 된 글자수를 구분하고 있지만 동등한 수준의 대체텍스트는 제공하지 않은 것이다.

추가로, 기능이 없는 암호 글자수 영역을 button 요소로 구성하였기 때문에 음성으로 정보를 인지하는 시각장애인이 콘텐츠의 목적이나 용도를 오해하게 되는 또 다른 이슈를 가지고 있다.
다른 서비스의 동일한 콘텐츠는 마크업이 조금 다르지만 거의 같은 이슈를 발견할 수 있다.

```
<a href="#none" class="masking-box" id="masking-box">
  <span class="active">첫번째 마스킹 문자</span>
```

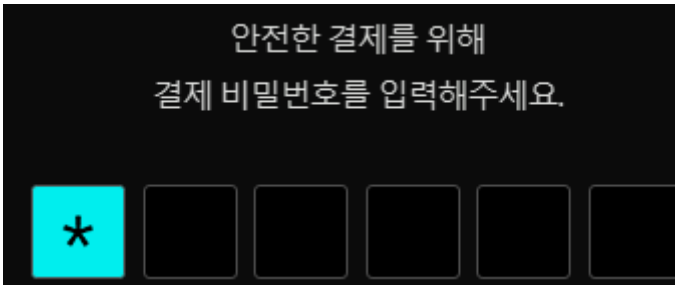
```
  <span class="active">두번째 마스킹 문자</span>
  <span class="">세번째 마스킹 문자</span>
  <span class="">네번째 마스킹 문자</span>
  <span class="">다섯번째 마스킹 문자</span>
  <span class="">여섯번째 마스킹 문자</span>
</a>
```

이 서비스에서는 암호 글자수 영역에 대체텍스트를 제공하고 있으나, 이전 사례와 같이 입력된 것과 그렇지 않은 것을 구분하는 텍스트 정보를 제공하지 않아, 현재 입력된 글자수를 확인하는 것은 불가능한 상태이다.

또한 이 사례에서도 기능이 없는 암호 글자수 영역에 a 요소를 사용하여 시각장애인 사용자가 해당 콘텐츠를 링크로 오해하는 문제를 야기할 수도 있다.

암호 키패드에 대체텍스트를 제공하지 않고, 키보드 초점의 접근을 막았으며, 현재 입력된 글자수를 확인할 수 없는 문제는 앞에서 언급한 간편 결제 서비스에서 모두 동일하게 나타나고 있어 장애인이 동일 유형의 서비스에 접근하는 것이 불가능한 상태이다.

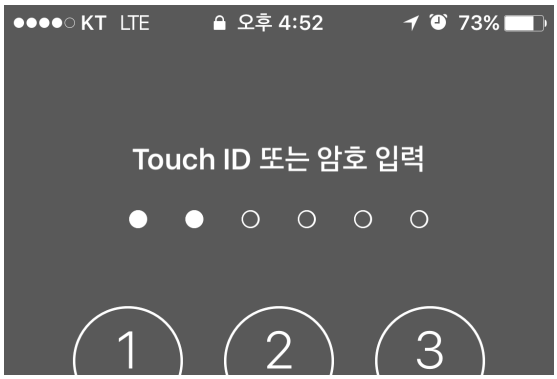
더구나 해당 간편결제 서비스들은 추가 적립 등의 혜택을 같이 제공하고 있어 서비스 접근성뿐만 아니라 사용자의 금전적인 손해까지 발생하고 있는 상황이다.



개선책

보안 키패드를 제공하는 암호 입력 화면에서 가장 심각한 이슈는 암호를 입력할 수 없다는 점이고 다음 이슈는 입력 글자수에 대한 확인이다

우선 현재 입력된 글자수를 확인할 수 있도록 대체텍스트를 제공해야 한다. 이런 유형의 콘텐츠는 모바일 운영체제의 암호입력 필드의 제공 방법을 참고하는 것이 좋다.



점으로 표시된 입력 글자수 표시 영역을 음성으로 확인해보면 "암호필드 6개 중 2개의 값 입력됨"과 같은 형태로 음성이 출력되고 있다.

이와 동일하지는 않더라도 유사한 형태로 간결하고 명확하게 현재 글자수와 전체 글자수 정보를 인식할 수 있도록 하는 것이 적절할 것이다.

웹에서 제공하는 가상 키패드는 기계적인 방식으로 접근하는 것을 차단하는 형태로 구현되어 있다. 하지만 시각장애인은 보조기술을 통해 콘텐츠에 접근해야 하기 때문에, 시각장애인을 위한 접근성을 제공하는 것이 현실적으로 불가능한 상태이다.

기술적으로 불가능하다고 해서 장애인이 보안 키패드를 이용할 수 없는 이슈를 그대로 방치한다면 장애가 있는 사용자는 해당 서비스를 전혀 이용할 수 없는 상태로 방치하는 것과 다르지 않다.

결국 장애를 이유로 서비스를 사용할 권리를 박탈하고 차별하는 행위로 이는 장애인차별금지법의 취지에 반하는 것이 된다.

이 문제에 대한 해결방안을 장애인도 차별 없이 사용 가능한 서비스를 디자인한다는 관점에서 생각해보면, 기술적 측면이 아니라 서비스의 이용 가능성을 고민하는 정책적인 면에서 새롭게 접근하는 것이 더 바람직한 방향이 될 것이다.

보안 키패드는 사용자를 검증하기 위한 인증수단의 한 종류이며 일정한 보안 수준을 보장한다. 보안 키패드 외에도 동등한 보안 수준을 보장하는 다른 인증수단을 함께 제공한다면 사용자는 자신에게 적합한 인증수단을 선택적으로 이용할 수 있게 된다.

이 때 함께 제공되는 또 다른 인증수단(또는 그 조합)은 특히 장애인의 접근성을 보장하는 수단으로 선택되어야 한다. 보안 정책과 인증수단 제공을 다변화하여 장애가 있는 사용자도 서비스 이용에 문제가 없도록 하는 것이 가장 적절한 개선방안이 될 것이다.

대체수단을 함께 제공하는 키패드

개인 뱅킹 서비스에서 암호와 같이 입력 값의 보호가 필요한 입력필드는 가상 키패드를 사용하도록 제공하고 있다. 이런 가상 키패드 또한 이전의 사례와 마찬가지로 스크린리더 사용자와 키보드 사용자가 사용할 수 없는 형태로 구현되어 동일한 접근성 이슈가 존재한다.

하지만 사용자의 입력 값을 보호하는 방법으로 가상 키패드만 사용하는 것은 아니다. 금융분야에서는 사용자의 입력 값 보호에 가상 키패드와, 키보드 보안프로그램을 함께 제공하여 사용자 환경에 따라 적절한 수단을 선택적으로 사용할 수 있도록 제공하고 있다.

이것은 단순히 기술적으로만 접근하지 않고, 보안이슈와 함께 사용자 환경을 고려한 정책으로 인한 것이다.

만약 사용자의 PC가 키보드 보안 프로그램이 설치된 환경이라면, 보안이 필요한 입력양식에서 가상 키패드를 해제하고 일반 키보드로 편집 창에 직접 입력 할 수 있다.

이러한 정책적인 접근을 통해 보안 키패드를 이용할 수 없는 사용자라 하더라도 필요한 정보를 입력하는데 전혀 방해를 받지 않는 환경이 제공되었다.

이런 사용 환경은 단순히 정보를 입력할 수 있다는 하나의 기능적인 측면이 아닌 서비스 전체를 이용할 수 있는 서비스 접근성을 보장하게 되는 큰 변화를 만들어낸다.

하지만, 키보드 보안프로그램을 설치하는 것으로 사용자 입력 값의 보호와 관련된 모든 접근성 이슈가 해결되는 것은 아니다.

먼저, 키보드 입력 값의 인식에 대한 문제를 살펴보자.

키보드 보안 프로그램은 키보드 입력 값을 모두 암호화하므로, 프로그램적으로는 어떤 정보가 입력된 것인지 알 수 없다.
결국 키보드 입력 값을 확인하여 음성 출력하는 스크린리더는 현재 입력하고 있는 글자를 정확히 읽어주는 것이 불가능하다.

예를 들어 “abcd1234”를 입력하더라도 스크린리더는 “aaaa1111”과 같은 형태의 음성만 들리게 되어 내가 어떤 글자를 입력하고 있는지 명확히 구분할 수 없다.

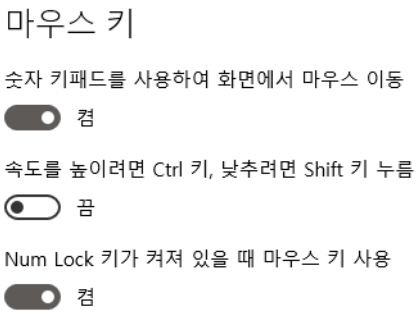
물론 입력 값을 시각적으로 구분할 수 없도록 마스킹하는 경우라면 비장애인과 크게 다르지 않은 상황이기 때문에 접근성 이슈가 아니라고 할 수 있다.

하지만 시각적으로 마스킹되지 않는 입력필드에 키보드 보안프로그램을 적용한다면 시각장애인은 비장애인과 달리 잘못된 정보(a와 1로 구성된)만을 인식하게 되므로 주의가 필요하다.

키보드 보안 프로그램은 필요한 영역에만 선택적으로 적용하여 시각장애가 있는 사용자가 상대적인 정보의 차이를 겪지 않도록 해야 한다.

이번에는 보조 기술의 사용을 제한한 경우를 살펴보자.

마우스 사용이 어려운 사용자들은 이를 보완하기 위해 다양한 보조기술을 활용한다. 그 중 가장 기본적인 수단은 윈도우에서 기본으로 제공하는 접근성 기능이다.



접근성 기능 중에서도 마우스 키는 별도의 하드웨어 장치를 필요로 하지 않기 때문에 마우스 대신 가장 쉽게 사용할 수 있는 보조기술이다.

이 기능은 키보드의 숫자패드를 이용하여 마우스 포인터를 상하좌우로 조작하기 때문에 키보드 만으로도 마우스 포인터를 조작할 수 있게 된다.

하지만 키보드 보안 프로그램을 설치하는 경우 해당 사이트에서는 마우스 키 기능을 활용한 마우스 조작이 불가능해진다.

편리하게 웹을 활용할 수 있는 기능이 있지만 사용할 수 없게 되어 장애 환경에 또 다른 장애를 추가로 야기하는 이슈가 발생한다.

그러므로 키보드보안 프로그램과 같이 운영체제의 접근성 기능을 제한할 가능성이 있는 프로그램을 사용해야 할 때는 주의가 필요하며, 반드시 필요한 영역에서만 선택적으로 적용하되, 다른 대체수단을 함께 제공하는 것이 바람직하다.

그렇다면 사용자 환경을 고려한 정책은 무엇일까?

보안 키패드와 키보드 보안 프로그램의 선택적인 사용 환경에서 마우스 키 기능 사용자에게는 키보드 보안 프로그램 설치 없이 보안 키패드를 사용하는 것이 더 편리한 결과를 나을 수 있다.

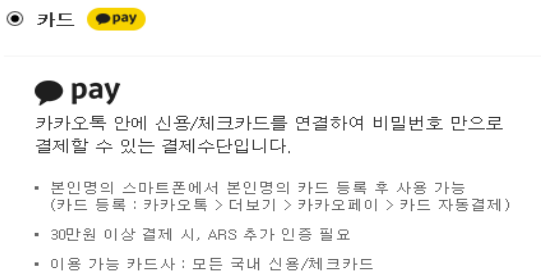
이는 키보드 보안 프로그램 설치를 통해 스크린리더 사용자에게 접근성을 보장하는 것과는 반대되는 상황이다.

이런 점에서 볼 때 보안에 대한 정책은 하나의 절대적인 수단을 제공하는 것이 아니라 사용자 환경에 맞게 선택하여 적용할 수 있는 여러 수단을 함께 제공하는 것이 가장 바람직하다.

모바일 애플리케이션을 이용한 대체수단

보안 키패드를 대체할 수 있는 보안 수단이 키보드 보안 프로그램뿐 일까?

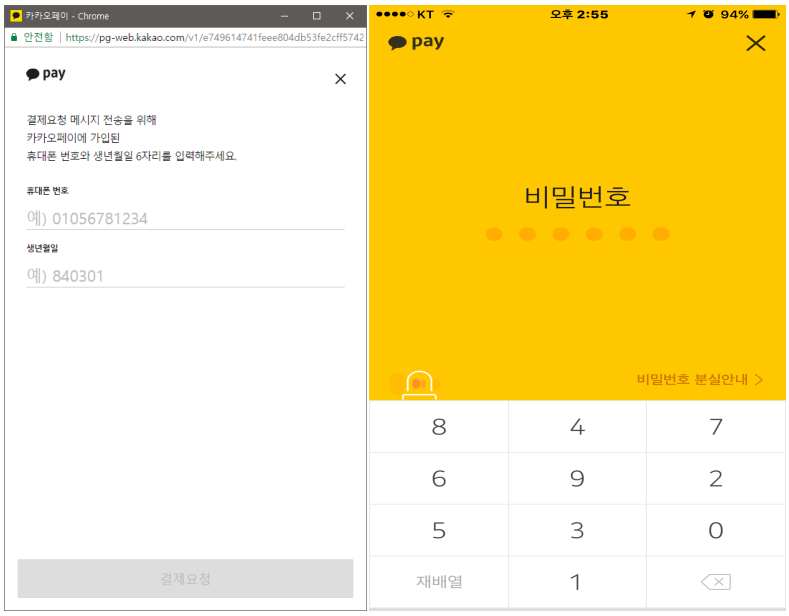
PC와 모바일에서 결제 서비스를 제공하는 다음의 간편결제 서비스 사례를 통해 또 다른 대체 수단 사용이 가능함을 확인할 수 있다.



타 결제서비스에서 암호입력 화면이 바로 나오는 것과 달리 해당 사례는 휴대폰 번호 입력 화면이 나타나고, PC 환경에서 결제를 요청했지만 모바일 애플리케이션의 결제수단과의 연동을 통한 서비스를 제공하고 있다.

서비스 초기에는 다른 간편결제와 동일하게 웹에서 가상 키패드로 암호를 입력하는 형태로 서비스 되었으나, PC 웹 환경에서는 가상 키패드의 기술적인 이슈로 접근성을 보장하기 어렵다는 이슈가 발견되었다.

그 후 모바일 애플리케이션 환경에서는 접근성을 보장할 수 있는 것을 확인하고 모바일 애플리케이션을 통해 암호 입력을 받도록 제공하고 있다.



PC 웹에서의 이슈를 모바일로 확장하여 해결하는 것이 과연 옳은 것이냐는 의문이 들기도 한다. 하지만 PC웹 환경에서 합리적인 대안을 찾기 어렵다면 접근성을 보장할 수 있는 다른 형태의 이용 수단을 제공하는 것도 최선은 아니지만 차선책으로 생각할 수 있다.

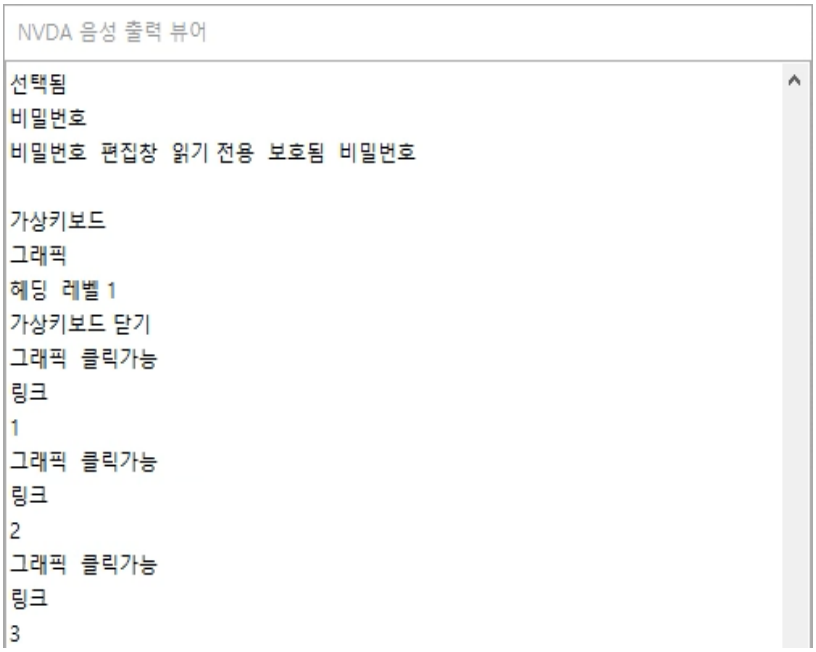
HTML로 구현된 가상 키보드

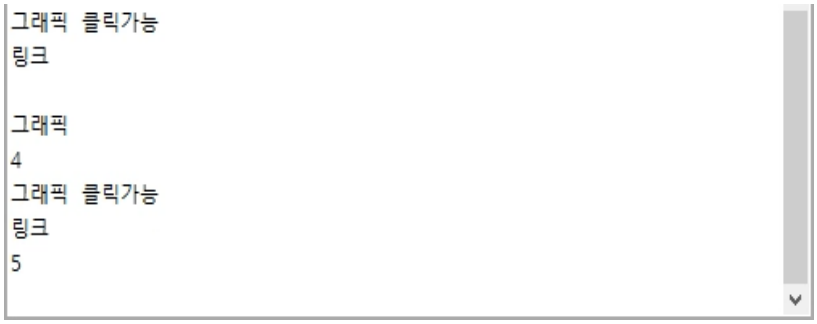
다음은 앞서 살펴 본 접근성 문제 중 일부가 개선 된 가상키보드로, ActiveX 나 플러그인 없이 웹 기술로만 구현된 사례이다.

어떤 점이 개선 되었는지를 다음 재연 절차에 따라 NVDA를 통해 알아보도록 하자.

재연절차(NVDA)

아래 방향키를 이용하여 계속 탐색





음성 출력과 같이 "가상 키보드"라는 제목을 통해 현재 접근된 영역부터가 가상 키보드 임을 알 수 있고, 각 키의 이름이 명확하게 읽혀지고 있음을 알 수 있다.

해당 부분의 마크업을 살펴보면, 다음과 같이 되어 있다.

```
<div>
  <a href="javascript:void(0)">
    <img src="btn_num_00.png" alt="1">
  </a>
  <a href="javascript:void(0)">
    <img src="btn_num_01.png" alt="2">
  </a>
  <a href="javascript:void(0)">
    <img src="btn_num_02.png" alt="3">
  </a>
  <img src="btn_blank.png" alt=" ">
  <a href="javascript:void(0)">
    <img src="btn_num_03.png" alt="4">
  </a>
  <a href="javascript:void(0)">
    <img src="btn_num_04.png" alt="5">
  </a>
  <a href="javascript:void(0)">
    <img src="btn_num_05.png" alt="6">
  </a>
  <a href="javascript:void(0)">
```

```
    <img src="btn_num_06.png" alt="7">
  </a>
  <img src="btn_blank.png" alt=" ">
  <a href="javascript:void(0)">
    <img src="btn_num_07.png" alt="8">
  </a>
  <a href="javascript:void(0)">
    <img src="btn_num_08.png" alt="9">
  </a>
  <a href="javascript:void(0)">
    <img src="btn_num_09.png" alt="0">
  </a>
</div>
```

모두 초점을 얻을 수 있도록 a 요소로 마크업 되어 있고, 각 키에 해당하는 이미지는 키가 가진 텍스트를 읽을 수 있도록 alt 속성에 해당 키 값을 가지고 있다.

영문과 한글이 공존해 있는 키의 경우를 살펴보면 마크업이 다음과 같이 되어 있는 것을 볼 수 있다.

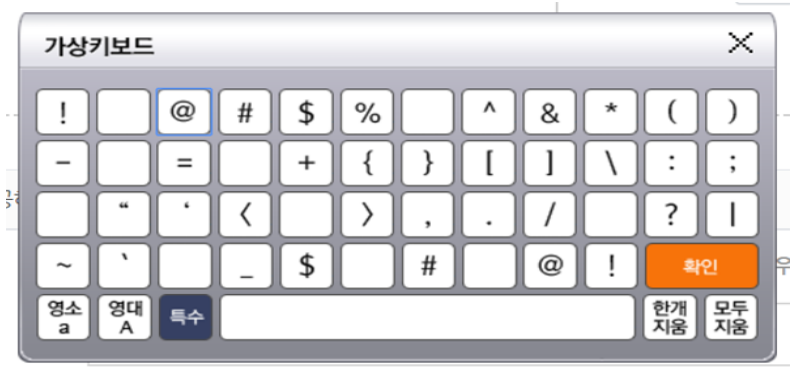
```
<div>
    <a href="javascript:void(0)">
        <img src="btn_lower_00.png" alt="q, 비읍">
    </a>
    <a href="javascript:void(0)">
        <img src="btn_lower_01.png" alt="w, 지읒">
    </a>
    <img src="btn_blank.png" alt=" ">
    <a href="javascript:void(0)">
        <img src="btn_lower_02.png" alt="e, 디귿">
    </a>
    <a href="javascript:void(0)">
        <img src="btn_lower_03.png" alt="r, 기역">
```

```
    </a>
    <a href="javascript:void(0)">
        <img src="btn_lower_04.png" alt="t, 시옫">
    </a>
    <a href="javascript:void(0)">
        <img src="btn_lower_05.png" alt="y, 요">
    </a>
    <img src="btn_blank.png" alt=" ">
    <a href="javascript:void(0)">
        <img src="btn_lower_06.png" alt="u, 여">
    </a>
    <a href="javascript:void(0)">
        <img src="btn_lower_07.png" alt="i, 야">
    </a>
    <a href="javascript:void(0)">
        <img src="btn_lower_08.png" alt="o, 애">
    </a>
    <a href="javascript:void(0)">
        <img src="btn_lower_09.png" alt="p, 에">
    </a>
</div>
```

각 키 마다 영문자와 한글을 모두 읽어주기 때문에 현재 명확하게 어느 키에 접근해 있는지를 인식하기가 쉽다.

많은 가상 키보드들이 가지고 있는 접근성에 대한 큰 문제점 중의 하나가 현재 접근 중인 키가 어떤 키인지를 알 수 없다는 점인데 반하여, 이 사례에서는 각 키에 대한 접근이 쉬울뿐더러 인식하는데 문제가 없어 보인다.

또한 키보드 만으로 접근 할 경우, 다음 그림과 같이 각 키 별로 초점을 얻는 것을 볼 수 있다.



때문에 현재 어느 키에 접근 중인지 파악이 가능하므로 키보드 사용자의 경우에도 무리 없는 사용을 기대할 수 있다.

다만 접근성에 대한 몇 가지 문제점을 여전히 가지고 있는데,

첫 번째 사진을 살펴보면 보안 상의 이유로 RCS를 통해 사용자가 선택하는 키를 캡쳐하는 것을 막기 위해 미러링 되는 두 개의 마우스 포인터가 제공되고 있는 것을 볼 수 있다. 하지만, 미러링 되는 마우스는 현재 접근 중인 키보드를 파악하는데 종종 혼란을 일으키기도 하며, 이는 인지 장애자에게 서비스를 사용하는데 어려움을 유발한다.

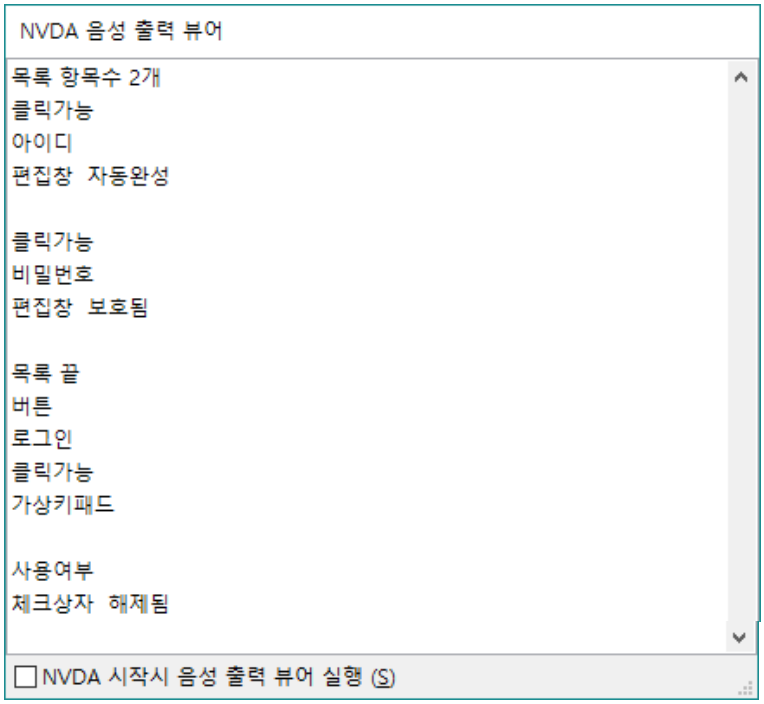
꼭 인지 장애가 아니더라도, 장애가 없는 사용자 역시도 잠시 한 눈을 팔고 다시 마우스를 사용하려면 현재 마우스의 실제 위치를 알아채기 위해서는 몇 번의 움직임의 시도가 이루어져야 확인 할 수 있는 어려움이 뒤따르게 된다.

또한, 가상 키보드 사용 여부의 체크상자가 입력상자 이후에 위치하고 있기 때문에, 사용자는 입력상자에 키를 입력한 이후에야 가상 키보드를 사용할 수 있음을 인지할 수 있다.

해당 부분을 NVDA로 읽어보면 다음과 같은 결과를 출력하게 된다.

재연 절차(NVDA)

아래 방향키를 이용하여 계속 탐색



일반적으로 편집 창에 접근하게 되면 이후를 다 읽어본 이후에 편집하기보다는 접근 즉시 값을 입력하게 될 터이다.

이 순서로 접근을 하게 되는 사용자는 실질적으로 로그인 버튼 이후에 무언가 입력과 관계된 추가적인 컨트롤이 있다는 것을 인지하기는 어렵기 때문에 보안 키보드를 사용하여 자신의 정보를 보호할 수 있는 수단이 있다는 것을 모르고 지나치기 쉽다.

그 외에도 스크린리더로 접근 시 레이어로 제공된 가상 키보드 외 다른 콘텐츠 영역에도 접근이 된다는 점도 문제이다.

가상 키보드가 화면 상에 표시되고 가상 키보드 패널이 초점을 얻을 때, 스크린리더가 브라우즈 모드(가상커서 모드)로 접근이 되기 때문에, 사용자는 포커스 모드에서 사용하듯 Tab키를 이용하여 탐색하지 않고 방향키를 이용하여 접근하게 된다.

이 경우, 가상 키보드가 열려 있는 상태에서 문서 탐색이 계속 이루어지는 문제가 발생되게 된다.

거론한 문제점이 해결 된다면, 접근 가능한 가상 키보드로서 사용자가 충분히 사용할 수 있지 않을까 기대된다.

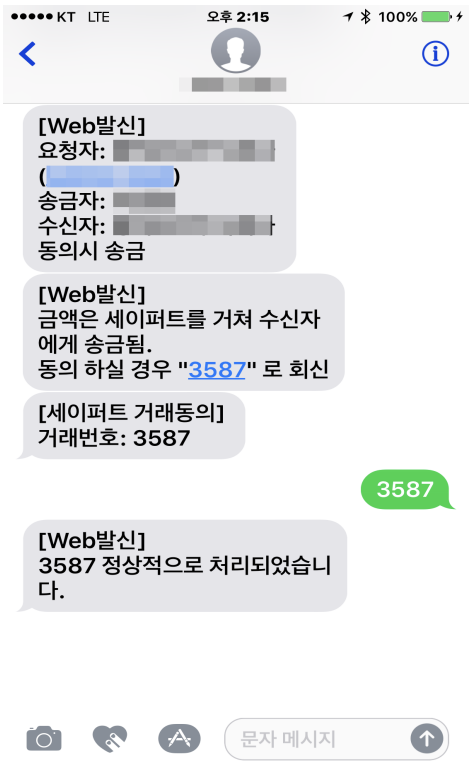
보안 키보드 대체 수단

앞의 보안 키패드 사례에서 살펴본 것처럼 단일 보안수단을 제공하여 발생하는 접근성 이슈를 대체수단을 활용하는 정책으로 해결할 수 있다는 것이 확인되었다.

그렇다면 현재 서비스에서 적용되고 있는 대체수단들에는 어떤 것들이 있는지 알아보자.

먼저, 사용자의 확인을 위해 페이지 상에서 어떤 값을 입력하는 것이 아니라, SMS를 통해 인증하는 사례를 살펴보자.

서비스 내에 예치 되어 있는 금액 중 일부를 사용하기 위해 다음 스크린 샷과 같이 사용자가 "동의함"이라는 키워드를 작성하고 버튼을 클릭하면 사용자의 핸드폰으로 SMS가 전송되고, 전송 받은 SMS에 기재되어 있는 특정한 번호를 회신하기만 하면 된다.



입력하기까지 제한 시간도 없고, SMS를 전송 받을 휴대폰만 있다면 모두 어려움 없이 사용이 가능하다.

지연이체 대체 수단

이 방법은 착오 송금 또는 금융 사기를 막기 위한 것으로 은행계좌에 있는 돈을 찾거나 보낼 때 일정 시간 '지연' 시켜 처리한다. 이 때 지연 시간이 금융 사기범들에게 넘어가는 돈을 지키는 역할을 한다.

국내 모든 금융기관에서는 2015년 10월 16일부터 "지연이체" 제도를 도입 시행하고 있다. 다만 신청자에 한해 이 서비스를 이용할 수 있고 기본적으로 지연이체가 사용되고 있지 않다.
반면 중국에서는 급증하는 전화 금융사기(보이스 피싱) 방지를 목적으로 현금 자동입출금기(ATM)를 이용한 송금 때 24시간 후에 이체되도록 하는 방안을 대다수 금융기관에 도입(2016년 12월)하였다. 은행 간 계좌 이체는 실시간 또는 길어야 2시간 이내 이루어 졌는데 기존의 제도를 대폭 조정한 것이다.

중국뿐 아니라 해외에서도 지연이체 방법을 사용하고 있는데 미국, 일본, 홍콩, 독일 사례를 살펴 보면 나라 별 금융기관마다 조금씩 다르긴 하지만 금융 사기 문제를 해결하기 위한 방책으로 지연 이체 처리하는 것을 알 수 있다.

구분	주요내용

미국	자행 이체는 실시간으로 가능. 타행 이체는 경우에 따라 1~3일 소요. 소요시간에 따라 수수료 차등 적용 (타행 이체 시 익일 기준 10달러 부과)
일본	자행 이체는 평일 19시 이전. 타행 이체는 평일 15시 이전 이체한 경우에 한해 당일 이체 가능. 휴일의 경우 다음 영업 일에 이체. 단, 이체 자금 및 수수료는 접수 시점에 인출.
홍콩	인터넷 뱅킹으로 이체 신청할 경우, 금융 직원 단말기에서 신청 내용을 불러와 건 별로 승인하는 절차 수행 (수신 계좌 유효 여부, 블랙리스트 계좌 여부 체크) 실 송금에는 몇 시간 지연 처리 한도 지정한 경우에만 이체 가능.
독일	자행/타행 이체 모두 1~2일 소요

지연이체 방법을 사용한다면 이체 과정에서 사용자가 입력한 내용을 보안 목적으로 차단하는 대신, 금융 기관 인증(홍콩 사례 참고)을 통해 금융 사고를 사전에 막을 수 있다.

알아 본 해외 사례와 같이 지연이체 제도를 별도의 신청 과정 없이 일관되게 모든 금융 서비스에 적용한다면 어떨까? 물론 기존의 실시간 이체에 익숙한 사용자가 불편해 할 수 있으나, 도입 취지와 목적을 충분히 수긍되도록 어필한다면 사용자도 받아들일 것이다.

이를 통해 금융 사고에 대한 보안 책임을 금융 기관에서 지게 되므로 책임의식을 강화하는 결과를 가져올 수 있고, 나아가 IT 기술 등을 접목한 다양한 금융 사기 피해 예방을 위한 제도나 서비스 개발에 보다 적극적으로 투자하게 될 것이다.

뿐만 아니라, 키보드 입력 단계에서 보안을 이유로 입력 정보를 차단할 필요가 없어져 접근성 문제를 보안과 결부하지 않고 해결할 수 있다.

시각장애인은 자신이 입력한 키 정보를 듣는데 문제가 발생 하지 않을 것이고, 입력해야 할 키 개수도 사전에 들어 알 수 있으므로 접근성이 크게 향상될 수 있다.

아래 코드를 살펴보면 각 input 요소마다 레이블(label)을 제공함으로 시각장애인 사용자가 스크린리더를 통해 해당 요소에 접근할 때 몇 번째 비밀번호 입력 창인지 듣고 인지하는데 어려움이 없다.

뿐만 아니라 사용자가 입력할 때마다 실시간으로 입력된 상태를 읽어주므로(WAI-ARIA Live Region 사용) 입력해야 할 비밀번호 개수와 입력된 개수를 확인 가능하다.

```
<h2 class="h2 center">비밀번호를 입력하세요</h2>
<div class="pw-input-group" role="group">
  <input type="password" class="inputted"
         required aria-label="첫번째 비밀번호">
  <input type="password" class="inputted"
         required aria-label="두번째 비밀번호">
  <input type="password" required aria-label="세번째 비밀번호">
  <input type="password" required aria-label="네번째 비밀번호">
  <input type="password" required aria-label="다섯번째 비밀번호">
  <input type="password" required aria-label="여섯번째 비밀번호">
   <span aria-live="polite">
     비밀번호 입력 6개 중, 2개 값 입력됨.
   </span>
</div>
```

일반 사용자 입장에서도 금융 사고에 대한 책임을 사용자가 아닌, 금융기관이 가지므로 보다 편리하고 안전하게 금융 서비스를 이용하는 결과를 가져올 수 있다.

금융 기관의 경우 단기적으로는 보안 투자가 불가피 하겠지만, 장기적인 안목에서는 별도의 광고 비용을 들이지 않고도 우수하고 안전한 보안 대책 및 사고 후 처리 사례

를 고객에게 각인 시켜 줌에 따라 신뢰감을 가진 충성 고객을 크게 확보하는 효과를 기대할 수 있을 것이다.

1회용 액세스 토큰 대체 수단

1회용 액세스 토큰(One-time Access Token 또는 Static Token으로 불림)은 보안 키보드를 대체하는 합리적인 접근법으로 클라이언트와 서버에서 토큰을 확인하는 과정을 거쳐 인증하는 방법이다.

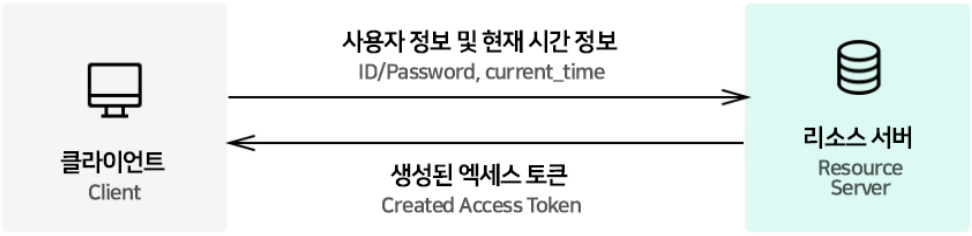
최신 인증 및 인증 솔루션은 프로토콜에 토큰 개념을 도입했다. 토큰은 사용자 권한 정보를 전달하는 특수하게 조작된 데이터 조각을 말한다. 다시 말해 토큰은 사용자 인증을 위한 정보 조각이다.
클라이언트 또는 서버에서 사용자 정보 토큰을 통해 인증 과정을 거친 후, 데이터 리소스에 접근할 권한을 부여한다.

액세스 토큰은 데이터 리소스에 직접 접근 가능한 정보를 담고 있다. 토큰 인증 과정을 설명하면 클라이언트가 데이터 리소스를 관리하는 서버에 액세스 토큰을 전달하면 해당 서버는 토큰에 포함된 정보를 확인한 후, 클라이언트에 데이터 리소스 접근 권한을 부여한다.

액세스 토큰은 만료 날짜를 제공할 수 있으므로 토큰 만료 시간(Expire Time)을 설정하여 1회용으로 사용할 수 있다.

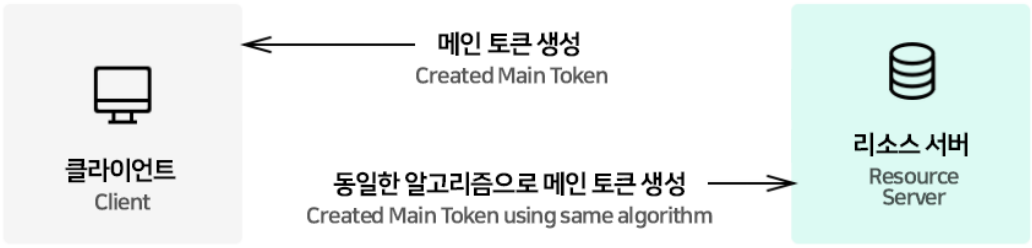
이해를 돕기 위해 클라이언트와 서버의 토큰을 사용한 보안 프로세스를 좀 더 상세히 풀어보자.

클라이언트는 사용자 계정 정보(ID/Password)와 current_time 변수 정보(서버의 데이터베이스, 클라이언트 스토리지에 모두 저장)를 API를 통해 서버에 토큰을 요청하면, 서버는 클라이언트의 입력을 해석하고 해시 토큰(예: 58f52c075aca5d3e07869598c4d66648)을 생성하고 이를 서버 측 데이터베이스에 저장하고, 클라이언트에 응답으로 전송한다.



클라이언트는 서버로부터 전송 받은 토큰을 클라이언트 스토리지(예: sessionStorage. 세션 스토리지는 세션이 만료되면 파기 된다)에 저장하고, 토큰 + 인증 요청 과정에서 보낸 current_time 변수를 사용하여 새 해시 토큰(main_token이라고 한다)을 생성한다.

서버 또한 동일한 알고리즘을 사용하여 작업을 수행하고 클라이언트와 일치하는 토큰을 만든다.



클라이언트가 서버 API를 통해 요청(Request) 할 때마다 생성된 메인 토큰(main_token)을 서버에 보내면, 서버는 서버에서 생성 된 메인 토큰과 클라이언트가 보낸 메인 토큰을 비교한다. 각 토큰이 일치하면 데이터 리소스에 접근 가능한 사용자임을 인증한다.

토큰을 이용한 인증 처리 시스템은 비교적 이해하기 쉬우며 구현하기 어렵지 않다. 뿐만 아니라 안정된 보안을 제공한다. 참고로 이 방법은 수 차례에 걸쳐 검증된 보안을 제공하는 시스템이다.

다만 위에서 설명한 프로세스에서 한가지 주의할 점은 액세스 토큰 생성 과정에서 사용자 정보와 함께 제공된 시간 정보(TTL)는 만료 시간이 1초에 가까울 정도로 짧아야 한다. (이메일의 경우 1초보다 오래 걸리기 때문)

TTL이란? Time to Live의 약자로 컴퓨터나 네트워크에서 데이터의 유효 기간을 나타내기 위한 방법이다. TTL은 계수기나 타임스탬프의 형태로 데이터에 포함되며, 정해진 유효기간이 지나면 데이터는 파기 된다.

1회용 액세스 토큰의 보안 프로세스를 살펴봤으니 정리해보자.

먼저 앞에서 다뤘던 만큼 보안 키보드를 사용하지 않았기에 접근성을 준수할 수 있는 것은 자명한 일이다. 사용자 인증 과정에 사용자 계정 정보와 함께 감춰진 1회용 액세스 토큰(1회 사용 후 자동 파기 됨)이 사용되기에 보안 또한 신뢰할 수 있다.

결론적으로 1회용 액세스 토큰은 사용자 편의와 사용성을 높이면서 접근성 또한 향상시키는 우수한 방법으로 볼 수 있다.

가상 키보드에 대한 제언

가상 키보드의 접근성 이슈는 단순히 가상 키보드를 사용하지 못한다는 점에서 끝나는 것이 아닌 대부분의 간편결제에서 가상 키보드를 보안요소로 사용함으로써 장애인이 결제 서비스를 사용할 수 없다는 점에서 큰 문제가 있다고 볼 수 있다.

하지만 현재 웹 기술로는 가상 키보드에 준하는 보안 문제를 해결할 수 없다. HTML 5 기반의 Crypto API가 인증 관련 문제를 해결할 수 있다고 하더라도 백신, 키보드 보안, 개인 방화벽 같은 보안 솔루션들은 HTML5로 구현이 어렵기 때문이다.

웹 표준 기술을 대체할 수 있는 다른 대안으로 1회용 액세스 토큰을 들 수 있다. 암호화된 1회용 토큰 정보를 제휴를 맺은 카드사나 은행에 생성 및 전송하고 이를 금융사에서 해석해서 결제하는 방식이다.

새로운 기술은 이 순간에도 계속 개발되고, 상상할 수 없는 새로운 보안 수단의 등장으로 지금과는 다른 새로운 접근성 이슈가 계속 나타날 것이다.

또한 단순히 기술적으로만 접근하면 해결할 수 없는 접근성 이슈도 계속 나올 수 있다.

생체 정보를 활용한 인증은 신체적 장애가 있는 사용자는 생체 정보를 인증 받는 과정이 쉽지 않고, 경우에 따라서는 생체 정보가 없는 경우까지 있다.

예를 들어 시각장애가 있는 사용자는 홍채인식을 위해 시선 고정을 할 수 없는 경우가 많고 심지어 인식이 불가능한 경우도 있다.

또 다른 예로 지문인증만 사용하는 서비스에서 지문이 지워지거나 손가락 절단 장애를 입은 사용자는 해당 서비스를 이용하는 것이 불가능하다.

이런 환경은 기술적으로 해결할 수 없는 접근성 이슈를 발생시킨다는 점에서 가상 키보드의 이슈와 매우 유사하다.

기술의 한계 또는 사용자의 한계라는 합리적인 것처럼 보이는 이유로 접근성 이슈를 계속 방치하는 것은 적절하지 않다.

가상 키보드와 같이 기술적 접근이 어려운 경우에는 보안수준을 고려한 정책적 접근을 통한 해결이 필요하다.

신체적 장애, 또는 사용환경의 제한 등으로 인해 기본으로 제공된 보안수단을 사용하지 못한다면, 대체할 수 있는 동등한 수준의 접근성 있는 보안수단(또는 조합)을 함께 제공하는 것이 기술적인 개선이 어려웠던 접근성 이슈를 보다 쉽게 해결하는 방법이 될 수 있다.

이런 정책적인 접근이 가능한 것은 금융감독원이 보안심의와 같은 의무사항을 폐지하고 자율적인 보안정책을 펼치고 있기 때문에 가능한 일이다.

이제는 일정 수준의 보안을 유지할 수 있다면 사용되는 수단이나 정책은 서비스 제공자가 자율적으로 선택할 수 있게 되었다.

궁극적으로 기술적인 해결 없이 보안 정책적인 측면에서의 접근이 필요한 이유는, 접근성의 목적은 현재 이슈가 되는 하나의 문제를 해결하는 것이 아니라, 장애가 있는 사용자라 하더라도 콘텐츠 또는 서비스를 동등한 수준으로 이용할 수 있도록 하는데 있기 때문이다.

보안키보드의 보안

같은 형태의 가상 키보드를 사용한다고 하더라도 웹 사이트에서 사용하는 것과 앱에서 사용하는 것의 뉘앙스가 다른 경우가 있다.

이는 보안에 대한 접근 방식이 다르기 때문인데 이 섹션에서는 그 부분에 대해서 다루어보도록 한다.

먼저 네이버에서 제공하는 네이버 페이 비밀번호 입력 부분의 HTML 코드이다.


```
<td>
  <a class="key" onclick="doClick('0')">
    <span class="number key1_1"></span>
  </a>
</td>
```

위 코드의 가장 큰 문제점은 해당 마크업 요소에 텍스트를 포함하고 있는 것이 존재하지 않아 시각장애인이 해당 사이트에 들어가서 스크린리더로 화면을 읽을 때 아무 내용도 들을 수 없다는 문제점을 가지고 있다.

하지만 이렇게 함으로써 보안이 정말 나아지고 있는가에 대한 근본적인 의문이 들어 코드를 좀 더 파헤쳐보았다.

우선 a 요소에서 doClick 이라는 함수를 호출하고 있는데 해당 함수가 하는 일을 간략하게 정리하면 다음과 같다.

doClick 함수의 인자로 숫자 값을 받는다.
authNo라는 id을 가진 input 요소를 찾아 그 값에 해당 숫자를 넣는다.
만약 authNo라는 id를 가진 input 요소가 6자인 경우 form을 submit 시켜준다.
이걸 코드로 보면 아래와 같다.

```
function doClick(obj) {
  var curUpw;

  curUpw = document.getElementById("authNo").value;
  document.getElementById("authNo").value = curUpw + obj;
}
```

이 코드가 실제로 보안적으로 아무런 문제가 없어 보일 수 있지만 실제로는 input hidden을 넣어둔 상태로 그저 해당 값에 숫자를 넣는 형태에 불과하기 때문에 이 값을 해킹하는 것은 그렇게 어렵지 않다.

하지만 가상 키보드가 마냥 보안 이슈를 해결하지 못하는 것은 아니다. 예를 들자면 위 스크린샷 에서 숫자 6을 클릭할 경우 doClick 이벤트에서 실제로 가져가는 숫자 값은 0이기 때문에 유저가 실제로 클릭한 것과 숫자 값이 다르기 때문이다.

예를 들어, 숫자 5자리를 입력한 후 authNo라는 id를 가진 input 요소를 살펴보면 아래와 같이 값을 가지고 있다.

```
<input type="hidden" id="authNo" value="32710">
```

위의 코드는 가상 키보드를 테스트할 때 12345를 입력하였을 때 나온 값이며 해당 값은 네이버의 가상키보드가 새로 열릴 때마다 갱신된다.

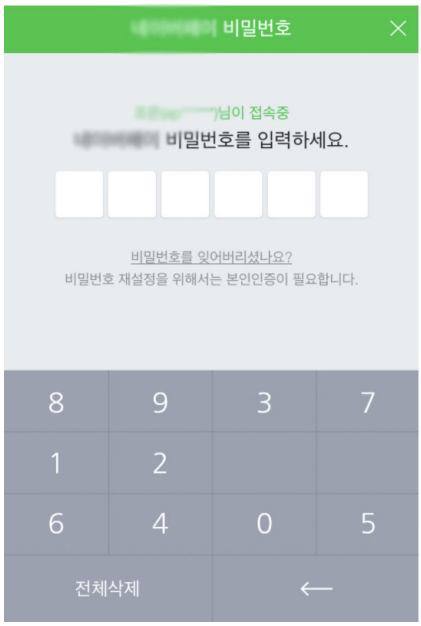
웹에서는 유저의 키보드 입력 값을 가져오는 게 쉬운 일이기 때문에 단순히 키보드 입력을 받게 된다면 비밀번호가 유출될 가능성이 있을 수 있다.

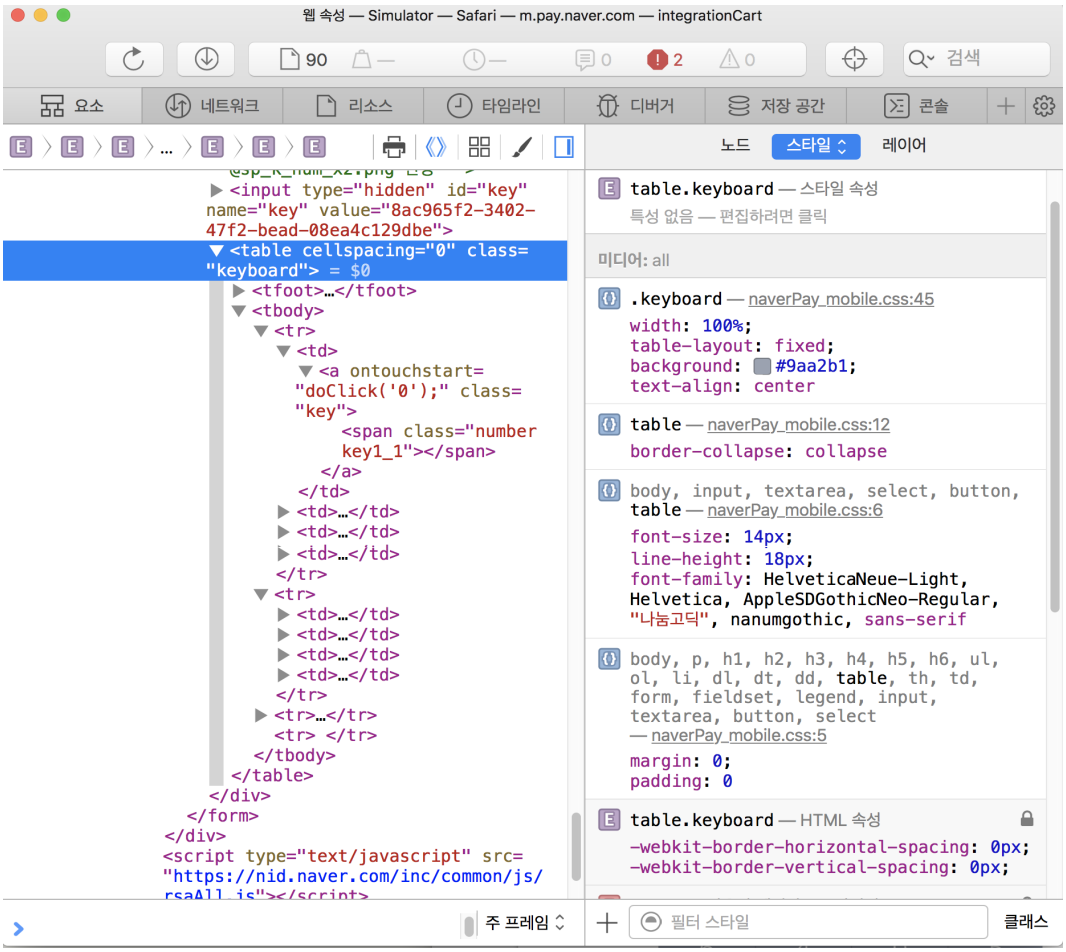
따라서 결제와 관련된 부분에서 키보드 보안을 하는 것은 어떻게 보면 당연한 일일 수 있다.

하지만 키보드 보안으로 인해 사용자가 해당 서비스를 사용하지 못한다면 키보드 보안 로직을 바꿀 필요가 있으리라 생각한다.
하지만 앱에서의 보안 방식은 웹에서 가져가는 그것과 많이 다를 수 있다. 실제로 앱에서 사용하는 가상 키보드는 스크린리더에서 그 값을 읽어주는 데 그렇게 할 수 있는 가장 큰 이유는 앱의 폐쇄성을 들 수 있다.

대표적으로 모바일 웹이라고 하더라도 웹 사이트는 개발자 도구를 실행함으로써 웹에서 바로 접근하는 것이 가능하며 만약 그렇게 한다고 한다면 유저가 입력하고 있는 값이 모두 노출되는 이슈가 발생할 수 있다.

예를 들어 iOS 사파리에서 아래와 같은 화면을 노출시키고 있다고 했을 때, Safari의 개발자 도구를 통해 어떻게 입력되고 있는 지 확인하는 것이 가능하다.





따라서 개방형 플랫폼인 웹에서는 조금 더 민감하게 해당 문제에 접근할 수 밖에 없지만 한편으로는 내가 개발자도구를 실행시킬 수 있었던 것은 어디까지나 내 기기까지의 한계로 다른 사람의 기기를 제어하는 것은 매우 어려운 일이다.

단적인 예로 PC에서는 크롬용 익스텐션을 하나만이라도 잘못 설치하면 바로 컴퓨터가 해킹될 수 있는 데에 반해 모바일에서는 크롬이나 Safari에 익스텐션 등을 설치해 바로 접근하는 방식을 사용할 수 없기 때문이다.

위의 사례는 PC 웹에서 사용하고 있는 키보드를 모바일에서 동일하게 사용하려고 하는 문제에서 비롯된 것으로 추정되며 PC와 모바일의 차이점은 스프라이트 이미지의 배율과 디자인 차이만 있었다.

만약 PC에서 가상키보드의 대체수단을 제공하기가 어렵다라고 한다면 모바일에서라도 올바른 HTML을 사용하여 해당 부분을 대체할 수 있으리라고 생각한다.

```
<td>
  <a class="key" onclick="doClick('0')">
    <span class="number key1_1"></span>
  </a>
</td>
```

다시 처음 살펴본 코드를 본다면, 위와 같이 작성되어있는 접근성을 지키지 않은 코드를 아래 표처럼 작성하여 더 올바른 형태의 HTML을 제공해줄 수 있으리라 생각한다.

```
<td>
  <button type="button" class="key" onclick="doClick('0')">
    <span class="number key1_1">8</span>
  </button>
</td>
```

만약에 위와 같은 형태로 제공하지 않으면 해당 문제는 이미지를 제공해주는 곳 (즉 서버)에서 클라이언트에 해당 숫자 값이 어떤 구조를 가지고 있는 지 알려주지 않았을 때 문제가 발생할 수 있다.

그렇다고 한다면 해당 문자열을 JSON 형태의 Ajax로 보내준다면 해당 문제를 해결할 수 있으리라 보여진다.
키보드 보안은 아주 어려운 문제 중 하나이며 많은 프론트엔드 개발자들이 골머리를 앓고 있는 영역이기도 하다.

PC 웹에서는 아직 존재하는 다양한 개방성 문제가 있기 때문에 시기가 적절하지 못할 수 있고 다른 인증 방식 (2 Factor 인증)을 사용하여 인증하는 것이 어떻게 보면 가장 안전한 방법일 수 있다.

그러나 모바일 웹에서는 PC에서 제공하는 그것과 달리 많이 폐쇄적으로 운영되고 있기 때문에 PC와 다른 접근방식을 가져갈 수 있을 것이라 생각한다.

참고자료

reCAPTCHA, Google
reCAPTCHA, Wikipedia
10초 리캡챠
당신이 인간임을 증명하시오. 이데일리 카드뉴스
캡챠의 개념과 안정성 분석 사례
복수의 이미지를 합성하여 사용하는 이미지 기반의 캡차와 이를 위한 안전한 운용 방법
캡차, 리캡차, 노캡차 리캡차
Limiting Automated Access
reCAPTCHA: Tough on Bots, Easy on Humans, Great on Mobile
수학식 캡챠
금융사 35곳 중 31곳 가상키보드, RCS에 뚫렸다.
전자금융감독규정시행세칙 (시행 2010년 8월 12일)
전자금융감독규정시행세칙 (시행 2012년 5월 29일)
모바일 대민서비스 구축 가이드(2016년 12월)
국방전자조달시스템 관리규정(시행 2016년 8월10일)