Available *online* at : http://bit.ly/InfoTekJar

InfoTekJar : Jurnal Nasional Informatika dan Teknologi Jaringan

ISSN (Print) 2540-7597 | ISSN (Online) 2540-7600

Dampak dan Pencegahan Serangan *File Inclusion*: Perspektif *Developer*

Muhammad Koprawi

Prodi Teknik Komputer, Fakultas Ilmu Komputer, Universitas Amikom Yogyakarta

KEYWORDS

File Inclusion, LFI, RFI, Web Security

CORRESPONDENCE

E-mail: koprawi@amikom.ac.id

A B S T R A C T

File Inclusion adalah salah satu celah keamanan yang memiliki dampak cukup besar terhadap *website* dan *server*. *File Inclusion* sendiri terdiri dari *Local File Inclusion* (LFI) dan *Remote File Inclusion (RFI)*. Celah keamanan ini terjadi salah satunya karena kurangnya kesadaran terhadap *secure programming* atau bagaimana menuliskan kode program dengan cara yang aman. Dampak serangan yang paling bisa dirasakan adalah diambil alihnya akses terhadap *website* ataupun *server*, jika *server* sudah berhasil diambil alih, otomatis *database* beserta hak akses yang lainnya pun berhasil dikuasai. Untuk itu pentingnya seorangan *developer* memahami dampak yang ditimbulkan oleh serangan *File Inclusion* dan memahami bagaimana menuliskan kode yang aman serta pengetahuan tambahan untuk pencegahan terjadinya serangan ini disisi *server*. Pada penelitian ini akan dijelaskan mengenai skenario, dampak dan pencegahan serangan *File Inclusion* dalam perspektif seorangan *developer*. Penelitian ini setidaknya akan membantu *developer-developer* muda dalam memahami dan menuliskan kode yang aman. Salah satu contohnya adalah menerapkan konsep pengujian kode *website* dengan pola *attack*, *defense* dan validasi sebelum *website* tersebut masuk ke fase produksi atau *live*.

PENDAHULUAN

Pada era digital seperti sekarang untuk melakukan pertukaran informasi salah satunya adalah melalui website, sebuah website biasanya terdiri dari beberapa komponen seperti adanya nama domain, adanya *server/hosting* (tempat meletakkan file website) dan terakhir adanya konten yang disajikan di website tersebut. Perkembangannya pada saat ini, website bisa dibangun atau dibuat tanpa harus mengetahui atau memahami lebih jauh tentang *programming*, tidak perlu mengambil kursus/kuliah yang berhubungan dengan programming, website tersebut bisa dibuat secara instan dan cepat. Kebanyakan *developer-developer* muda kurang *aware* atau peduli tentang keamanan terhadap sebuah website, yang terpenting adalah website tersebut bisa diakses oleh banyak orang. Menurut OWASP (*Open Web Application Security Project*) ada 10 ancaman atau celah-celah keamanan yang paling banyak terjadi. OWASP sendiri adalah sebuah organisasi yang fokus pada keamanan web, OWASP membuat beberapa standarisasi dan dokumen yang bisa digunakan sebagai rujukan pada bidang keamanan website. Adapun 10 ancaman menurut OWASP adalah sebagai berikut [1][2]:

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities
5. Broken Access Control
6. Security Missconfiguration
7. Cross Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components With Known Vulnerabilities
10. Insufficient Logging and Monitoring

Selain 10 celah keamanan di atas, masih adalagi celah-celah keamanan yang juga didokumentasikan OWASP salah satunya adalah *File Inclusion* [1], [2].

File Inclusion adalah salah satu celah keamanan yang memungkinkan penyerang untuk menyisipkan file-file tertentu atau bisa jadi file-file berbahaya dari komputer lokal (*Local File Inclusion)* atau dari komputer *remote (Remote File Inclusion)*[3]. Adapun hal-hal yang mungkin bisa dilakukan oleh penyerang adalah mengakses file-file sensitif yang tersedia di server web atau bisa jadi mengeksekusi file berbahaya yang

https://doi.org/10.30743/infotekjar.v4i2.2332

sudah ditambahkan di server melalui celah *file inclusion*. *File Inclusion* terbagi menjadi dua yaitu *Local File Inclusion (LFI)* dan *Remote File Inclusion (RFI)*[3].

Local File Inclusion (LFI)

Local File Inclusion adalah celah keamanan yang memungkinkan penyerang untuk bisa membaca atau melihat file-file yang ada di server termasuk file-file sensitif[4], [5]. LFI biasanya muncul karena kesalahan pada *coding*, salah satunya disebabkan oleh sebuah fungsi seperti fungsi include() yang tidak divalidasi dan difilter dengan baik dan benar. Fungsi include() adalah fungsi dari bahasa pemrograman PHP yang berfungsi untuk memasukkan atau mengaitkan sesuatu seperti file ke dalam sebuah *page* (halaman) di website[6]. Ketika fungsi ini tidak divalidasi dengan benar, serangan LFI bisa dijalankan pada halaman-halaman tertentu. Salah satu dampak dari serangan LFI adalah terbacanya file-file sensitif yang ada di server, contohnya file-file sensitif yang ada di server linux. Pada server linux ada beberapa file yang dianggap dan bersifat sensitif antara lain:

- /etc/passwd
- /etc/shadow
- /etc/group
- /etc/issue
- dll

Salah satu contohnya adalah file /etc/passwd, pada file ini terdapat informasi-informasi sensitif seperti *username*, *password* yang dienkripsi, user id, group id, dan lain sebagainya lagi[7]. Informasi tersebut tidak boleh diketahui oleh orang yang tidak berhak atau tidak memiliki akses ke server.

Remote File Inclusion (RFI)

Remote File Inclusion adalah celah keamanan yang memungkinkan penyerang untuk menyisipkan file berbahaya dari luar server dan mekeksekusinya, file ini biasanya berisi *evil code* atau kode jahat yang bisa digunakan untuk mengendalikan komputer atau server korbannya[5], [6]. Celah ini muncul salah satunya karena konfigurasi di server yang tidak benar dan *coding* yang tidak divalidasi dengan baik dan benar. Dampak dari celah ini salah satunya adalah penyerang bisa mengakses secara langsung file-file sensitif bahkan bisa memanipulasi file-file tersebut, melihat *database*, mengubah hak akses dan yang paling parah adalah mengambil alih server.

Pengetahuan dan pemahaman terhadap dampak yang dihasilkan oleh celah ini dirasa penting untuk diketahui. Selain dampak yang dihasilkan, pencegahannya pun juga tidak kalah penting untuk diketahui. Oleh karena itu tujuan penelitian ini adalah untuk memberikan pandangan-pandangan baru kepada *developer-developer* muda agar selalu memperhatikan dan menuliskan *code* dengan baik dan aman sehingga celah keamanan seperti ini bisa diminimalisir bahkan dihindari.

METODE

Penelitian ini menggunakan metode eksperimental, pengujian akan dilakukan menggunakan mesin virtual (*virtual machine)*, disini peneliti menggunakan *software virtual machine* virtualbox, untuk itu akan dibuat beberapa *environment* yang akan digunakan dalam pengujian ini:

Mesin Attacker

Mesin *attacker* adalah mesin yang disiapkan untuk melakukan serangan/pengujian terhadap mesin victim/target yang sudah terpasang sebuah *vulnerability web*, adapun spesifikasinya sebagai berikut:

- Sistem Operasi Kali Linux 2019.3 64 bit
- Konfigurasi jaringan menggunakan *Host Only Adapter*
- IP Address: 192.168.100.3

Kali Linux dipilih sebagai sistem operasi pada mesin *attacker* ini adalah karena sistem operasi ini sudah dilengkapi dengan banyak *tools* yang biasa digunakan untuk keperluan pengujian terhadap server ataupun website.

Mesin Victim

Mesin victim/target adalah mesin yang disiapkan sebagai mesin target yang sudah terpasang sebuah *vulnerability web* pada pengujian ini, adapun spesifikasinya sebagai berikut:

- Sistem Operasi Ubuntu Server 18.04.3 64 bit
- Konfigurasi jaringan menggunakan Host Only Adapter
- IP Address: 192.168.100.2

Tools

Adapun *tools* yang dimaksud adalah tools yang terpasang di mesin victim seperti:

- PHP 5.6.21
- Apache2
- MySQL

untuk mempermudah instalasi *tools* di atas maka dipilihlah lampp sebagai *tools* yang diinstal di server victim, lampp sudah di*bundle* dengan tools di atas, tujuan instalasi *tools* ini adalah untuk keperluan menjalankan website yang nantinya akan dijadikan target serangan.

Celah Local File Inclusion

Beberapa cara yang bisa digunakan untuk mendapatkan celah LFI adalah sebagai berikut:

a. Menggunakan parameter yang tersedia
   Menggunakan parameter disini maksudnya adalah memanfaatkan parameter buatan contoh parameter page=, pada parameter page= akan dilakukan percobaan apakah parameter tersebut divalidasi atau tidak, parameter ini menggunakan *method* GET sehingga penyerang bisa langsung mengisi parameter tersebut. Contoh disini adalah untuk mengakses halaman *vulnerability web* menggunakan URL http://192.168.100.2/simple_blog/index.php?page=tech.php, jika file tech.php ada di server maka file tersebut akan dimuat melalui fungsi include(), jika tidak ada seharusnya yang ditampilkan adalah page lain. Untuk menguji apakah parameter tersebut sudah divalidasi atau tidak, cukup mengganti tech.php dengan nama file yang ada di server contohnya file /etc/passwd jadi bisa diakses dengan URL seperti berikut http://192.168.100.2/simple_blog/index.php?page=/etc/passwd.

 https://doi.org/10.30743/infotekjar.v4i2.2332

b. Directory traversal (*dot dot slash*)
Jika cara pertama tidak berhasil, bisa menggunakan cara directory traversal. Caranya hampir sama dengan cara pertama tapi ditambahkan *dot-dot slash* di depan file sensitif yang akan di baca. Adapun penulisan URL sebagai berikut http://192.168.100.2/simple_blog/index.php?page=../../../../../../../../etc/passwd banyaknya *dot-dot slash* yang dimasukkan tidak ada batasan tertentu sampai file yang dimaksud bisa dibaca.

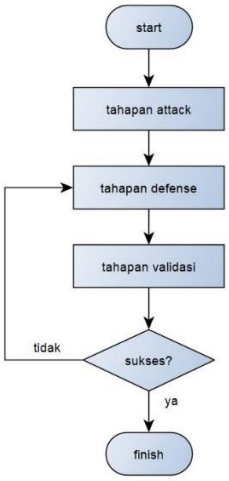
c. PHP Wrapper
PHP wrapper bisa digunakan untuk membaca file-file tertentu yang ada di server tidak harus file sensitif. Setiap website dinamis biasanya mempunyai sebuah file konfigurasi yang menyimpan informasi seperti *username*, *password*, nama *database*. Kemudian kebiasaan penamaan file konfigurasi tersebut seperti config.php, koneksi.php, wp-config dan masih banyak lainnya. Dengan memanfaatkan PHP wrapper dan kebiasaan penamaan file konfigurasi dapat mengakibatkan terbacanya isi file konfigurasi tersebut. Contoh PHP wrapper seperti file://, http://, ftp://, php:// dan sebagainya. Untuk menggunakan PHP wrapper agar bisa membaca isi dari sebuah file dengan ekstensi PHP memerlukan trik tersendiri seperti mengkonversi isi file menjadi base64 encode kemudian langkah selanjutnya adalah melakukan base64 decode sehingga isi file bisa terbaca sepenuhnya.

Celah Remote File Inclusion

Celah RFI muncul karena penulisan *code* yang tidak aman seperti tidak divalidasinya parameter yang menggunakan *method* GET, kemudian celah ini juga muncul karena kesalah konfigurasi di server seperti pada file php.ini dengan membiarkan konfigurasi seperti allow_url_fopen dan allow_url_include diset menjadi On.

Tahapan Pengujian

Pada Penelitian ini akan dilakukan pegujian terhadap *vulnerability web* yang dilakukan dengan beberapa kali tahapan yaitu *attack*, *defense* dan *validasi*.



Gambar 1. *Flowchart* Tahapan pengujian

- Pada tahapan *attack*, *vulnerability web* akan diserang menggunakan mesin *attacker*, semua celah *file inclusion* akan dicari.
- Pada tahapan *defense*, *vulnerability web* akan diperbaiki celah keamanannya dengan fokus dibagian *coding* atau sesuai dengan laporan pada tahapan *attack*.
- Pada tahapan validasi, *vulnerability web* akan diserang kembali setelah sebelumnya sudah diperbaiki celah-celah yang didapatkan pada tahapan *attack*.

Tahapan validasi sangat penting karena tahapan ini adalah tahapan akhir untuk memastikan celah yang ada di *vulnerability web* seharunsya sudah tidak ada.

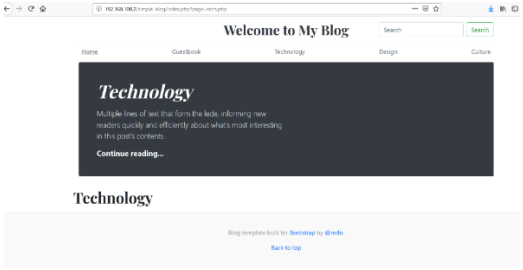
HASIL DAN PEMBAHASAN

Pada bagian ini akan dijelaskan tahapan-tahapan pengujian yang dilakukan seperti yang telah dijelaskan pada bagian metode yaitu tahapan *attack*, tahapan *defense* dan tahapan validasi.

Tahapan Attack

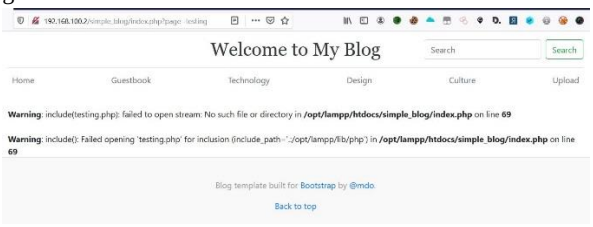
Mencari Celah Kerentanan Local File Inclusion (LFI)

a. Menggunakan parameter yang ada
Pada *vulnerability web* ditemukan link berikut http://192.168.100.2/simple_blog/index.php?page=tech.php, ketika diakses menampilkan konten seperti pada Gambar 2.



Gambar 2. Halaman tech.php

Kemudian dilakukan pengujian dengan menambahkan sembarangan path/url/kata, disini dicoba kata *testing*, sehingga menjadi http://192.168.100.2/simple_blog/index.php?page=testing dan respon dari website tersebut adalah seperti pada gambar 3.

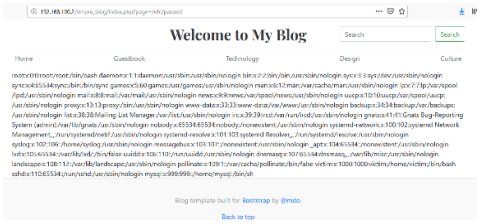


Gambar 3. Respon kata testing

Respon yang muncul adalah error yang sebernarnya bisa menjadi informasi untuk penyerang, informasi tersebut seperti:

- File atau direktori yang dimaksud tidak ada
- Fungsi include() yang terlihat
- *Path* atau tempat website tersebut diletakkan di server

Selanjutnya dilakukan pengujian untuk membaca file sensitif yang ada di server, disini akan dicoba untuk membaca file /etc/passwd yang ada di server sehingga menjadi http://192.168.100.2/simple_blog/index.php?page=/etc/passwd dan respon dari website tersebut adalah seperti pada gambar 4.



Gambar 4. File /etc/passwd terbaca

Respon yang muncul adalah file sensitif /etc/passwd bisa terbaca langsung dan hal ini disebabkan karena fungsi include() yang tidak difilter dan divalidasi.

b. Directory traversal (*dot dot slash*)
Pengujian selanjutnya adalah dengan menggunakan teknik *directory traversal*, caranya adalah dengan menambahkan ../../(*dot dot slash*) pada URL seperti berikut http://192.168.100.2/simple_blog/index.php?page=../../etc/passwd dan untuk banyaknya *dot dot slash* tidak ditentukan tetapi terus dicoba sampai file yang dimaksud bisa terbaca seperti pada gambar 5.

Gambar 5. Pengujian menggunakan directory traversal

Respon yang ditampilkan berhasil membaca file sensitif /etc/passwd yang ada di server.

c. PHP Wrapper
Setelah berhasil membaca file sensitif, langkah selanjutnya adalah pengujian membaca file lain selain file sensitif yang ada di server. Misalkan disini akan dilakukan pengujian untuk membaca file konfigurasi website tersebut, biasanya di dalam file konfigurasi terdapat *username, password* dan *database.* Pada tahapan ini dilakukan beberapa kali percobaan untuk menebak nama file konfigurasi tapi secara umum biasanya nama file konfigurasi seperti koneksi.php, konek.php, config.php, wp-config.php dan lain-lain. Pada pengujian ini untuk file dengan ekstensi .php tidak bisa langsung dibaca seperti http://192.168.100.2/simple_blog/index.php?page=koneksi.php tapi untuk bisa membaca isi file tersebut bisa memanfaatkan PHP wrapper salah satunya adalah php://filter yang kemudian isi file tersebut di *encode* menggunakan base64 seperti pada gambar 6.

Gambar 6. Hasil encode base64 file koneksi.php

Hasil respon tersebut kemudian di decode kembali dan akan menampilkan *username, password* dan nama *database*.

Dampak

Dari beberapa pengujian di atas, dampak yang lebih besar lagi adalah terjadinya RCE (*Remote Code Execution*)[8], penyerang memiliki akses ke server dengan memanfaatkan celah LFI tadi dengan memanfaatkan PHP Wrapper php://input dan php shell_exec(). Respon dari RCE adalah terbacanya isi file-file lain yang ada di direktori server sesuai dengan *code* yang dimasukkan.

Mencari Celah Kerentanan Remote File Inclusion (RFI)

Seperti halnya LFI, untuk mencari celah RFI bisa menggunakan parameter yang sudah ada misalkan disini parameter page. Sehingga URL yang diakses seperti berikut http://192.168.100.2/simple_blog/index.php?page=http://attackersite.com/evilcode dan pada pengujian ini menggunakan URL seperti berikut http://192.168.100.2/simple_blog/index.php?page=http://192.168.100.3/evilscript.txt. Evilscript disini berupa *code* yang yang sudah disiapkan seperti membaca isi direktori di server.

Dampak

Dampak yang cukup besar pada celah RFI ini adalah server target bisa diambil alih, contohnya dengan memanfaatkan *shell code* yang banyak beredar di internet.

Tahapan Defense

a. Menutup Celah Kerentanan Local File Inclusion (LFI)
Berdasarkan pengujian sebelumnya dan melihat *code* yang diterapkan pada website tersebut seperti pada ganbar 7.

```php
<?php

  if (isset($_GET['page'])) {

    include $_GET['page'];

  }
```

Gambar 7. Potongan *code*

Dari *code* di atas terlihat tidak ada filter apapun terhadap parameter page, beberapa hal yang bisa diterapkan untuk menutup celah pada *code* ini adalah:

- Menambahkan direktori dan ekstensi file yang diperbolehkan untuk dieksekusi seperti pada gambar 8.

```php
<?php

  if (isset($_GET['page'])) {

    include "pages/".$_GET['page'].".php";

  }
```

Gambar 8. menambahkan ekstensi file

- Membuat *whitelist* file-file yang boleh dieksekusi oleh fungsi include() seperti pada gambar 9.

```php
<?php

  $whitelist_page = array('tech.php', 'design.php', 'culture.php');
  if (isset($_GET['page'])) {
    if(!in_array($_GET['page'], $whitelist_page)) {
      die('file tidak ditemukan');
    } else {
```

Gambar 9. menambahkan whitelist page

https://doi.org/10.30743/infotekjar.v4i2.2332

b. Menutup Celah Kerentanan Remote File Inclusion (RFI)

Celah RFI muncul salah satunya adalah kesalahan konfigurasi di server seperti php.ini dengan membiarkan konfigurasi seperti allow_url_fopen dan allow_url_include diset menjadi On untuk itu pastikan untuk merubah settingan allow_url_fopen dan allow_url_include menjadi Off dan menonaktifkan beberapa fungsi PHP seperti exec, passthru, shell_exec, system yang ada pada php.ini dan ditambahkan pada disable_functions.

Tahapan Validasi

a. Validasi Kerentanan Local File Inclusion (LFI)

Berikut hasil pengujian setelah dilakukan tahapan *defense* untuk celah yang muncul pada *Local File Inclusion*.

Tabel 1. Validasi kerentanan *Local File Inclusion*

Nama Pengujian	Pengujian Pada Tahapan Attack	Pengujian Pada Tahapan Defense
Eksekusi File Sensitif pada URL	Berhasil	Tidak Berhasil
Eksekusi File Sensitif Melalui *directory traversal*	Berhasil	Tidak Berhasil
Membaca isi file konfigurasi di server	Berhasil	Tidak Berhasil

b. Validasi Kerentanan Remote File Inclusion (RFI)

Berikut hasil pengujian setelah dilakukan tahapan *defense* untuk celah yang muncul pada *Remote File Inclusion*.

Tabel 2. Validasi kerentanan *Remote File Inclusion*

Nama Pengujian	Pengujian Pada Tahapan Attack	Pengujian Pada Tahapan Defense
Melakukan *Remote Code Execution* memanfaatkan PHP Wrapper	Berhasil	Tidak Berhasil
Eksekusi *evilscript* dari melalui website penyerang	Berhasil	Tidak Berhasil
Eksekusi *reverse shell code* melalui website penyerang	Berhasil	Tidak Berhasil

Dari tahapan validasi celah-celah yang sebelumnya muncul sudah berhasil ditutup melalui tahapan *defense*.

KESIMPULAN

Berdasarkan hasil penelitian yang telah melalui tahapan *attack*, tahapan *defense* dan tahapan validasi dapat ditarik beberapa kesimpulan sebagai berikut:

1. Sebelum sebuah website masuk ke fase *live* atau bisa diakses oleh orang banyak hendaknya dilakukan pengujian terlebih dahulu.
2. Dampak dari file inclusion adalah terbacanya file sensitif di server, terbacanya file konfigurasi website di server, remote code execution dan berhasilnya penyerang masuk ke server dan mengakses shell di server.
3. Salah satu dampak yang paling berbahaya pada celah file inclusion adalah dapat diambil alihnya server jika penyerang sudah berhasil mengakses shell di server.
4. *File inclusion* terjadi karena code yang dituliskan tidak difilter dan divalidasi dengan benar.
5. Local File Inclusion dapat ditutup dengan menambahkan validasi seperti menambahkan ekstensi file yang akan dimuat, menambahkan *whitelist* file apa saja yang boleh dimuat.
6. *Remote File Inclusion* dapat ditutup dengan mengganti konfigurasi allow_url_fopen dan allow_url_include menjadi Off dan menambahkan disable_functions seperti exec, shell_exec, passthru dan system pada file php.ini.

DAFTAR PUSTAKA

[1] Owasp 2013, OWASP Top 10-2013 "The Ten Most Critical Web Application Security Risk" from https://www.owasp.org/images/f/f8/OWASP_Top_10_-_2013.pdf

[2] Owasp 2017, OWASP Top 10-2017 "The Ten Most Critical Web Application Security Risk" from https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_%28en%29.pdf.pdf.

[3] A. Prof, P. V Ami, and A. P. S. C. Malav, "Top Five Dangerous Security Risks over Web Application," vol. 2, no. 1, pp. 41–43, 2013.

[4] M. M. Hassan, T. Bhuyian, M. K. Sohel, M. H. Sharif, and S. Biswas, "SAISAN: An automated Local File Inclusion vulnerability detection model," *Int. J. Eng. Technol.*, vol. 7, no. 2, pp. 4–8, 2018, doi: 10.14419/ijet.v7i2.3.9956.

[5] A. Begum, M. M. Hassan, T. Bhuiyan, and M. H. Sharif, "RFI and SQLi based local file inclusion vulnerabilities in web applications of Bangladesh," *IWCI 2016 - 2016 Int. Work. Comput. Intell.*, no. December, pp. 21–25, 2017, doi: 10.1109/IWCI.2016.7860332.

[6] D. R. Sahu and D. S. Tomar, "Defensive Programming to Reduce PHP Vulnerabilities," no. July, 2014.

[7] K. Das, "Linux command line for you and me Documentation," 2019.

[8] S. Biswas, M. Sohel, M. M. Sajal, T. Afrin, T. Bhuiyan, and M. M. Hassan, "A Study on Remote Code Execution Vulnerability in Web Applications," *Int. Conf. Cyber Secur. Comput. Sci.*, no. October, pp. 1–8, 2018.