Desain dan Implementasi Simulator Kernel *Exploitation* pada Rasberry Pi Menggunakan Docker dan Qemu

Lambang Akbar Wijayadi, Ridho Rahman Hariadi, dan Hatma Suryotrisongko
Departemen Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)
e-mail: ridho@if.its.ac.id

***Abstrak*—Jumlah penggunaan dan pengembangan piranti IoT semakin meningkat. Implementasi Raspberry Pi dalam IoT juga menunjukan efek yang positif dengan beberapa keuntungan misalnya harga yang murah, kekuatan komputasi yang mumpuni, dan kustomisasi. Beberapa developer juga mengembangkan kernel module pada Raspberry Pi untuk menyelesaikan permasalahan komputasi, hal ini nyatanya memperluas *attack surface* bagi keamanan IoT itu sendiri. Namun keamanan menjadi aspek yang sering tertinggal maupun ditinggalkan, sebagai contohnya praktisi keamanan siber masih begitu awam dengan pengembagan eksploitasi kernel pada arsitektur yang ARM yang digunakan pada Raspberry Pi. Salah satu cara untuk menyelesaikan permasalahan tersebut dengan mengembangkan pelatihan yang mampu memberikan pengalaman *hands-on* eksploitasi seperti di keadaan nyata dan edukasi yang interaktif. Edukasi di bidang keamanan siber sudah dikembangkan dengan memertimbangkan aspek gamifikasi dan *hands-on*, salah satu produknya adalah CTFd. Namun praktik *hands-on* dalam edukasi keamanan siber cukup rumit. Misalnya pada eksploitasi kernel, jika peserta salah berinteraksi dengan kernel dan mengakibatkan sistem *crash*, maka perlu melakukan *reboot* secara berulang. Sehingga dalam penelitian ini, penulis mengembangkan simulator eksploitasi kernel pada Raspberry Pi menggunakan Docker dan QEMU, pada sistem sudah memilik kernel modul yang didesain memilik celah keamanan yang bisa digunakan untuk mendapatkan hak akses *root*.**

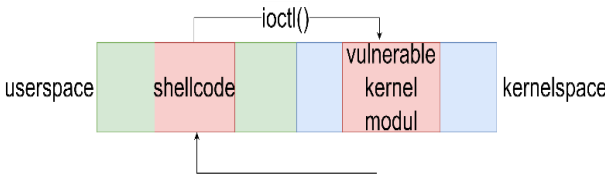
***Kata Kunci*—Eksploitasi Kernel, Simulasi Eksploitasi, *Cyber Ranges*, Eksploitasi ARM.**

I. PENDAHULUAN

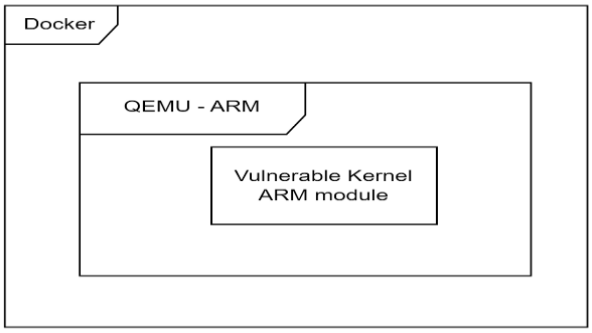
LEBIH banyak piranti *Internet of Things* yang di-*deploy* akhir akhir ini, pada tahun 2021 terdapat 13 Milyar piranti IoT yang ada di internet. Seiring dengan bertambahanya jumlah perangkat IoT, serangan *cyber* yang mengarah ke perangkat IoT juga semakin bertambah, pada tahun 2021 terdapat lebih dari 1 milyar serangan menyasar piranti IoT.

SoC atau *system-on-chip* merupakan teknologi yang memungkinkan pengembangan perangkat IoT menjadi lebih marak. Beberapa perangkat *SoC* menggunakan linux kernel sebagai sistem operasi, salah satunya adalah Raspberry Pi [1]. Pengembangan kernel modul pada Raspberry Pi juga menjadi salah satu solusi untuk masalah komputasi yang sering dihadapi [2]. Namun, luasnya pengembangan kernel modul pada Raspberry Pi juga menambah *attack vector* pada keamanan kernel, sehingga satu kesalahan atau celah keamanan pada kernel modul dapat berakibat fatal pada sistem secara keseluruhan [3].

Kernel merupakan *controller* dari sistem operasi, artinya ialah yang mengatur segala adminstrasi yang berjalan didalam sebuah sistem. Operasi – operasi yang diatur kernel



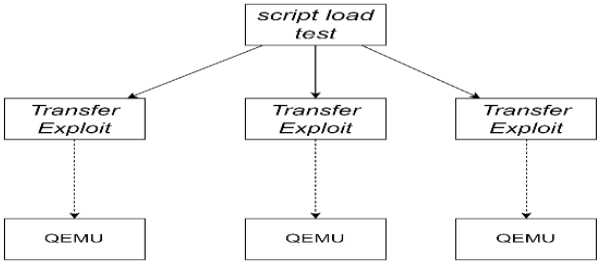
Gambar 1. Ilustrasi teknik ret2usr.



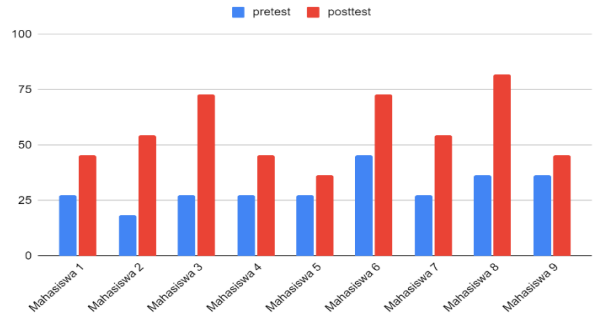
Gambar 2. Desain implementasi QEMU dan Docker untuk menjalankan vulnerable kernel modul.

begitu beragam, mulai dari menampilkan input dari *user* sampai mengatur alokasi memori ataupun membuka dan menutup port. Untuk menjalankan tugas ini, kernel memiliki *privilege* yang sangat tinggi didalam sistem operasi, kernel harusnya menjadi tempat yang paling aman dan dapat dipercaya. Sedangkan kernel hanyalah program yang awalnya dikembangkan dengan bahasa pemrograman C, dan beberapa penelitian sudah membuktikan bahwasanya bahasa C lebih sulit untuk diamankan karena fleksibilitasnya [4-5]. Terdapat 3003 celah keamanan pada kernel, diantaranya adalah CVE-2010-3904, CVE-2010-1146, CVE-2021-26708, dirty cow *exploit* yang bisa digunakan untuk mendapatkan *privilege* akses dari user root [6].

Dengan banyaknya serangan *cyber* dan kebutuhan akan sumber daya manusia di bidang keamanan siber, pembelajaran mengenai pemahaman dan kemampuan teknis di bidang keamanan siber menjadi lebih penting dari sebelumnya. Pengajaran *cyber security* yang diselingi dengan praktik secara langsung dibuktikan efektif [7]. Namun pembelajran mengenai eksploitasi kernel dan secara spefisik pada arsitektur ARM belum begitu banyak diimplementasikan. Praktik *hands-on* pada bidang keamanan siber, memiliki tantanganya tersendiri, misalnya ketika mempraktikan eksploitasi kernel, tak jarang kernel akan *crash* dan mengakibatkan sistem operasi menjadi *stuck* serta memerlukan *reboot* berkali – kali. Oleh karena itu, diperlukan lingkungan simulasi untuk menjalankan eksploitasi kernel



Gambar 3. Penggunaan load test.



Gambar 4. Diagram kolom hasil pretest dan posttest.

Tabel 1.
Pertanyaan Kuesioner dan Korelasi Parameter

No	Pertanyaan	Parameter
Q1	Apakah lebih suka menyelesaikan CTF atau tugas *homework* biasa di kelas keamanan siber mendatang?	Preferensi CTF dan tugas tradisional
Q2	Apakah 3 minggu pembelajaran kernel exploitation membuatmu tertarik dengan bidang ini?	Ketertarikan
Q3	Apakah kamu merasakan mendapat skill baru setelah pembelajaran kernel exploit?	Kompetensi Baru
Q4	Apakah simulator secara akurat mensimulasikan lingkungan perangkat Raspberry Pi dalam eksploitasi kernel?	Akurasi dari simulator
Q5	Menurut anda seberapa mempermudah simulator qemu dan docker dibandingkan dengan perangkat Raspberry Pi asli dalam praktik eksploitasi kernel?	Kegunaan simulator pada eksploit kernel dibandingkan perangkat keras Raspberry Pi
Q6	Bagaimana Anda menilai keefektifan dan kemudahan penggunaan berlatih di simulator dibandingkan dengan berlatih di perangkat Raspberry Pi yang sebenarnya?	Efektifitas dan kemudahan simulator

pada arsitektur ARM dengan tujuane untuk memberikan pemahaman praktik bagi peserta pembelajaran.

II. TEORI PENUNJANG

Platform pembelajaran eksploitasi khususnya pada *binary file* cukup banyak bersebaran di internet. Namun cukup jarang menemui pembelajaran mengenai kernel khususnya pada arsitektur ARM. Eksploitasi pada arsitektur ARM berbeda dengan arsitektur X86_64 yang prosessornya lebih sering kita jumpai, karena menggunakan set instruksi yang berbeda.

Penelitian yang menggunakan emulasi dengan sistem VDI untuk menghindari terjadinya *crash* dan membandingkan pembelajaran tersebut dengan pembelajaran tradisional [7].

Penggunaan *capture the flag (CTF)* pada pembelajaran keamanan siber sudah sangat umum digunakan. *CTF* awalnya digunakan sebagai ajang kompetisi, namun beberapa penelitian yang menggunakan *CTF* pada pembelajaran terbukti lebih disukai oleh peserta [8-9]. Salah satu platform *CTF* yang fleksibel dan populer adalah CTFd [10].

III. PERANCANGAN

A. Menentukan Teknik Kernel Eksploitasi

Teknik yang dapat digunakan pada eksploitasi kernel beragam, misalnya *null pointer dereference, ret2usr, ret2dir, use-after-free, buffer overflow, write-what-where*, dan *race condition*. Teknik – teknik tersebut dapat digunakan untuk melakukan *privilege escalation* ataupun leak informasi sensitif yang tersimpan di dalam kernel. Penulis melakukan analisa terhadap ketuju teknik tersebut dan menarik kesimpulan bahwasanya teknik *ret2usr* memiliki kompleksitas yang cukup rendah dibandingkan teknik yang lainya. Terdapat teknik lain yang juga memiliki konsep yang sederhana misalnya *null pointer dereference*, namun teknik ini tidak dapat digunakan pada arsitektur ARM dikarenakan tidak dapat menyalin *payload* ke *null pointer*. Teknik *ret2usr* akan mengarahkan kernel untuk mengeksekusi program yang sudah dibuat penyerang dan berada pada daerah memori yang dikontrol oleh penyerang. Gambar 1 merupakan gambaran penyerangan teknik *ret2usr*.

B. Pembuatan Vulnerable Kernel Module

Secara garis besar, tahapan pembuatannya sebagai berikut:

1) Pengembangan Vulnerable Kernel Modul

Kernel modul dikembangkan sedemikian hingga dapat dieksploitasi dengant teknik *ret2usr*. Untuk memungkinkan hal tersebut terjadi maka kernel perlu diarahkan untuk mengeksekusi code yang berada disisi *userspace.* Pada kasus ini, kernel berkomunikasi dengan user menggunakan fungsi *ioctl.* Dari fungsi ioctl akan mengirimkan sebuah struct dengan parameter *address* dan tipe data *unsigned long int*. Parameter tersebut akan *dicopy* ke *kernelspace* dengan fungsi *copy_from_user*, yang disimpan dalam *struct param* *. Selanjutnya, *address input* dari *user* akan dipindahkan ke fungsi *blank()*, dimana fungsi *blank()* ini akan dieksekusi oleh *kernelspace*.

2) Konfigurasi Kernel

ARM kernel memilik *security feature* bernama PAN atau *privilege access never.* Fitur ini tidak memungkinkan kernel untuk mengeksekusi *code* yang berada pada sisi *user*. Untuk membuat kernel modul dapat dieksploitasi menggunakan teknik *ret2usr*, maka fitur ini perlu dimatikan dengan cara menambahkan CONFIG_CPU_SW_DOMAIN_PAN=n ketika me-*compile* kernel.

3) Eksploitasi Kernel Modul

Kernel modul yang sudah didesain memiliki celah keamanan perlu dilakukan *testing* apakah modul tersebut benar dapat dieksploitasi atau tidak. Pengembangan eksploitasi pada set instruksi CISC dan RISC berbeda, sehingga diperlukan pemahaman yang mendalam tentang bagaimana set instruksi RISC yang digunakan pada ARM. Tujuan dari eksploitasi kernel adalah dengan mendapatkan *privilege escalation*, salah satu caranya adalah dengan menajalankan fungsi yang dapat merubah *privilege* yaitu commit_creds(prepare_kernel_cred(0).

Tabel 1.
Hasil Evaluasi Pertama dan Perbaikan

Mahasiswa	Nilai Evaluasi Pertama	Nilai Evaluasi Perbaikan
Mahasiswa 1	40	90
Mahasiswa 2	90	100
Mahasiswa 3	80	100
Mahasiswa 4	0	90
Mahasiswa 5	40	-
Mahasiswa 6	100	-
Mahasiswa 7	20	90
Mahasiswa 8	100	-
Mahasiswa 9	100	-

Tabel 2.
Jumlah Peserta Mengumpulkan Jawaban Salah

Mahasiswa	*Wrong flag* Evaluasi I	*Wrong flag* evaluasi perbaikan
Mahasiswa 1	39 (88 %)	12 (55 %)
Mahasiswa 2	22 (69 %)	2 (15 %)
Mahasiswa 3	18 (69 %)	6 (37 %)
Mahasiswa 4	0 (0 %)	58 (86 %)
Mahasiswa 5	16 (80 %)	-
Mahasiswa 6	15 (57 %)	-
Mahasiswa 7	9 (81 %)	10 (53 %)
Mahasiswa 8	28 (71 %)	-
Mahasiswa 9	11 (50 %)	-

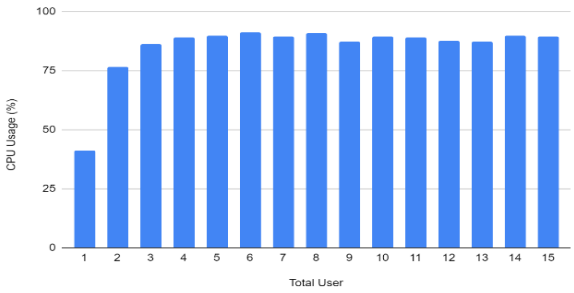
Terdapat beberapa perbedaan pada shellcode x86 dengan ARM, utamanya pada baris ketiga dan keempat. Pada baris ketiga *attacker* perlu merubah nilai dari register M_CSPR, register tersebut berguna untuk mengetahui status dari proses yang sedang berjalan, termasuk apakah proses dijalankan pada *usermode* atau *kernelmode*. Sedangkan pada baris keempat merupakan salah satu *address* dari fungsi pada program *exploit* yang menjalankan proses baru untuk *spawning shell*.

C. Desain Simulator Kernel Exploitation

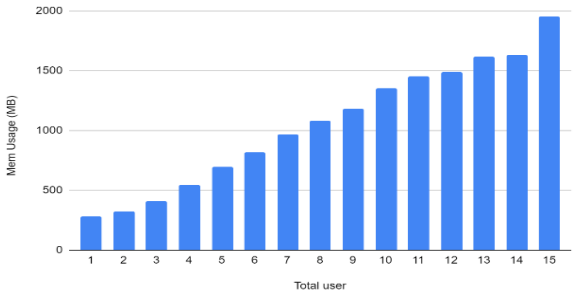
Dalam penelitian ini, kernel akan dikontainerisasi menggunakan docker dan qemu untuk membuat *isolated environment* yang aman dan mudah dikelola. Kontainerisasi ini akan membantu dalam meningkatkan keamanan sistem operasi dan mengelola *resource* dengan lebih efisien. Penggunaan docker dan qemu akan memisahkan kernel dari host sistem operasi, sehingga apabila terdapat celah keamanan pada kernel, tidak akan merusak sistem operasi utama. Alat ini juga memudahkan portabilitas penelitian dan manajemen resource pada kernel. Penggunaan qemu sebagai isolasi kernel modul sudah diterapkan pada beberapa pembelajaran, misalnya pada pwn.college (Gambar 2).

D. Pembuatan Learning Path dan Deployment CTFd

Dalam evaluasi, peserta akan diberikan soal pada masing masing tahapan pengerjaan eksploitasi kernel. Hal ini ditujukan supaya peserta sadar bahwasanya mereka sedang berada pada jalur yang tepat dan tidak malah masuk kedalam *rabbit hole*. Dalam implementasinya peserta akan diberikan soal dalam beberapa tahapan, tahapan selanjutnya hanya bisa dibuka jika peserta sudah menyelesaikan tahapan yang sekarang. Sebagai contoh, semisal evaluasi dilakukan dalam 3 tahapan dengan 4 soal per-masing – masing tahapan, maka untuk dapat mengerjakan soal nomer 5 maka peserta perlu menyelesaikan soal no 1 – 4 terlebih dahulu. Penulis disini membuat sebuah panduan untuk mengerjakan eksploitasi kernel pada akun github penulis beserta video pengerjaan, panduan didesain untuk mudah diikuti dan mudah dipahami.



Gambar 5. Diagram kolom penggunaan CPU oleh kontainer.



Gambar 6. Diagram kolom penggunanaan memory oleh kontainer.

Platform yang digunakan untuk melakukan evaluasi adalah CTFd versi 3.5.0. Tidak semua soal – soal yang diberikan pada saat evaluasi berbentuk *flag based*, ada juga soal yang menanyakan nilai dari memori, atau perintah yang perlu dijalankan dan sebagainya. Untuk menangani permasalahan ini, maka penulis memanfaatkan fitur *regular expression* pada platform dan juga menambahkan kemungkinan *flag* lain yang dikumpulkan oleh peserta yang dianggap benar oleh penulis namun terdapat perbedaan penulisan

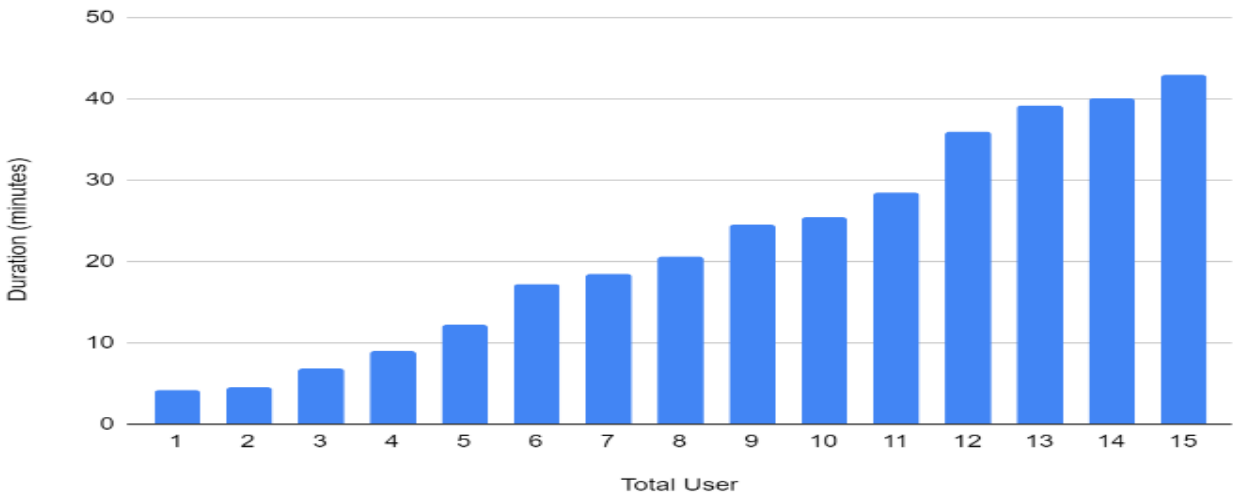
E. Evaluasi Pembelajaran

Penelitian ini mengevaluasi pemaham peserta mengenai eksploitasi kernel pada arsitektur ARM, Praktik kernel eksploitasi dijalankan sebagai evaluasi akhir semester pada kelas *malware* dan *hacking* dengan jumlah mahasiswa adalah 9, terdiri dari 8 mahasiswa angkatan 2019 dan 1 mahasiswa angkatan 2018, semua mahasiswa berjenis kelamin laki – laki. Dalam evaluasi pembelajaran kami mengumpulkan beberapa data melalui kuesioner, *multiple choice pre-* dan *post-test*, dan log dari platform ujian.

F. Load Test Infrastruktur

Untuk mengetahui resiliensi dari infrastruktur yang digunakan untuk evaluasi, utamanya adalah *virtual machine* yang akan di-*deploy* ketika peserta terhubung ke server. Untuk memastikan berjalanya evaluasi pembelajaran lancar, maka dalam penelitian ini akan melakukan *load testing* dengan menjalankan pengiriman eksploitasi secara pararel dengan menggunakan beberapa user. Hal ini dapat terlaksana dengan mengembangkan *script* dengan *threading* untuk menjalankan program lain yang digunakan untuk *transfer exploit*, Gambar 3 menunjukan bagaimana program menjalankan *load test*.

Script load testing ini akan dijalankan pada *virtual machine* ubuntu versi 20.04 dengan spesifikasi RAM 3218 MB dan 1 core CPU Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz. Selama menjalankan program *load testing*, kita dapat mengetahui penggunaan kontainer dalam docker dengan menjalankan docker stats, dari perintah tersebut kita



Gambar 7. Diagram kolom waktu pengiriman eksploit.

bisa mendapatkan penggunaan CPU, penggunaan memori, data yang dikirimkan melalui jaringan (Network I/O), dan data yang dibaca ataupun ditulis ke sistem operasi utama. Untuk memantau penggunaan docker secara otomatis selama proses pengiriman eksploit, kami menggunakan program docker-stat-graph dan akan dikumpulkan nilai rata – rata dari penggunaan CPU, memori, dan durasi pengiriman eksploit.

IV. HASIL DAN PEMBAHASAN

A. Hasil

Hasil dari penelitian ini meliputi hal-hal berikut:

1) Hasil Evaluasi Pembelajaran

Praktik kernel eksploitasi dijalankan sebagai evaluasi akhir semester pada kelas malware dan hacking dengan jumlah mahasiswa adalah 9, terdiri dari 8 mahasiswa angkatan 2019 dan 1 mahasiswa angkatan 2018, semua mahasiswa berjenis kelamin laki – laki. Dalam evaluasi pembelajaran kami mengumpulkan beberapa data melalui kuesioner, *multiple choice pre-* dan *post-test,* dan log dari platform ujian.

2) Hasil Evaluasi dari Kuisioner

Pada akhir pembelajaran penulis menyempatkan membuat dua kuesioner dengan kuesioner pertama ditanyakan kepada 9 mahasiswa dan kuesioner kedua ditanyakan kepada 4 mahasiswa yang sudah pernah melakukan eksploitasi pada perangkat keras Raspberry Pi. Tabel 1 menunjukan daftar pertanyan beserta parameter yang ingin didapatkan pada masing – masing pertanyaan.

Pertanyaan dalam kuesioner dijawab dengan basis skala dengan nilai 1 sebagai respon paling negatif dan 5 sebagai respon paling positif. Pada pertanyaan Q1, terdapat 3 (33%) mahasiswa memilih skala 3, 2 (22 %) mahasiswa memilih skala 4, dan 4 (44 %) mahasiswa memilih skala 5. Pada pertanyaan Q2, terdapat 1 (11 %) mahasiswa memilih skala 2, 4 (44 %) mahasiswa memilih skala 4, dan 4 (44 %) mahasiswa memilih skala 5. Pada Q3, terdapat 1 (11 %) mahasiswa memilih skala 2, 1 (11 %) mahasiswa memilih skala 3, 2 (22 %) mahasiswa memilih skala 4, dan 5 (56 %) mahasiswa memilih skala 5.

Selanjutnya pada pertanyaan Q4 – Q6 hanya ditanyakan kepada 4 mahasiswa. Pada Q4, terdapat 2 (50 %) mahasiswa memilih skala 3, dan 2 (50 %) mahasiswa memilih skala 4. Pada Q5, semua peserta menjawab dengan skala 5, dan pada Q6, 1 mahasiswa menjawab skala 4 dan sisanya menjawab skala 5.

3) Hasil Evaluasi dari Nilai Pre- dan Post-test

Dalam penelitian ini dikumpulkan nila *pre-* dan *post-test* bagi ke-9 mahasiswa tersebut setelah evaluasi perbaikan berlangsung dan dengan menggunakan *paired t-test* didapatkan hasil yang secara statistik signifikan. Terdapat 11 pertanyaan dalam *pretest* dan *postest* dikategorikan menjadi 4 domain pengetahuan, yaitu *assembly* atau pengetahuan umum mengenai *excutables file*, *binary exploitation*, kernel, dan *kernel exploitation*. Pada Gambar 4 menunjukan Data yang disajikan sudah diubah dalam bentuk presentase, dimana pada *pretest* nilai rata – rata yang didapatkan adalah 30.3 dengan *standard deviation* 7.9, dan pada *posttest* nilai rata – rata yang didapatkan adalah 56.6 dengan *standard deviation* 15.6. Dengan menggunakan *paired t-test* didapatkan nilai t sebesar 5.6402 dengan derajat kebebasan sebesar 8, menggunakan *two-side tail* dengan alfa = 0.01 didapatkan nilai p sebesar 0.0004869, dikarenakan nilai perbedaan rata rata tersebut positif maka secara statistik terdapat perbedaan yang signifikan antara nilai *pretest* dan *postest.*

4) Hasil Evaluasi dari Platform

Ujian dilaksanakan dua kali pada tanggal 13 desember 2022 dan 16 desember 2022 selama 150 menit. Seperti dijalaskan pada subbab sebelumnya, bahwasanya terdapat 8 tahapan dalam pengerjaan dan dua eksploitasi yang harus diselesaikan.

Ujian kedua diberikan bagi mahasiswa yang menginginkan perbaikan nilai, ujian kedua memilik pertanyaan berbeda dengan ujian pertama. Selain mengamati hasil akhir nilai mahasiswa ketika ujian, penulis juga mengamati apakah mahasiswa mencoba panduan atau mengikut video *walktrough* yang sudah diberikan. Pada Tabel 2 menunjukan nilai mahasiswa yang sudah dikonversi dalam bentuk persentase. Mahasiswa 6 dan 8 pernah bertanya mengenai eksploitasi kernel sebelum ujian secara pribadi maupun dikelas sebelum evaluasi pertama, dan mahasiswa 2 dan 3 pernah bertanya sebelum evaluasi perbaikan. Ujian pertama diikuti oleh 9 mahasiswa dan ujian kedua diikuti oleh 5 mahasiswa.

Dari platform evaluasi didapatkan data berapa kali peserta salah *submit flag* selama evaluasi berlangsung. Pada Tabel 3 menunjukan jumlah peserta salah memasukan *flag* dari evaluasi pertama dan evaluasi perbaikan.

5) Hasil Evaluasi dari Feedback

Setelah evaluasi peserta akan diminta untuk memberikan feedback mengenai pembelajaran kernel eksploitasi pada arsitektur ARM yang sudah dijalankan selama 3 minggu.

6) Hasil Load Test Kontainer

Pada penelitian kali ini, selain mengamati luaran dari pembelajaran penulis juga meneliti bagaimana resiliensi dari server atau container yang digunakan untuk sebagai emulasi kernel. Perlu diketahui bahwa spesifikasi server yang digunakan adalah sebagai berikut 4 vCPU dengan 2.8 GHz, 4 GB RAM, 80 GB Disk Storage.

Load test dijalankan dengan melakukan transfer eksploit program dengan menggunakan 1 – 15 user secara bersamaan.

Pada Gambar 5 menunjukan penggunaan CPU yang sudah dikonversi dalam presentase, ketika digunakan oleh 1 – 15 peserta yang sedang mengirimkan program *exploit.* Penggunaan CPU paling tinggi adalah 91,43 ketika digunakan secara paralel oleh 9 user dan dengan nilai rata rata sebesar 84,98 (Gambar 5).

Pada Gambar 6 menunjukan penggunaan memory (RAM), ketika digunakan oleh 1 – 15 peserta yang sedang mengirimkan program *exploit.* Didapatkan penggunaan RAM paling tinggi sebesar 1954,45 MB ketika dijalankan oleh 15 user secara bersamaan, dan rata rata penggunaan RAM sebesear 1054,11 MB.

Pada Gambar 7 menunjukan waktu yang diperlukan hingga eksploit dari semua user dapat terkirim ke server dalam satuan menit. Didapatkan nilai maksimal dikirimkan ketika terdapat 15 *user* yang mengirimkan eksploit secara bersamaan yaitu 42.9 menit, nilai terendah dengan 1 *user* yaitu 4.1 menit dan nilai rata rata yang dibutuhkan untuk mengirimkan eksloit adalah 21.98 menit.

B. Analisa

Didapatkan dari hasil nilai *pretest* dan *posttest*, bahwasanya pemahaman peserta naik sangat signifikan yaitu 86% persen dimana hal ini dibuktikan juga dari *paired t-test* yang didapatkan nilai p 0.000 dengan nilai sigma alfa 0.01 karena p kurang dari sigma alfa, maka terdapat perbedaan signifikan. Untuk hasil nilai rata – rata evaluasi pertama adalah 63.3/100 dan evaluasi kedua 94/100. Semua mahasiswa berhasil menyelesaikan tantangan pada tahap pertama (*prequel* 0 – 4) kecuali 2 mahasiswa yang memiliki kendala perangkat ditengah evaluasi, Sedangkan pada evaluasi kedua, peserta memiliki kenaikan yang signifikan yaitu mayoritas mahasiswa berhasil menyelesaikan tahapan pertama dan kedua, selain itu 2 peserta berhasil mendapat nilai sempurna pada evaluasi kedua. Dibuktikan juga bahwasanya *engagement* menjadi faktor penting dalam pembelajaran hal ini dibuktikan dari ke 4 mahasiswa yang berhasil mendapat nilai sempurna pernah berkomunikasi ke penulis tentang praktik eksploitasi kernel.

Mahasiswa yang mengumpulkan *flag* yang salah bisa dijadikan indikasi kepemahaman peserta, dimana mahasiswa yang mendapatkan nilai lebih tinggi dengan waktu lebih cepat akan memiliki jumlah *submit wrong flags*-nya lebih sedikit. Jika melihat perilaku peserta yang mengumpulkan jumlah jawaban salah cukup tinggi terdapat dua hal yang perlu dianalisa, yang pertama adalah mahasiswa menebak jawaban dan yang kedua adalah mahasiswa salah menggunakan format *flag.* Sebagai contoh, mahasiswa 1 pada evaluasi pertama melakukan 39 (88 %) kali mengumpulkan jawaban yang salah dikarenakan kesalahan format flag dan melakukan *brute force* jawaban, sedangkan mahasiswa 6, sebagai mahasiswa dengan jumlah mengumpulkan jawaban saling paling sedikit dengan total 13 (53 %) jawaban salah dikarenakan kesalahan penulis dalam pengaturan *flag* pada server.

Berdasarkan hasil dari kuesioner didapatkan bahwa mayoritas mahasiswa lebih menyukai penugasan dalam bentuk CTF dibandingkan bentuk tugas tradisional. Mahasiswa juga memiliki ketertarikan yang tinggi dan merasa mendapatkan *ketrampilan* baru pada bidang eksploitasi kernel selama tiga minggu pembelajaran. Dari hasil kuesioner, 44% menilai preferensi mereka untuk CTF sebagai 5, dan presentase yang sama menilai minat mereka dalam bidang ini sebagai 4 atau 5. Selain itu, 56 % mahasiswa menilai perolehan *skill* mereka dalam kernel eksploit sebagai 5. Dari kuesioner kedua mengenai penggunaan simulator eksploitasi kernel. Dari hasil kuesioner yang diberikan kepada 4 mahasiswa dapat disimpulkan bahwa simulator sangat membantu pada eksploitasi kernel dengan 4 mahasiswa memberikan skala 5. Peserta simulator yang dikembangkan efektif dan mudah digunakan dengan 3 mahasiswa menjawab 5. Namun dari sisi akurasi tidak mendapatkan hasil yang tinggi, terdapat 2 mahasiswa menjawab skala 3 dan 2 mahasiswa menjawab skala 4.

Selama 3 minggu pembelajaran, peserta memberikan respon positif terhadap pembelajaran kernel eksploitasi pada arsitektur ARM yang dilakukan. Beberapa peserta mengatakan bahwa *pembelajaran* tersebut “keren”, “menarik” dan “memberikan pengalaman (yang) seru”. Namun, ada beberapa peserta yang merasa bahwa pembelajaran terlalu kompleks dan mengharapkan terdapat materi dasar yang diajarkan terlebih dahulu sebelum masuk ke eksploitasi kernel. Beberapa peserta merasa bingung saat memelajari kernel *exploitation* pada pertemuan pertama, tetapi setelah terdapat video tutorial atau *writeup,* mereka merasa lebih mudah memahaminya. Ada juga beberapa peserta yang merasa bahwa materi yang diajarkan cukup menarik tetapi mereka mengalami kesulitan memahami materi yang *advance* dalam waktu yang padat. Secara keseluruhan, dalam waktu pembelajaran yang singkat dan dapat mencapai setidaknya pemahaman tingkat dasar mengenai eksploitasi kernel kepada mahasiswa yang belum pernah mempraktikan *binary exploitation* adalah luaran yang diharapkan dalam penelitian ini.

Dari sisi infrastruktur yang diimplementasikan maka dapat disimpulkan bahwa dengan kapasitas server yang sudah disebutkan sebelumnya dapat digunakan untuk praktik pembelajaran dengan catatan mengenai berapaa lama waktu yang dibutuhkan dan berapa jumlah peserta pembelajaran. Penulis berpendapat jika evaluasi dilakukan selama 180 menit, bagain pengiriman eksploit harusnya dilakukan kurang dari 20% dari total waktu evaluasi atau 36 menit. Pada data yang telah ditampilkan, infrastruktur ini akan sesuai jika jumlah peserta kurang dari 12 dikarenakan ketika terdapat 13

peserta atau lebih yang mengirimkan eksploit maka masing – masing peserta akan membutuhkan waktu lebih dari 39 menit.

V. KESIMPULAN

Berdasarkan penelitian yang sudah dilakukan dapat disimpulkan beberapa hal sebagai berikut: (1) Simulator eksploitasi kernel pada Raspberry Pi yang menggunakan arsitektur ARM 32 bit dapat diterapkan dengan menggunakan Docker, dan QEMU. Simulator yang dikembangkan terbukti sangat membantu peserta memelajari dan mempraktikan eksploitasi kernel dibandingkan dengan secara langsung menggunakan perangkat keras Raspberry Pi. Peserta juga menilai efektifitas dan kemudahan dalam menjalankan simulator eksploitasi kernel Raspberry Pi. (2) Edukasi eksploitasi kernel dengan arsitektur ARM dapat diimplementasikan dengan bentuk *capture-the-flag* dimana peserta diberikan petunjuk dan *step-by-step* pengerjaan yang jelas untuk menemukan *flag*. Dari hasil nilai *pretest* dan *posttest* peserta mendapatkan kenaikan nilai rata–rata sebesar 86%. Kuesioner dan umpan balik yang diberikan peserta mendapat respon positif seperti testimoni tentang peserta yang merasakan keseruan dalam pembelajaran maupun minat dan rasa penambahan skill peserta pasca pembelajaran.

DAFTAR PUSTAKA

[1] G. K. Adam, "DALI LED driver control system for lighting operations based on Raspberry Pi and kernel modules," *Electronics*, vol. 8, no. 9, p. 1021, 2019, doi: 10.3390/electronics8091021.

[2] J. Bermúdez-Ortega, E. Besada-Portas, J. A. López-Orozco, J. A. Bonache-Seco, and J. M. la Cruz, "Remote web-based control laboratory for mobile devices based on EJsS, Raspberry Pi and Node. js," *IFAC-PapersOnLine*, vol. 48, no. 29, pp. 158–163, 2015, doi: 10.1016/j.ifacol.2015.11.230.

[3] Z. Li, J. Wang, M. Sun, and J. C. S. Lui, "Securing the Device Drivers of Your Embedded Systems: Framework and Prototype," in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, 2019, pp. 1–10. doi: 10.1145/3339252.3340506.

[4] E. H. Boudjema, C. Faure, M. Sassolas, and L. Mokdad, "Detection of security vulnerabilities in C language applications," *Secur. Priv.*, vol. 1, no. 1, p. e8, 2018, doi: 10.1002/spy2.8.

[5] S. Nagarakatte, J. Zhao, M. M. K. Martin, and S. Zdancewic, "SoftBound: Highly Compatible and Complete Spatial Memory Safety for C," in *Proceedings of the 30th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2009, pp. 245–258. doi: 10.1145/1542476.1542504.

[6] H. Chen, Y. Mao, X. Wang, D. Zhou, N. Zeldovich, and M. F. Kaashoek, "Linux Kernel Vulnerabilities: State of The Art Defenses and Open Problems," in *Proceedings of the Second Asia-Pacific Workshop on Systems*, 2011, pp. 1–5. doi: 10.1145/2103799.2103805.

[7] M. Bhatt, I. Ahmed, and Z. Lin, "Using Virtual Machine Introspection for Operating Systems Security Education," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 2018, pp. 396–401. doi: 10.1145/3159450.3159606.

[8] J. Vykopal, V. Švábensk`y, and E.-C. Chang, "Benefits and Pitfalls of Using Capture the Flag Games in University Courses," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 2020, pp. 752–758. doi: 10.1145/3328778.3366893.

[9] K. Leune and S. J. Petrilli Jr, "Using Capture The Flag to Enhance The Effectiveness of Cybersecurity Education," in *Proceedings of the 18th Annual Conference on Information Technology Education*, 2017, pp. 47–52. doi: 10.1145/3125659.3125686.

[10] K. Chung, "Lowering the Barriers to Capture the Flag Administration and Participation," in *Proceedings of the USENIX Workshop on Advances in Security Education*, 2017, pp. 1–8.