「Javaアプリケーション脆弱性事例調査資料」について

- この資料は、Javaプログラマである皆様に、脆弱性を身近な問題として感じてもらい、セキュアコーディングの重要性を認識していただくことを目指して作成しています。

- 「Javaセキュアコーディングスタンダード CERT/Oracle版」と合わせて、セキュアコーディングに関する理解を深めるためにご利用ください。

JPCERTコーディネーションセンター
セキュアコーディングプロジェクト
secure-coding@jpcert.or.jp

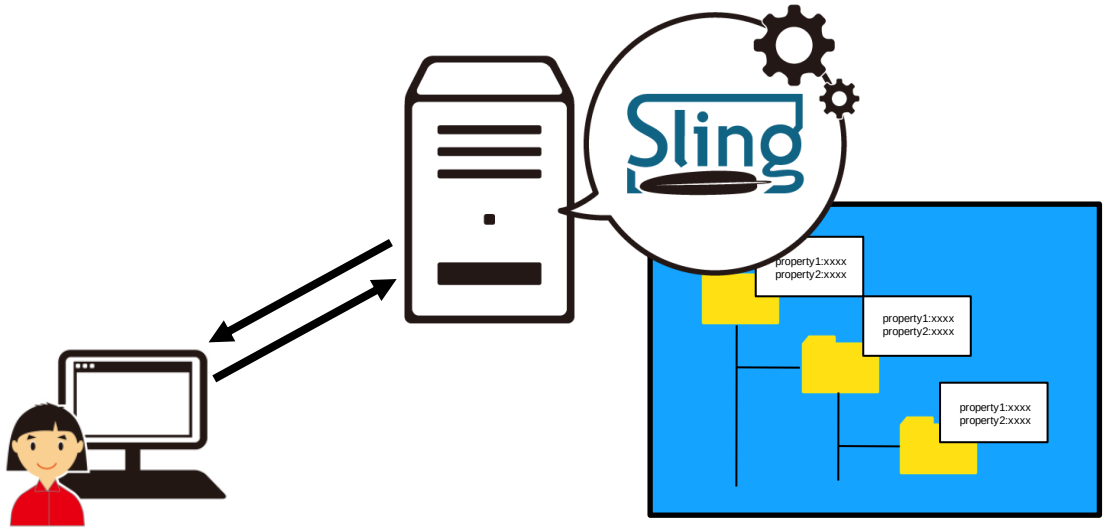
Apache Sling におけるサービス運用妨害 (無限ループ) の脆弱性

CVE-2012-2138

JVNDB-2012-003033

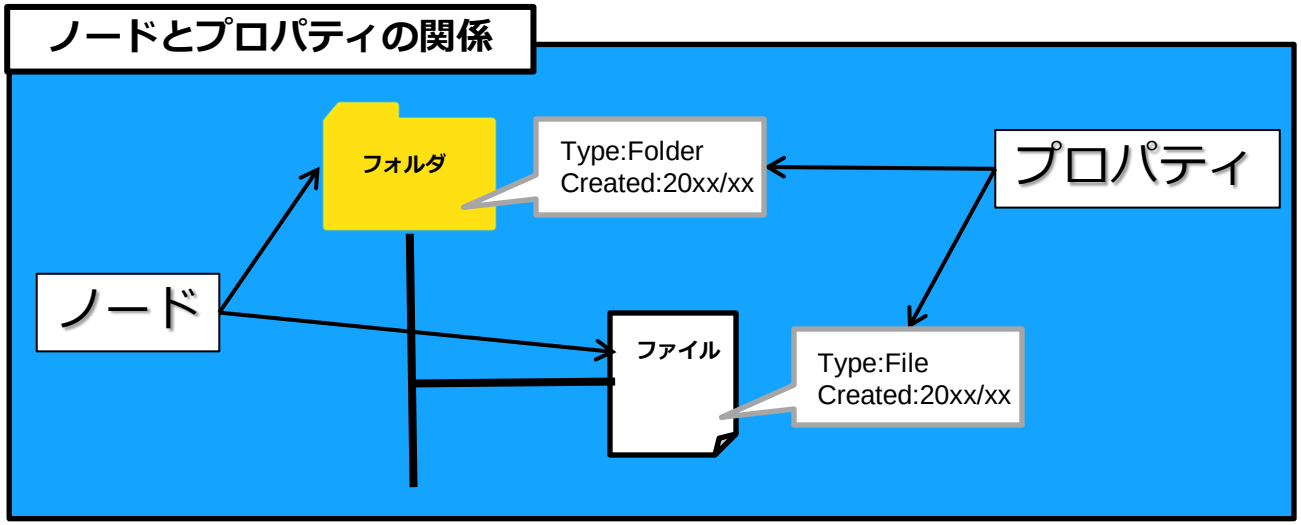
Apache Sling とは

- Webサイトのコンテンツ管理を目的としたオープンソースのWebフレームワーク
- コンテンツの管理方式としてApache Jackrabbitと同様のJavaコンテンツリポジトリ(JCR)を採用している。



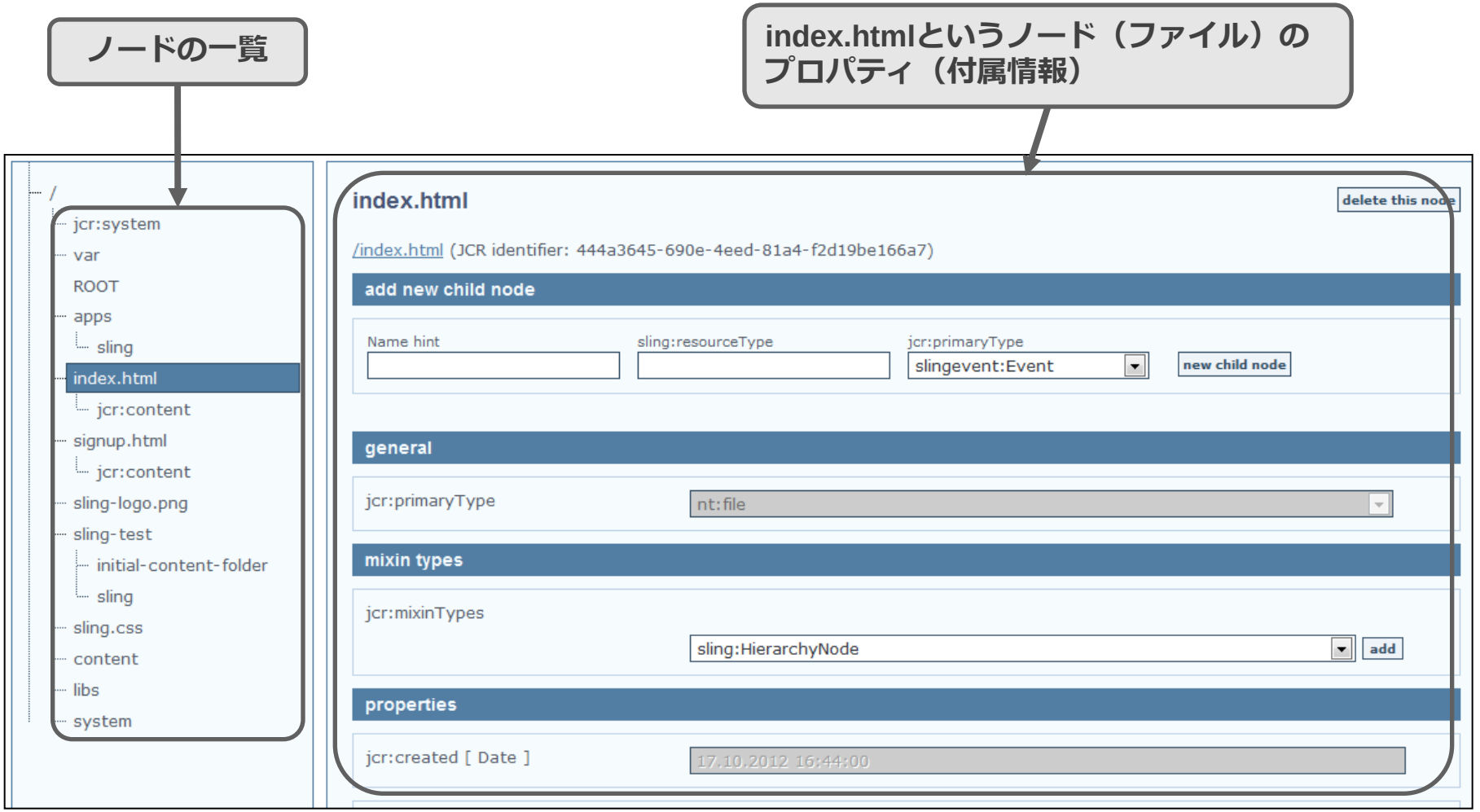
JCR(Javaコンテンツリポジトリ)

- Javaコンテンツリポジトリはノードとプロパティのツリーで構成される。
  - ノード：ファイルシステムにおけるファイルやフォルダ
  - プロパティ：ノードの付属情報



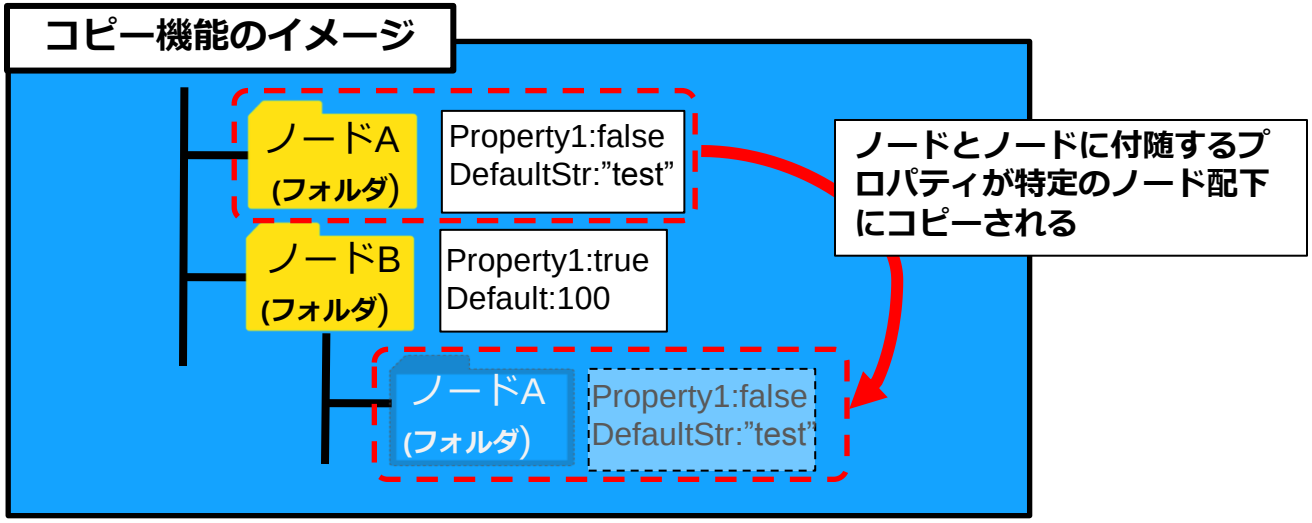
JCR(Javaコンテンツリポジトリ)

Apache Slingのノード管理画面



Apache Sling のコンテンツ管理機能

- Slingにはノードの作成・編集・削除・コピー・移動などを行う機能が存在する。
- コピー機能を使うと、指定されたノードと付随するプロパティを特定のノード配下にコピーすることができる。

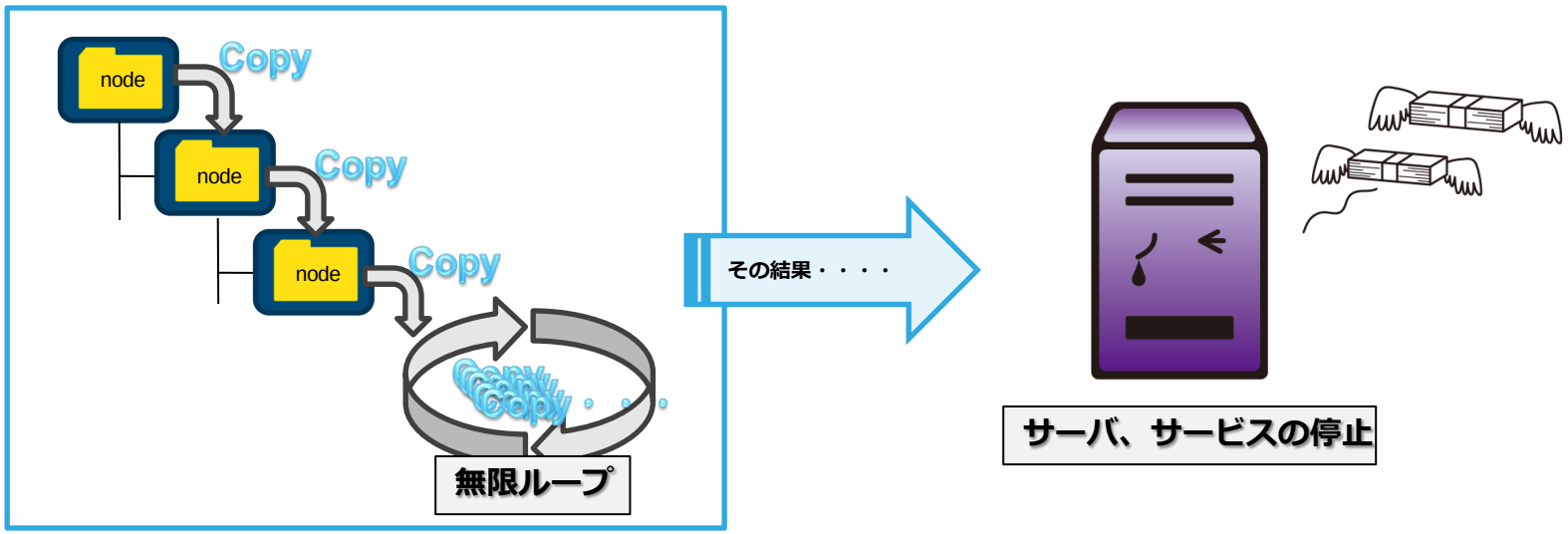


脆弱性の概要

- コピー機能にはサービス運用妨害(DoS)攻撃を受ける脆弱性が存在する。
- コピーの際に、コピー元とコピー先に同一のファイルパスを指定すると、無限ループが発生し、サービス運用妨害が成立する。

脆弱性が悪用された場合のリスク

- 無限ループによりCPU/メモリリソースが枯渇し、サーバが応答しなくなる。
- その結果、提供されているサービスが停止する。提供しているサービスによっては金銭的な被害が発生する可能性がある。



通常の処理

コンテンツをコピーする際の処理フロー

① クライアントからリクエストが送信される。

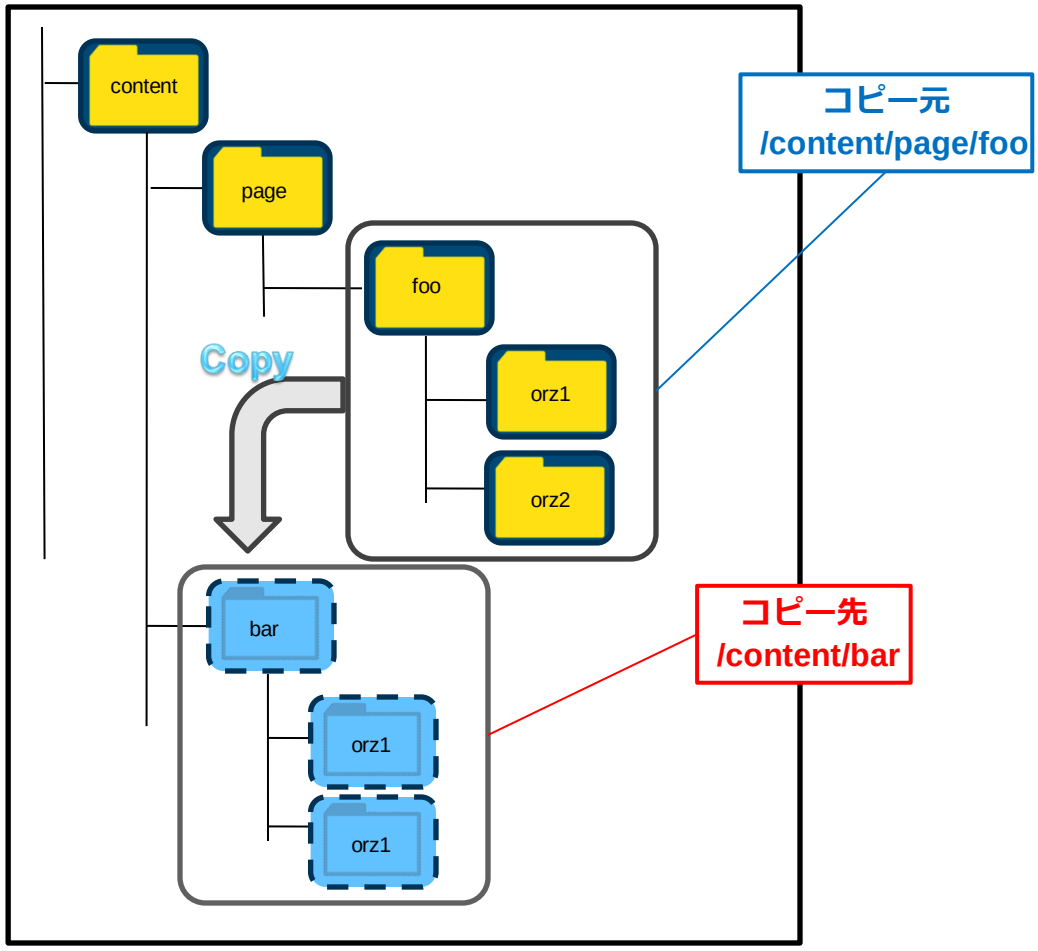
② アプリケーションがリクエストを受信し、コピー元とコピー先のパスの情報を取り出す。

③ CopyOperationクラスのCopyメソッドでコピー処理を行う。

④ 結果を含むレスポンスがクライアントへ送信される。

①クライアントからリクエストが送信される

ここでは、ノード「/content/page/foo」とその配下のノード「orz1」「orz2」を「/content」配下のノード「bar」としてコピーするリクエストが送られたとする。



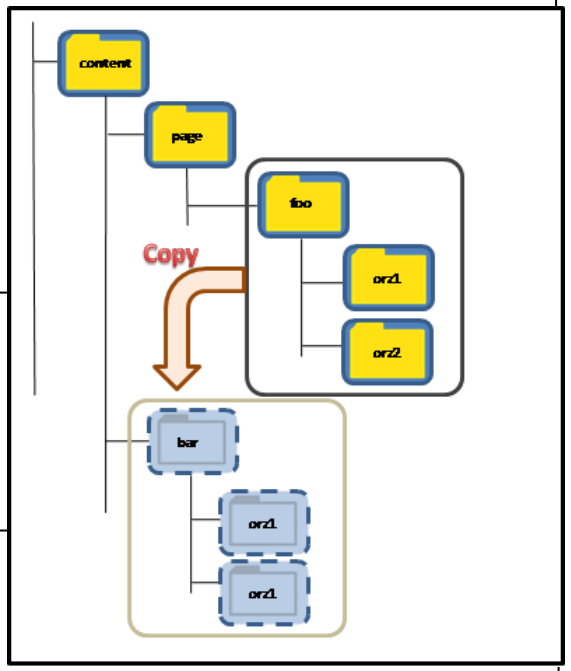
①クライアントからリクエストが送信される

HTTPリクエスト

コンテンツのコピー先のノードをパスで指定する

コンテンツのコピー元のノードをパラメータ@CopyFromで指定する。

```
POST /content/bar HTTP/1.1
Host: localhost:8080
:
@CopyFrom=/content/page/foo
```



上記HTTPリクエストを送信するためのHTML

```
<form action='/content/bar' method='POST'>
:
<input type='hidden' name='@CopyFrom' value='/content/page/foo'>
:
</form>
```

②アプリケーションがリクエストを受信し、コピー処理を開始する

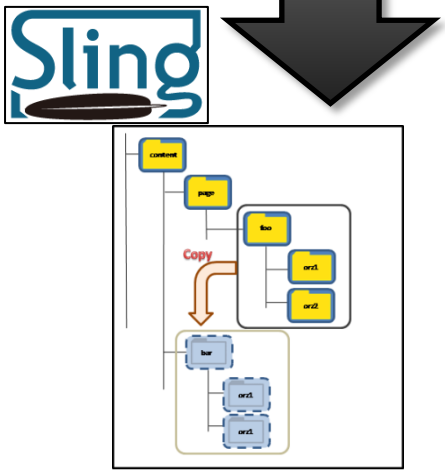
コピー処理はCopyOperationクラスのCopyメソッドで行われる。

HTTPリクエスト

```
POST /content/bar HTTP/1.1
Host: localhost:8080
:
@CopyFrom=/content/page/foo
```

経由するメソッド

SlingPostServlet.doPost()
ModifyOperation().run()
ModifyOperation.doRun()
ModifyOperation.processCopies()
ModifyOperation.processMovesCopiesInternal()
CopyOperation.copy()



CopyOperation.java

/content/page/foo

/content/bar

```
public class CopyOperation extends
AbstractCopyMoveOperation{
        :
        static Item copy(Node src, Node dstParent, String name)
```

③CopyOperationクラスのCopyメソッドでコピー処理を行う

CopyOperation.java

```
static Item copy(Node src, Node dstParent, String name)
        throws RepositoryException {

    // ensure destination name                                    ①
    if (name == null) {
        name = src.getName();
    }

    // ensure new node creation                                   ②
    if (dstParent.hasNode(name)) {
        dstParent.getNode(name).remove();
    }

    // create new node                                            ③
    Node dst = dstParent.addNode(name, src.getPrimaryNodeType().getName());
    for (NodeType mix : src.getMixinNodeTypes()) {
        dst.addMixin(mix.getName());
    }

    // copy the properties                                        ④
    for (PropertyIterator iter = src.getProperties(); iter.hasNext();) {
        copy(iter.nextProperty(), dst, null);
    }

    // copy the child nodes                                       ⑤
    for (NodeIterator iter = src.getNodes(); iter.hasNext();) {
        Node n = iter.nextNode();
        if (!n.getDefinition().isProtected()) {
            copy(n, dst, null);
        }
    }
    return dst;
}
```

CopyOperationクラスのCopyメソッドでは下記の5つの処理が行われる。

① コピー先のノード名を確認

② コピー先に既に同様の名前のノードが存在しないかを確認し、存在すれば削除

③ コピー先のノード作成

④ コピー元のノードからコピー先のノードへプロパティ情報をコピー

⑤ コピー元の子ノードをコピー先にコピー(同じメソッドが再帰的に呼ばれる)

③CopyOperationクラスのCopyメソッドでコピー処理を行う

CopyOperation.java

```
static Item copy(Node src, Node dstParent, String name)
        throws RepositoryException {

    // ensure destination name
    if (name == null) {
        name = src.getName();
    }

    // ensure new node creati
    if (dstParent.hasNode(na
        dstParent.getNode(name)
    }

    // create new node
    Node dst = dstParent.addNode(n                    odeType().getName());
    for (NodeType mix : src.getMixin
        dst.addMixin(mix.getName());
    }

    // copy the propert
    for (PropertyItera
        copy(iter.nextP
    }

    // copy the child no
    for (NodeIterator
        Node n = iter.ne
        if (!n.getDefini
            copy(n, dst, n
        }
    }
    return dst;
}
```

CopyOperationクラスのCopyメソッドでは下記の5つの処理が行われる。

① **コピー先のノード名を確認**

② コピー先に既に同様の名前のノードが存在しないかを確認し、存在すれば削除

③ コピー[illegible]ド作成

④ コピー元のノードからコピー先[illegible]コ[illegible]先[illegible]帰

コピー元
/content/page/foo

コピー先
/content/bar

```
static Item copy(Node src, Node dstParent, String name)
        throws RepositoryException {
    // ensure destination name
    if (name == null) {
        name = src.getName();
    }
```

コピー先のノード名(bar)

コンテンツのコピー先のノード名が指定される

③CopyOperationクラスのCopyメソッドでコピー処理を行う

CopyOperation.java

```
static Item copy(Node src, Node dstParent, String name)
        throws RepositoryException {

    // ensure destination name
    if (name == null) {
        name = src.getName();
    }

    // ensure new node creation
    if (dstParent.hasNode(name)) {
        dstParent.getNode(name).remove();
    }

    // create new node
    Node dst = dstParent.addNode(name, src.getPrimaryNodeType().getName());
    for (NodeType mix : src.getMixinNodeTypes()) {
        dst.addMixin(mix.getName());
    }

    // copy the properties
    for (PropertyIterator iter = src.getProperties(); iter.hasNext();) {
        copy(iter.nextProperty(), dst, null);
    }

    // copy the child nodes
    for (NodeIterator iter = src.getNodes(); iter.hasNext();) {
        Node n = iter.nextNode();
        if (!n.getDefinition().isProtected()) {
            copy(n, dst, null);
        }
    }
    return dst;
}
```

CopyOperationクラスのCopyメソッドでは下記の5つの処理が行われる。

① コピー先のノード名を確認

② **コピー先に既に同様の名前のノードが存在しないかを確認し、存在すれば削除**

③ コピー先のノード作成

④ コピー元のノードからコピー先のノードへプロパティ情報をコ

コピー先 /content/bar

コピー先のノード名(bar)

```
// ensure new node creation
if (dstParent.hasNode(name)) {

dstParent.getNode(name).remove();
}
```

コピー先(/content/)に同名のノード(bar)が存在しないか確認し、存在すれば削除する。

③CopyOperationクラスのCopyメソッドでコピー処理を行う

CopyOperation.java

```
static Item copy(Node src, Node dstParent, String name)
        throws RepositoryException {

    // ensure destination name
    if (name == null) {
        name = src.getName();
    }

    // ensure new node creation
    if (dstParent.hasNode(name)) {
        dstParent.getNode(name).remove();
    }

    // create new node
    Node dst = dstParent.addNode(name, src.getPrimaryNodeType().getName());
    for (NodeType mix : src.getMixinNodeTypes()) {
        dst.addMixin(mix.getName());
    }

    // copy the properties
    for (PropertyIterator
                                                    Next();) {
```

CopyOperationクラスのCopyメソッドでは下記の5つの処理が行われる。

① コピー先のノード名を確認

② コピー先に既に同様の名前のノードが存在しないかを確認し、存在すれば削除

③ **コピー先のノード作成**

④ コピー元のノードからコピー先のノードへプロパティ情報をコピー

コピー先 /content/bar

コピー先のノード名(bar)

```
// create new node
Node dst = dstParent.addNode(name, src.getPrimaryNodeType().getName());
for (NodeType mix : src.getMixinNodeTypes()) {
        dst.addMixin(mix.getName());
}
```

コピー先(/content/)にnameで指定されたノード(bar)を作成する。

③CopyOperationクラスのCopyメソッドでコピー処理を行う

CopyOperation.java

```
static Item copy(Node src, Node dstParent, String name)
        throws RepositoryException {

    // copy the properties
        for (PropertyIterator iter = src.getProperties(); iter.hasNext();) {
            copy(iter.nextProperty(), dst, null);
        }

    // create new node
    Node dst = dstParen                                  tPrimaryNodeType().getName());
    for (NodeType mix :                                  Types()) {
        dst.addMixin(mix.ge
    }

    // copy the properties
    for (PropertyIterator ite = src.getProperties(); iter.hasNext();) {
        copy(iter.nextProperty(), dst, null);
    }

    // copy the child nodes
    for (NodeIterator iter = src.getNodes(); iter.hasNext();) {
        Node n = iter.nextNode();
        if (!n.getDefinition().isProtected()) {
            copy(n, dst, null);
        }
    }
    return dst;
}
```

コピー元
/content/page/foo

コピー先
/content/bar

コピー元(/content/page/foo)のプロパティをコピー先(/content/bar)にコピーする。ここで呼び出されているcopyメソッドは実行中のメソッドとは別のオーバーロードされたメソッド。

③ コピー先のノード作成

④ **コピー元のノードからコピー先のノードへプロパティ情報をコピー**

⑤ コピー元の子ノードをコピー先にコピー(同じメソッドが再帰的に呼ばれる)

③CopyOperationクラスのCopyメソッドでコピー処理を行う

CopyOperation.java

```
static Item copy(Node src, Node dstParent, String name)
        throws RepositoryException {
    // copy the child nodes
    for (NodeIterator iter = src.getNodes(); iter.hasNext();) {
        Node n = iter.nextNode();
        if (!n.getDefinition().isProtected()) {
            copy(n, dst, null);
        }
    }
```

コピー元
/content/page/foo

コピー先
/content/bar

コピー元に子ノードが存在する場合は再帰的にcopyメソッドがコールされ、コピー処理が行われる。

```
    copy(iter.nextProp
  }

  // copy the child nodes
  for (NodeIterator iter = src.getNodes(); iter.hasNext();) {
      Node n = iter.nextNode();
      if (!n.getDefinition().isProtected()) {
          copy(n, dst, null);
      }
  }
  return dst;
}
```

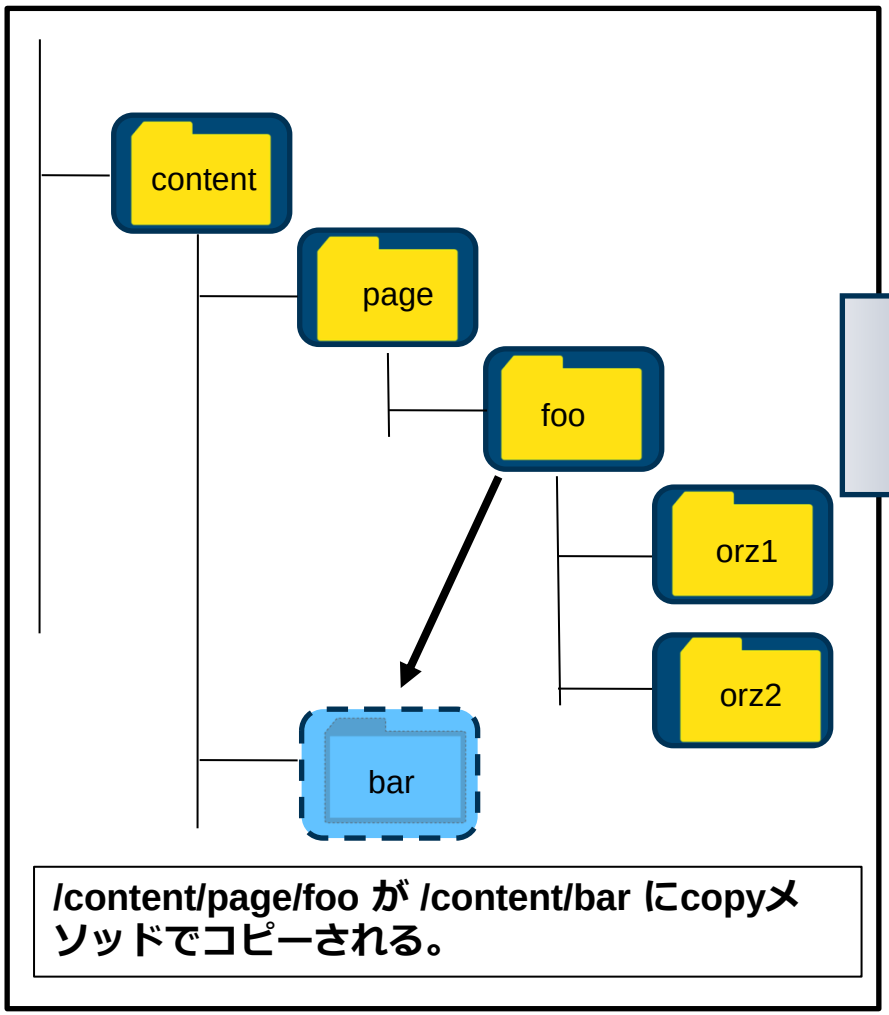
③ コピー先のノード作成

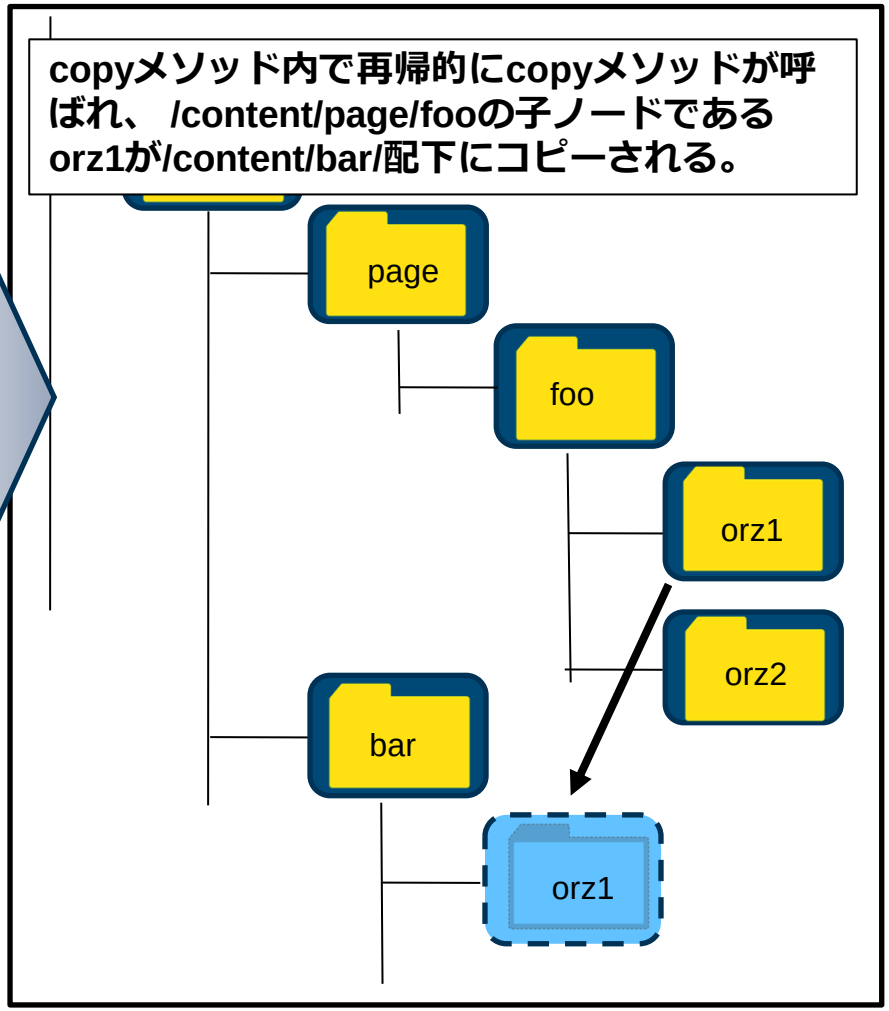
④ コピー元のノードからコピー先のノードへプロパティ情報をコピー

⑤ **コピー元の子ノードをコピー先にコピー(同じメソッドが再帰的に呼ばれる)**

③CopyOperationクラスのCopyメソッドでコピー処理を行う

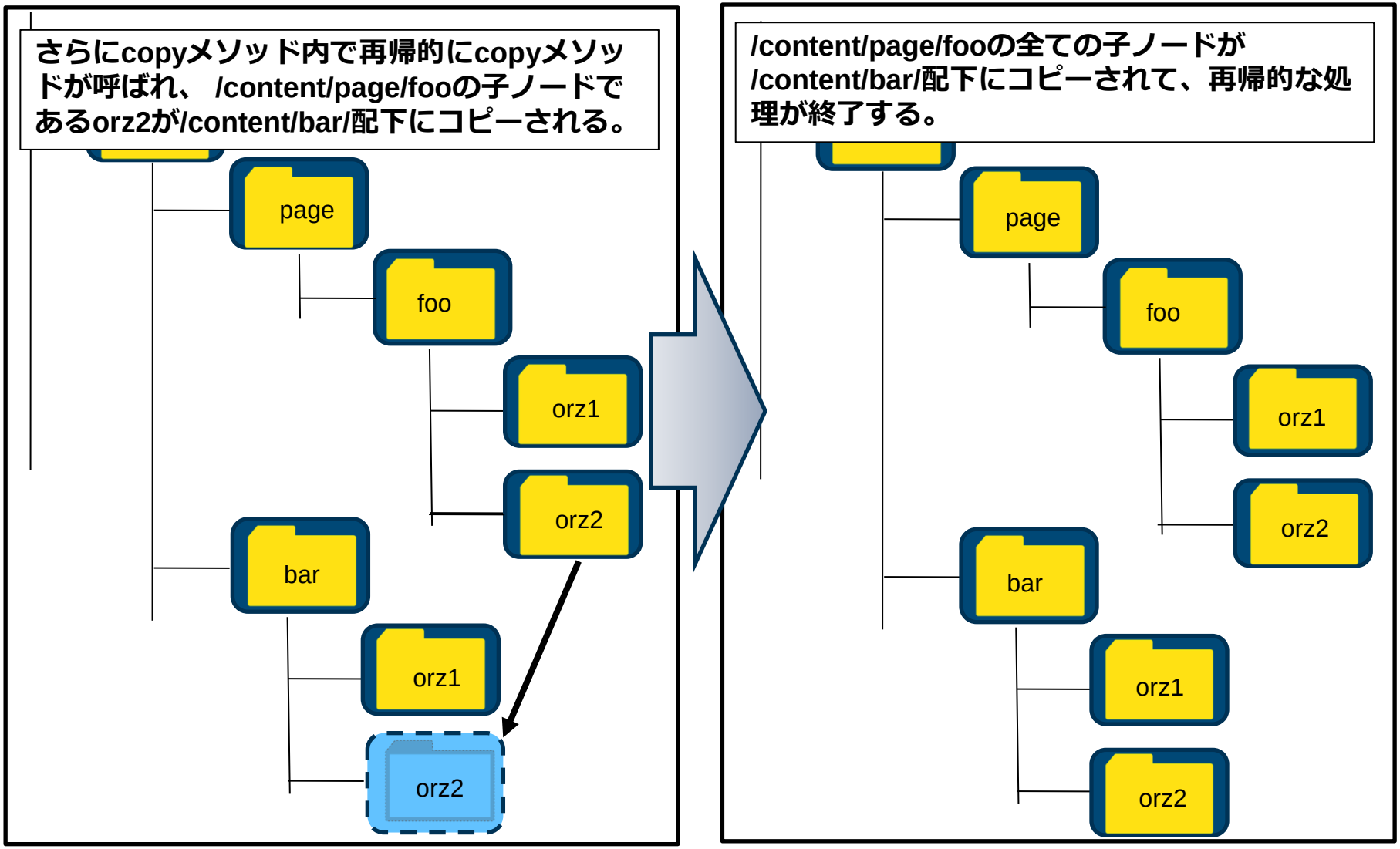
■子ノードに対する再帰的なコピー処理





③CopyOperationクラスのCopyメソッドでコピー処理を行う

■子ノードに対する再帰的なコピー処理



④レスポンスがクライアントへ送信される。

コピー処理が終了し、結果をレスポンスとしてクライアントへ送信する。

Content created /content/bar/3_1350523654050

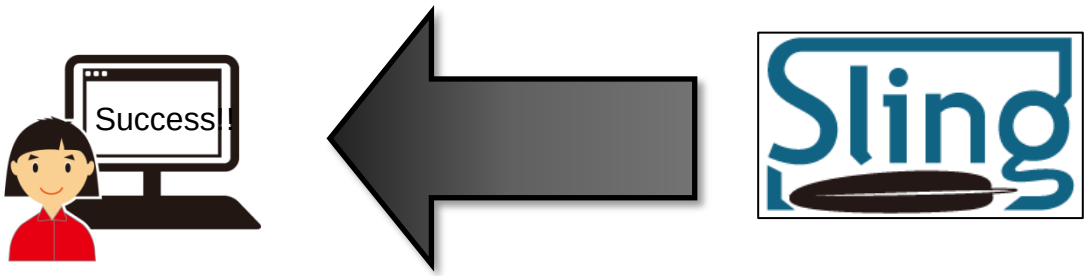
Status	201
Message	Created
Location	/bar/3_1350523654050
Parent Location	/bar
Path	/content/bar/3_1350523654050
Referer	

ChangeLog	<pre>created("/content/bar/3_1350523654050");
	deleted("/content/bar/3_1350523654050");
	copied("/content/page/foo", "/content/bar/3_1350523654050");
	</pre>

Go Back

Modified Resource

Parent of Modified Resource



攻撃コード

攻撃コードのHTTPリクエスト

コピー先ノード

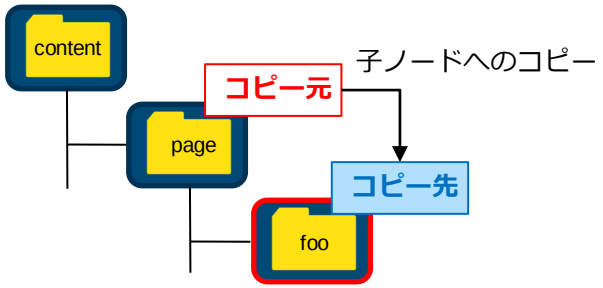
```
POST /content/page/foo HTTP/1.1
Host: localhost:8080
:
@CopyFrom=/content/page/
```

コピー元ノード

攻撃コードのHTTPリクエストを送信するためのHTML

```
<form action='/content/page/foo' method='POST'>
:
<input type='hidden' name='@CopyFrom' value='/content/page/'>
:
</form>
```

■攻撃コードのポイント
コピー対象のノードに対し、自分自身の子ノード配下に自身をコピーするように指定している。
(親ノード→子ノードのコピー)



攻撃コード: 攻撃コードを受け取った際の処理

コンテンツをコピーする際の処理フロー

① クライアントからリクエストが送信される

② アプリケーションがリクエストを受信し、コピー元とコピー先の値を取り出す

③ コピー処理を行う

④ 結果を含むレスポンスがクライアントへ送信される

攻撃コードが実行された際の処理のフローは③のコピー処理にて無限ループが発生する。

攻撃コード実行時

③CopyOperationクラスのCopyメソッドでコピー処理を行う

CopyOperation.java

```
static Item copy(Node src, Node dstParent, String name)
        throws RepositoryException {

    // ensure destination name
    if (name == null) {
        name = src.getName();
    }

    // ensure new node creation
    if (dstParent.hasNode(name)) {
        dstParent.getNode(name).remove();
    }

    // create new node
    Node dst = dstParent.addNode(name, src.getPrimaryNodeType().getName());
    for (NodeType mix : src.getMixinNodeTypes()) {
        dst.addMixin(mix.getName());
    }

    // copy the properties
    for (PropertyIterator iter = src.getProperties(); iter.hasNext();) {
        copy(iter.nextProperty(), dst, null);
    }

    // copy the child nodes
    for (NodeIterator iter = src.getNodes(); iter.hasNext();) {
        Node n = iter.nextNode();
        if (!n.getDefinition().isProtected()) {
            copy(n, dst, null);
        }
    }
    return dst;
}
```

CopyOperationクラスのCopyメソッドでは下記の5つの処理が行われる。

① コピー先のノード名を確認

② コピー先に既に同様の名前のノードが存在しないかを確認し、存在すれば削除

③ コピー先のノード作成

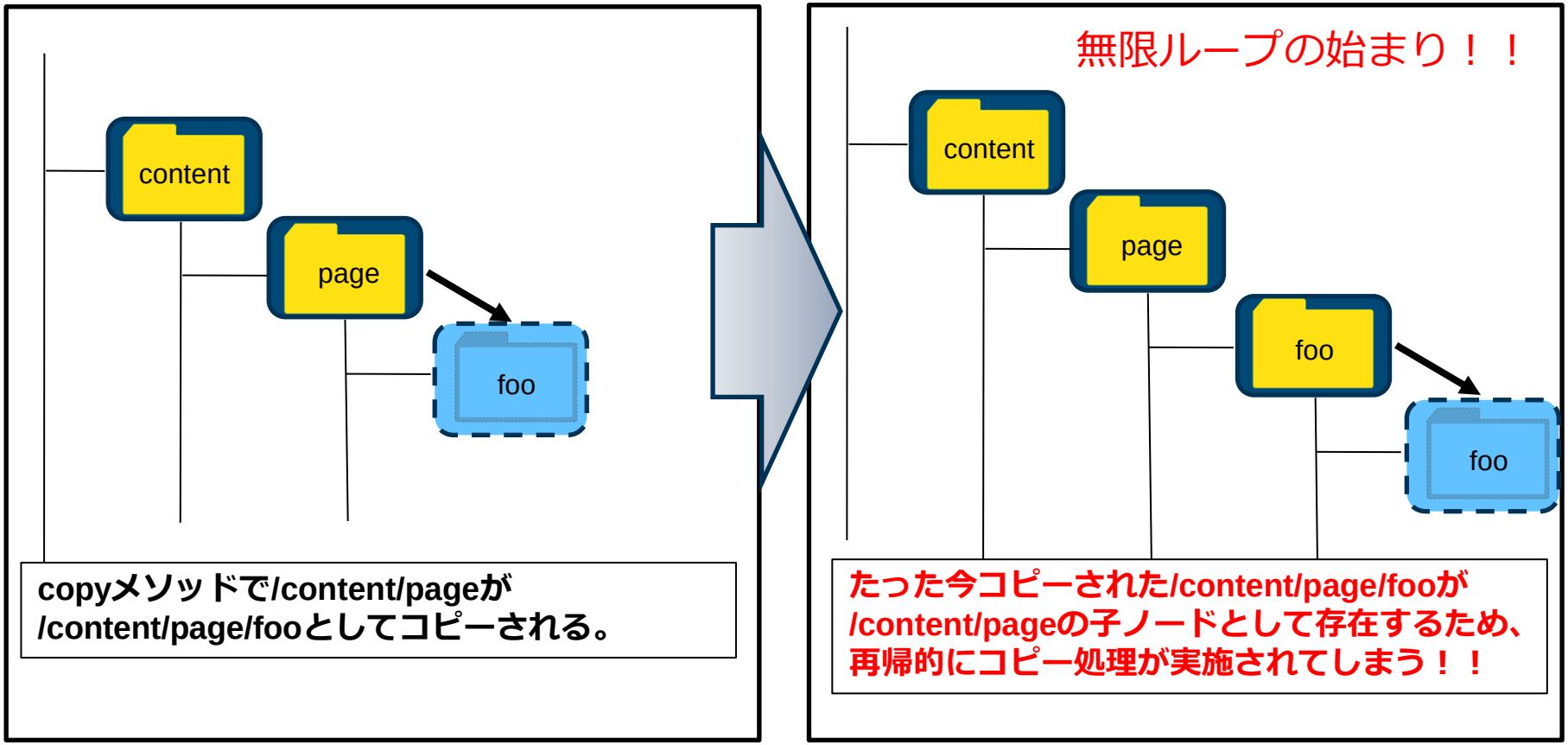
④ コピー元のノードからコピー先のノードへプロパティ情報をコピー

⑤ **コピー元の子ノードをコピー先にコピー(同じメソッドが再帰的に呼ばれる)**

copyメソッド内の5番目の処理である、子ノードを再帰的にコピーする処理でループが発生する！！

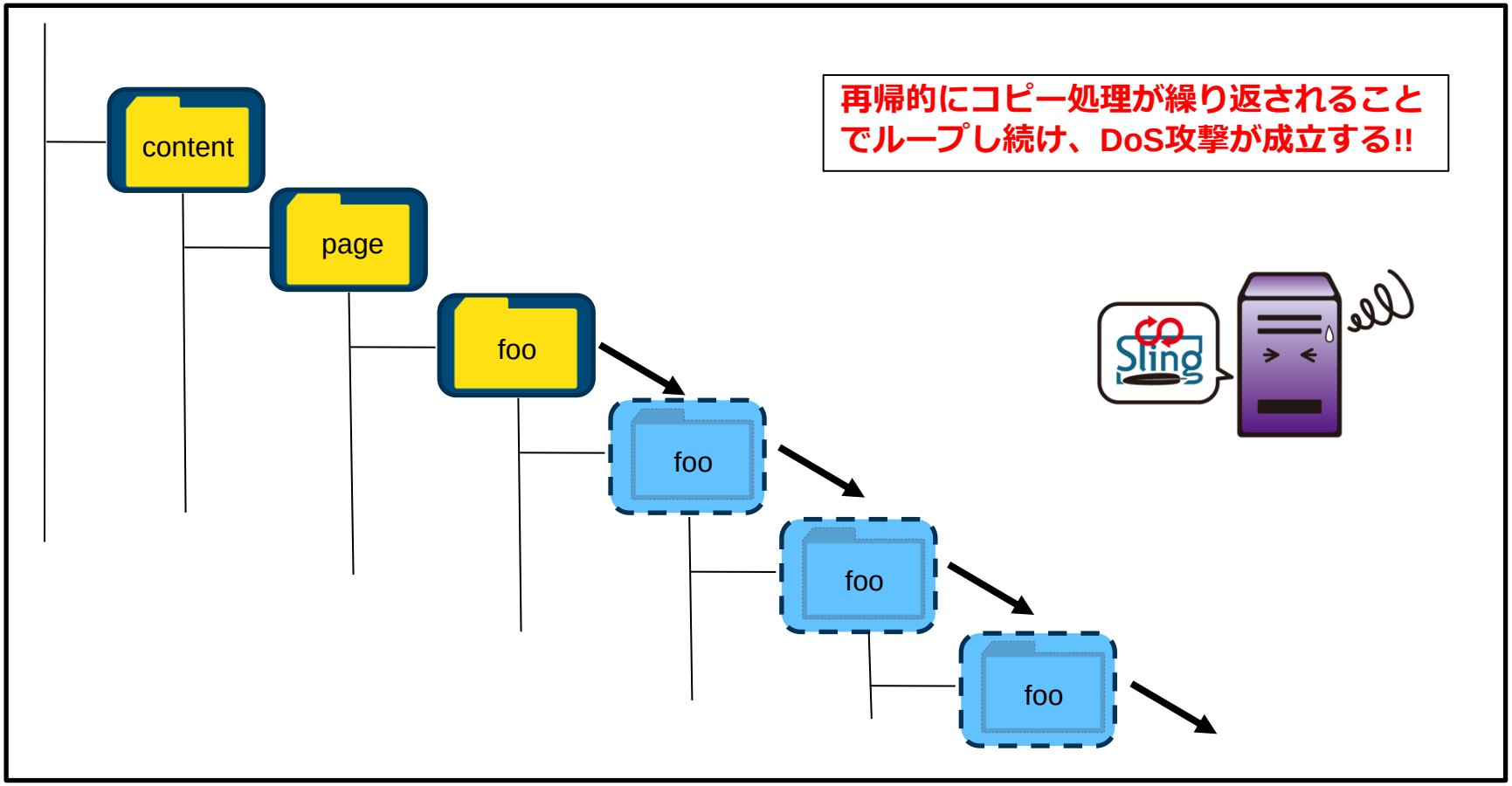
攻撃コード実行時
③CopyOperationクラスのCopyメソッドでコピー処理を行う

■コピー元に「/content/page/」、コピー先に「/content/page/foo」を指定した際のコピー処理



攻撃コード実行時
③CopyOperationクラスのCopyメソッドでコピー処理を行う

■コピー元に「/content/page/」、コピー先に「/content/page/foo」を指定した際のコピー処理



問題点とその対策

- **今回のアプリケーションにおける具体的な問題点**

  copyメソッドではメソッドの引数を検証せずにコピー処理を行っていた。

以下のコーディングガイドに違反している！！

「MET00-J メソッドの引数を検証する」

- **問題点に対してどうすべきだったか**

  処理対象のデータ構造（今回の場合はツリー構造）を考慮して、copyメソッドの引数であるdstParentノードがsrc以下のノードでないことを確認した上で、コピー処理を行うべきだった。

修正版コード

脆弱性はバージョン2.1.3にて修正が適用されている

コンテンツをコピーする際の処理フロー

① クライアントからリクエストが送信される。

② アプリケーションがリクエストを受信し、コピー元とコピー先の値を取り出す

③ コピー処理を行う。

④ 結果を含むレスポンスがクライアントへ送信される。

ループが発生するコピー処理のコードが修正されている。

修正前/修正後の処理比較

③CopyOperationクラスのCopyメソッドでコピー処理を行う

修正前の処理

CopyOperationクラスのCopyメソッドでは下記5つの処理が行われる。

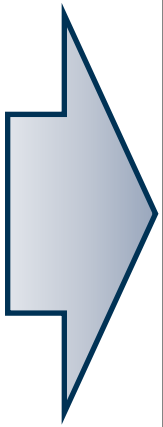
① コピー先のノード名を確認

② コピー先に既に同様の名前のノードが存在しないかを確認し、存在すれば削除

③ コピー先のノード作成

④ コピー元のノードからコピー先のノードへプロパティ情報をコピー

⑤ コピー元の子ノードをコピー先にコピー（同じメソッドが再帰的に呼ばれる）



修正後の処理

CopyOperationクラスのCopyメソッドでは下記6つの処理が行われる。

⓪ isAncestorOrSameNodeメソッドでコピー元とコピー先が同じツリーでないことを確認する

① コピー先のノード名を確認

② コピー先に既に同様の名前のノードが存在しないかを確認し、存在すれば削除

③ コピー先のノード作成

④ コピー元のノードからコピー先のノードへプロパティ情報をコピー

⑤ コピー元の子ノードをコピー先にコピー(同じメソッドが再帰的に呼ばれる)

修正版コード

③CopyOperationクラスのCopyメソッドでコピー処理を行う

CopyOperation.java

```
static Item copy(Node src, Node dstParent, String name)
        throws RepositoryException {

    if(isAncestorOrSameNode(src, dstParent)) {
        throw new RepositoryException(
       "Cannot copy ancestor " + src.getPath() + " to descendant " +
       dstParent.getPath());
    }

    // ensure destination nam
    if (name == null) {
        name = src.getName
    }
:
}
```

CopyOperationクラスのCopyメソッドでは下記6つの処理が行われる。

⓪ isAncestorOrSameNodeメソッドでコピー元とコピー先が同じツリーでないことを確認する

① コピー先のノード名を確認

② コピー先に既に同様の名前のノードが存在しないかを確認し、存在すれば

```
if(isAncestorOrSameNode(src, dstParent)) {
        throw new RepositoryException(
        "Cannot copy ancestor " + src.getPath() + " to descendant " +
        dstParent.getPath());
    }
```

コピー先のノード情報

コピー元のノード情報

修正版コード

③CopyOperationクラスのCopyメソッドでコピー処理を行う

■isAncestorOrSameNode**メソッドではコピー元とコピー先のノードに対して３つのチェックを行っている。**

isAncestorOrSameNodeメソッド

```
static boolean isAncestorOrSameNode(Node src, Node dest) throws
RepositoryException {

    if(src.getPath().equals("/")) {
        return true;

    } else if(src.getPath().equals(dest.getPath())) {
        return true;

    } else if(dest.getPath().startsWith(src.getPath() + "/")) {
        return true;

    }
    return false;
}
```

①コピー元がルートディレクトリでないか

②コピー元とコピー先が同一のパスでないか

③コピー元とコピー先が同一のツリーでないか

修正版コード

③CopyOperationクラスのCopyメソッドでコピー処理を行う

■isAncestorOrSameNodeメソッドでのチェック内容のまとめ

下記の3つのチェックにより、無限ループが発生するようなコピー元/コピー先の組み合わせが指定されていたら、エラーとして処理する。

① コピー元がルートディレクトリでないか
② コピー元とコピー先が同一のパスでないか
③ コピー元とコピー先が同一のツリーでないか

■修正版コードに攻撃コードを実行するとどうなるか

コピー元に「/content/page/」、コピー先に「/content/page/foo」が指定されている

↓

上記②のチェックでエラーとなり、無限ループは発生しない!!

まとめ

■ この脆弱性から学べるプログラミングの注意点
- アプリケーションの処理内容や、扱うデータ構造に応じた引数の検証をすべき
    - 今回のコピー処理は再帰的な動作であり、コピー対象の指定の仕方によっては無限ループが発生する可能性があることを考慮すべきであった。

■ 上記への対策
- 再帰的な処理を実装する場合、無限ループにならない条件(あるいは無限ループになる条件)を明確にし、その条件を確認するコードを必ず入れる

著作権・引用や二次利用について

■本資料の著作権はJPCERT/CCに帰属します。

■本資料あるいはその一部を引用・転載・再配布する際は、引用元名、資料名および URL の明示をお願いします。

記載例

引用元：一般社団法人JPCERTコーディネーションセンター
Java アプリケーション脆弱性事例解説資料
Apache Sling におけるサービス運用妨害(無限ループ)の脆弱性
https://www.jpcert.or.jp/securecoding/2012/No.01_Apache_Sling.pdf

■本資料を引用・転載・再配布をする際は、引用先文書、時期、内容等の情報を、JPCERT コーディネーションセンター広報(office@jpcert.or.jp)までメールにてお知らせください。なお、この連絡により取得した個人情報は、別途定めるJPCERT コーディネーションセンターの「プライバシーポリシー」に則って取り扱います。

本資料の利用方法等に関するお問い合わせ

JPCERTコーディネーションセンター
広報担当
E-mail：office@jpcert.or.jp

本資料の技術的な内容に関するお問い合わせ

JPCERTコーディネーションセンター
セキュアコーディング担当
E-mail：secure-coding@jpcert.or.jp