

From DOS to Remote Command Execution: ScriptFTP Unicode Stack Overflow (0day)

A Whitepaper by Spentera Security

<http://www.spentera.com>



Ringkasan Eksekutif

Stack atau Buffer Overflow adalah sebuah anomali di mana sebuah program, ketika menulis data ke dalam buffer, melewati batas buffer dan menimpa memori yang berdekatan dengan program memory. Untuk membuat kondisi Buffer Overflow menguntungkan, biasanya dengan mengisi register EIP (Instruction Pointer) dengan sebuah Return Address dan memulainya dari sana.

ScriptFTP adalah program FTP client yang menggunakan batch script untuk memproses permintaan terhadap FTP server. Program ScriptFTP menggunakan Unicode (UTF-16) sebagai encoding data pada program.

Paper ini akan memaparkan kelemahan buffer overflow pada ScriptFTP yang berbasis Unicode dan SEH.

Disclaimer:

Exploit ini belum dipublikasikan dan masih berstatus **Coordinated Disclosure** dengan pihak ScriptFTP. Tentunya tulisan ini juga berstatus privat milik Spentera Security dan ScriptFTP.

Latar Belakang

Bagaimana sebuah program ScriptFTP diketahui memiliki bug? Tentu saja hal ini tidak bisa diketahui dengan hanya menebak, tapi dengan melakukan riset (fuzzing framework) maka akan didapatkan hasil yang signifikan. Program ScriptFTP diketahui kemudian mengalami kegagalan dalam mengatasi perintah LIST yang anomali. Kemudian hasil tersebut bisa dikembangkan untuk menjadi studi kasus. Secara umum, program FTP client akan melakukan koneksi ke FTP server, proses awalnya selalu sama, koneksi ke port 21, mengirimkan username dan password, melihat opsi/perintah yang bisa dijalankan di FTP server, lalu menunggu perintah selanjutnya.

Percobaan dapat dilakukan dengan mengirimkan paket yang anomali atau sesuatu yang tidak wajar. Misalkan kita mau login, tentu saja username dan password akan kita isi dengan karakter-karakter "a-z", "A-Z", atau "0-9", dan memiliki panjang yang sesuai (karena kita memang mau login secara sah ke FTP server). Namun bagaimana jika kita mengirimkan karakter yang panjang dan anomali, misalkan mengirimkan username dengan 5000 karakter A dan password dengan 10000 karakter '{', apakah FTP client akan merespon dengan normal? Atau malah terjadi hal yang lain, crash misalnya? Teknik inilah yang dinamakan fuzzing.

ScriptFTP mengalami kegagalan dalam mengatasi perintah LIST yang anomali, hal ini dibuktikan dengan tertimpanya nilai pada alamat SE Handler dan Pointer to Next SEH. Dengan tertimpanya alamat SE Handler, berarti membuktikan bahwa aplikasi ScriptFTP mengalami Stack atau Buffer Overflow. Pada tipe buffer overflow seperti ini, alamat EIP akan dapat dikuasai setelah kondisi *exception* diteruskan.

Eksplotit Berbasis SEH

Pembuatan eksploit berbasis SEH berbeda dengan pembuatan eksploit biasa karena SEH berperan dalam menangani *exception* yang terjadi ketika program mengalami buffer overflow. Namun demikian, SEH tetap dapat diatasi dengan memanfaatkan instruksi POP POP RET untuk menarik kembali User Controlled Buffer yang berhasil masuk ke memory.

Ketika SE Handler tertimpa dengan alamat yang memiliki urutan POP POP RET, maka perintah POP POP akan tereksekusi dan perintah terakhir, yaitu RET akan membawa User Controlled Buffer kembali ke ESP. Dengan demikian, ESP akan menunjuk ke alamat Pointer to Next SEH. Biasanya yang terjadi berikutnya adalah menimpa alamat

Pointer to Next SEH dengan alamat JMP SHORT agar dapat melewati alamat SE Handler. Setelah berhasil melewati SE Handler, perintah berikutnya bisa diawali dengan NOP sebelum akhirnya sampai ke shellcode.

Masalah Unicode

Mengapa developer menggunakan Unicode sebagai encoding? Unicode memungkinkan kita untuk secara visual memanipulasi teks sebagian besar sistem di seluruh dunia secara konsisten. Jadi aplikasi dapat digunakan di seluruh dunia, tanpa harus khawatir bagaimana teks tersebut akan terlihat ketika ditampilkan di komputer - hampir semua komputer - di seluruh dunia. Dan perlu diingat, bahwa ketika bermain-main dengan exploit berbasis Unicode, maka karakter yang bisa digunakan untuk keperluan instruksi assembly (opcodes) dan pembuatan shellcode menjadi sangat terbatas. Batas aman tersebut dimulai dari 01 – 7F, yang merupakan representasi karakter pada keyboard.

Unicode based exploit pada awalnya ditinggalkan karena banyak bug hunter yang belum tahu bagaimana mengatasi Unicode (membuat exploit), namun paper Chris Anley (2002) menjawab itu semua dan keluarlah istilah Venetian Shellcode. Pada tahun 2003, Obscou menulis di Phrack magazine tentang bagaimana teknik Chris Anley dipakai untuk menulis shellcode, beberapa minggu kemudian Dave Aitel dari Immunity.Inc membuat script untuk mempercepat penulisan shellcode Obscou. Tahun berikutnya (2004), FX berhasil menyempurnakan script yang ada sebelumnya. SkyLined kemudian menyempurnakan penulisan shellcode dalam bentuk Uppercase dan Unicode compatible yang kita kenal dengan alpha2 encoder. Tahun 2009, Peter Van Eeckhoutte menyempurnakan semua dokumen tersebut menjadi 1 dokumen, sehingga lebih menarik dan mudah dipahami.

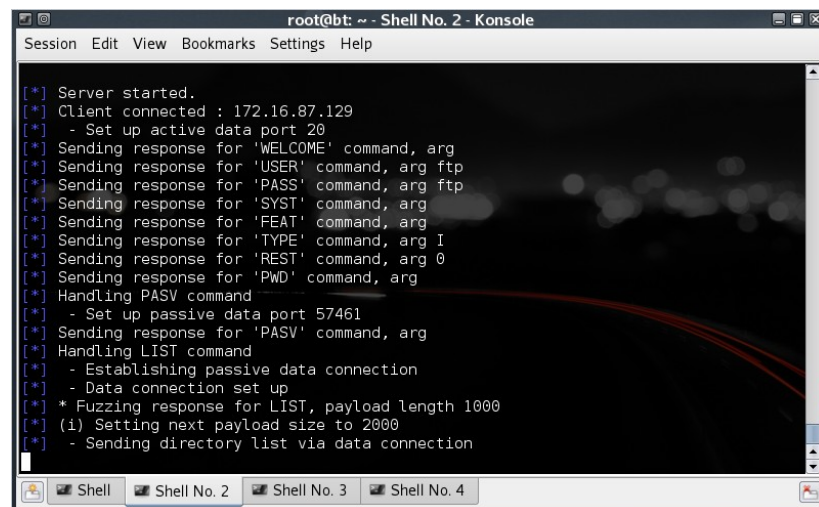
Program ScriptFTP berbasis Unicode, sehingga teknik pembuatan exploit biasa (ordinary SEH or direct RET) tidak akan berjalan lancar. Beberapa teknik dan pengetahuan tentang CPU register, perintah/instruksi di Assembly akan sangat diperlukan untuk membuat Venetian Shellcode, yaitu sebuah shellcode yang dipersiapkan untuk menjadi jalan bagi shellcode lainnya.

Exploit berbasis Unicode merupakan exploit yang tidak sederhana, dan membutuhkan ketelitian untuk bereksperimen dengan stack dan register.

Proses Overflow

Proses fuzzing pada program ScriptFTP menggunakan fuzzer dari Metasploit. Bisa

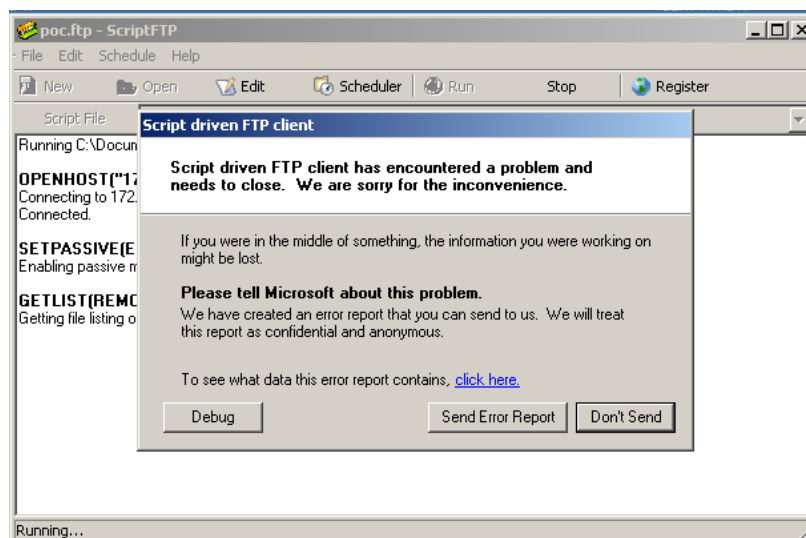
dilihat dari gambar 1.0 berikut, Metasploit client ftp fuzzer bertindak sebagai server dummy yang akan mengirimkan data yang anomali terhadap perintah-perintah yang diminta oleh ftp client (ScriptFTP). Ketika program ScriptFTP dijalankan dan terkoneksi dengan Metasploit fuzzer, maka yang terjadi adalah:



```
root@bt: ~ - Shell No. 2 - Konsole
Session Edit View Bookmarks Settings Help

[*] Server started.
[*] Client connected : 172.16.87.129
[*] - Set up active data port 20
[*] Sending response for 'WELCOME' command, arg
[*] Sending response for 'USER' command, arg ftp
[*] Sending response for 'PASS' command, arg ftp
[*] Sending response for 'SYST' command, arg
[*] Sending response for 'FEAT' command, arg
[*] Sending response for 'TYPE' command, arg I
[*] Sending response for 'REST' command, arg 0
[*] Sending response for 'PWD' command, arg
[*] Handling PASV command
[*] - Set up passive data port 57461
[*] Sending response for 'PASV' command, arg
[*] Handling LIST command
[*] - Establishing passive data connection
[*] - Data connection set up
[*] * Fuzzing response for LIST, payload length 1000
[*] (i) Setting next payload size to 2000
[*] - Sending directory list via data connection
```

Gambar 1.0 – Metasploit FTP Client Fuzzing Tool



Gambar 2.0 – Program ScriptFTP Crash

Program ScriptFTP crash saat mencoba untuk membaca file/folder pada Metasploit dummy ftp server. Perintah LIST yang diminta oleh ftp client direspon dengan nama file yang anomali (panjang karakter 2000 bytes) oleh ftp server, ternyata hal ini membuat program ftp client crash.

Verifikasi Bug Aplikasi

Verifikasi bug biasanya gampang-gampang susah. Biasanya pembuat exploit akan membuat sederet karakter-karakter secara acak, lalu mengirimkannya ke posisi dimana program tersebut tidak dapat menampung data-data tersebut. Proses ini

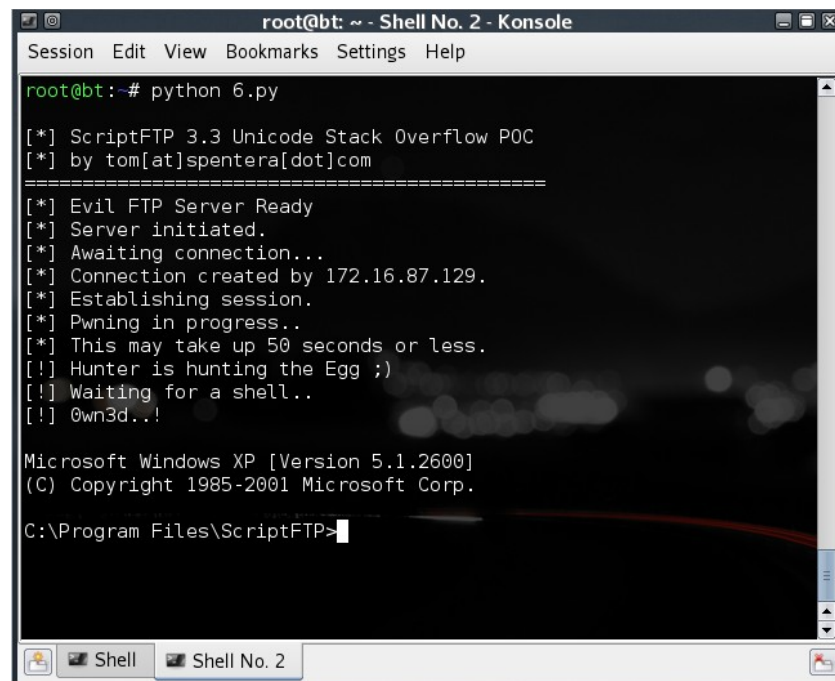
biasanya sebentar, namun ada kalanya ketika melakukan verifikasi ini terjadi hal yang tidak diinginkan, seperti misalnya buffer yang tertimpa ketika melakukan fuzzing dan ketika melakukan verifikasi ternyata berbeda.

Tujuan verifikasi ini untuk menentukan beberapa bytes yang menempa instruction pointer (EIP) atau menempa SE Handler (jika berbasis SEH).

Eksplorasi Unicode Buffer Overflow

Proses eksploitasi akan membuat Venetian Shellcode, agar shellcode yang kita siapkan bisa dieksekusi dengan baik. Selain itu, masalah yang paling sering terjadi dalam pembuatan exploit adalah ketika jumlah buffer tidak mencukupi untuk menampung shellcode yang kita inginkan.

Pada percobaan ini kali ini penggunaan egghunter tidak dapat dihindari, hal ini karena terbatasnya buffer yang dapat kita manfaatkan untuk mengisi shellcode. Namun sekali lagi, karena ini berbasis unicode, segalanya harus dilakukan dengan cara unicode :)



```
root@bt: ~ - Shell No. 2 - Konsole
Session Edit View Bookmarks Settings Help

root@bt:~# python 6.py

[*] ScriptFTP 3.3 Unicode Stack Overflow POC
[*] by tom[at]spentera[dot]com
=====
[*] Evil FTP Server Ready
[*] Server initiated.
[*] Awaiting connection...
[*] Connection created by 172.16.87.129.
[*] Establishing session.
[*] Pwning in progress..
[*] This may take up 50 seconds or less.
[!] Hunter is hunting the Egg ;)
[!] Waiting for a shell..
[!] 0wn3d...!

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Program Files\ScriptFTP>
```

Gambar 3.0 – Command Execution pada ScriptFTP

Mencegah Buffer Overflow

Pengembang software dapat mencegah serangan buffer overflow pada saat fase desain dalam tahap Software Development Life Cycle (SDLC). Selama fase desain, pengembang dapat menentukan spesifikasi tertentu mengenai pelaksanaan prosedur panggilan untuk input dan manipulasi data, serta menambahkan exception handling ke dalam program desain. Selain itu, audit juga harus dilakukan pada interval yang teratur ketika proses SDLC dilakukan sehingga dapat memastikan bahwa hanya prosedur yang didefinisikan yang akan diterima, dan memastikan hanya proses pengendalian yang dikerjakan selama proses pengembangan.

Menggantungkan program pada proteksi yang dimiliki sistem operasi merupakan kesalahan yang fatal dan sering terjadi. Proteksi pada sistem operasi seperti Structured Exception Handling (SEH), SafeSEH (Safe Exception Handlers), Stack Cookies, Stack-Smashing Protection (ProPolice), Address Space Layout Randomization (ASLR), atau Data Execution Prevention (DEP) tidak sepenuhnya aman karena sudah banyak peneliti yang mencoba melewati proteksi tersebut dan mereka pun mengklaim berhasil. Untuk itu, proteksi tambahan pada program sangat disarankan pada proses SDLC.

Referensi

1. Building IA32 'Unicode-Proof' Shellcodes - obscou
<http://www.phrack.org/issues.html?issue=61&id=11#article>
2. Vivisection of an Exploit : What To Do When It Isn't Easy – Dave Aitel
<http://www.blackhat.com/presentations/win-usa-03/bh-win-03-aitel/bh-win-03-aitel.pdf>
3. Vulnerability Finding in Win32—A Comparison - FX
<http://www.blackhat.com/presentations/win-usa-04/bh-win-04-fx.pdf>
4. ALPHA2: Zero tolerance, Unicode-proof uppercase alphanumeric shellcode encoding.
http://skypher.com/wiki/index.php?title=Www.edup.tudelft.nl/~bjwever/documentation_alpha2.html.php
5. Exploit writing tutorial part 7 : Unicode – from 0x00410041 to calc
<http://www.corelan.be/index.php/2009/11/06/exploit-writing-tutorial-part-7-unicode-from-0x00410041-to-calc/>

Tentang Penulis

Penulis adalah seorang yang antusias dengan dunia IT security :)
What do you expected?

Hanny Haliwela

hanny@spentera.com

Identitas GPG: 0xEDE1709E

Sidik jari: 5D5A 02F8 4F70 6EE9 8652 C423 6E0B 78DF EDE1 709E

Tom Gregory

tom@spentera.com

Identitas GPG: 0xEE56FBFD

Sidik jari: CDCF 009A 82DC 624A 4D52 6154 06BA D3D4 EE56 FBFD