サイバーセキュリティ Assignment 3
Incident 2

Group 6

2020 年 7 月 14 日

担当者 SubgroupB

学生番号	氏名	所属研究室
2011156	成 泰鏞	ソフトウェア工学研究室
2011231	久田 祥平	ソーシャルコンピューティング研究室
2011271	森本 康太	情報セキュリティ工学研究室

Step 1

Question

For each incident, provide a detailed summary of the incident report toward cybersecurity professionals, by fully mobilizing concepts that you learned through these lectures. Create a timeline of the incident, as well as a fishbone diagram, that illustrates root cause and other important events in order to help with the understanding.

Answer

我々は，2017 年 2 月に発生した Web コンテンツ管理システム WordPress を利用する Web ページが改ざんされるインシデントについて調査した．世界中で 150 万以上の Web ページが改造され，国内でも行政や議員の Web サイトが改ざん被害を受けた．改ざんの被害自体はいたずらのような改変のみで，フィッシングサイトやコンテンツにマルウェアを仕込むような改変は報告されておらず，個人情報の流出といった深刻なものはなかった．しかしながら，病院や当時の五輪相といった公的な情報を提供するホームページも改ざんされた[*1] ことから，影響は小さくないと言える．WordPress の脆弱性が原因となって引き起こされたインシデントレポートも報告されている．日本小児循環器学会のインシデントレポート[*2] によると，ホームページのタイトルが改ざんなどの被害が報告されている．この脆弱性に対して，Web コンテンツセキュリティ会社，Sucuri 社が WordPress のチームに脆弱性を通知した．1 週間後のアップデートによって，該当する脆弱性は修正された．WordPress は，脆弱性の内容を公開しないことで被害を最小限にしようと試みた．しかしながら，脆弱性公開から 2 日以内に概念実証 (以下 PoC) 方法が，Web 上で共有されたことによって，攻撃手法が広く拡散され，改ざんされる事例は急増した．また，この改ざんの二次被害としては，サーチエンジンが改ざんされたページを発見することで，ページランクが下がることが考えられる．WordPress のこの脆弱性に対して，CVE-2017-1001000 [*3] が発行されている．これによると，機密性や可用性への影響はなく，完全性への影響も限定的ではあるもの，脆弱性へのアクセスが非常に容易なことより CVSS3.0 スコアは 7.5 となっている．

[*1] サイト改ざん相次ぐ　病院や大学，五輪相も被害: 日本経済新: https://r.nikkei.com/article/DGXLASDG06HFY_W7A200C1CC1000?s=5

[*2] http://jspccs.jp/wp-content/uploads/140206.pdf

[*3] NVD - CVE-2017-1001000 : https://nvd.nist.gov/vuln/detail/CVE-2017-1001000

表 1　インシデントタイムライン

2016 年 8 月 16 日	WordPress4.7 がリリース，REST API が追加される
2016 年 12 月 6 日	WordPress4.7 がリリース，REST API コンテンツエンドポイントが追加される
2017 年 1 月 11 日	WordPress4.7.1 がリリース，4.7 から 61 件のバグが修正される
2017 年 1 月 20 日	Sucuri 社が Wordpress に脆弱性について警告
2017 年 22〜23 日	Sucuri や SiteLock，Cloudflare，Incapsula のような Web セキュリティー会社が悪意のある攻撃を防ぐ Firewall をそのカスタマーに追加
2017 年 26 日	WordPress4.7.2 をリリース
2017 年 2 月 1 日	WordPress4.7 や 4.7.1 の脆弱性の内容が公開される．
2017 年 2 月 2 日	脆弱性に対する PoC が Web ページの改ざん方法がオンライン上で共有され，攻撃を試みる事例が多数発生し始める．
2017 年 2 月 6 日	日本国内における注意喚起が行われる．

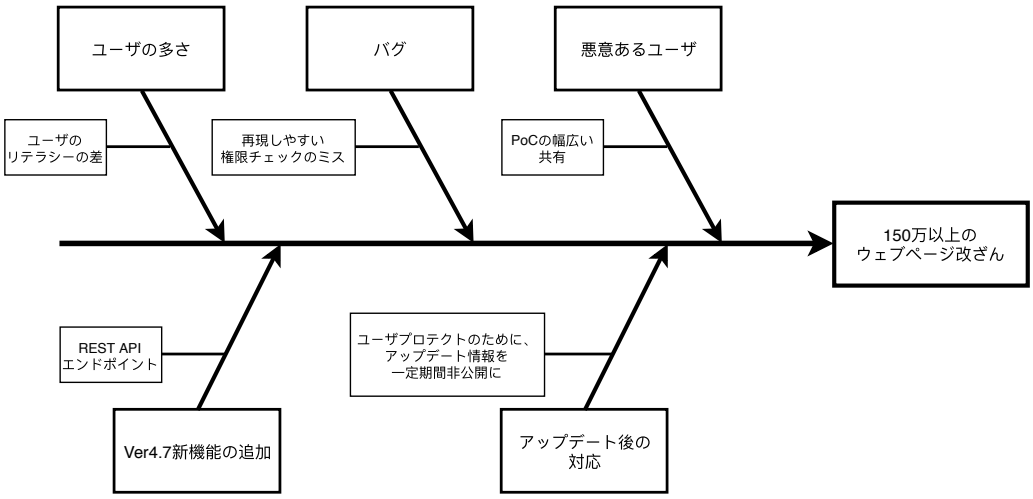

図 1　特性要因図

Step 2

Question

Identify technical factors that led to each incident. Based on the real timeline of the particular incident, identify two critical periods where technical intervention was necessary.

Answer

WordPress の REST API が，インシデントの原因として最大のものであった．WordPress では以前から REST API がプラグインとして用意されていたが，WordPress 4.7 以降で WordPress Core にバンドルしたことで，API で様々な機能が実行できるようになった．今回の脆弱性は，この REST API にあった．次に，REST　API のコードを用いて，脆弱性の技術的要因について述べる．攻撃者は，認証を回避して，id=1 のコンテンツの改変を試みたとする．WordPress の REST API の内部では，以下の 2 つのメソッドが処理を行う．

- `update_item_permissions_check` (権限の確認)
- `update_item` (コンテンツの変更)

update_item_permissions_check メソッド[*4] のソースコード 1 において，id に指定されたコンテンツの性質による，メソッドの返り値は表 2 の通りである．

表 2　メソッドの返り値

コンテンツの性質	返り値
存在しないコンテンツ	true
存在し権限のあるコンテンツ	true
存在し権限のないコンテンツ	false

ソースコード 1　update_item_permissions_check メソッド

```
497:  public function update_item_permissions_check( $request ) {
498:    $post = get_post( $request['id'] );
499:    $post_type = get_post_type_object( $this->post_type );
500:    if ( $post && ! $this->check_update_permission( $post ) ) {
501:      return new WP_Error( 'rest_cannot_edit', __( 'Sorry, you are not allowed to edit this post...
502:    }
503:    if ( ! empty( $request['author'] ) && get_current_user_id() !== $request['author'] &&
                ! current_user_can( $post_type->cap->edit_others_posts ) ) {
504:      return new WP_Error( 'rest_cannot_edit_others', __( 'Sorry, you are not allowed to update ...
505:    }
506:    if ( ! empty( $request['sticky'] ) && ! current_user_can( $post_type->cap->edit_others_posts ) ) {
507:      return new WP_Error( 'rest_cannot_assign_sticky', __( 'Sorry, you are not allowed to make ...
508:    }
509:    if ( ! $this->check_assign_terms_permission( $request ) ) {
510:      return new WP_Error( 'rest_cannot_assign_term', __( 'Sorry, you are not allowed to assign ...
511:    }
512:    return true;
513:  }
```

表 2 から，存在しないコンテンツが指定された場合，update_item_permissions_check メソッド最後の return 文 にて true が返されることになる．しかし，この「存在しないコンテンツ」については，ソースコード 2 の update_item メソッドの 526 行目にてエラーが返される予定であった．

ソースコード 2　update_item メソッド

```
523:  public function update_item( $request ) {
524:    $id   = (int) $request['id'];
525:    $post = get_post( $id );
526:    if ( empty( $id ) || empty( $post->ID ) || $this->post_type !== $post->post_type ) {
527:      return new WP_Error( 'rest_post_invalid_id', __( 'Invalid post ID.' ), array( 'status' => 404 ) );
528:    }
529:    $post = $this->prepare_item_for_database( $request );
530:    if ( is_wp_error( $post ) ) {
531:      return $post;
532:    }
```

ここで，id=1A が指定されたときに，update_item_permissions_check メソッドと update_item メソッドの両方で呼ばれている get_post 関数が受け取るパラメータを確認すると， update_item_permissions_check では，get_post('1A') が呼ばれ，1A を ID とするコンテンツはないため，「コンテンツは存在しない」が返され，チェック結果は true となる．一方，update_item メソッドは，$id を整数にキャストしているため，get_post(1) が呼ばれ，ID=1 のコンテンツが変更されることなる．これにより，本来権限のないコンテンツ ID=1 に対する更新ができてしまう．これより，不正な id に対して，REST API 内で id の処理に不整合が生じる．これが今回のインシデントの技術的要因である[*5] ．

次に，技術的に介入が必要とされた時期について述べる．1 つ目の介入点は，Ver4.7 のリリース前であると考えられる．WordPress4.7 では RestAPI コンテンツエンドポイントが追加されている．Ver4.6 比較し，Rest API を通じて以下の機能を編集できるようになっていた．

1. 投稿記事 (Post) [一覧の取得・表示・追加・更新・削除・revision 取得]
2. 固定ページ (Page) [一覧の取得・表示・追加・更新・削除・revision 取得]

[*4] wp-includes/rest-api/endpoints/class-wp-rest-posts-controller.php (ver4.7.1)

[*5] 参考 https://blog.tokumaru.org/2017/02/wordpress-4.7.1-Privilege-Escalation.html

3. メディア [一覧の取得・表示・追加・更新・削除]
4. コンテントタイプの種類 [表示]
5. カテゴリー [一覧の取得・表示・追加・更新・削除]
6. タグ [一覧の取得・表示・追加・更新・削除]
7. カスタムタクソノミー [表示・編集]
8. WordPress ユーザ [一覧の取得・表示・追加・更新・削除]
9. コメント [一覧の取得・表示・追加・更新・削除]

これらの機能を維持するために，REST API はデフォルトの状態で ON となっており，これが被害の拡大につながったと考えられる．一般のユーザは，このような API を頻繁に利用するとは考えにくい．それゆえ，デフォルトでは REST API の状態を OFF にするべきであり，必要な場合のみ ON にするべきであると考えられる．

2 つ目の介入点は，4.72 のリリース時と考えられる．アップデートから情報公開までに猶予期間を設けたり，Web コンテンツセキュリティ会社を通じて WAF を実装したりと対策はなされていたが，アップデートを強制したり，以前のバージョンの利用を禁止したりと，ユーザーに強制介入できる要素を設けておく必要はあったと考えられる．HP 管理者も，セキュリティーアップデートと言う形の Ver4.72 が公開された時点でアップデートすれば，攻撃が実際に起こるまではタイムラグがあるので，Web ページの改ざんを未然に防ぐことはできたと考えられる．

Step 3

Question

Identify human factors that led to each incident. Based on the real situation of the particular incident, describe how best you would deliver risk messages, as an external security consultant, in the two critical periods that you identified in the step 2.

Answer

インシデントの原因となった人的要因は, 以下の 3 つが考えられる．

- 開発者
- 悪意のあるユーザ
- WordPress 使用者の HP 管理者

まず，開発者について述べる．プログラムの傾向として，一部の変数を利用直前に int に変更している．これが潜在的なバグや脆弱性の原因となっていた．入力値について,「数値以外のものをエラーとする」などの変数管理の習慣が必要ではあった．今回はチェックと更新とで異なる入力を用いてたため，両者の不整合がチェック漏れの原因となっている．アプリケーションにおいて API は外部との境界線になるため, 入念なチェックが必要である．次に悪意のあるユーザーについて述べる．悪意のあるユーザーが脆弱性に対する PoC が Web ページの改ざん方法がオンライン上で共有を行った．その結果, 他の悪意のあるユーザーが改ざんを行い被害が広がった．最後に WordPress 使用者の HP 管理者について述べる．サイトを放置している HP 管理者や仕様変更を拒みアップデートを行わない管理者が多く存在した．実際, 最新のバージョンを使用しているユーザーは, 42.8% と半分も存在しない[*6] ．そのためユーザーのリテラシーが低く, 攻撃対象となるサイトが増えてしまい被害が広がった．

次に，リスクメッセージについて，以下の時点で配信するべきと考える．まず，2016 年 8 月 16 日: WordPress4.7 リリース時点．WordPress 開発者に対して, API のリスクについて伝え, 脆弱性のチェックを行うことを推奨する. 特に，ver 4.7 では REST　API を通じて多くの機能が追加されている．また，過去にも API の追加が脆弱性となり，様々なインシデントが発生している点を考慮し，脆弱性のチェックや通常時よりも入念なテストを呼びかけるメッ

[*6] https://ja.wordpress.org/about/stats/

セージが必要であると考える．次に，2017 年 1 月 26 日 WordPress4.72 のリリース時点．WordPress 使用者に対してアップデートを強く促すメールを配信することを推奨する．通常，WordPress はアップデートの度に更新通知をメールで配信している．ver 4.72 のリリースは，脆弱性を修正する点で，通常より緊急性が高いアップデートであると言える．それゆえ，通常の更新通知ではなく，ユーザに重要度と緊急度高いアップデートであることを強調する必要があったと考える．