BEAST et TLS, la fin du monde ?

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 23 septembre 2011. Dernière mise à jour le 25 septembre 2011

https://www.bortzmeyer.org/beast-tls.html

La presse a comme d'habitude annoncé n'importe quoi à propos de la faille de sécurité TLS qui a été officiellement publiée le samedi 24, vers 00h00 UTC (mais l'article n'est pas encore disponible officiellement). Les titres les plus sensationnalistes se sont succédés, « Le protocole SSL aurait été craqué », « Une faille dans le chiffrement SSL et des millions de sites HTTPS vulnérables », « Des pirates [sic] brisent le cryptage [re-sic <https://www.bortzmeyer.org/cryptage-n-existe-pas.html>] SSL utilisé par des millions de sites » et autres fariboles du même genre. C'est la loi habituelle des médias officiels : parler sérieusement et étudier son sujet ne rapportent rien. Faire de la mousse est bien plus efficace... et sera de toute façon oublié dans deux jours. Mais, derrière cette ridicule campagne de presse, qu'y a-t-il comme information fiable ?

La faille a été « pré-annoncée » par ses découvreurs, Juliano Rizzo et Thai Duong, qui ont pris soin de s'assurer que leur exposé à une conférence de sécurité peu connue ne passerait pas inaperçu. Leur plan comm' a été mis au point avec autant de soin que leur logiciel, même le nom de celui-ci (BEAST pour "*Browser Exploit Against SSL/TLS*") a probablement fait l'objet d'une étude soignée, suivant les mœurs habituels dans le monde de la sécurité informatique. L'article n'a pas été officiellement publié mais des copies (apparemment incomplètes) ont fuité, par exemple sur Google Doc <https://docs.google.com/viewer?a=v&pid=explorer&chrome=true&srcid=0B2J69udptdboYzFmMTQxM2MtZGEwYi00YjA5LWEwM2Q hl=en> ou bien dans cette archive RAR <http://insecure.cl/Beast-SSL.rar>. Comme, à l'heure où j'écris, il n'y a pas eu de publication scientifique sérieuse complète, on ne peut que deviner. Donc, en se fiant aux différentes rumeurs qui circulent et au code qui a été publié pour corriger la faille, que sait-on ?

L'attaque est apparemment de type « texte en clair choisi ». L'attaquant génère du texte de son choix et le fait chiffrer. L'observation du résultat lui donne alors des indications sur la clé de chiffrement utilisée. Les bons protocoles de chiffrement ne sont pas vulnérables à cette attaque, mais TLS **1.0** (et seulement cette version, normalisée dans le RFC 2246 [1]) a quelques particularités dans le chiffrement

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc2246.txt

en mode CBC qui le rendent vulnérables. Ces vulnérabilités ont été corrigées dans la version 1.1 du protocole, publiée dans le RFC 4346 il y a plus de cinq ans (aujourd'hui, on en est à la 1.2, normalisée dans le RFC 5246). La section 1.1 du RFC 4346 expliquait les changements avec TLS 1.0 et notait déjà « *"The implicit Initialization Vector (IV) is replaced with an explicit IV to protect against CBC attacks."* ». Mais corriger un protocole à l'IETF est une chose, déployer le correctif en est une autre.

Outre le changement de protocole, on pouvait aussi corriger le problème en bricolant les données à certains endroits, ce que fait la bibliothèque OpenSSL depuis... avant 2004 ! C'est d'ailleurs la publication de ce problème par l'équipe d'OpenSSL <http://www.openssl.org/~bodo/tls-cbc.txt> qui avait mené à TLS 1.1. (Un article plus détaillé sur ce sujet est celui de Gregory Bard en 2004 <http://eprint.iacr.org/2004/111.pdf>. Le premier message rendant publique cette vulnérabilité est de 2002 <http://www.mail-archive.com/openssl-dev@openssl.org/msg10664.html>.) On le voit, il n'y a rien de nouveau, la plupart des failles de sécurité sont connues depuis longtemps lorsque la presse fait semblant de nous révéler des choquantes nouveautés.

Est-ce que cela veut dire que la nouvelle attaque n'est que du pschiiit ? Non, il y a au moins deux nouveautés, une mise en œuvre logicielle de l'attaque, et qui fonctionne (le fameux BEAST), et un certain nombre d'astuces et d'optimisations qui font toute la différence entre une faille connue et une faille attaquée. Le passage de la théorie à la pratique est donc un évenement important.

Faut-il alors paniquer ? Comme d'habitude, non. La faille ne semble pas triviale à exploiter. Il faut que l'attaquant puisse produire des données (le texte en clair choisi) sur le poste de la victime. BEAST peut utiliser plusieurs techniques pour cela (un navigateur Web moderne est un monstre bien trop compliqué et qui fournit plein de failles de sécurité à un attaquant), par exemple un code Java envoyé à la victime. Il faut aussi que l'attaquant puisse monter une attaque de l'homme du milieu, par exemple en prenant le contrôle d'un réseau WiFi mal protégé. L'attaque est donc possible mais pas simple et quelques précautions la rendent très difficile : ne faire que du HTTPS avec les sites sensibles, ne pas se connecter à ces sites sensibles depuis un réseau inconnu (genre le Wifi de Starbucks)... Ce sont des simples règles de défense en profondeur (certes, TLS était censé protéger contre le réseau WiFi dangereux, mais il n'est pas interdit de porter ceinture **et** bretelles).

Mais, puisque j'ai dit que la version 1.1 de TLS, qui corrige la faille fondamentale, date de 2006, et que le contournement dans OpenSSL, qui fonctionne même en version TLS 1.0, date de 2004, pourquoi l'attaque est-elle encore possible en 2011 ? Pour un simple problème de déploiement. D'abord, pour OpenSSL : cette bibliothèque est de loin la plus utilisée dans les serveurs HTTP, nettement devant GnuTLS. Mais chez les navigateurs Web, c'est plutôt NSS qui domine, et cette bibliothèque n'a pas le contournement. Le patch fait <https://src.chromium.org/viewvc/chrome?view=rev&revision=90643> par les auteurs de Chromium à la bibliothèque qu'ils utilisent met d'ailleurs en œuvre la même technique qu'avec OpenSSL (une variante existe <http://src.chromium.org/viewvc/chrome?view=rev&revision=97269> ; le "*patch*" pour Firefox est discuté dans la bogue #665814 <https://bugzilla.mozilla.org/show_bug.cgi?id=665814>). Le problème est pourtant connu depuis longtemps (bogue Firefox #480514 <https://bugzilla.mozilla.org/show_bug.cgi?id=480514>).

Actuellement, un très grand nombre de navigateurs Web n'a pas TLS 1.1 <http://en.wikipedia.org/wiki/Transport_Layer_Security#Browser_implementations> et aucune mesure prise uniquement sur le serveur ne pourra résoudre cela. Si l'administrateur du serveur veut interdire le vieux et dangereux TLS 1.0 (avec le module Apache de GnuTLS <http://modgnutls.sourceforge.net/>, c'est apparemment `GnuTLSPriorities NORMAL:!VERS-TLS1.0`), il cassera tout puisque la plupart des navigateurs <http://isc.sans.edu/diary.html?storyid=11629> ne pourront pas se connecter. Une autre approche pour le gérant de site Web est plutôt de ne pas utiliser les algorithmes de chiffrement en CBC. Elle est détaillé dans l'article de PhoneFactor <http://www.phonefactor.com/

resources/CipherSuiteMitigationForBeast.pdf> (c'est cette méthode qu'utilisent les serveurs HTTP de Google). Côté client, une bonne compilation de techniques limitant les risques est disponible sur StackExchange <http://security.stackexchange.com/questions/7450/tls-1-0-javascript-inject

C'est donc moins sexy qu'une nouvellement découverte faille du protocole mais la réalité est là : le problème principal est celui d'une non-mise à jour des logiciels, pour parer aux vulnérabilités connues. Les raisons de cette non-mise à jour sont nombreuses. Une d'elles est que gérer les nouvelles versions du protocole peut empêcher les connexions avec les serveurs bogués <http://blogs.msdn.com/b/ieinternals/archive/2011/03/25/misbehaving-https-servers-impair-tls-1.1-and-tls-1.2.aspx>. En effet, le protocole TLS est conçu pour que les anciens clients puissent interagir avec les nouveaux serveurs et réciproquement, mais cela suppose que toutes les implémentations respectent la norme (qui décrit avec précision comment assurer la compatibilité des différentes versions, cf. annexe E du RFC 5246). Or, il existe beaucoup de programmes bogués, qui bloquent toute évolution du protocole (la faille de renégociation de TLS <https://www.bortzmeyer.org/tls-renego.html>, corrigée par le RFC 5746, connaît des problèmes de déploiement similaires, un tiers des serveurs étaient encore vulnérables en 2010). Résultat, la plupart des serveurs TLS coupent TLS 1.1 et 1.2 pour éviter tout problème comme le montre l'excellente enquête de Qualys publiée à Blackhat <https://media.blackhat.com/bh-us-10/presentations/Ristic/BlackHat-USA-2010-Ristic-Qualys-SSL-Survey-HTT pdf>.

Pour savoir la version maximale que permet un serveur TLS, vous pouvez utiliser l'outil en ligne de commande `gnutls-cli`, livré avec GnuTLS (l'outil équivalent d'OpenSSL ne semble pas capable de distinguer les différentes versions mineures de TLS.) GnuTLS va tenter d'utiliser la version la plus élevée possible. Avec un GnuTLS récent, qui gère TLS 1.2, contre un Apache récent, lui-même avec GnuTLS :

```
% gnutls-cli  --verbose  --port 443 svn.example.org
...
- Version: TLS1.2
```

Avec un serveur traditionnel (ici, celui de l'IETF) :

```
% gnutls-cli  --verbose  --port 443 www.ietf.org
...
- Version: TLS1.0
```

Vous pouvez aussi utiliser sslscan <http://sourceforge.net/projects/sslscan/> pour tester un serveur mais, comme openssl, il ne sait pas distinguer les différentes versions de TLS 1.0.

Vous pouvez aussi tester votre propre site TLS en ligne depuis Qualys <https://www.ssllabs.com/> (il ne semble pas gérer SNI, hélas). Quelques lectures pour approfondir le sujet : une bonne explication pédagogique notamment du CBC <http://practicalcrypto.blogspot.com/2011/09/brief-diversion-beast-attack-on-tlsssl.html>, une autre explication bien faite <https://blog.torproject.org/blog/tor-and-beast-ssl-attack>, notamment dans le contexte de Tor, une très bonne explication détaillée <http://luxsci.com/blog/is-ssltls-really-broken-by-the-beast html> du problème, la déclaration des développeurs <https://lists.gnu.org/archive/html/help-gnutls/2011-09/msg00018.html> de GnuTLS, un article détaillé <https://www.imperialviolet.org/2011/09/23/chromeandbeast.html> d'un des développeurs de Google Chrome, un très bon résumé et analyse des conséquences <http://article.gmane.org/gmane.network.openvpn.user/32566> pour le logiciel OpenVPN, un bon résumé par Microsoft <http://blogs.technet.com/b/srd/archive/2011/09/26/is-ssl-broken-more-about-security-advisory-2588513.

aspx>, un autre chez SANS <http://isc.sans.edu/diary.html?storyid=11722>, un article plus détaillé de Bard <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.61.5887&rep=rep1&type=pdf> (qui notait déjà les points essentiels, à savoir que la vulnérabilité était close par TLS 1.1 ou bien par l'astuce de l'enregistrement vide de OpenSSL), l'analyse du gourou TLS de l'IETF <http://www.educatedguesswork.org/2011/09/security_impact_of_the_rizzodu.html> (très clair, et un des rares articles écrits **après** l'exposé de Rizzo et Duong), le très intéressant récit du développement de BEAST <http://vnhacker.blogspot.com/2011/09/beast.html> par un de ses auteurs, un article <http://blog.ivanristic.com/2011/10/mitigating-the-beast-attack.html> consacré uniquement aux techniques permettant de limiter l'effet de la faille, la discussion dans le groupe de travail TLS de l'IETF <http://www.ietf.org/mail-archive/web/tls/current/msg08032.html> et, bien sûr, l'article original, « "*Here Come The [Caractère Unicode non montré [2]] Ninjas*" », dès qu'il sera officiellement publié... En français, l'article de SecurityVibes <http://www.securityvibes.fr/menaces-alertes/ssl-faille-beast/> est intéressant. Ceux qui aiment lire le code source pourront trouver une exploitation (je ne l'ai pas testée) de la faille en Java en <http://pastebin.com/pM3ZbCdV> ou bien dans l'archive RAR citée plus haut <http://insecure.cl/Beast-SSL.rar>.

Merci à Mathieu Pillard <http://www.virgule.info/> pour ses commentaires.

2. Car trop difficile à faire afficher par LaTeX