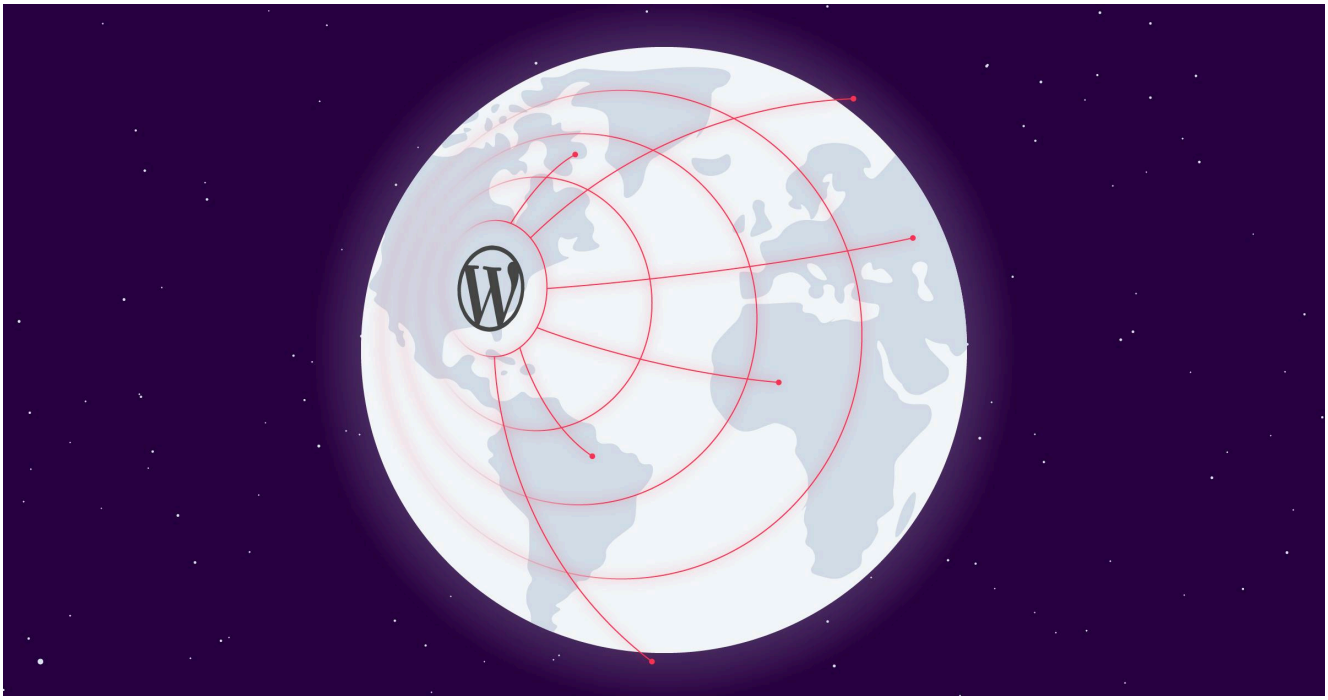
WordPress Core – Unauthenticated Blind SSRF



WordPress Core – Unauthenticated Blind SSRF

Our security researchers were surprised to discover a low-hanging code vulnerability in WordPress Core that we will discuss in this blog post.
by simon scannell and thomas chauchefoin|September 06, 2022

WordPress ist das weltweit beliebteste Content-Management-System und wird von über 40 % aller Websites verwendet . Diese breite Akzeptanz macht es zu einem Top-Ziel für Bedrohungsakteure und Sicherheitsforscher, die für das Melden von Sicherheitsproblemen über ihr öffentliches Bug-Bounty-Programm bezahlt werden.

Vulnerability Broker sind auch sehr daran interessiert, ungepatchte Schwachstellen zu erwerben, die es ihnen ermöglichen, WordPress-Instanzen zu übernehmen, und bieten manchmal bis zu 300.000 US-Dollar für kritische

Schwachstellen. Als solches hat WordPress eine stark überprüfte Codebasis, in der von Forschern nicht mehr erwartet wird, dass sie niedrig hängende Früchte finden. Unsere bisherigen Recherchen zu diesem Ziel erforderten umfangreiche Fachkenntnisse und Anstrengungen, um Sicherheitsprobleme aufzudecken.

Dieser Blogbeitrag beschreibt eine überraschend einfache Schwachstelle in der WordPress-Implementierung von Pingbacks. Während die Auswirkungen dieser Schwachstelle im Falle von WordPress für die meisten Benutzer gering sind, ist das damit verbundene anfällige Codemuster ziemlich interessant zu dokumentieren, da es wahrscheinlich auch in den meisten Webanwendungen vorhanden ist. Das Ziel dieses Blogbeitrags ist es, über dieses Muster aufzuklären und das Bewusstsein zu schärfen.

Offenlegung

Diese Schwachstelle wurde WordPress am 21. Januar gemeldet; es ist noch keine Lösung verfügbar. Bitte lesen Sie den Abschnitt *Patch* , um eine Anleitung zu möglichen Korrekturen zu erhalten, die Sie auf Ihre WordPress-Instanzen anwenden können.

Es ist das erste Mal, dass wir Details über eine ungepatchte Schwachstelle veröffentlichen, und diese Entscheidung wurde uns nicht leicht gemacht. Dieses Problem wurde erstmals vor etwa sechs Jahren im Januar 2017 von einem anderen Forscher und zahlreichen anderen im Laufe der Jahre gemeldet. Nach unserem Bericht und weiteren Untersuchungen konnten wir auch mehrere öffentliche Blog-Posts identifizieren, die dasselbe Verhalten dokumentieren wie der, über den wir heute berichten werden.

Aufgrund der geringen Auswirkungen in der vorliegenden Form, der vorherigen Veröffentlichung und der Notwendigkeit, sie mit zusätzlichen Schwachstellen in Software von Drittanbietern zu

verketten, glauben wir, dass diese Version WordPress-Benutzer nicht gefährdet und ihnen nur helfen kann, ihre Instanzen zu härten.

Einfluss

Wir konnten keine Möglichkeiten finden, dieses Verhalten zu nutzen, um anfällige Instanzen zu übernehmen, ohne auf andere anfällige Dienste angewiesen zu sein.

Es könnte die Ausnutzung anderer Schwachstellen im internen Netzwerk der betroffenen Organisation erleichtern, beispielsweise durch die Verwendung einer der jüngsten Confluence OGNL-Injektionen, der epischen Remote-Code-Ausführung in Jenkins, die von @orange_8361 gefunden wurde , oder einer der anderen von AssetNote dokumentierten Ketten .

Technische Details

Verwendung des anfälligen Konstrukts in der Pingback-Funktion

Pingbacks sind eine Möglichkeit für Blogautoren, benachrichtigt und angezeigt zu werden, wenn andere „befreundete“ Blogs auf einen bestimmten Artikel verweisen: Sie werden neben Kommentaren angezeigt und können frei akzeptiert oder abgelehnt werden. Unter der Haube müssen Blogs HTTP-Anfragen aneinander senden, um das Vorhandensein von Links zu identifizieren. Auch Besucher können diesen Mechanismus auslösen.

Diese Funktion wurde vielfach kritisiert, da sie es Angreifern ermöglicht, verteilte Denial-of-Service-Angriffe durchzuführen, indem sie böswillig Tausende von Blogs auffordern, auf einem einzelnen Server des Opfers nach Pingbacks zu suchen. Pingbacks sind auf WordPress-Instanzen immer noch standardmäßig aktiviert, da soziale und Community-

Funktionen für das persönliche Bloggen wichtig sind. Es wird jedoch nicht erwartet, dass diese Anfragen an andere interne Dienste gesendet werden, die auf demselben Server oder lokalen Netzwerksegment gehostet werden.

Die Pingback-Funktionalität wird auf der XML-RPC-API von WordPress bereitgestellt. Zur Erinnerung: Dies ist ein API-Endpunkt, der XML-Dokumente erwartet, in denen der Client eine aufzurufende Funktion zusammen mit Argumenten auswählen kann.

Eine der implementierten Methoden ist pingback.ping und erwartet die Argumente pagelinkedfrom und pagelinkedto : Das erste ist die Adresse des Artikels, der auf das zweite verweist.

pagelinkedto muss auf einen bestehenden Artikel der lokalen Instanz zeigen, hier http://blog.tld/?p=1 , und pagelinkedfrom auf die externe URL, die einen Link zu pagelinkedto enthalten soll .

Unten sehen Sie, wie eine Anfrage an diesen Endpunkt aussehen würde:

```
POST /xmlrpc.php HTTP/1.1
Host: blog.tld
[...]
<methodCall>
   <methodName>pingback.ping</methodName>
  <params>
    <param>
      <value><string>http://evil.tld</string></value>
    </param>
    <param>
      <value><string>http://blog.tld/?p=1</string></value>
    </param>
  </params>
</methodCall>
```

Implementierung der URL-Validierung

Die WordPress-Core-Methode wp_http_validate_url() führt einige Überprüfungen der vom Benutzer bereitgestellten URLs durch, um das Missbrauchsrisiko zu verringern. Zum Beispiel:

1. Das Ziel darf keinen Benutzernamen und kein Passwort enthalten;
2. Der Hostname darf folgende Zeichen nicht enthalten: #:?[]
3. Der Domänenname sollte nicht auf eine lokale oder private IP-Adresse wie 127.0.0.1, 192.168.* usw. verweisen.
4. Der Zielport der URL muss entweder 80, 443 oder 8080 sein.

Der dritte Schritt kann das Auflösen von Domänennamen beinhalten, falls sie in der URL vorhanden sind (z. B. http://foo.bar.tld). In diesem Fall wird die IP-Adresse des Remote-Servers ermittelt, indem die URL analysiert **[1]** und später aufgelöst wird **[2]** , bevor sie validiert wird, um nicht öffentliche IP-Bereiche auszuschließen:

src/wp-includes/http.php

```
$parsed_url = parse_url( $url ); // [1]
// [...]
$ip = gethostbyname( $host );    // [2]
        if ( $ip === $host ) {
           // Error condition for gethostbyname().
                return false;
        }
     // IP validation happens here
}
// [...]
```

Der Validierungscode scheint korrekt implementiert zu sein, und die URL gilt jetzt als vertrauenswürdig. Was passiert als nächstes?

Implementierung des/der HTTP-Client(s)

Zwei HTTP-Clients können Pingback-Anfragen verarbeiten, nachdem sie die URL validiert haben, basierend auf verfügbaren PHP-Funktionen: Requests_Transport_cURL und Requests_Transport_fsockopen . Sie sind beide Teile der Requests- Bibliothek, die unabhängig voneinander unter dem Dach von WordPress entwickelt wurden.

Werfen wir einen Blick auf die Implementierung des letzteren. Wir wissen, dass es die PHP-Streams-API von seinem Namen verwendet. Es arbeitet auf der Transportebene, und der Client muss die HTTP-Anforderung manuell erstellen. Die URL wird erneut mit parse_url() geparst und dann wird ihr *Host* – Teil verwendet, um ein Ziel zu erstellen, das mit der PHP-Streams-API kompatibel ist (z. B. tcp://host:port):

wp-includes/Requests/Transport/fsockopen.php

```
public function request($url, $headers = array(), $data = array(), $options = array()) {
    // [...]
    $url_parts = parse_url($url);
    // [...]
    $host = $url_parts['host'];
    else {
        $remote_socket = 'tcp://' . $host;
    }
    // [...]
    $remote_socket .= ':' . $url_parts['port'];
```

Weiter entfernt wird dieses Ziel verwendet, um mit stream_socket_client() einen neuen Stream zu erstellen , und die HTTP-Anforderung wird erstellt und dorthin geschrieben:

wp-includes/Requests/Transport/fsockopen.php

```
□□$socket = stream_socket_client($remote_socket, $errno, $errstr, ceil($options['connect_timeout']), STREAM_CLIENT_CONNECT, $context);
// [...]
```

```
$out = sprintf("%s %s HTTP/%.1F\r\n", $options['type'], $path,
$options['protocol_version']);
// [...]
if (!isset($case_insensitive_headers['Host'])) {
    $out .= sprintf('Host: %s', $url_parts['host']);
    // [...]
}
// [...]
fwrite($socket, $out);
```

Wie wir sehen können, impliziert dieser Prozess eine andere DNS-Auflösung, sodass stream_socket_client() die IP des Hosts identifizieren kann, um die Pakete zu senden.

Das Verhalten des anderen HTTP-Clients, cURL, ist sehr ähnlich und wird hier nicht behandelt.

Die Schwachstelle

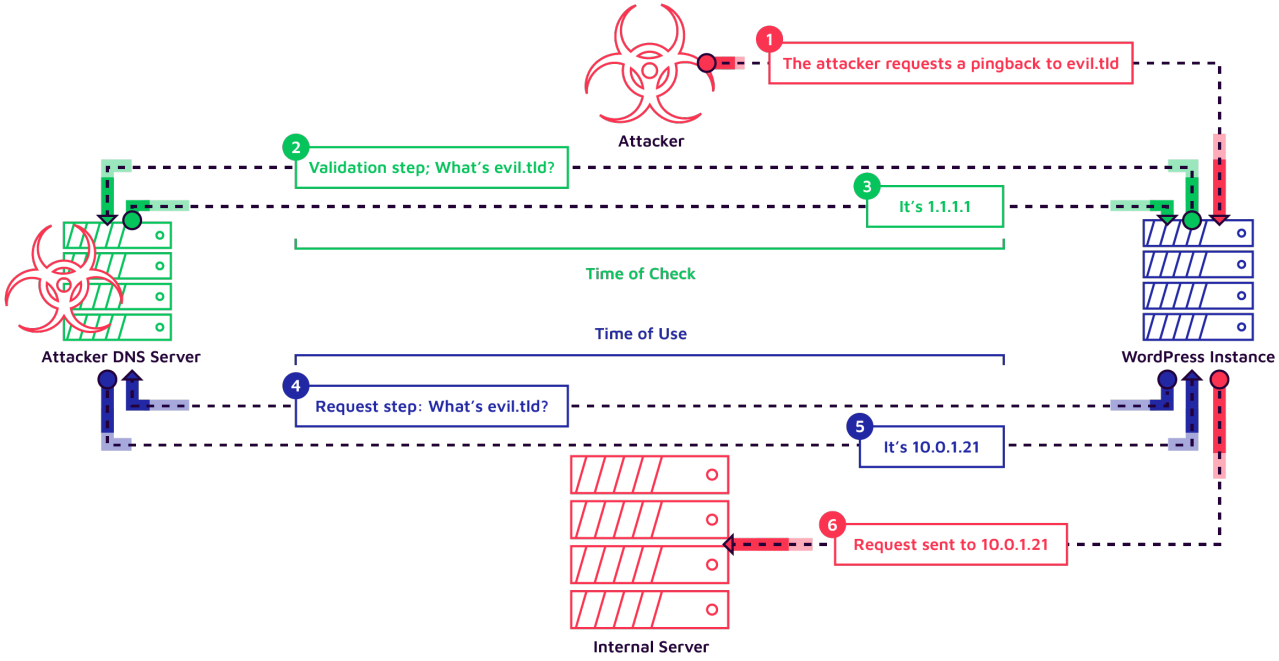
Dieses Konstrukt hat ein Problem: Der HTTP-Client muss die URL erneut analysieren und den Hostnamen erneut auflösen, um seine Anfrage zu senden. **In der Zwischenzeit könnte ein Angreifer die Domain so geändert haben, dass sie auf eine andere Adresse als die zuvor validierte verweist!**

Diese Fehlerklasse wird auch Time-of-Check-Time-of-Use genannt: Eine Ressource wird validiert, kann aber später vor ihrer effektiven Verwendung geändert werden. Es ist üblich, solche Schwachstellen in Abwehrmaßnahmen gegen serverseitige Anforderungsfälschungen (SSRF) zu finden. [Wir haben in unserem Code Security Adventskalender 2021 sogar eine Herausforderung veröffentlicht, die auf diesem anfälligen Codemuster basiert.]

https://platform.twitter.com/embed/Tweet.html?creatorScreenName=sonarsource&dnt=false&embedId=twitter-widget-0&features=eyJ0ZndfdGltZWxpbmVfbGlzdCI6eyJidWNrZXQiOlsibGlua3RyLmVlIiwidHIuZWUiLCJ0ZXJyYS5jb20uYnIiLCJ3d3cubGlua3RyLmVlIiwid3d3LnRyLmVlIiwid3d3LnRlcnJhLmNvbS5iciJdLCJ2ZXJzaW9uIjpudWxsfSwidGZ3X2hvcml6b25fdGltZWxpbmVfMTIwMzQiOnsiYnVja2V0IjoidH

JlYXRtZW50IiwidmVyc2lvbiI6bnVsbH0sInRmd190d2VldF9lZGl0X2JhY2tl
bmQiOnsiYnVja2V0Ijoib24iLCJ2ZXJzaW9uIjpudWxsfSwidGZ3X3JlZnNyY1
9zZXNzaW9uIjp7ImJ1Y2tldCI6Im9uIiwidmVyc2lvbiI6bnVsbH0sInRmd19z
aG93X2J1c2luZXNzX3ZlcmlmaWVkX2JhZGdlIjp7ImJ1Y2tldCI6Im9uIiwidm
Vyc2lvbiI6bnVsbH0sInRmd19jaGluX3BpbGxzXzE0NzQxIjp7ImJ1Y2tldCI6
ImNvbG9yX2ljb25zIiwidmVyc2lvbiI6bnVsbH0sInRmd190d2VldF9yZXN1bH
RfbWlncmF0aW9uXzEzOTc5Ijp7ImJ1Y2tldCI6InR3ZWV0X3Jlc3VsdCIsInZl
cnNpb24iOm51bGx9LCJ0Zndfc2Vuc2l0aXZlX21lZGlhX2ludGVyc3RpdGlhbF
8xMzk2MyI6eyJidWNrZXQiOiJpbnRlcnN0aXRpYWwiLCJ2ZXJzaW9uIjpudWxs
fSwidGZ3X2V4cGVyaW1lbnRzX2Nvb2tpZV9leHBpcmF0aW9uIjp7ImJ1Y2tldC
I6MTIwOTYwMCwidmVyc2lvbiI6bnVsbH0sInRmd19kdXBsaWNhdGVfc2NyaWJl
c190b19zZXR0aW5ncyI6eyJidWNrZXQiOiJvbiIsInZlcnNpb24iOm51bGx9LC
J0ZndfdmlkZW9faGxzX2R5bmFtaWNfbWFuaWZlc3RzXzE1MDgyIjp7ImJ1Y2tl
dCI6InRydWVfYml0cmF0ZSIsInZlcnNpb24iOm51bGx9LCJ0Zndfc2hvd19ibH
VlX3ZlcmlmaWVkX2JhZGdlIjp7ImJ1Y2tldCI6Im9uIiwidmVyc2lvbiI6bnVs
bH0sInRmd19zaG93X2dvdl92ZXJpZmllZF9iYWRnZSI6eyJidWNrZXQiOiJvZm
YiLCJ2ZXJzaW9uIjpudWxsfSwidGZ3X3Nob3dfYnVzaW5lc3NfYWZmaWxpYXRl
X2JhZGdlIjp7ImJ1Y2tldCI6Im9mZiIsInZlcnNpb24iOm51bGx9LCJ0ZndfdH
dlZXRfZWRpdF9mcm9udGVuZCI6eyJidWNrZXQiOiJvbiIsInZlcnNpb24iOm51
bGx9fQ%3D%3D&frame=false&hideCard=false&hideThread=false&id=14
6824893937984716 8&lang=en&origin=https%3A%2F%2Fblog.sonarsourc
e.com%2F%2Fwordpress-core-unauthenticated-blind-
ssrf%2F&sessionId=b21903e53850f875895c48efb872fe3f48f36be0&sit
eScreenName=blog_SonarSource&theme=light&widgetsVersion=a3525f
077c700%3A1667415560940&width=550px

Wie diese aufeinanderfolgenden Schritte aussehen, haben wir im folgenden Diagramm zusammengefasst:



Ausbeutungsszenarien

Wir haben den Code in der Hoffnung geprüft, differenzielle Parser-Fehler zu finden, die es ermöglichen würden, unbeabsichtigte Ports zu erreichen oder POST-Anforderungen ohne Erfolg durchzuführen: Die anfänglichen URL-Validierungsschritte sind restriktiv genug, um ihre Ausnutzung zu verhindern. Wie bereits erwähnt, müssten Angreifer dieses Verhalten mit einer anderen Schwachstelle verketten, um die Sicherheit der angegriffenen Organisation erheblich zu beeinträchtigen.

Patch

Zum Zeitpunkt der Erstellung dieser Veröffentlichung sind uns keine öffentlichen Patches bekannt; Die obigen Details basieren auf einem Zwischenpatch, der uns während des Offenlegungsprozesses mitgeteilt wurde.

Das Beheben solcher Schwachstellen erfordert das Beibehalten der validierten Daten, bis sie zum Ausführen der HTTP-Anforderung verwendet werden. Es sollte nach dem Validierungsschritt nicht verworfen oder transformiert werden.

Die WordPress-Betreuer folgten diesem Weg, indem sie ein zweites, optionales Argument in wp_http_validate_url() einführten . Dieser Parameter wird als Referenz übergeben und enthält die IP-Adressen, auf denen WordPress die Validierung durchgeführt hat. Der endgültige Code ist etwas ausführlicher, um ältere Versionen von PHP zu berücksichtigen, aber die Hauptidee ist hier.

Als vorübergehende Problemumgehung empfehlen wir Systemadministratoren, den Handler pingback.ping des XMLRPC-Endpunkts zu entfernen. Eine Möglichkeit, dies zu tun, besteht darin, die functions.php des verwendeten Designs zu aktualisieren, um den folgenden Aufruf einzuführen:

```
add_filter('xmlrpc_methods', function($methods) {
  unset($methods['pingback.ping']);
  return $methods;
});
```

Es ist auch möglich, den Zugriff auf xmlrpc.php auf Webserver-Ebene zu blockieren.

Zeitleiste

Datum	Handlung
2022-01-21	Wir übermitteln die Schwachstelle an die Betreuer mit einer 90-tägigen Offenlegungsrichtlinie.
2022-01-21	Unsere Einreichung wird als Duplikat gegen einen Bericht geprüft, der ursprünglich vor (genau) 5 Jahren (2017-01-21) gesendet wurde.
2022-04-11	WordPress beantragt eine Verlängerung unserer 90-tägigen Offenlegungsrichtlinie um 30 Tage, da sie mehr Zeit benötigen, um an Backports zu arbeiten. Sind wir uns einig.
2022-05-23	Maintainer teilen einen Patch für WordPress 5.9.3.
2022-06-01	Wir haben positives Feedback zum Patch gegeben.

Datum	Handlung
2022-07-16	Wir teilen unsere Absicht mit, diese Veröffentlichung am 6. September zu veröffentlichen.
2022-09-01	Abschließende Hinweise zur bevorstehenden Veröffentlichung.
2022-09-06	Dieser Artikel wird 228 Tage nach unserem Bericht und 2054 Tage nach dem ersten Bericht eines anderen Forschers veröffentlicht.

Zusammenfassung

In diesem Artikel haben wir eine blinde SSRF-Schwachstelle beschrieben, die WordPress Core betrifft. Während die Auswirkungen in diesem Fall als gering angesehen werden, handelt es sich um ein weit verbreitetes anfälliges Codemuster, dem wir selbst bei großen Projekten immer wieder begegnen. Wir empfehlen Entwicklern, ihre eigenen Codebasen auf diese Art von Codeschwachstellen zu überprüfen, die sich, wie wir gezeigt haben, sogar in sehr populärem und gut überprüftem Code verstecken können.

Wir möchten den WordPress-Betreuern für ihre Hilfe bei der Lösung dieses Problems danken, auch wenn wir nicht das bestmögliche Ergebnis erzielen konnten.

Verwandte Blog-Beiträge

- WordPress 5.8.3 – Schwachstelle durch Objektinjektion
- WordPress 5.8.2 – Gespeicherte XSS-Schwachstelle
- WordPress 5.7 – XXE-Schwachstelle
- WordPress 5.1 – CSRF zur Remotecodeausführung
- WordPress 5.0.0 – Remote-Code-Ausführung

Simon Scannell und Thomas Chauchefoin Schwachstellenforscher