4.17.3 postMessage呼び出しの不備

Webストレージとは

ブラウザの機能で、クッキーよりも高性能なストレージ（保存庫）を提供します
Webストレージには、永続保存のlocalStorage と ブラウザセッション保存のsessionStorage があります

	クッキー	LocalStorage	sessionStorage
データの区分け	ブラウザ内共通	ブラウザ内共通	タブごと
アクセス制御	ドメイン＋パス	同一オリジンポリシー	同一オリジンポリシー
有効期間	expires指定	無期限	ブラウザセッションまで
サーバへの送信	自動的に送信	送信されない	送信されない
JavaScriptからのアクセス	httpOnly属性で制御	アクセス可能	アクセス可能
利用者参照	参照可能	参照可能	参照可能
利用者変更	変更可能	変更可能	変更可能

運用上の注意点

XSS脆弱性があると情報漏洩につながるので、重要な情報はWebストレージに保存しない

post Messageとは

iframeやwindows.openで開いたウィンドウなど、複数の異なるオリジン・ウィンドウが連携して、メッセージやデータをやり取りを行う汎用的な仕組み

postMessage呼び出しの不備

メッセージ送信先が未検証の場合	win.postMessage(message, origin);	「origin」に明確に指定する。「＊」の指定は危険。
メッセージ送信元が未検証の場合	event.source.postMessage("メッセージ", event.origin);	送信元に送信する場合、「event.origin」を指定する。「＊」の指定は危険。

4h-010 :postMessageの例

【ブラウザ】

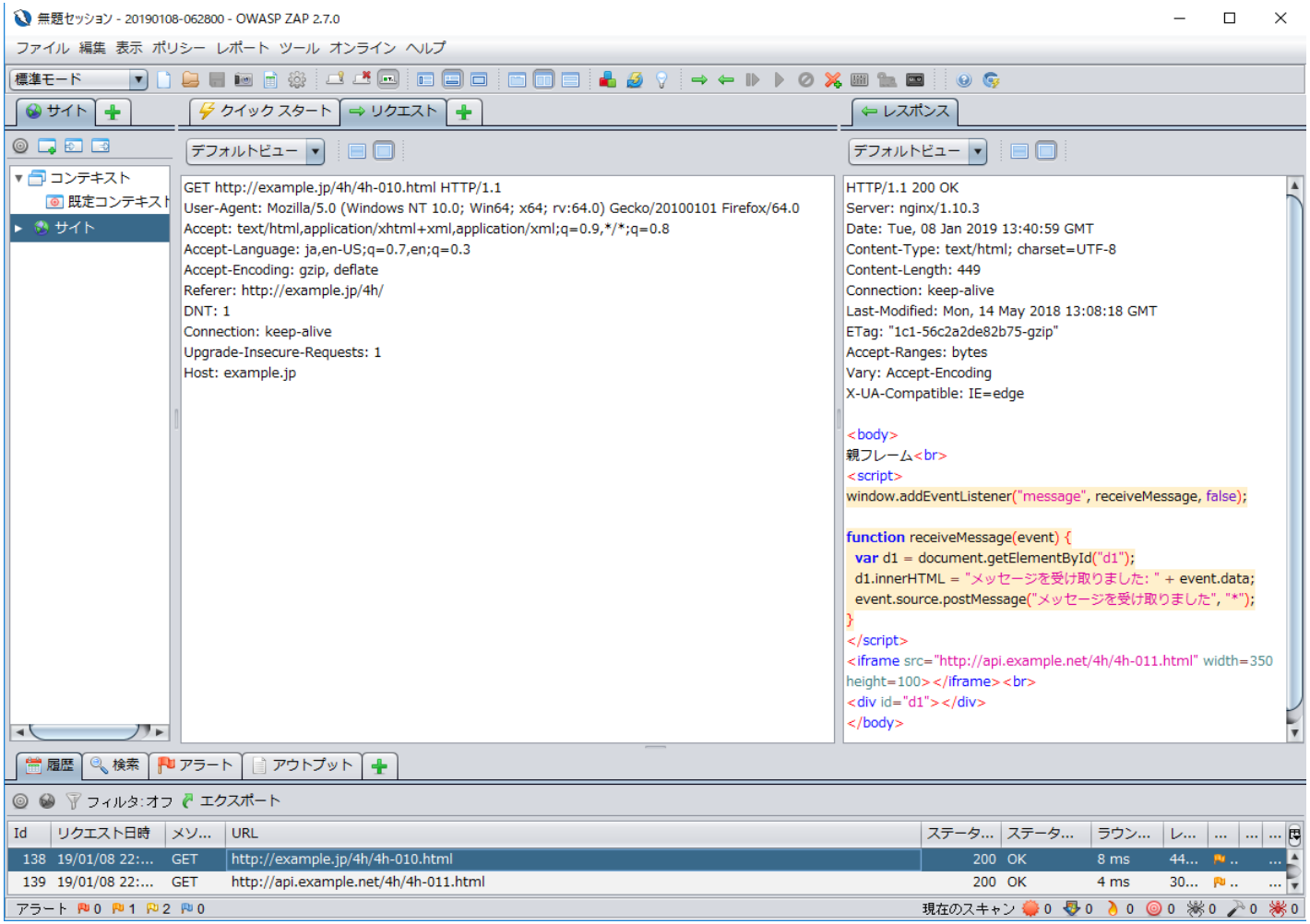
- 4.17.3 postMessage呼び出しの不備
  1. 4h-010 :postMessageの例
  2. 4h-010 :postMessageの脆弱性の悪用
  3. 4h-010a:postMessageの悪用（対策版）
  4. 4h-910 :メッセージ送信元の未確認（攻撃）
  5. 4h-910c:メッセージ送信元の確認版（攻撃）

```
<li>4.17.3 postMessage呼び出しの不備</li>
<ol>
<li><a href="4h-010.html">4h-010 :postMessageの例</a></li>
<li><a href="http://trap.example.com/4h/4h-010.html">4h-010 :postMessageの脆弱性の悪用</a></li>
<li><a href="http://trap.example.com/4h/4h-010a.html">4h-010a:postMessageの悪用（対策版）</a></li>
<li><a href="http://trap.example.com/4h/4h-910.html">4h-910 :メッセージ送信元の未確認（攻撃）</a></li>
<li><a href="http://trap.example.com/4h/4h-910c.html">4h-910c:メッセージ送信元の確認版（攻撃）</a></li>
</ol>
```

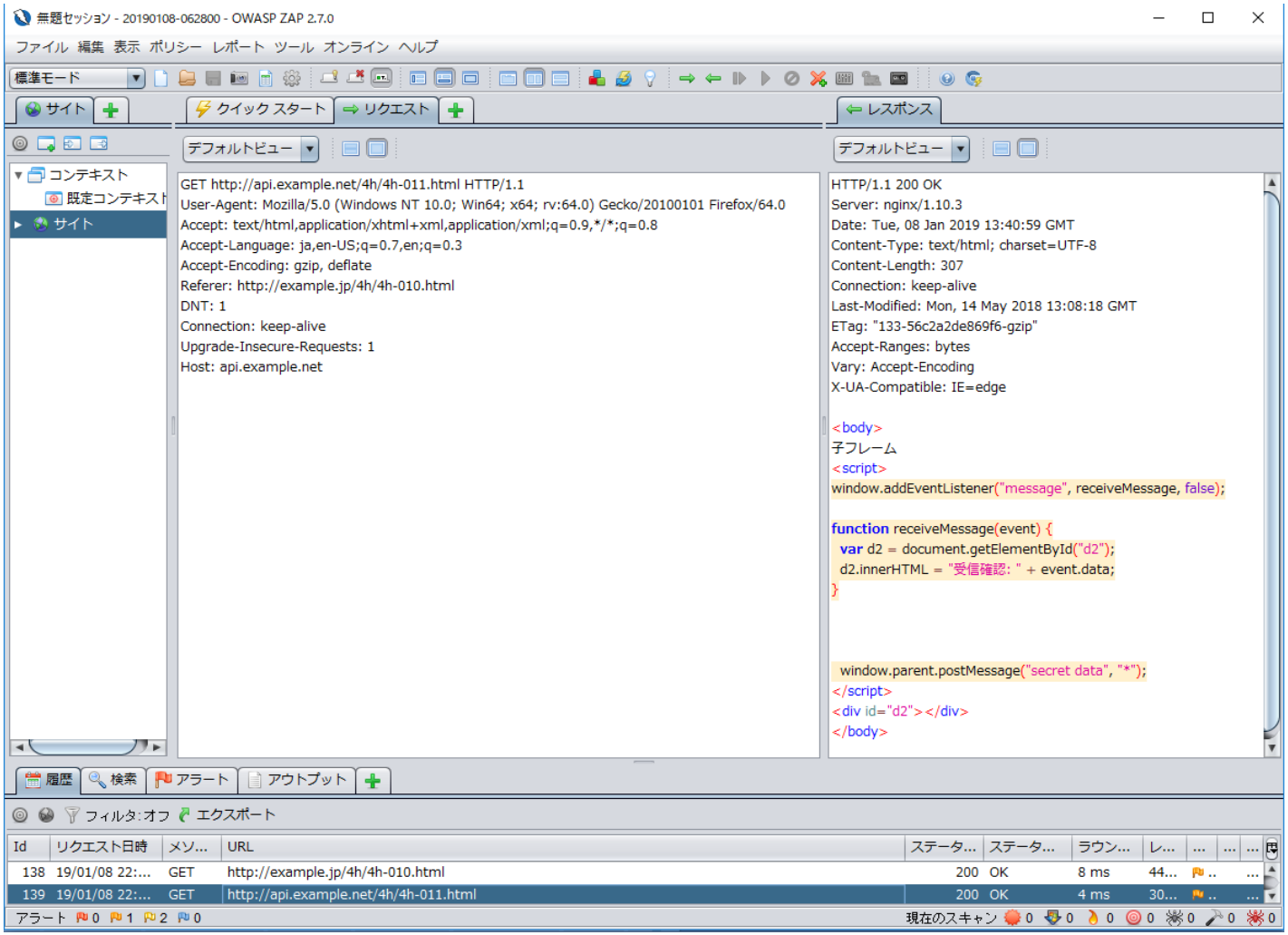
【サーバ: 4h/4h-010.html 】

【サーバ: 4h/4h-011.html 】

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



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



4h-010 :postMessageの例

【ブラウザ】

- 4.17.3 postMessage呼び出しの不備
  1. 4h-010 :postMessageの例
  2. 4h-010 :postMessageの脆弱性の悪用
  3. 4h-010a:postMessageの悪用（対策版）
  4. 4h-910 :メッセージ送信元の未確認（攻撃）
  5. 4h-910c:メッセージ送信元の確認版（攻撃）

```
31 <li>4.17.3 postMessage呼び出しの不備</li>
32 <ol>
33 <li><a href="4h-010.html">4h-010 :postMessageの例</a></li>
34 <li><a href="http://trap.example.com/4h/4h-010.html">4h-010 :postMessageの脆弱性の悪用</a></li>
35 <li><a href="http://trap.example.com/4h/4h-010a.html">4h-010a:postMessageの悪用（対策版）</a></li>
36 <li><a href="http://trap.example.com/4h/4h-910.html">4h-910 :メッセージ送信元の未確認（攻撃）</a></li>
37 <li><a href="http://trap.example.com/4h/4h-910c.html">4h-910c:メッセージ送信元の確認版（攻撃）</a></li>
38 </ol>
```

trap.example.com/4h/4h-010.html

親フレーム

子フレーム
受信確認: メッセージを受け取りました

メッセージを受け取りました: secret data

trap.example.com/4h/4h-010.html　http://trap.example.com/4h/4h-010

view-source:http://trap.example.com/4h/4h-010.html

```
1 <body>
2 親フレーム<br>
3 <script>
4 window.addEventListener("message", receiveMessage, false);
5
6 function receiveMessage(event) {
7   var d1 = document.getElementById("d1");
8   d1.innerHTML = "メッセージを受け取りました: " + event.data;
9   event.source.postMessage("メッセージを受け取りました", "*");
10 }
11 </script>
12 <iframe src="http://api.example.net/4h/4h-011.html" width=350 height=100></iframe><br>
13 <div id="d1"></div>
14 </body>
```

【サーバ： 4h/4h-010.html 】

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

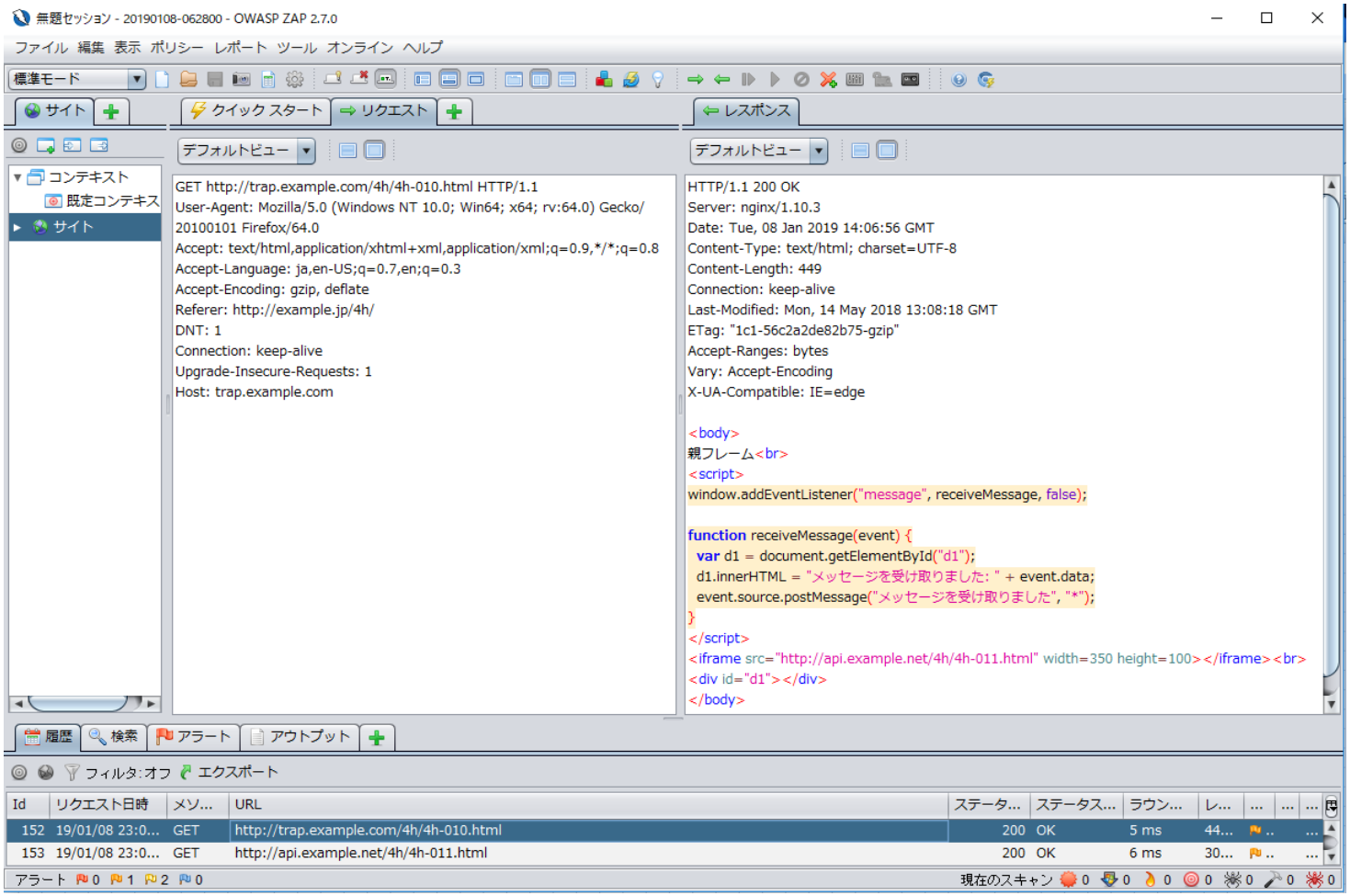
```
<body>
親フレーム<br>
<script>
window.addEventListener("message", receiveMessage, false);

function receiveMessage(event) {
  var d1 = document.getElementById("d1");
  d1.innerHTML = "メッセージを受け取りました: " + event.data;
  event.source.postMessage("メッセージを受け取りました", "*");
}
</script>
<iframe src="http://api.example.net/4h/4h-011.html" width=350 height=100></iframe><br>
<div id="d1"></div>
</body>
```

【サーバ： 4h/4h-011.html 】

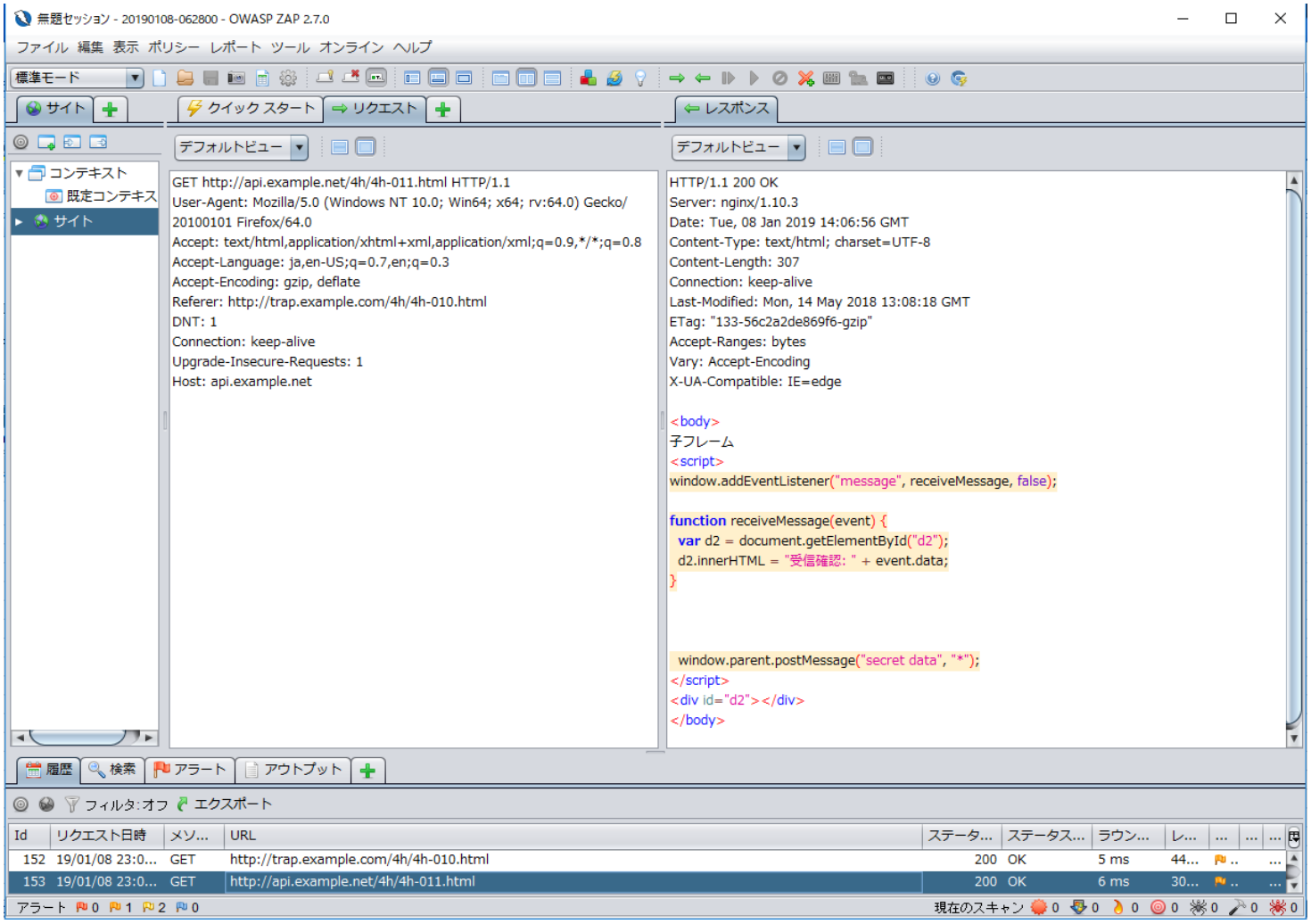
/var/www/html/4h/4h-011.html - wasbook@example.jp - エディタ - WinSCP

```
<body>
子フレーム
<script>
window.addEventListener("message", receiveMessage, false);

function receiveMessage(event) {
  var d2 = document.getElementById("d2");
  d2.innerHTML = "受信確認: " + event.data;
}


  window.parent.postMessage("secret data", "*");
</script>
<div id="d2"></div>
</body>
```

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



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



4h-010a:postMessageの悪用（対策版）

【ブラウザ】

- 4.17.3 postMessage呼び出しの不備
  1. 4h-010 :postMessageの例
  2. 4h-010 :postMessageの脆弱性の悪用
  3. 4h-010a:postMessageの悪用（対策版）
  4. 4h-910 :メッセージ送信元の未確認（攻撃）
  5. 4h-910c:メッセージ送信元の確認版（攻撃）

```
<li>4.17.3 postMessage呼び出しの不備</li>
<ol>
<li><a href="4h-010.html">4h-010 :postMessageの例</a></li>
<li><a href="http://trap.example.com/4h/4h-010.html">4h-010 :postMessageの脆弱性の悪用</a></li>
<li><a href="http://trap.example.com/4h/4h-010a.html">4h-010a:postMessageの悪用（対策版）</a></li>
<li><a href="http://trap.example.com/4h/4h-910.html">4h-910 :メッセージ送信元の未確認（攻撃）</a></li>
<li><a href="http://trap.example.com/4h/4h-910c.html">4h-910c:メッセージ送信元の確認版（攻撃）</a></li>
</ol>
```

【サーバ： 4h/4h-010a.html 】

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

```
<body>
親フレーム<br>
<script>
window.addEventListener("message", receiveMessage, false);

function receiveMessage(event) {
  var d1 = document.getElementById("d1");
  d1.innerHTML = "メッセージを受け取りました: " + event.data;
  event.source.postMessage("メッセージを受け取りました", event.origin);
}
</script>
<iframe src="http://api.example.net/4h/4h-011a.html" width=350 height=100></iframe><br>
<div id="d1"></div>
</body>
```

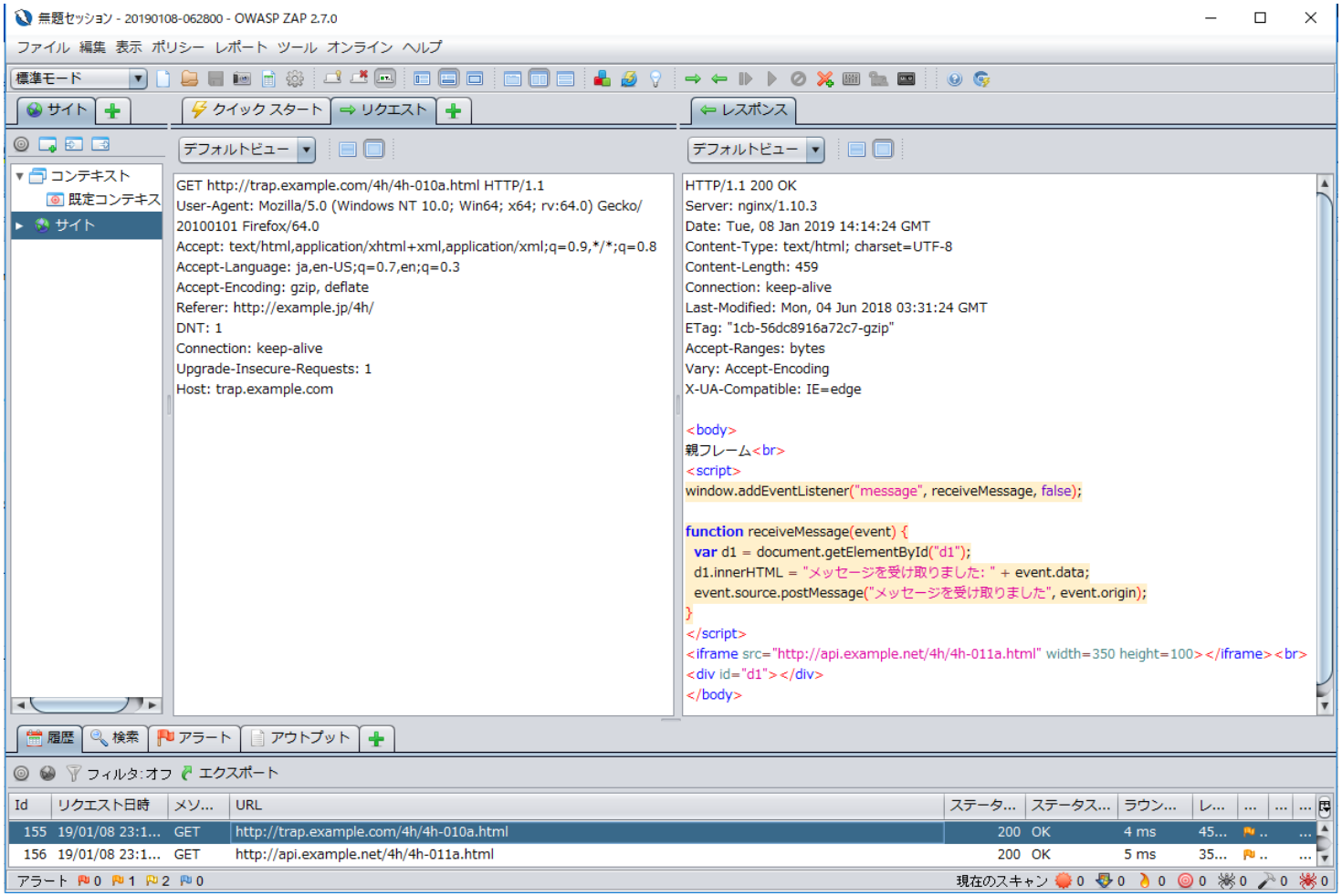

```
<body>
親フレーム<br>
<script>
window.addEventListener("message", receiveMessage, false);

function receiveMessage(event) {
  var d1 = document.getElementById("d1");
  d1.innerHTML = "メッセージを受け取りました: " + event.data;
  event.source.postMessage("メッセージを受け取りました", event.origin);
}
</script>
<iframe src="http://api.example.net/4h/4h-011a.html" width=350 height=100></iframe><br>
<div id="d1"></div>
</body>
```

【サーバ： 4h/4h-010a.html 】

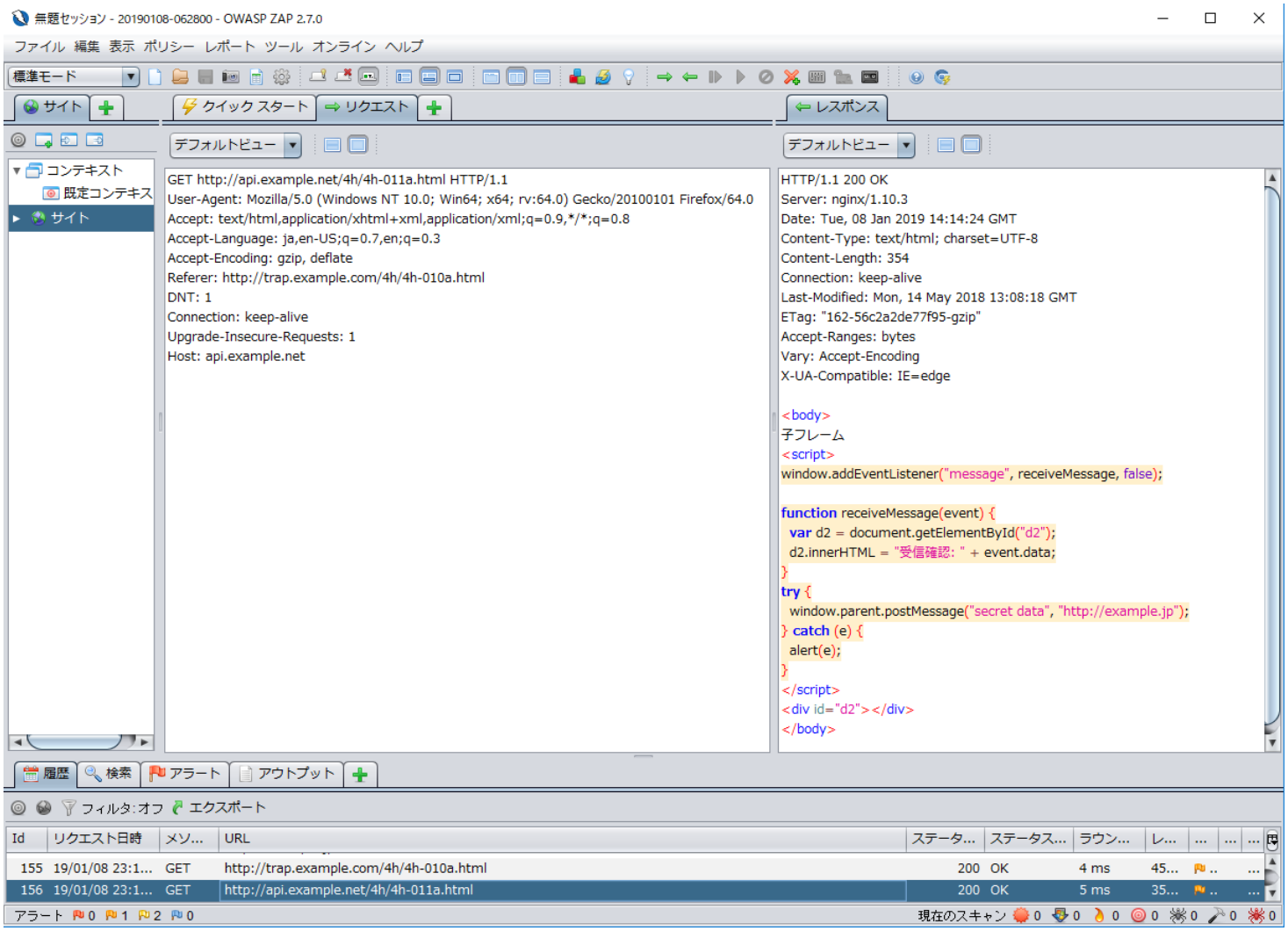
/var/www/html/4h/4h-011a.html - wasbook@example.jp - エディタ - WinSCP

```
<body>
子フレーム
<script>
window.addEventListener("message", receiveMessage, false);

function receiveMessage(event) {
  var d2 = document.getElementById("d2");
  d2.innerHTML = "受信確認: " + event.data;
}
try {
  window.parent.postMessage("secret data", "http://example.jp");
} catch (e) {
  alert(e);
}
</script>
<div id="d2"></div>
</body>
```

【ブラウザ→サーバ： リクエスト trap.example.com/4h/4h-010a.html → レスポンス 】



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

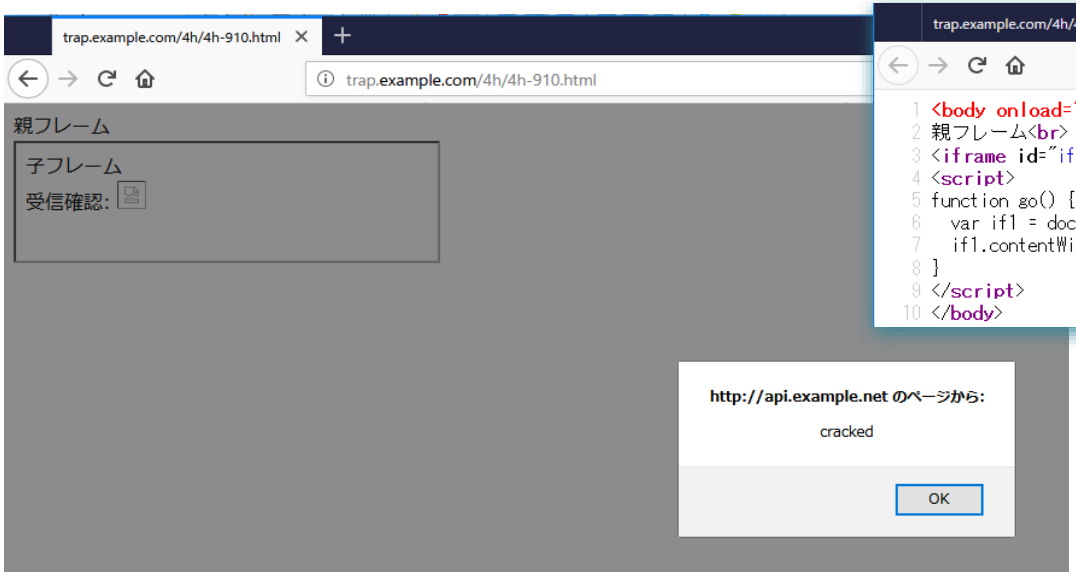


4h-910 :メッセージ送信元の未確認（攻撃）

【ブラウザ】

- 4.17.3 postMessage呼び出しの不備
  1. 4h-010 :postMessageの例
  2. 4h-010 :postMessageの脆弱性の悪用
  3. 4h-010a:postMessageの悪用（対策版）
  4. 4h-910 :メッセージ送信元の未確認（攻撃）
  5. 4h-910c:メッセージ送信元の確認版（攻撃）

```
<li>4.17.3 postMessage呼び出しの不備</li>
<ol>
<li><a href="4h-010.html">4h-010 :postMessageの例</a></li>
<li><a href="http://trap.example.com/4h/4h-010.html">4h-010 :postMessageの脆弱性の悪用</a></li>
<li><a href="http://trap.example.com/4h/4h-010a.html">4h-010a:postMessageの悪用（対策版）</a></li>
<li><a href="http://trap.example.com/4h/4h-910.html">4h-910 :メッセージ送信元の未確認（攻撃）</a></li>
<li><a href="http://trap.example.com/4h/4h-910c.html">4h-910c:メッセージ送信元の確認版（攻撃）</a></li>
</ol>
```



【サーバ： 4h/4h-910.html 】

/var/www/html/4h/4h-910.html - wasbook@example.jp - エディタ - WinSCP

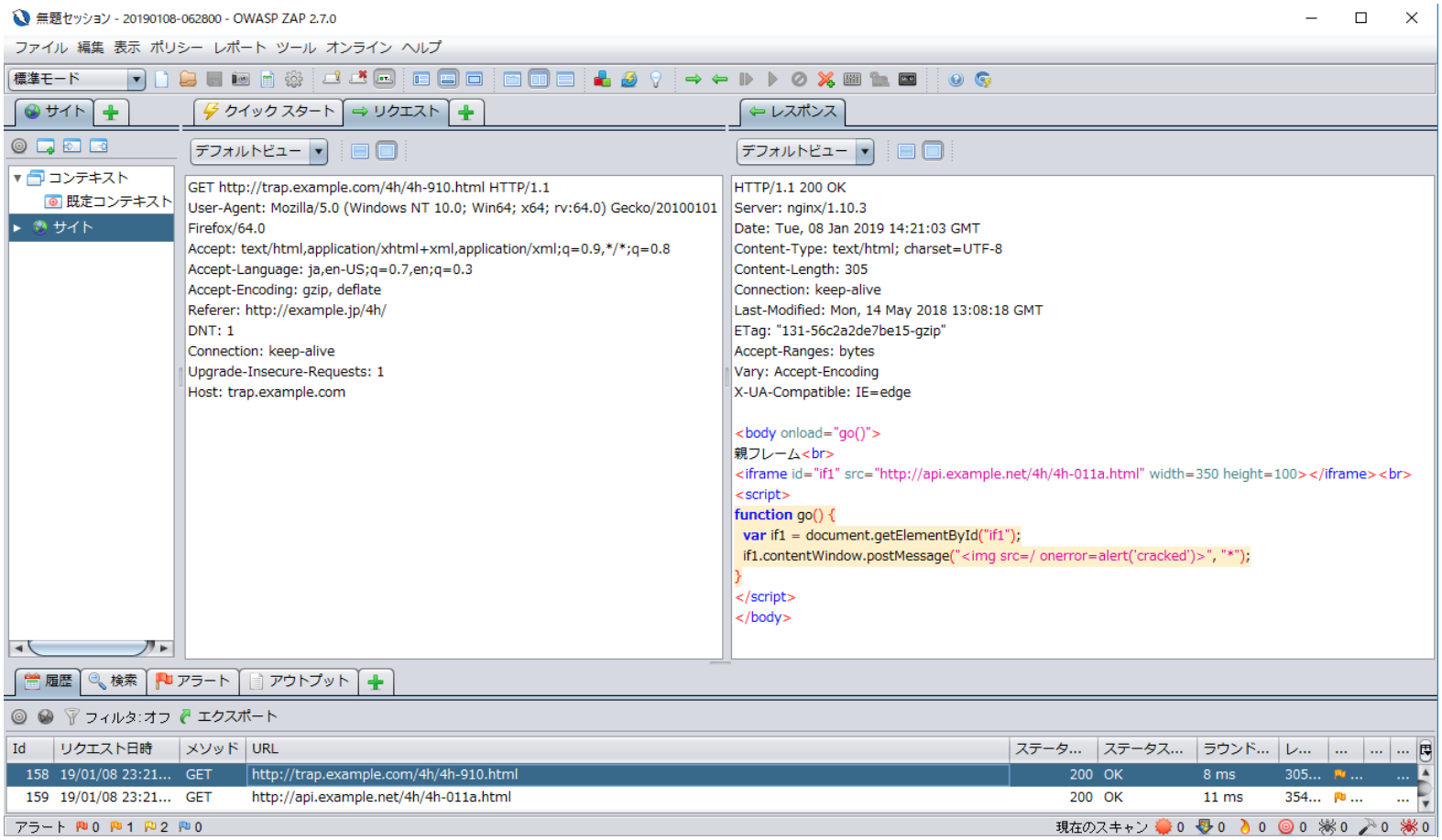
```
<body onload="go()">
親フレーム<br>
<iframe id="if1" src="http://api.example.net/4h/4h-011a.html" width=350 height=100></iframe><br>
<script>
function go() {
  var if1 = document.getElementById("if1");
  if1.contentWindow.postMessage("<img src=/ onerror=alert('cracked')>", "*");
}
</script>
</body>
```

【サーバ： 4h/4h-011a.html 】

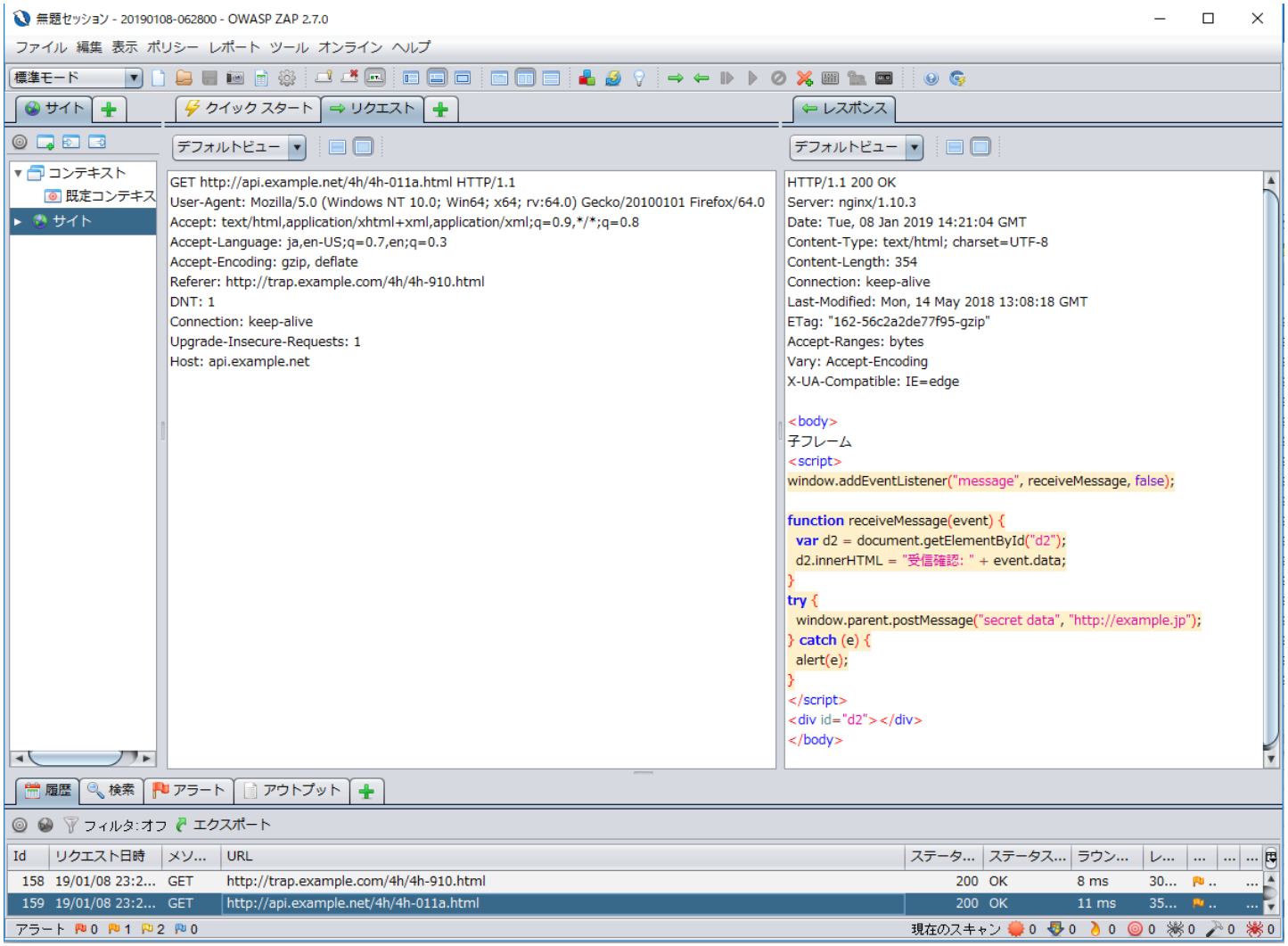
/var/www/html/4h/4h-011a.html - wasbook@example.jp - エディタ - WinSCP

```
<body>
子フレーム
<script>
window.addEventListener("message", receiveMessage, false);

function receiveMessage(event) {
  var d2 = document.getElementById("d2");
  d2.innerHTML = "受信確認: " + event.data;
}
try {
  window.parent.postMessage("secret data", "http://example.jp");
} catch (e) {
  alert(e);
}
</script>
<div id="d2"></div>
</body>
```

【ブラウザ→サーバ： リクエスト trap.example.com/4h/4h-910.html → レスポンス 】



【ブラウザ→APIサーバ： リクエスト 4h/4h-011a.html → レスポンス 】



4h-910c:メッセージ送信元の確認版（攻撃）

【ブラウザ】

- 4.17.3 postMessage呼び出しの不備
  1. 4h-010 :postMessageの例
  2. 4h-010 :postMessageの脆弱性の悪用
  3. 4h-010a:postMessageの悪用（対策版）
  4. 4h-910 :メッセージ送信元の未確認（攻撃）
  5. 4h-910c:メッセージ送信元の確認版（攻撃）

```
<li>4.17.3 postMessage呼び出しの不備</li>
<ol>
<li><a href="4h-010.html">4h-010 :postMessageの例</a></li>
<li><a href="http://trap.example.com/4h/4h-010.html">4h-010 :postMessageの脆弱性の悪用</a></li>
<li><a href="http://trap.example.com/4h/4h-010a.html">4h-010a:postMessageの悪用（対策版）</a></li>
<li><a href="http://trap.example.com/4h/4h-910.html">4h-910 :メッセージ送信元の未確認（攻撃）</a></li>
<li><a href="http://trap.example.com/4h/4h-910c.html">4h-910c:メッセージ送信元の確認版（攻撃）</a></li>
</ol>
```

【サーバ： 4h/4h-910c.html 】

/var/www/html/4h/4h-910c.html - wasbook@example.jp - エディタ - WinSCP

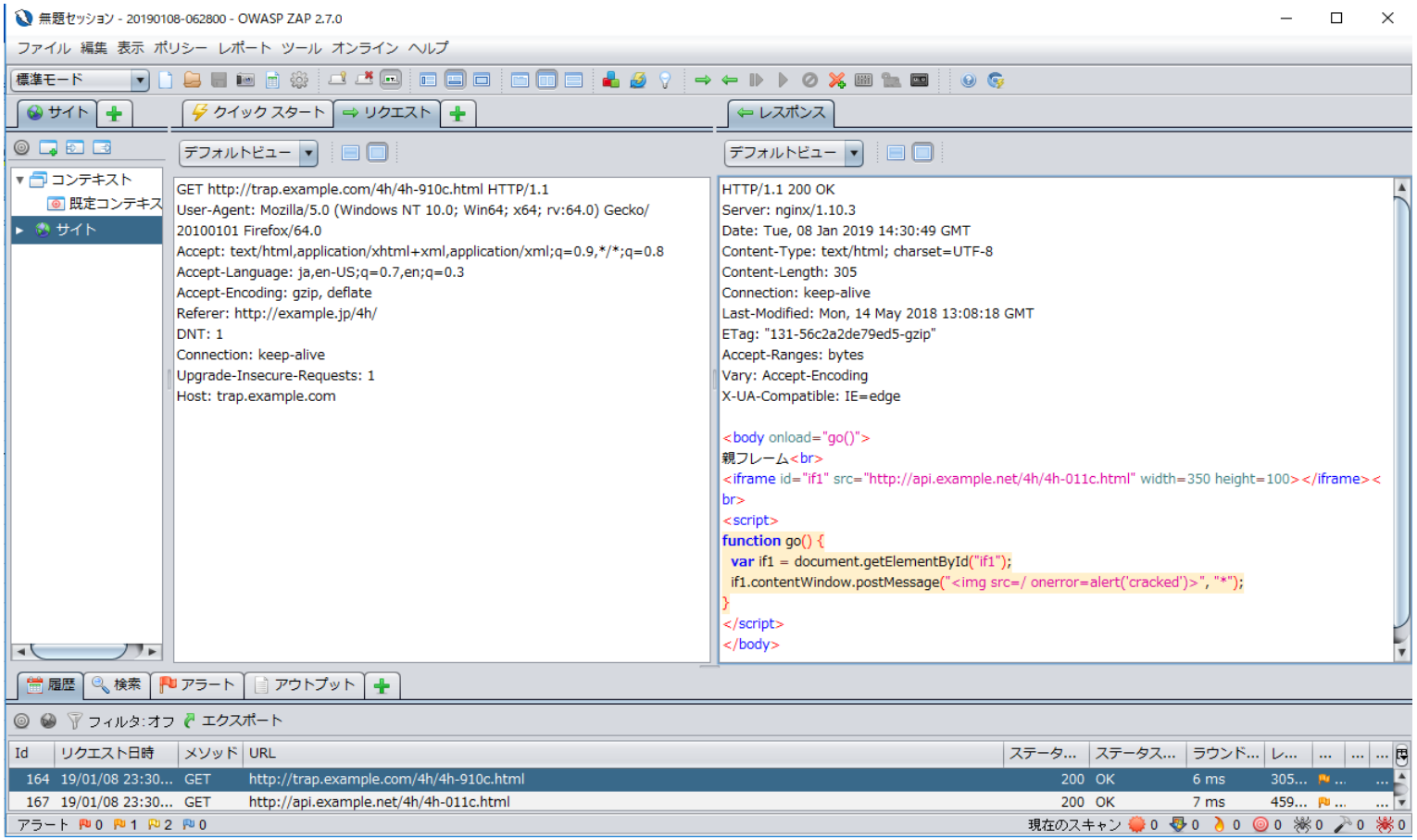
```
<body onload="go()">
親フレーム<br>
<iframe id="if1" src="http://api.example.net/4h/4h-011c.html" width=350 height=100></iframe><br>
<script>
function go() {
  var if1 = document.getElementById("if1");
  if1.contentWindow.postMessage("<img src=/ onerror=alert('cracked')>", "*");
}
</script>
</body>
```

【サーバ： 4h/4h-011c.html 】

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

```
<body>
子フレーム
<script>
window.addEventListener("message", receiveMessage, false);

function receiveMessage(event) {
  var d2 = document.getElementById("d2");
  if (event.origin !== "http://example.jp") {
    d2.textContent = "オリジン違反";
    return;
  }
  d2.innerHTML = "受信確認: " + event.data;
}
try {
  window.parent.postMessage("secret data", "http://example.jp");
} catch (e) {
  alert(e);
}
</script>
<div id="d2"></div>
</body>
```

【ブラウザ→サーバ： リクエスト trap.example.com/4h/4h-910c.html → レスポンス 】



【ブラウザ→APIサーバ： リクエスト 4h/4h-011c.html → レスポンス 】