4-15 共有資源やキャッシュに関する問題

4.15.1 競合状態の脆弱性

競合状態（レース・コンデション）の脆弱性

共有資源として、複数のプロセスやスレッドから同時に利用している変数、共有メモリ、ファイル、データベースなどの排他制御が不十分な場合、競合状態（レース・コンデション）の脆弱性が発生します
※　Java のサーブレットクラスのインスタンス変数は共通資源

競合状態の脆弱性の影響

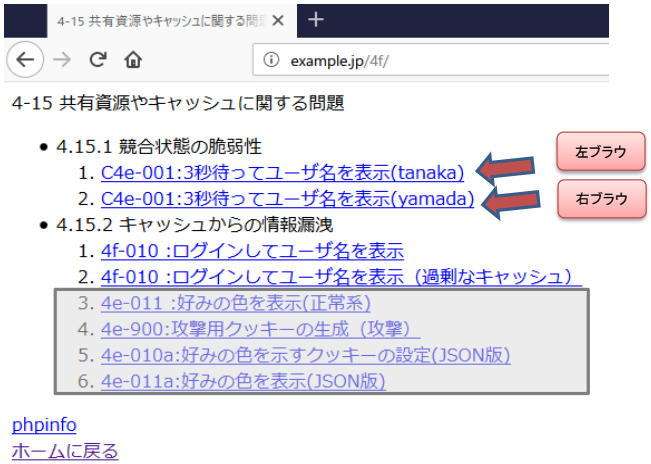
別人の個人情報などが画面に表示される（別人問題）
データベースの不整合、ファイル内容破壊

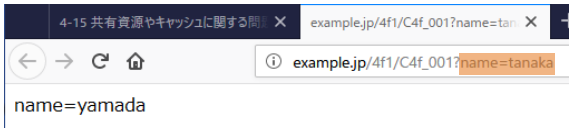
競合状態の脆弱性の対策

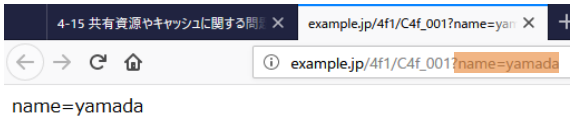
共有資源の利用を避ける	Javaの場合、変数name は、ローカル変数 String　name に変える　※JSPも似た表記
共有資源の排他制御を行う	Javaのマルチスレッド環境で排他制御を行うには、synchronized文やsynchronizedメソッドを使う ※　待ち合わせが発生するので、サービス妨害攻撃の要因になりえます ※　JavaのSingleThreadModelインターフェイスはシングルスレッドで動くため、インスタンス変数をロックせずに済みますが、 Srevlet2.4以降では非推奨になっているので、今後は使用しないほうが無難

C4e-001:3秒待ってユーザ名を表示(tanaka)
C4e-001:3秒待ってユーザ名を表示(yamada)

【ブラウザ】







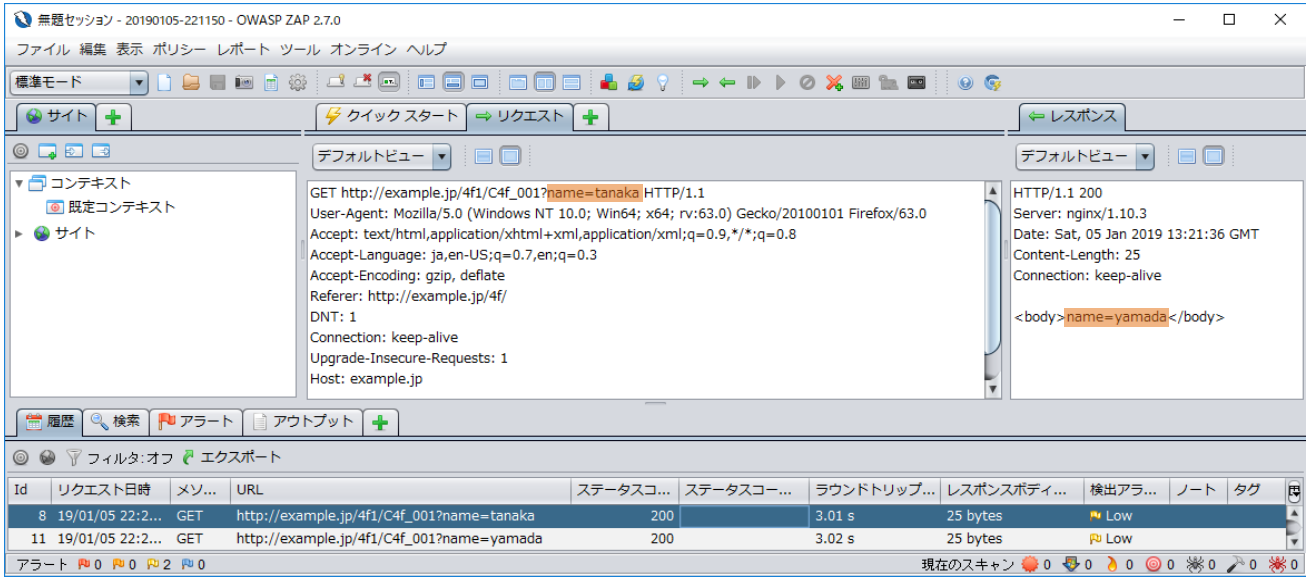
【サーバ： 4f/C4f_001.java 】

/var/www/html/4f/C4f_001.java - wasbook@example.jp - エディタ - WinSCP

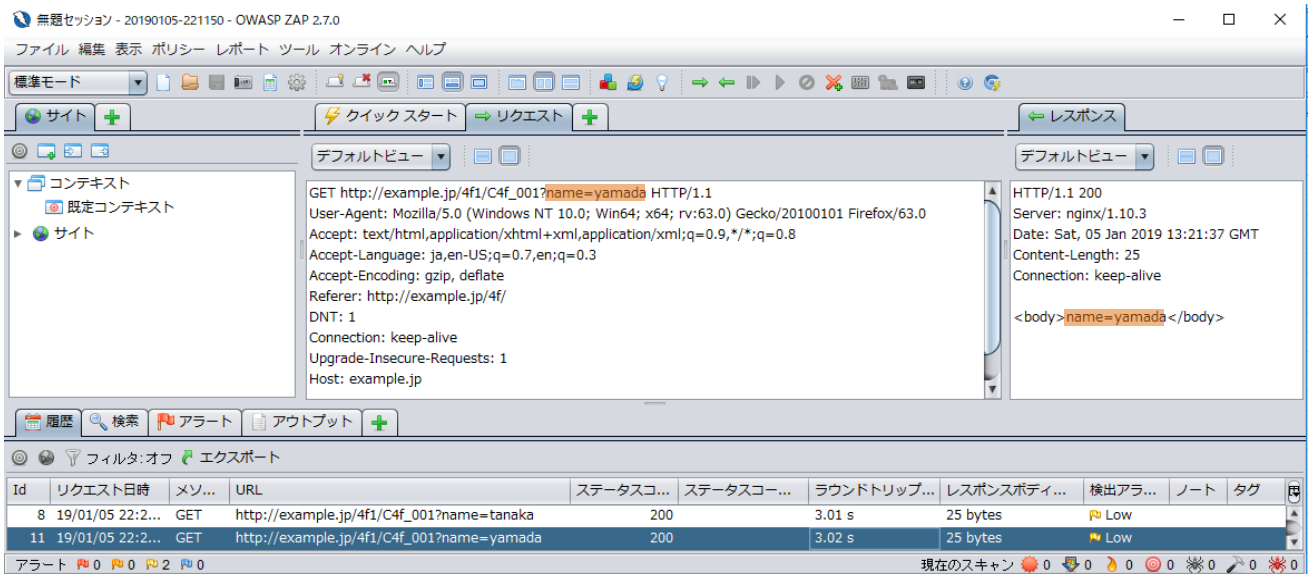
```
import java.io.*;
import javax.servlet.http.*;
import org.apache.commons.lang3.StringEscapeUtils;

public class C4f_001 extends HttpServlet {
  String name; // インスタンス変数として宣言
  protected void doGet(HttpServletRequest req,
                       HttpServletResponse res)
       throws IOException {
    PrintWriter out = res.getWriter();
    out.print("<body>name=");
    try {
      name = req.getParameter("name"); // クエリストリングname
      Thread.sleep(3000); // 3秒待つ（時間のかかる処理のつもり）
      out.print(StringEscapeUtils.escapeHtml4(name)); // ユーザ名の表示
      // out.print(name); // ユーザ名の表示
    } catch (InterruptedException e) {
      out.println(e);
    }
    out.println("</body>");
    out.close();
  }
}
```

【ブラウザ→サーバ： リクエスト 4f/C4f_001.java（左ブラウザ） → レスポンス 】



【ブラウザ→サーバ： リクエスト 4f/C4f_001.java（右ブラウザ） → レスポンス 】



4.15.2 キャッシュからの情報漏洩

キャッシュからの情報漏洩の脆弱性

Webアプリケーションは処理高速化や負荷軽減のために、多くの箇所でキャッシュを使いますが、キャッシュが過剰に働くと個人情報漏洩などの原因になる場合があります

キャッシュからの情報漏洩の影響

別人の個人情報などが画面に表示される（別人問題）

キャッシュからの情報漏洩の原因

アプリケーション側のキャッシュ機能不備

Cache-Controlヘッダのキャッシュ方法の種類（ディレクティブ）

ディレクティブ	説明
no-store	全くキャッシュしない
no-cache	キャッシュの有効性を毎回、サーバに確認する
private	１人のユーザのためのキャッシュを許可する。典型的にはブラウザのキャッシュは許可するが、キャッシュサーバのキャッシュは許可しない。
public	全てのキャッシュを保存する。
must-revalidate	リソースを使う場合にキャッシュが無効化していないことを確認する
max-age	キャッシュの有効期限を設定する（秒数）

PHPの session_cache_limiter関数は「public」指定で、以下ヘッダを返す

```
Expires:  session.cache_expireだけ未来の日時
Cache-Control:  piblic,max-age=session.cache_expireを分⇒秒の数値
```

キャッシュサーバ側のキャッシュ機能不備

```
proxy_ignore_headers Cache-Control Expires Set-Cookie;
```

（Cache-Control Expires Set-Cookie）を無視する。アプリ側の設定を無視して、キャッシュサーバ側の設定を返す。（/etc/nginx/conf.d/defaultなどの設定）

キャッシュからの情報漏洩の対策

リバースプロキシの他に、CDNやロードバランサもキャッシュ機能があるので、インフラ担当者との調整が必要

アプリケーション側の対策

ブラウザやキャッシュサーバの挙動を検討し、適切と思われる設定は以下である。Pragma:no-cache は古いソフト用伝統的記法だが、絶対安全ではない。

```
Cache-Control:  private,no-store,no-cache,must-revalidate
Pragma: no-cache
```

PHP対策（ただし、デフォルト設定）

```
session_cache_limiter('nocache');
```

→ レスポンス

```
Cache-Control:  no-store,no-cache,must-revalidate,post-check=0,
pre-check=0
Pragma: no-cache
```

※ private 設定されないが、通常はこれで問題ない。
キャッシュサーバやCDN側で必要な場合は別途、header関数で設定するとよい。

キャッシュサーバ側のキャッシュ対策

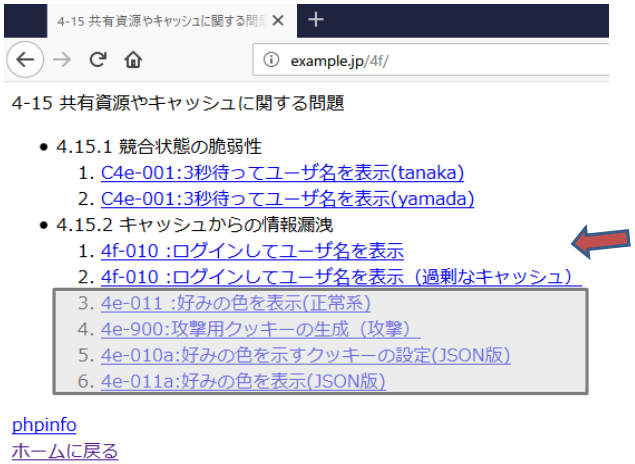
```
proxy_ignore_headers Cache-Control Expires Set-Cookie;    # この行を削除
```

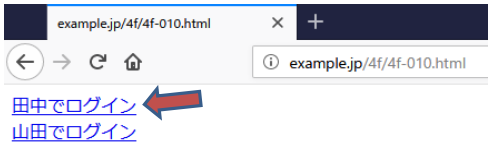
URLに乱数を付与する対策（キャッシュバスタ）

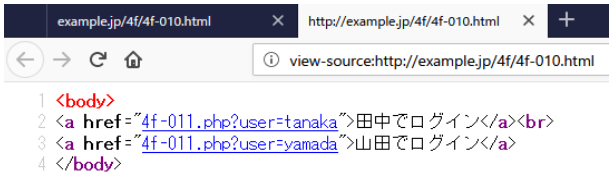
キャッシュはURL毎に別リソースとして扱われるので、URLを強制的に変えることで、キャッシュ経由の情報漏洩を防止できます
ただし、キャッシュ用ストレージの無駄使いであり、URLが知られると情報漏洩につながります

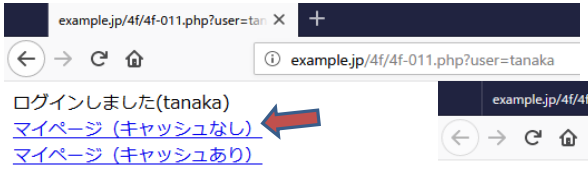
4f-010 :ログインしてユーザ名を表示

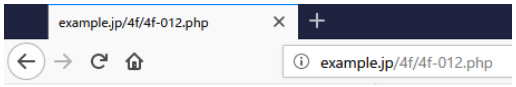
【ブラウザ】

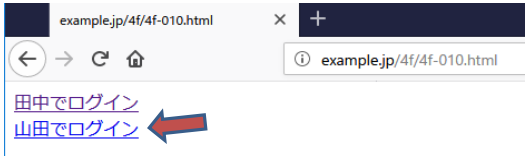


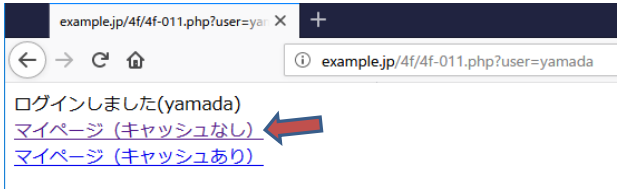


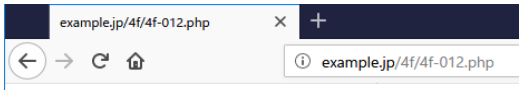












【サーバ： 4f/4f_010.html 】

/var/www/html/4f/4f-010.html - wasbook@example.jp - エディタ - WinSCP

```
<body>
<a href="4f-011.php?user=tanaka">田中でログイン</a><br>
<a href="4f-011.php?user=yamada">山田でログイン</a>
</body>
```

【サーバ： 4f/4f_011.php 】

/var/www/html/4f/4f-011.php - wasbook@example.jp - エディタ - WinSCP

```
<body><?php
  $user = $_GET['user'];
  if ($user === 'tanaka' || $user === 'yamada') {
    session_start();
    session_regenerate_id(true);
    $_SESSION['user'] = $user;
    echo 'ログインしました(' . htmlspecialchars($user) . ')<br>';
    echo '<a href="4f-012.php">マイページ（キャッシュなし）</a><br>';
    echo '<a href="4f-012a.php">マイページ（キャッシュあり）</a>';
  } else {
    echo 'ユーザ名が違います';
  }
?>
</body>
```

【サーバ： 4f/4f_012.php 】

/var/www/html/4f/4f-012.php - wasbook@example.jp - エディタ - WinSCP

```
<body><?php
  session_start();
  if (empty($_SESSION['user'])) {
    die("ログインしていません");
  }
  echo "ユーザ {$_SESSION['user']} でログイン中です";
?></body>
```

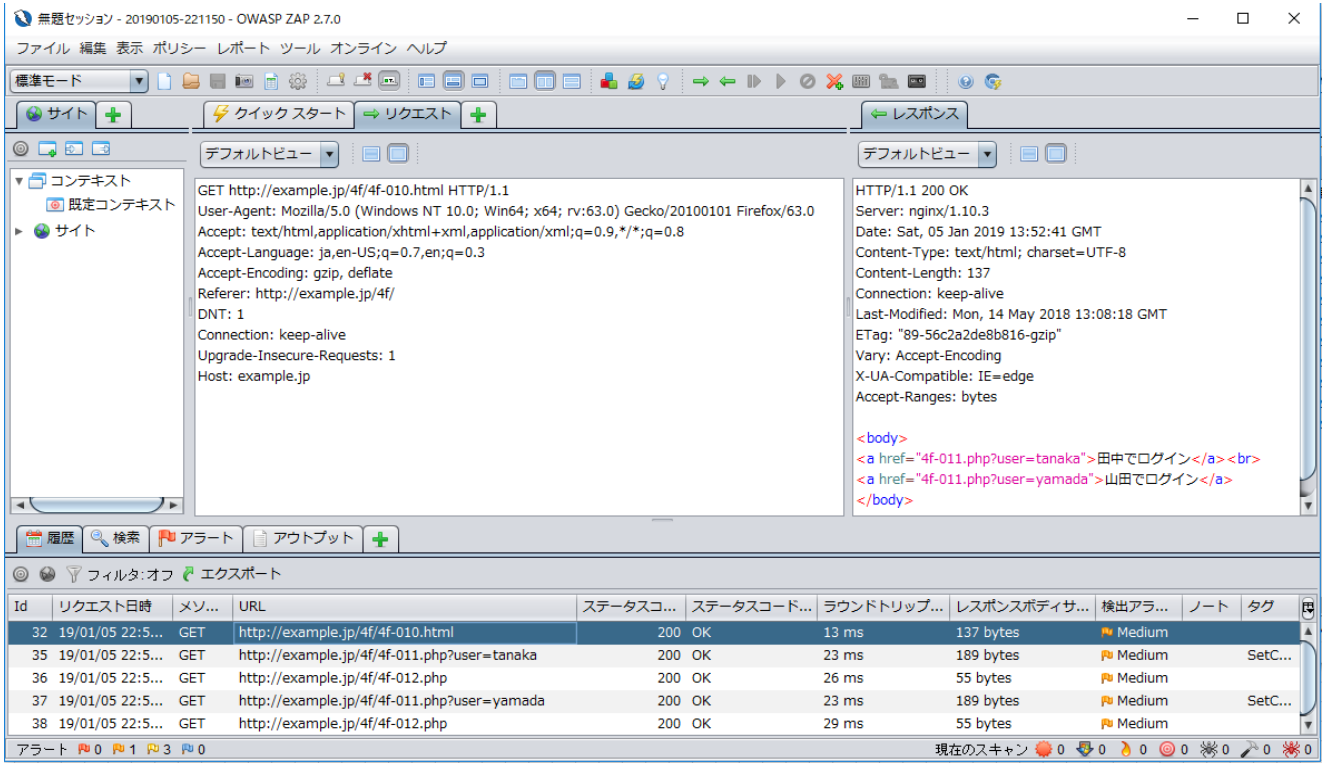
【サーバ： 4f/4f_012a.php 】

/var/www/html/4f/4f-012a.php - wasbook@example.jp - エディタ - WinSCP

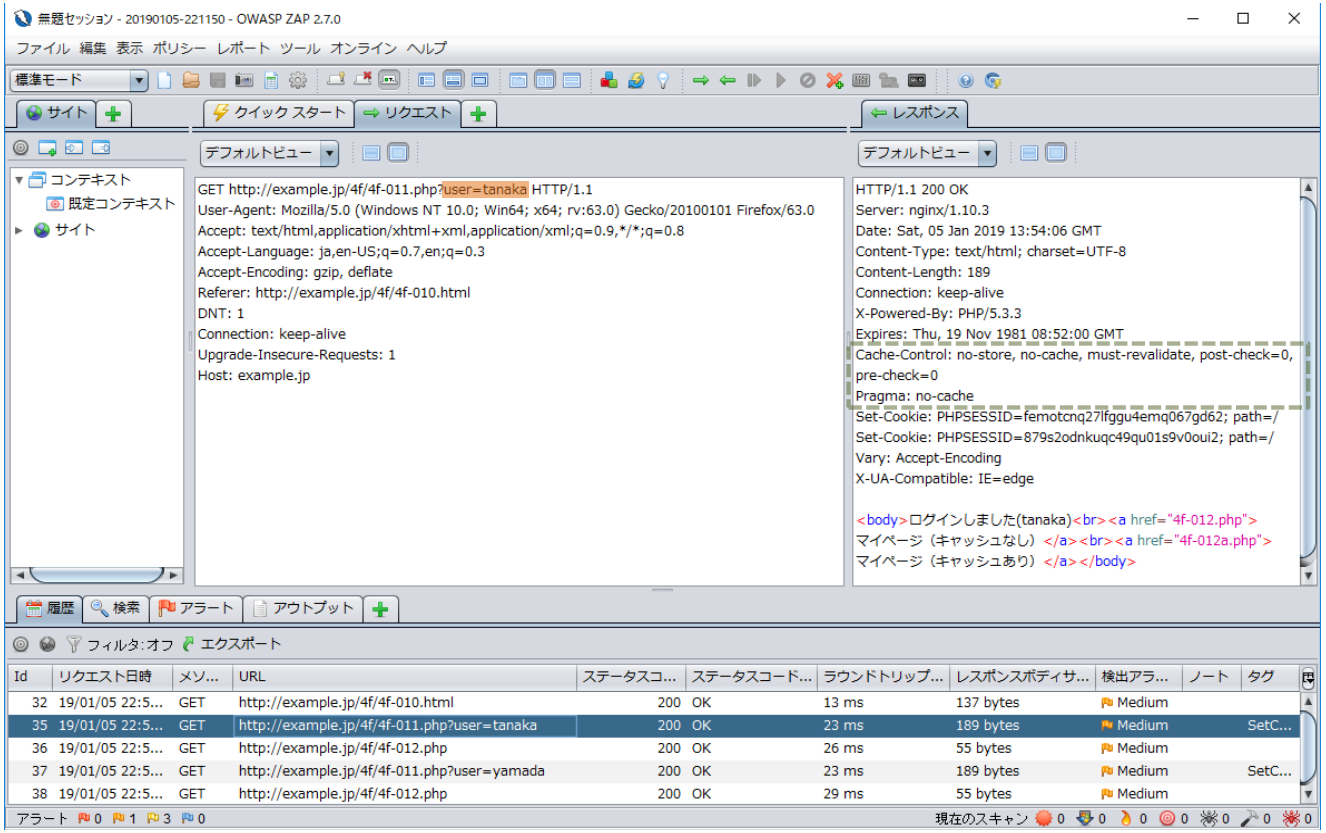
```
<body><?php
  session_cache_limiter('public');
  session_cache_expire(1);
  session_start();
  if (empty($_SESSION['user'])) {
    die("ログインしていません");
  }
  echo "ユーザ {$_SESSION['user']} でログイン中です";
?></body>
```

PHPの session_cache_limiter関数は「public」指定で、以下ヘッダを返す
Expires: session.cache_expireだけ未来の日時
Cache-Control: piblic,max-age=session.cache_expireを分⇒秒の数値

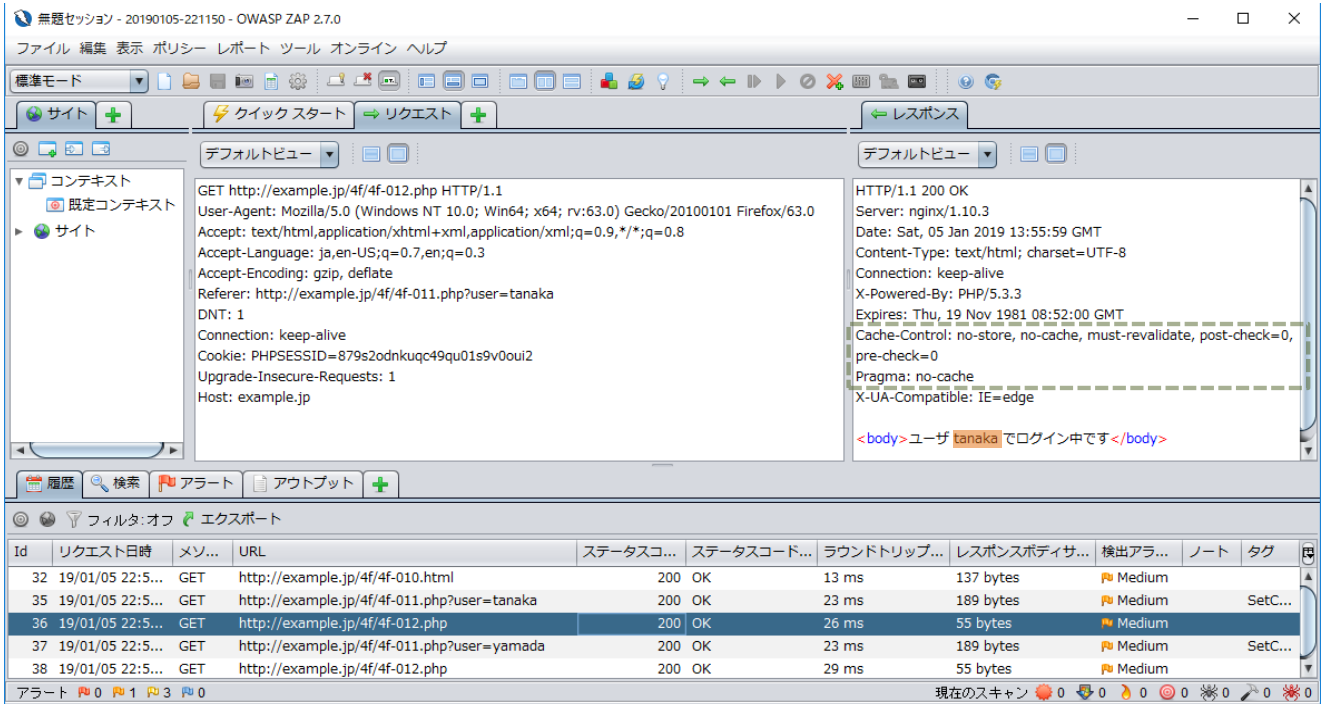
【ブラウザ→サーバ： リクエスト 4f/4f-010.html → レスポンス 】



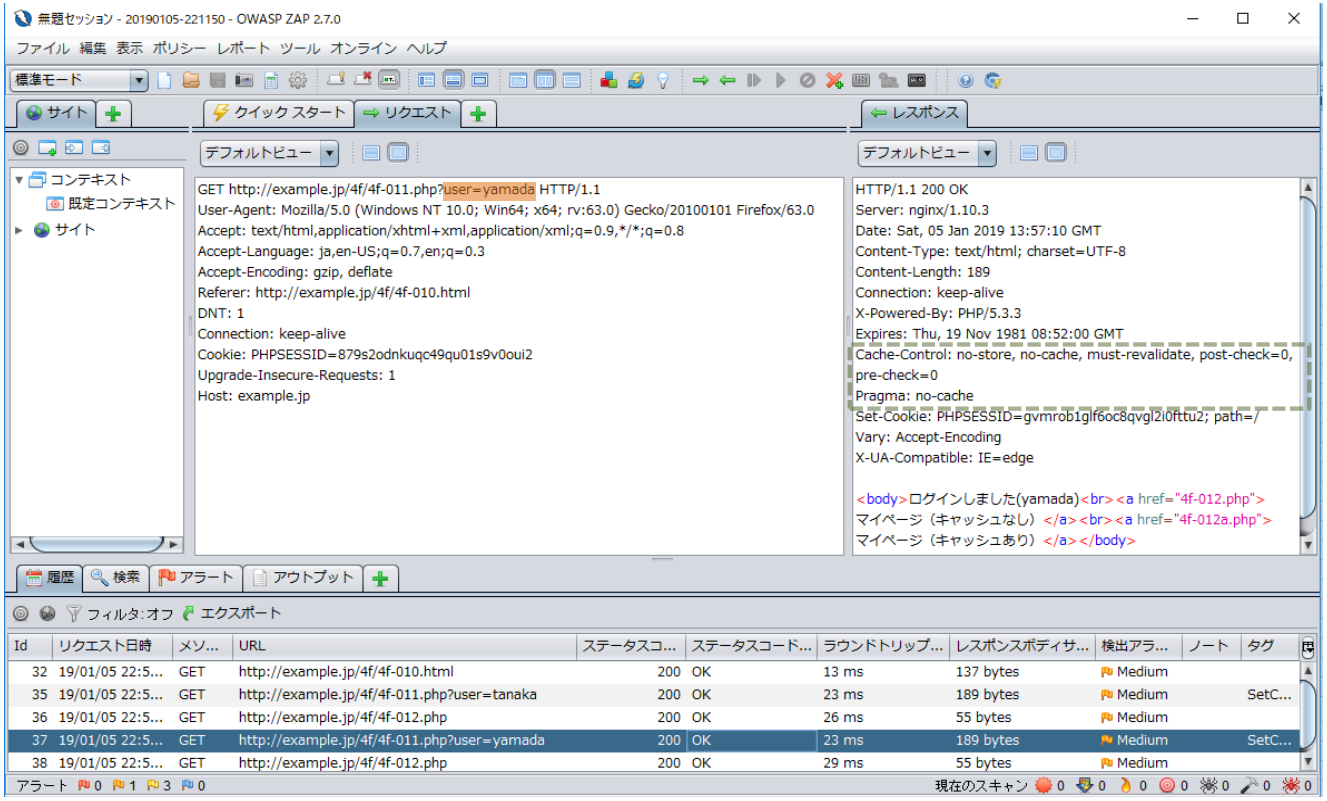
【ブラウザ→サーバ： リクエスト 4f/4f-011.php → レスポンス 】



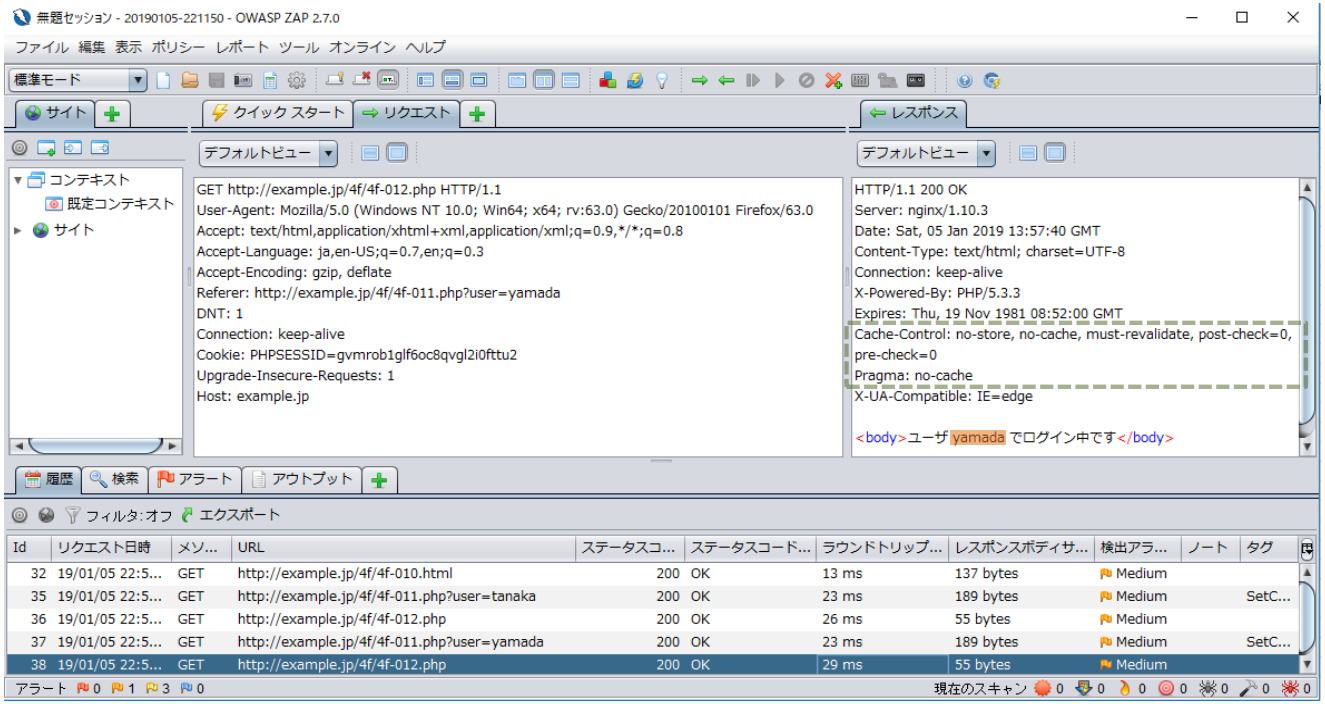
【ブラウザ→サーバ： リクエスト 4f/4f-012.php → レスポンス 】



【ブラウザ→サーバ： リクエスト 4f/4f-011.php → レスポンス 】

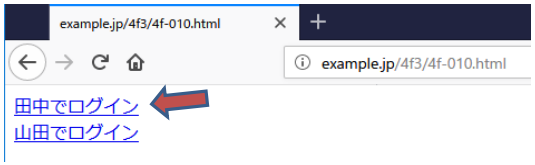


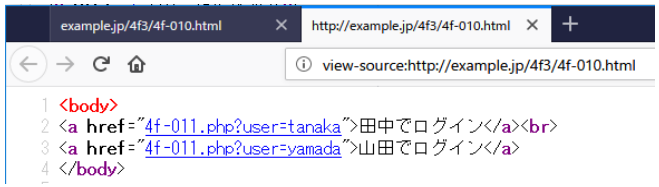
【ブラウザ→サーバ： リクエスト 4f/4f-012.php → レスポンス 】

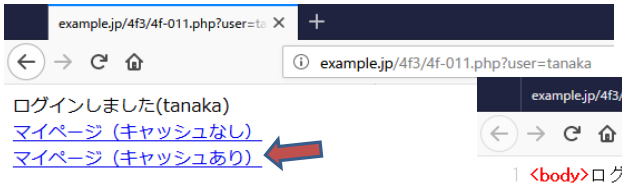


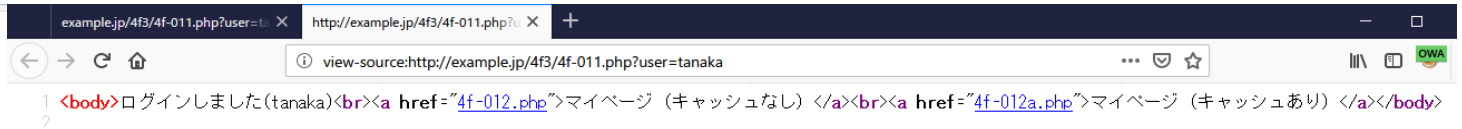
4f-010 :ログインしてユーザ名を表示（過剰なキャッシュ）

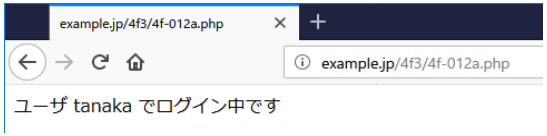
【ブラウザ】

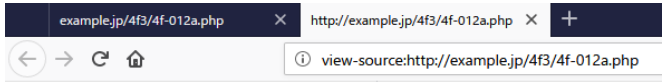

example.jp/4f3/4f-010.html
example.jp/4f3/4f-010.html
田中でログイン
山田でログイン

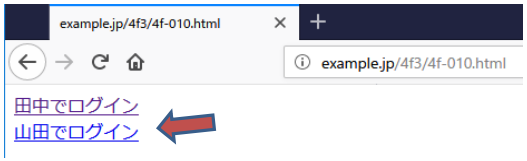

example.jp/4f3/4f-010.html
http://example.jp/4f3/4f-010.html
view-source:http://example.jp/4f3/4f-010.html
<body>
田中でログイン
山田でログイン
</body>

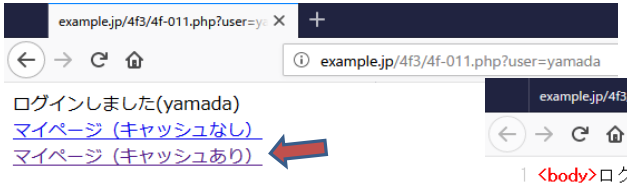

example.jp/4f3/4f-011.php?user=tanaka
ログインしました(tanaka)
マイページ（キャッシュなし）
マイページ（キャッシュあり）


example.jp/4f3/4f-011.php?user=tanaka
http://example.jp/4f3/4f-011.php?user=tanaka
view-source:http://example.jp/4f3/4f-011.php?user=tanaka
<body>ログインしました(tanaka)
マイページ（キャッシュなし）
マイページ（キャッシュあり）</body>

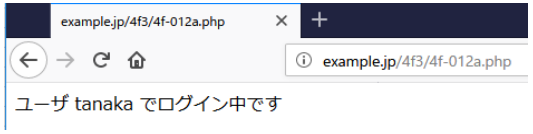

example.jp/4f3/4f-012a.php
example.jp/4f3/4f-012a.php
ユーザ tanaka でログイン中です

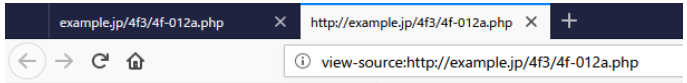

example.jp/4f3/4f-012a.php
http://example.jp/4f3/4f-012a.php
view-source:http://example.jp/4f3/4f-012a.php
<body>ユーザ tanaka でログイン中です</body>


example.jp/4f3/4f-010.html
example.jp/4f3/4f-010.html
田中でログイン
山田でログイン


example.jp/4f3/4f-011.php?user=yamada
ログインしました(yamada)
マイページ（キャッシュなし）
マイページ（キャッシュあり）


example.jp/4f3/4f-011.php?user=yamada
http://example.jp/4f3/4f-011.php?user=yamada
view-source:http://example.jp/4f3/4f-011.php?user=yamada
<body>ログインしました(yamada)
マイページ（キャッシュなし）
マイページ（キャッシュあり）</body>


example.jp/4f3/4f-012a.php
example.jp/4f3/4f-012a.php
ユーザ tanaka でログイン中です


example.jp/4f3/4f-012a.php
http://example.jp/4f3/4f-012a.php
view-source:http://example.jp/4f3/4f-012a.php
<body>ユーザ tanaka でログイン中です</body>

【サーバ： 4f/4f_010.html 】

/var/www/html/4f/4f-010.html - wasbook@example.jp - エディタ - WinSCP

```
<body>
<a href="4f-011.php?user=tanaka">田中でログイン</a><br>
<a href="4f-011.php?user=yamada">山田でログイン</a>
</body>
```

【サーバ： 4f/4f_011.php 】

/var/www/html/4f/4f-011.php - wasbook@example.jp - エディタ - WinSCP

```
<body><?php
  $user = $_GET['user'];
  if ($user === 'tanaka' || $user === 'yamada') {
    session_start();
    session_regenerate_id(true);
    $_SESSION['user'] = $user;
    echo 'ログインしました(' . htmlspecialchars($user) . ')<br>';
    echo '<a href="4f-012.php">マイページ（キャッシュなし）</a><br>';
    echo '<a href="4f-012a.php">マイページ（キャッシュあり）</a>';
  } else {
    echo 'ユーザ名が違います';
  }
?>
</body>
```

【サーバ： 4f/4f_012.php 】

/var/www/html/4f/4f-012.php - wasbook@example.jp - エディタ - WinSCP

```
<body><?php
  session_start();
  if (empty($_SESSION['user'])) {
    die("ログインしていません");
  }
  echo "ユーザ {$_SESSION['user']} でログイン中です";
?></body>
```

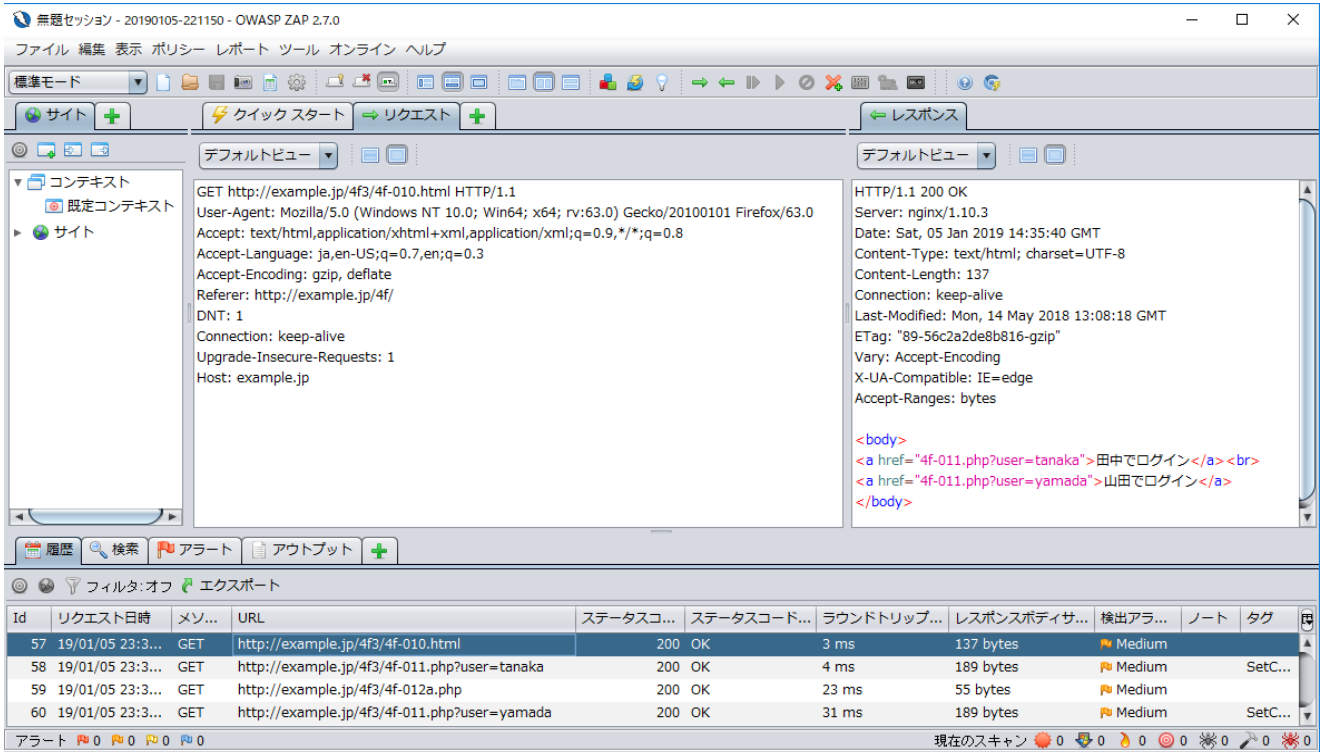
【サーバ： 4f/4f_012a.php 】

/var/www/html/4f/4f-012a.php - wasbook@example.jp - エディタ - WinSCP

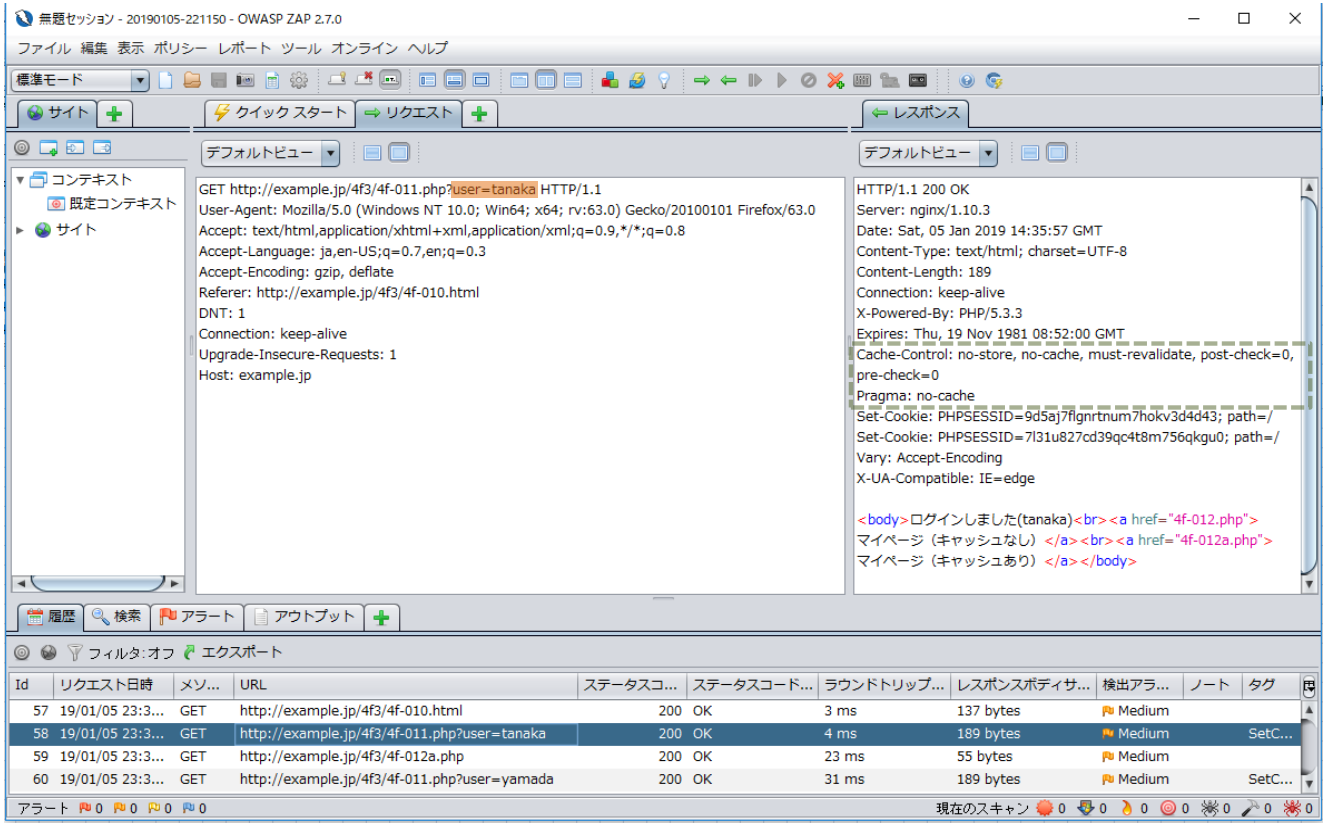
```
<body><?php
  session_cache_limiter('public');
  session_cache_expire(1);
  session_start();
  if (empty($_SESSION['user'])) {
    die("ログインしていません");
  }
  echo "ユーザ {$_SESSION['user']} でログイン中です";
?></body>
```

PHPの session_cache_limiter関数は「public」指定で、以下ヘッダを返す
Expires: session.cache_expireだけ未来の日時
Cache-Control: piblic,max-age=session.cache_expireを分⇒秒の数値

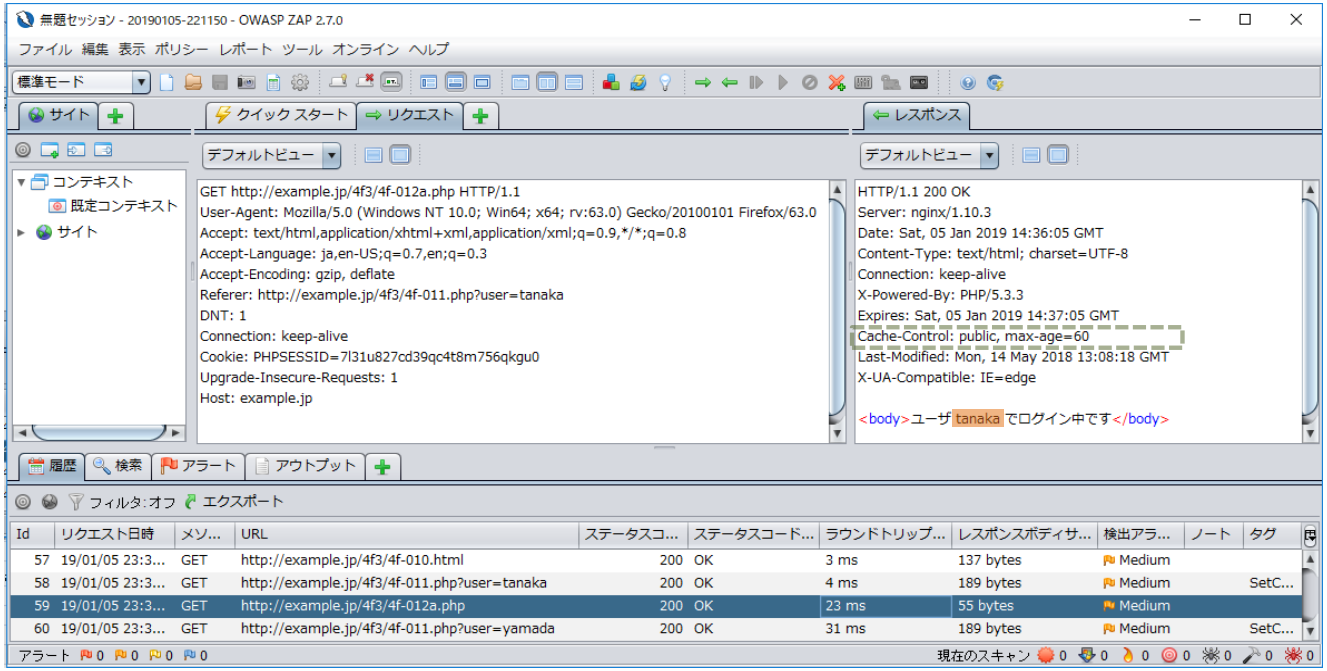
【ブラウザ→サーバ： リクエスト 4f/4f-010.html → レスポンス 】



【ブラウザ→サーバ： リクエスト 4f/4f-011.php → レスポンス 】



【ブラウザ→サーバ：リクエスト 4f/4f-012a.php → レスポンス 】



キャッシュは常に有効（最長60秒）

【ブラウザ→サーバ： リクエスト 4f3/4f-011.php → レスポンス 】