ສຶກສາປະສິດທິພາບຂອງເຄື່ອງມືກວດສອບຊ່ອງໂຫວ່ເພື່ອຮັບປະກັນຄວາມປອດໄພການໃຊ້ງານເວັບແອັບພລິເຄຊັນຈາກໄພຂົ່ມຂູ່ທາງໄຊເບີ

**ກົງມະນີ ສົມສະໝອນ, ທາ ບຸນທັນ,ຄຳພັດ ບຸນນະດີ, ວິມົນທາ ຊຽວວົງພະຈັນ ແລະ ພອນໄຊ ວິລະກອນ*

ພາກວິຊາວິສະວະກຳຄອມພິວເຕີ ແລະ ເຕັກໂນໂລຊີຂໍ້ມູນ, ຄະນະວິສະວະກຳສາດ, ມະຫາວິທະຍາໄລແຫ່ງຊາດ

ບົດຄັດຫຍໍ້

ການພັດທະນາລະບົບ ເວັບແອັບພລິເຄຊັນ (Web Application) ໃຫ້ມີປະສິດທິພາບສູງນັ້ນ ນອກຈາກປະສິດທິພາບການເຮັດວຽກໄດ້ດີແລ້ວ ຄວາມປອດໄພກໍ່ແມ່ນສິ່ງສຳຄັນອີກປະການໜຶ່ງທີ່ຕ້ອງຫຼຸດຜ່ອນຊ່ອງໂຫວ່ໃຫ້ຫນ້ອຍທີ່ສຸດເທົ່າທີ່ເປັນໄປໄດ້ ຫຼື ບໍ່ມີເລີຍກໍ່ຍິ່ງດີ. ສະນັ້ນ, ໃນການຄົ້ນຄ້ວານີ້ຈຶ່ງໄດ້ສຶກສາປະສິດທິພາບໃນກວດສອບຫາຊ່ອງໂຫວ່ໃນ ເວັບ ແອັບພລິເຄຊັນຂອງ 2 ເຄື່ອງມື ຄື: ແຊັບ (ZAP(Zed Attack Proxy)) ແລະ ເວກາ (Vega) ເພາະວ່າເຄື່ອງມືເຫຼົ່ານີ້ ສະໜັບສະໜຸນການນຳໃຊ້ໃນຫຼາຍລະບົບປະຕິບັດການທັ້ງ ວິນໂດ (windows OS), ແມັກ ໂອເອັສ (macOS) ແລະ ລີນຸກ (Linux), ສາມາດສະແກນຊອກຫາຊ່ອງໂຫວ່ໄດ້ເກືອບທຸກ ເວັບ ແອັບພລິເຄຊັນ ບໍ່ວ່າຈະພັດທະນາມາຈາກໂປຣແກຣມພາສາ (Programing Language) ຫຍັງກໍ່ຕາມ ແລະ ເປັນເຄື່ອງມືເປັນທີ່ນິຍົມໃຫ້ໃຊ້ລ້າ (Free). ຈຸດປະສົງໃນການຄົ້ນຄ້ວາແມ່ນເພື່ອ 1.) ສຶກສາປະສິດທິພາບການຊອກຄົ້ນຫາຈຳນວນຊ່ອງໂຫວ່, 2.) ສຶກສາປະສິດທິພາບຄວາມໄວຂອງການຊອກຫາ ແລະ 3.) ສຶກສາປະສິດທິພາບການແກ້ໄຂຂໍ້ບົກຜ່ອງທີ່ພົບຂອງເຄື່ອງມື, ໂດຍວິທີການຄົ້ນຄ້ວາທົດລອງແມ່ນຕິດຕັ້ງເຄື່ອງມື 2 ເຄື່ອງມືລົງໃສ່ຄອມພິວເຕີ ວິນໂດ 11 ໂປຣເຟສຊັນນອລ ເວີຊັນ (Windows 11 Professional version) ຈາກນັ້ນເຮັດການສະແກນຫາຊ່ອງໂຫວ່ 10 ຄັ້ງຕໍ່ເວັບ, ໃນ 5 ເວັບ ແອັບພລິເຄຊັນ ທີ່ຖືກພັດທະນາຈາກຫຼາກຫຼາຍພາສາທີ່ແຕກຕ່າງກັນ. ຜົນການທົດລອງແມ່ນໄດ້ບັນທຶກແຍກລົງໃສ່ໃນຕາຕະລາງເອັກເຊວຟາຍ (Excel file), ຫຼັງຈາກນັ້ນ , ເຮັດການຄິດໄລ່ຫາຄ່າສະເລ່ຍລວມຂອງທັງ 3 ຫົວຂໍ້ຂອງແຕ່ລະເຄື່ອງມື, ແລ້ວນຳຜົນສະເລ່ຍລວມມາ ວິເຄາະ ແລະ ສົມທຽບກັນ. ຈາກຜົນຄຳນວນສາມາດສະຫຼຸບໄດ້ວ່າ: ເຄື່ອງມື ແຊັບ ແມ່ນເຮັດໄດ້ກວ່າທາງດ້ານ ປະສິດທິພາບໃນການກວດຈັບຫາຈຳນວນຊ່ອງໂຫວ່ (ດ້ວຍຜົນຄະແນນສະເລ່ຍ (ZAP) 60.3 ຕໍ່ 11.3 (Vega)) ແລະ ຄຳແນະນຳໃນການແກ້ໄຂບັນຫາ (ດ້ວຍຜົນຄະແນນສະເລ່ຍ (ZAP) 123 ຕ່ຳ 33 (Vega)). ສ່ວນເຄື່ອງມື ເວກາ ກໍ່ສາມາດສະແກນຫາຊ່ອງໂຫວ່ໄດ້ໄວກວ່າ (ດ້ວຍຜົນເວລາສະເລ່ຍ (ZAP) 41,364.01 Sec ຕໍ່ 125.80 Sec (Vega)). ຈາກຜົນທົດລອງທີ່ໄດ້ຮັບສາມາດຊ່ວຍນັກພັດທະນາໃນການເລືອກເຄື່ອງມືຊ່ວຍໃນການຊອກຫາ ແລະ ຫຼຸດຜ່ອນຈຳນວນຊ່ອງໂຫວ່ໃນການພັດທະນາ *ເວັບ* ແອັບພລິເຄຊັນ ໄດ້ເປັນຈຳນວນຫຼາຍພໍສົມຄວນ, ພ້ອມກັນນັ້ນ ຍັງຊ່ວຍໃຫ້ ນັກພັດທະນາ ສາມາດແກ້ບັນຫາໄດ້ຖືກຈຸດ ແລະ ວ່ອງໄວຕໍ່ເຫດການອີກດ້ວຍ.

ຄຳສັບສຳຄັນ: *ເວັບແອັບພລິເຄຊັນ, ZAP, Vega, ເຄື່ອງມືຊອກຫາຊ່ອງໂຫວ່, ຄວາມປອດໄພ.*

**ຕິດຕໍ່ພົວພັນ: ກົງມະນີ ສົມສະໝອນ; ໂທ: 020 57656565; ອີເມວ: kongmanyssm@gmail.com*

A Study on the Effectiveness of Vulnerability Tools in Protecting Web Applications from Cyber Threats

**Kongmany SOMSAMONE, Khampheth BOUNNADY, Tha BOUNTHANH, Vimontha KHIEOVONGPACHANH and Phonexay VILAKONE*

Computer Engineering and IT, Faculty of Engineering, National University of Laos (NUOL)

Abstract

In developing a highly efficient web application system, security is just as important as performance, as it must minimize or ideally eliminate vulnerabilities. Therefore, this research examines the effectiveness of two tools—ZAP (Zed Attack Proxy) and Vega—in detecting vulnerabilities in web applications. Both tools are free, widely used, and compatible with multiple operating systems, including Windows, macOS, and Linux. They can scan for vulnerabilities in almost any web application, regardless of the programming language used. The objectives of this research are to: 1) study the effectiveness of detecting vulnerabilities; 2) evaluate the effectiveness of scanning speed, and 3) examine the effectiveness of fixing defects in web applications. The experimental method involved installing both tools on a Windows 11 Professional computer and scanning for vulnerabilities 10 times per website across five web applications developed in different programming languages. The results were recorded in Excel, and the average values for all three objectives were calculated for each tool. Finally, the averages were analyzed and compared. The findings show that ZAP is more effective than Vega at detecting vulnerabilities (average score: 60.3 for ZAP versus 11.3 for Vega) and at providing recommendations for resolving issues (average score: 123 for ZAP versus 33 for Vega). However, Vega outperformed ZAP in scanning speed (average time: 125.80 seconds for Vega versus 41,364.01 seconds for ZAP). These experimental results can help developers select the most appropriate tools for reducing vulnerabilities in web application development and for resolving issues quickly and accurately.

Keywords: *Web applications, ZAP, Vega, Vulnerability Scanning tools, Security*

**Correspondence: Kongmany Somsamone*; *Tel: 020 57656565; Email: kongmanyssm@gmail.com*

1. ພາກສະເໜີ

1.1 ຄວາມເປັນມາ ແລະ ຄວາມສຳຄັນຂອງບັນຫາ

ໃນຍຸກສະໄໝທີ່ເຕັກໂນໂລຊີມີການຂະຫຍາຍຕົວຢ່າງວ່ອງໄວ ແລະ ມີການພັດທະນາແບບກ້າວກະໂດດ ເຊິ່ງມີບົດບາດສຳຄັນໃນຊີວິດປະຈຳວັນ, ເວັບ ແອັບພລິເຄຊັ່ນ (Web applications) ໄດ້ກາຍເປັນສ່ວນໜຶ່ງໃນວຽກງານປະຈຳວັນຂອງຫຼາຍສາຂາອາຊີບ,ໜ່ວຍງານ ແລະ ອົງການຈັດຕັ້ງຕ່າງໆທີ່ຈຳເປັນຕ້ອງມີ ແລະ ຈຳເປັນຕ້ອງໄດ້ໝຸນໃຊ້ເຂົ້າໃນການເພີ່ມປະສິດທິພາບ ຂອງວຽກງານໃຫ້ໄດ້ຮັບຜົນດີ (Khiaovongphachanh et al. 2024). ເຖິງຢ່າງໃດກໍ່ຕາມ, ຫຼຽນແມ່ນມີ 2 ດ້ານສະເໝີ ໃນທາງກົງກັນຂ້າມ, ມັນກໍ່ກາຍເປັນເປົ້າໝາຍໃຫຍ່ອັນໜຶ່ງໃນໂລກໄຊເບີທີ່ຜູ້ບໍ່ຫວັງດີ ຫຼື ແຮັກເກີ (Hacker) ທີ່ສອຍໃຊ້ຜົນປະໂຫຍດແບບບໍ່ຖືກຕ້ອງ ແລະ ໂຈມຕີ ລະບົບການເຮັດວຽກງານ ທາງອິນເຕີເນັດ (Sengudone, 2017; Hoffman, 2020; Mongkolsawat, 2016). ອີງຕາມຜົນສະຫຼຸບທຸກໆ 4ປີ ຂອງ 10 ອັນດັບໄພຄຸກຄາມຕົ້ນຕໍ ຕໍ່ ເວັບ ແອັບພລິເຄຊັ່ນ ໃນປີ 2021 ຈາກອົງກອນທີ່ບໍ່ຫວັງຜົນກຳໄລ ໂອວາສພ OWASP (The Open Worldwide Application Security Project) ທີ່ເຮັດວຽກເພື່ອປັບປຸງຄວາມປອດໄພຂອງຊອບແວ ໄດ້ຊີ້ໃຫ້ເຫັນວ່າ ຊ່ອງໂຫວ່ແບບການລະເມີດການຄວບຄຸມການເຂົ້າເຖິງ (Broken Access Control) ແມ່ນຍັບຂຶ້ນຈາກອັນດັບທີ່ 5 ໃນປີ 2017 ມາເປັນອັນດັບທີ່ 1 ໃນປີ 2021 ຍ້ອນສາຍເຫດເກີດມາຈາກການອະນຸຍາດສິດຂອງຜູ້ໃຊ້ຫຼາຍກວ່າລະດັບທີ່ຜູ້ໃຊ້ຄວນຈະສາມາດເຮັດໄດ້ເຊັ່ນວ່າ: ບໍ່ໄດ້ເຂົ້າສູ່ລະບົບແບບລັອກອິນ (Login) ແຕ່ພິມທີ່ຢູ່ ຫຼື ຢູອາຣແອລ (URL) ໂດຍກົງໃສ່ທີ່ເປັນພາກສ່ວນຂອງຜູ້ດູແລລະບົບ (Admin) ເພື່ອເບິ່ງຂໍ້ມູນຂອງເວັບໄຊທ໌ທັງຫມົດ, ຫຼື ແກ້ໄຂບາງ ລະຫັດ (ID) ຂອງຜູ້ໃຊ້ ຫຼື ລະຫັດ ທີ່ເປັນເອກະລັກ (Unique id) ໂດຍເຂົ້າເຖິງຂໍ້ມູນທີ່ບໍ່ໄດ້ເປັນເຈົ້າຂອງໄດ້ແບບງ່າຍໆ, ນອກເໜືອຈາກນີ້ ເວັບ ແອັບພລິເຄຊັ່ນ ຈຳນວນຫຼາຍຍັງມີການຕັ້ງຄ່າ CORS (Cross-Origin Resource Sharing) ທີ່ບໍ່ຖືກຕ້ອງ ທີ່ອະນຸຍາດໃຫ້ຄົນພາຍນອກເຂົ້າເຖິງ API

ຈາກເວັບໄຊທ໌ພາຍນອກໄດ້, ເຊິ່ງມັນເປັນຊ່ອງໂຫວ່ທີ່ໃຫຍ່ ແລະ ອັນຕະລາຍທີ່ຮຸນແຮງຫຼາຍ. ໃນຂະນະດຽວກັນຊ່ອງໂຫວ່ແບບສັກຢາ (Injection) ເຊິ່ງເປັນອັນດັບທີ່ 1 ມາຍາວນານຈົນກະທັ້ງປີ 2017 ເຖິງຈະມີຈຳນວນຫຼຸດລົງກໍ່ຍັງຕິດຢູ່ອັນດັບທີ່ 3 ຂອງປີ 2021, ເຊິ່ງ ຊ່ອງໂຫວ່ທີ່ຖືກໂຈມຕີໄດ້ແບ່ງອອກເປັນ 2 ປະເພດຕົ້ນຕໍຄື: Injection ຜ່ານລະຫັດທົ່ວໄປເຊັ່ນ: ຊ່ອງໂຫວ່ທີ່ນິຍົມໃນການຖືກ ໂຈມຕີຫຼາຍທີ່ສຸດແມ່ນ SQL Injection ເກີດມາຈາກໂປຣແກຣມ ຍອມຮັບການປ້ອນຂໍ້ມູນ ແບບຕົວໜັງສື (String) ຈາກຜູ້ໃຊ້ເພື່ອ ເຮັດວຽກຢູ່ໃນລະບົບ, ແຕ່ຜູ້ໃຊ້ສົ່ງ string ທີ່ເປັນຄຳສັ່ງທີ່ເປັນ ອັນຕະລາຍທີ່ເຮັດໃຫ້ ຕົວແລ່ນໂປຣແກຣມ (Compiler) ເຂົ້າໃຈຜິດ ວ່າມັນເປັນຄຳສັ່ງທີ່ຜູ້ສ້າງໄດ້ກຳນົດໄວ້. ຫຼັງຈາກນັ້ນ, ຜູ້ໃຊ້ກໍ່ສາມາດ ເຂົ້າເຖິງຂໍ້ມູນທີ່ບໍ່ຄວນຈະເຂົ້າເຖິງໄດ້ ຫຼື ແກ້ໄຂຂໍ້ມູນທີ່ບໍ່ຄວນຈະຖືກ ແກ້ໄຂ ຫຼື ຮ້າຍແຮງໄປກວ່ານັ້ນເຖິງຂັ້ນສາມາດລຶບຕາຕະລາງໃນຖານ ຂໍ້ມູນໄດ້ເລີຍ ແລະ ແບບທີ່ 2 ກໍ່ຄື Injection ຜ່ານ XSS (Cross Site Scripting) ເປັນວິທີການລັບລອບໃສ່ຄຳສັ່ງລະຫັດລັບໄປ ກັບໄຟລ໌ທີ່ເຮັດວຽກ ໃນລັກສະນະແບບປົກະຕິ ພຽງແຕ່ເມື່ອເຖິງ ຈຸດໝາຍປາຍທາງການເຮັດວຽກຈະແຕກຕ່າງອອກໄປໂດຍການ ໃສ່ຄຳສັ່ງ JavaScript ເພື່ອລັກລອບເອົາ Cookies ໄປຢືນຢັນຕົວ ຕົນຂອງຜູ້ໃຊ້ອື່ນ ແລະ ປອມຕົວເປັນຜູ້ໃຊ້ອື່ນເພື່ອດຳເນີນການຢ່າງໃດ ຢ່າງໜຶ່ງແທນ ຫຼື ທັ້ງໝົດ. ນີ້ມັນສະແດງໃຫ້ວ່າໄພຮ້າຍໃນໂລກໄຊເບີ ແມ່ນມີຄວາມຮຸນແຮງ, ມີຫຼາກຫຼາຍແບບ, ອັນຕະລາຍ, ມີການ ປ່ຽນແປງຕາມຍຸກສະໄໝ ແລະ ການລະເວລາ ຢ່າງບໍ່ຢຸດຢັ້ງ ຕະຫຼອດ ເວລາ (The Open Worldwide Application Security Project, 2021; Common Weakness Enumeration Organization, 2024; Morimoto, 2022; Adapt, Adept, and Adopt, 2022).

ສະນັ້ນ ຄວາມສຳຄັນອັນດັບໜຶ່ງທີ່ນັກພັດທະນາຄວນຄຳນຶງ ເຖິງ ແລະ ປະຕິເສດບໍ່ໄດ້ ກໍ່ຄືຄວາມປອດໄພທີ່ແຂງແກ່ນຂອງລະບົບ ຕໍ່ກັບການໂຈມຕີທາງໄຊເບີ, ດ້ວຍຄຳນyຍາມທີ່ວ່າຂໍ້ມູນທີ່ດີນັ້ນຈະ ຮັກສາໄວ້ເຊິ່ງຫຼັກການຂອງ ຊີໄອເອ (CIA) ຄື: ການຮັກສາຄວາມລັບ (Confidentiality), ຄວາມຖືກຕ້ອງ (Integrity) ແລະ ຄວາມ ພ້ອມໃຊ້ງານ (Availability) ຂອງຂໍ້ມູນໃນລະບົບ. Douangphachanh (2018) ໄດ້ຄົ້ນຄວ້າປະສິດທິພາບຂອງເຄື່ອງມື ເພື່ອເພີ່ມປະສິດທິພາບຄວາມປອດໄພໃນການພັດທະນາ ເວັບແອັບພ ລິເຄຊັນ ເພື່ອເຮັດໃຫ້ນັກພັດທະນາມີຄວາມຮູ້ ແລະ ນຳໃຊ້ເຄື່ອງມື ທີ່ຈຳເປັນ ເພື່ອເສີມສ້າງຄວາມປອດໄພຂອງ ເວັບແອັບພລິເຄຊັນ ໃຫ້ ມີຄວາມປອດໄພສູງຕໍ່ກັບການລະເມີດ ຫລື ການໂຈມຕີລະບົບທີ່ອາດ ຈະເກີດຂຶ້ນ, ເຊິ່ງເຄື່ອງມືທີ່ໃຊ້ໃນການວິເຄາະຫາຊ່ອງໂຫວ່ ໃນ ເວັບແອັບພລິກເຄຊັນ ແມ່ນມີຫຼາຍຊະນິດເຊິ່ງແບ່ງອອກເປັນ 2 ປະເພດຄື: ແບບໃຊ້ເປັນທາງການຄ້າທີ່ຕ້ອງໃຫ້ຊື້ (Commercial license) ແລະ ແບບ ໂອເພັນຊອສ (Open-Source) ທີ່ໃຫ້ໃຊ້ລ້ຳ (Free) ທີ່ມີຫລາຍແບບໃຫ້ເລືອກໃຊ້ຕາມຄວາມເໝາະສົມ ແລະ ກົງກັບຈຸດປະສົງຂອງວຽກງານຄວາມປອດໄພນັ້ນໆເຊັ່ນ: ເນັ້ນໃສ່ການ ຫາຊ່ອງໂຫວ່ທາງດ້ານເຄືອຂ່າຍ (Network Vulnerability Scanning), ການຫາຊ່ອງໂຫວ່ໃນ ເວັບ ແອັບພລິເຄຊັນ (Web application vulnerabilities), ການທົດສອບຄວາມປອດໄພຂອງ ເຄືອຂ່າຍ (Network discovery and security auditing), ການ ຫາຊ່ອງໂຫວ່ສ່ວນບຸກຄົນ ແລະ ຂະໜາດນ້ອຍ (Personal and small-scale vulnerability scanning) ແລະ ອື່ນໆ (Anas et al. 2023; Stuttard et al. 2008). ແຕ່ວ່າໃນການສຶກສາຄັ້ງນີ້ແມ່ນ ສຸມ ໃສ່ການຫາຊ່ອງໂຫວ່ໃນ ເວັບ ແອັບພລິເຄຊັນ (Web application vulnerabilities) ທີ່ໃຊ້ເຄື່ອງມື ແຊັບ ແລະ ເວກາ, ເຊິ່ງທັ້ງສອງ ເຄື່ອງມືແມ່ນສາມາດໃຊ້ໃນລະບົບປະຕິບັດການວິນໂດ (Windows OS) ແລະ ສາມາດສະແກນຫາຊ່ອງໂຫວ່ໄດ້ເກືອບທຸກ ເວັບແອັບພ ລິເຄຊັນ ບໍ່ວ່າຈະພັດທະນາມາຈາກພາສາຫຍັງກໍ່ຕາມ, ພ້ອມກັບຕອບ ສະໜອງຄຸນສົມບັດຂັ້ນສູງສຳລັບຜູ້ທົດສອບການໂຈມຕີທີ່ຊັບຊ້ອນ, ໃຊ້ງ່າຍ ແລະ ເປັນເຄື່ອງມືແບບ ໂອເພັນຊອສ (Open-Source) ໃຫ້ ໃຊ້ລ້ຳ (Matti, 2021; BorntoDev, 2022; Abdulghaffar et al. 2023; Pentest Tools, 2024). ໂດຍທີ່ເຄື່ອງມື ແຊັບ (ZAP - Zed Attack Proxy) ແມ່ນເປັນເຄື່ອງມືສຳລັບນັກພັດທະນາ ເວັບ ແອັບພ ລິເຄຊັນ, ຜູ້ດູແລລະບົບ, ແລະ ນັກເຈາະລະບົບ ທີ່ຕ້ອງການສະແກນ ແລະ ກວດສອບຊ່ອງໂຫວ່ຂອງ ເວັບ ແອັບພລິເຄຊັນ, ເຊິ່ງຖືກ ພັດທະນາດ້ວຍພາສາ Java ສິ່ງຜົນໃຫ້ ສາມາດເຮັດວຽກໄດ້ໃນທຸກ ລະບົບປະຕິບັດການ, ລວມທັ້ງ Raspberry Pi. ແລະ ເຄື່ອງມື ເວ ກາ (Vega) ເປັນເຄື່ອງມືທົດສອບຄວາມປອດໄພຂອງ ເວັບ ແອັບພ ລິເຄຊັນ ແລະ ສາມາດຊ່ວຍຊອກຫາ ຫຼື ສະແກນຫາຊ່ອງໂຫວ່ຢູ່ໃນ ເວັບໄຊທ໌ທີ່ເຊື່ອມຕໍ່ກັບຖານຂໍ້ມູນ ແລະ Cross-Site Scripting (XSS) ທີ່ສາມາດເປີດເຜີຍຂໍ້ມູນລະອຽດອ່ອນ ແລະ ຊ່ອງໂຫວ່ອື່ນໆ ໂດຍບໍ່ຕັ້ງໃຈ ແລະ ຍັງສາມາດສະແກນການໂຕ້ຕອບລູກຂ່າຍ

(Client) - ເຊີບເວີ (Server) ສຳລັບເວັບໄຊທ໌ HTTP. ທັງ 2 ເຄື່ອງມືລ້ວນແຕ່ເປັນເຄື່ອງມືທີ່ສຳຄັນຫຼາຍທີ່ນິຍົມໃຊ້ກັນຢ່າງກວ້າງຂວາງທີ່ຊ່ວຍໃຫ້ນັກພັດທະນາ ແລະ ຜູ້ຊ່ຽວຊານດ້ານຄວາມປອດໄພກວດພົບ ແລະ ແກ້ໄຂຊ່ອງໂຫວ່ໃນ ແອັບ ພລິເຄຊັນເວັບ ໃນການພັດທະນາ (Program OWASP ZAP, 2016; Kritsada et al. 2015). ເຖິງຢ່າງໃດກໍ່ຕາມ, ນອກຈາກ 2 ເຄື່ອງມືນີ້ແລ້ວກໍ່ຍັງມີອີກຫຼາຍເຄື່ອງມືເຊັ່ນ: ອາປິຕິ (Wapiti) ທີ່ເຮັດການສະແກນຊອກຫາຊ່ອງໂຫວ່ແບບ ເອສຄິວແອວ ອິນເຈັກຊັນ (SQL injection(SQLi)) ແລະ ຄຣອສໄຊສະຄຣິບຕິ້ງ (cross-site scripting (XSS)) ແຕ່ບໍ່ແທດເໝາະກັບການຄົ້ນຄວ້າຄັ້ງນີ້ເນື່ອງຈາກມັນໃຊ້ລະບົບປະຕິບັດການ ລີນຸກສ (Linux) (Matti, 2021). ອາຣັກນີ (Arachni) ກໍ່ເປັນອີກເຄື່ອງມືດ້ານຄວາມປອດໄພໃນການຊອກຫາຊ່ອງໂຫວ່ໃນ ເວັບ ແອັບພລິເຄຊັນ ທີ່ຖືກພັດທະນາດ້ວຍພາສາ ຣູບີ (Ruby) ແຕ່ມັນບໍ່ສະໜັບສະໜຸນລະບົບປະຕິບັດການອີນໂດ ແລະ ໃຫ້ໃຊ້ລ້າສະເພາະບາງລາຍການຄຸນສົມບັດຂອງເຄື່ອງມືເທົ່ານັ້ນ (Abdulghaffar et al. 2023).

1.2. ຄຳຖາມຂອງການຄົ້ນຄ້ວາ

1. ການຄົ້ນຫາຊ່ອງໂຫວ່ທີ່ຈະເປັນບ່ອນຈູດອ່ອນສຳລັບການໂຈມຕີຂອງ ເວັບ ແອັບພລິເຄຊັນ ດ້ວຍເຄື່ອງມື ແຊັບ (ZAP) ແລະ ເວກາ (Vega) ໂຕໃດດີກວ່າກັນ?
2. ປະສິດທິພາບໃນການກວດຈັບຫາຊ່ອງໂຫວ່ ແຊັບ (ZAP) ແລະ ເວກາ (Vega) ໂຕໃດເຮັດໄດ້ໄວກວ່າກັນ?
3. ແນວທາງໃນການແກ້ໄຂບັນຫາທີ່ພົບຂອງແຕ່ລະເຄື່ອງມື ແຊັບ (ZAP) ແລະ ເວກາ (Vega) ໂຕໃດດີກວ່າກັນ?

1.3. ຈຸດປະສົງຂອງການຄົ້ນຄ້ວາ

ບົດຄົ້ນຄວ້ານີ້ແມ່ນມີຈຸດປະສົງໃນການສຶກສາປະສິດທິພາບຂອງເຄື່ອງມືໃນ 3 ດ້ານຄື:

1. ເພື່ອສຶກສາປະສິດທິພາບຂອງເຄື່ອງມືໃນການກວດຈັບ ຫຼື ຊອກຫາຈຳນວນຊ່ອງໂຫວ່ ໃນ ເວັບແອັບພລິເຄຊັນ
2. ວັດແທກປະສິດທິພາບຄວາມໄວໃນການກວດຈັບຊ່ອງໂຫວ່ ໃນ ເວັບ ແອັບພລິເຄຊັນ
3. ປະເມີນປະສິດທິພາບທາງດ້ານວິທີການແກ້ໄຂບັນຫາທີ່ພົບ ໃນ ເວັບ ແອັບພລິເຄຊັນ

ທັງໝົດນີ້ແມ່ນເພື່ອປະກອບສ່ວນເຂົ້າໃນການຕັດສິນໃຈໃນການເລືອກໃຊ້ເຄື່ອງມືຊ່ວຍຊອກຫາຊ່ອງໂຫວ່ໃຫ້ຖືກຕ້ອງ ແລະ ເໝາະສົມກັບສະພາບເຫດການນັ້ນໆ ໃນຂະນະຂອງການພັດທະນາ ລະບົບ ເວັບ ແອັບພລິເຄຊັນ.

2. ບົດຄົ້ນຄວ້າທີ່ກ່ຽວຂ້ອງ

ໃນພາກສ່ວນນີ້ເປັນການສັງລວມການທົບທວນນະວັດຕະກຳກ່ຽວກັບວັດຖຸປະສົງຂອງການຄົ້ນຄ້ວາໂດຍສະເພາະການປະເມີນຄວາມ ຕ້ອງການໃນການພັດທະນາລະບົບ ເວັບ ແອັບພລິເຄຊັນ ໃຫ້ມີຄວາມປອດໄພໃຫ້ສູງທີ່ສຸດ ລາຍລະອຽດດັ່ງລຸ່ມນີ້

2.1 ເພີ່ມຄວາມປອດໄພໃນ ເວັບ ແອັບພລິເຄຊັນ ດ້ວຍເຄື່ອງສະແກນຊ່ອງໂຫວ່ແບບຫຼາກຫຼາຍ

ການທີ່ຈະຮູ້ໄດ້ວ່າ ເວັບ ແອັບພລິເຄຊັນ ມີຕວາມປອດໄພສູງພຽງໃດນັ້ນ ວິທີທີ່ງ່າຍ ແລະ ບໍ່ເປືອງແຮງແມ່ນຈຳເປັນຕ້ອງໃຊ້ເຕື່ອງມືສະແກນຊ່ອງໂຫວ່ແບບຫຼາກຫຼາຍ ແລະ ເຮັດວຽກແບບອັດຕະໂນມັດ, ສະນັ້ນ Abdulghaffar et al. (2023) ໄດ້ເຮັດການວິໄຈ ການປັບປຸງຄວາມປອດໄພຂອງ ເວັບ ແອັບພລິເຄຊັນ ຜ່ານການທົດສອບການເຈາະລະບົບແບບອັດຕະໂນມັດ ດ້ວຍເຄື່ອງສະແກນຊ່ອງໂຫວ່ແບບຫຼາກຫຼາຍ (Enhancing Web Application Security through Automated Penetration Testing with Multiple Vulnerability Scanners) ໂດຍການອອກແບບເພື່ອເຮັດໃຫ້ເປັນອັດຕະໂນມັດໃນການດຳເນີນງານຂອງຫຼາຍ ຕົວສະແກນຫາຊ່ອງໂຫວ່ໃນ ເວັບ ແອັບພິເຄຊັນ (Web Application Vulnerability Scanners (WAVS)) ພາຍໃຕ້ ແພລັດຟອມດຽວ, ເຊິ່ງໄດ້ນຳເອົາຄຸນສົມບັດຂອງ 2 ເຄື່ອງມືສະແກນຄື: ອາຣັກນີ (Arachni) ແລະ ແຊັບ (ZAP) ມາຮວມເຂົ້າກັນ. ຜົນການທົດສອບສະແດງໃຫ້ເຫັນວ່າ ໃຫ້ເຫັນຄວາມຖືກຕ້ອງຫຼາຍກວ່າເກົ່າເມື່ອປຽບທຽບກັບການສະແກນແບບດ່ຽວຂອງແຕ່ລະເຄື່ອງມື.

2.2 ການທົດສອບການໂຈມຕີແບບເຈາະຈີ້ງລະບົບກ່ອນນຳໃຊ້ຕົວຈິງ

ທຸກໆຄັ້ງທີ່ມີການພັດທະນາລະບົບ ເວັບແອັບພິເຄຊັນ ຈຳເປັນຕ້ອງມີການທົດສອບທຸກໆຂັ້ນຕອນ ເລີ້ມຕັ້ງແຕ່ຂັ້ນການພັດທະນາ (Development State) ຫາ ຂັ້ນຕອນຜູ້ໃຊ້ທົດສອບ (UAT: User Acceptance Test) ແລະ ຂັ້ນຕອນການໃຊ້ງານຈິງ (Production) ເພື່ອຮັບປະກັນຄວາມຖືກຕ້ອງ ແລະ ປອດໄພຂອງລະບົບ, ເຊິ່ງສອດຕ່ອງກັບ Pasomsup, C. (2020) ໄດ້ເຮັດ

ການທົດສອບທີ່ໄດ້ໃຊ້ເຄື່ອງມື ZAP ເປັນເຄື່ອງມືການທົດສອບການປ້ອງກັນການໂຈມຕີທາງໄຊເບີ, ໂດຍເຮັດການທົດສອບການໂຈມຕີແບບມຸ່ງເນັ້ນໃສ່ຊ່ອງໂຫວ່ຊະນິດຄອສໄຊສະຄຣິບຕິ້ງ (Cross-Site Scripting (XSS)) ເຊິ່ງເປັນການຝັງສະຄຣິບ (Script) ເພື່ອໂຈມຕີຈາກຊ່ອງໂຫວ່ທີ່ມີເທິງໜ້າເວັບໄຊ ແລະ ການໂຈມຕີແບບເອສຄິວແອວ ອິນເຈັກຊັນ (SQL Injection (SQLi)) ລວມເຖິງການໂຈມຕີແບບບັຟເຟີໂອເວີໂຟຣ (Buffer Overflow) ຜົນການທົດສອບທີ່ໄດ້ຄື: ສາມາດປ້ອງກັນການໂຈມຕີໄດ້ເພາະວ່າຄ່າຮ້ອງຂໍ (Request value) ຂອງແຕ່ລະການໂຈມຕີຈະຂຶ້ນ ທີ່ມີລາຍລະອຽດການສະແດງຜົນຂອງໜ້າເວັບ (Webpage) ນັ້ນວ່າມີອົງປະກອບ (Element) ທີ່ກົງກັບການສະແກນ ເຊິ່ງຖືກກວດສອບຕາມທີ່ໂປຣແກຣມກຳນົດໄວ້.

2.3 ການສັ້ນຫາເຄື່ອງມືທີ່ເໝາະສົມໃນການຊອກຫາຊ່ອງໂຫວ່

ແນ່ນອນວ່າການເລືອກເຄື່ອງມືທີ່ເໝາະສົມໃນການໃຊ້ນັ້ນແມ່ນມີຄວາມສຳຄັນຫຼາຍທາງດ້ານຄຸນະພາບທີ່ໃຫ້ປະສິດທິພາບສູງ ແຕ່ບາງຄັ້ງມັນກໍ່ຈຳກັດທາງດ້ານງົບປະມານ. Matti (2021) ໄດ້ເຮັດການຄົ້ນຄ້ວາການປະເມີນເຄື່ອງສະແກນຊ່ອງໂຫວ່ຂອງເວັບແຫຼ່ງໂດຍການຈັດຫາເຄື່ອງສະແກນຫາຊ່ອງໂຫວ່ຂອງ ເວັບ ແອັບພິເຄຊັນ ແບບແຫຼ່ງເປີດ (Open-source) ມາ 7 ເຄື່ອງມື, ເຊິ່ງລ້ວນແລ້ວແຕ່ຖືກພັດທະນາຈາກຫຼາກຫຼາຍພາສາເຊັ່ນ: ພາສາ Java, Perl, Python, C ແລະ Ruby. ນາມາວິເຄາະ ແລະ ຄັດເລືອກ ເຄື່ອງມືທີ່ເໝາະສົມທີ່ສຸດໃນການທົດລອງ, ໂດຍເງື່ອນໃນການຄັດເລືອກຄື:

- ສາມາດດາວໂຫຼດແບບລ້ຽງໄດ້ງ່າຍ
- ມີຈຸດເດັ່ນສາມາດຊອກຫາ ຊ່ອງໂຫວ່ແບບ ເອສຄິວແອວ ອິນເຈັກຊັນ (SQL injection(SQLi)) ແລະ ຄອສໄຊສະຄຣິບຕິ້ງ (cross-site scripting (XSS))
- ສະໜັບສະໜຸນແບບຫຼາກຫຼາຍລະບົບປະຕິບັດການ
- ໃຊ້ງ່າຍ ແລະ ເປັນທີ່ຮູ້ຈັກໂດຍທົ່ວໄປ
- ສາມາດຕິດຕໍ່ ແລະ ຂໍຄຳປຶກສາຈາກເຈົ້າຂອງເຄື່ອງມືໄດ້

ໃນທີ່ສຸດກໍ່ເລືອກໄດ້ 3 ເຄື່ອງມືຄື: ແຊັບ (ZAP), ເວກາ (Vega) ແລະ ວາປິຕິ (Wapiti) ທີ່ຈະໃຊ້ໃນການສະແກນຊອກຫາຊ່ອງໂຫວ່ແບບ ເອສຄິວແອວ ອິນເຈັກຊັນ (SQL injection(SQLi)) ແລະ ຄອສໄຊສະຄຣິບຕິ້ງ (cross-site scripting (XSS)). ຜົນສະຫຼຸບສຸດທ້າຍໄດ້ສະແດງໃຫ້ເຫັນວິທີການ ແລະ ຄວາມສາມາດເດັ່ນທີ່ແຕກຕ່າງກັນຂອງແຕ່ລະເຄື່ອງມືສະແກນໃດໆແມ່ນສາມາດນຳໃຊ້ ແລະ ຂໍ້ຈຳກັດຂອງໃຜມັນ.

2.4 ການຊອກຫາຊ່ອງໂຫວ່ໃນ ເວັບ ແອັບພິເຄຊັນ ທີ່ໃຊ້ງານຈິງແລ້ວ

ບາງຄັ້ງການຮັບມອບລະບົບຈາກທີມພັດທະນາມາແລ້ວບໍ່ແນ່ໃຈວ່າທີມພັດທະນາໄດ້ທົດສອບແບບຮອບຄອບໃຫ້ບໍ່ ແລະ ລະບົບມີຄວາມປອດໄພບໍ່, ເຊິ່ງຜູ້ຮັບ ຫຼື ເຈົ້າຂອງໂຄງການເອງກໍ່ສາມາດ ທົດສອບເອງໂດຍໃຊ້ເຄື່ອງມືແບບແດັສທ (DAST: Dynamic Application Security Testing) ດັ່ງທີ່ Morimoto (2022) ໄດ້ເຮັດການທົດລອງກ່ຽວກັບການເຮັດ ວີເອ (VA: Vulnerability Assessment) ດ້ວຍເຄື່ອງມື ແຊັບ (ZAP) ເຮັດການສະແກນອັດຕະໂນມັດຊອກຫາຊ່ອງໂຫວ່ຈຸດອ່ອນເບື້ອງຕົ້ນ, ເຊິ່ງເນັ້ນໃສ່ການການໂປຣແກຣມເຮັດວຽກແທນຄົນຜົນຂອງການເຮັດ ວີເອ (VA) ແມ່ນໄດ້ເນັ້ນໃສ່ 4 ລາຍການຫຼັກຄື: ລາຍຊື່ຂອງຊ່ອງໂຫວ່ຈຸດອ່ອນ, ຄຳອະທິບາຍຂອງແຕ່ລະຊ່ອງໂຫວ່ຈຸດອ່ອນ, ຄ່າຄວາມສ່ຽງຂອງແຕ່ລະຊ່ອງໂຫວ່ຈຸດອ່ອນ ແລະ ແນວທາງໃນການແກ້ໄຂ. ມັນສະແດງໃຫ້ເຫັນວ່າການນຳໃຊ້ເຄື່ອງມື ແຊັບ (ZAP) ແມ່ນມີຄວາມງ່າຍຫຼາຍ, ວ່ອງໄວ ແລະ ໄດ້ຜົນດີພໍສົມຄວນ. ທ່ານຍັງໄດ້ຄຳຄິດເຫັນອີກວ່າການໃຊ້ໂປຣແກຣມດັ່ງກ່າວເຖິງແມ່ນວ່າຈະເປັນໂປຣແກຣມທີ່ໃຫ້ໃຊ້ລ້ຽງ ແຕ່ກໍ່ມີການອັບເດດ (Update) ໂປຣແກຣມຢ່າງສະໝ່ຳສະເໝີ ໂດຍຢູ່ພາຍໃນການເບິ່ງແຍງດູແລຈາກອົງກອນສາກົນທີ່ບໍ່ຫວັງຜົນກຳໄລ (SSP: The Software Security Project) ເຊິ່ງມີຄວາມໜ້າເຊື່ອຖືເປັນຢ່າງດີ.

2.5 ການປະເມີນຜົນຂອງເຄື່ອງມືຊອກຫາຊ່ອງໂຫວ່ແບບ Open Source

ເຄື່ອງມືຊອກຫາຊ່ອງໂຫວ່ແບບແຫຼ່ງເປີດທີ່ໃຫ້ໃຊ້ລ້ຽ (Open source vulnerability scanners) ເປັນທາງເລືອກໜຶ່ງທີ່ສຳຄັນສຳລັບນັກພັດທະນາ ເວັບ ແອັບພລິເຄຊັນ, ໂດຍສະເພາະໃນກໍລະນີທີ່ມີງົບປະມານອັນຈຳກັດ ເຊິ່ງມັນສາມາດປະຢັດຄ່າໃຊ້ຈ່າຍໄດ້ເປັນຢ່າງຫຼາຍສົມຄວນ, ສະນັ້ນ ການປະເມີນຫາເຄື່ອງມືຄວນຈະມີການສົມທຽບປະສິດທິພາບຫາຈຸດເດັ່ນຂອງແຕ່ລະເຄື່ອງມືເພື່ອມາໝຸນໃຊ້ໃຫ້ແທດເໝາະ ສອດຄ່ອງກັບ Hilmi (2020) ໄດ້ເຮັດການທົດລອງເພື່ອປະເມີນເຄື່ອງສະແກນຊ່ອງໂຫວ່ແບບແຫຼ່ງເປີດຂອງ 2 ສອງ ອັນຄື: Paros ແລະ ZAP ກ່າວໄວ້ວ່າ ເວັບ ແອັບພລິເຄຊັນ ໄດ້ຖືກນຳໃຊ້ໂດຍມະນຸດເພື່ອລວບລວມຂໍ້ມູນ, ການສື່ສານ, e-commerce ແລະ ກິດຈະກຳອື່ນໆ. ເນື່ອງຈາກຂໍ້ມູນທີ່ຖືກຈັດເກັບ

ໂດຍເວັບ ແອັບພລິເຄຊັນ ມີຄຸນຄ່າຫຼາຍ ແລະ ອ່ອນໄຫວ (Sensitive), ສະນັ້ນ ຂະບວນການໂຈມຕີຕໍ່ຂໍ້ມູນດັ່ງກ່າວໄດ້ເພີ່ມຂຶ້ນ ດ້ວຍການພະຍາຍາມຊອກຫາຊ່ອງໂຫວ່ ແລະ ລັກລອບເອົາຂໍ້ມູນ. ດ້ວຍເຫດນີ້, ມັນເປັນສິ່ງຈຳເປັນທີ່ຈະກວດເບິ່ງຈຸດອ່ອນ ເວັບ ແອັບພລິເຄຊັນ ເພື່ອຮັບປະກັນວ່າມັນປອດໄພ. ຢ່າງໃດກໍ່ຕາມ, ການກວດສອບຊ່ອງໂຫວ່ດ້ວຍຕົນເອງເປັນວຽກທີ່ໜ້າເບື່ອ ແລະ ໃຊ້ເວລາຫຼາຍ. ດັ່ງນັ້ນ, ມີຄວາມຈຳເປັນອັນໃຫຍ່ຫຼວງສຳລັບເຄື່ອງສະແກນຄວາມອ່ອນແອຂອງແອັບພລິເຄຊັນເວັບ. ນອກຈາກນີ້ (Matti, 2021) ຍັງກ່າວອີກວ່າ: ບາງຄັ້ງການປະເມີນ ຈຳເປັນຕ້ອງໄດ້ໃຊ້ວິທີການພິເສດແບບສະເພາະ (ເທັກນິກ) ຂອງເຄື່ອງມືນັ້ນໆ ສຸມໃສ່ຊ່ອງໂຫວ່ແບບສະເພາະ ເຈາະຈົງໃນສະຖານະການການແຜ່ລະບາດແບບກ້ວາງຂວາງໃນໄລຍະເວລາດັ່ງກ່າວ ເພື່ອໃຫ້ຮັບມືກັບເຫດການຢ່າງທັນເວລາ.

3. ວິທີດຳເນີນການຄົ້ນຄວ້າ

ຮູບທີ 1: ເຄື່ອງມືຊອກຫາຊ່ອງໂຫວ່ ແຊັບ ແລະ ເວກາ

ວິທີການທົດລອງແມ່ນໄດ້ດຳເນີນການຕິດຕັ້ງ ຊອບແວ ແຊັບ ແລະ ເວກາ ລົງໃສ່ອຸປະກອນຄອມພິວເຕີ ແລ້ວດຳເນີນການສະແກນຊອກຫາຊ່ອງໂຫວ່

ຮູບທີ 2: ແຜນວາດການໃຊ້ເຄື່ອງມືທົດລອງ

3.1. ປະຊາກອນ ແລະ ກຸ່ມຕົວຢ່າງ

ກຸ່ມຕົວຢ່າງໃນການຄົ້ນຄວ້າຄັ້ງນີ້ ເນື່ອງຈາກມີຂໍ້ຈຳກັດຫຼາຍດ້ານເຊັ່ນ: ທາງດ້ານລິຂະສິດຂອງເວັບໄຊທ໌ໃນການທົດລອງ, ເວລາສັ້ນ ແລະ ຂາດທຶນຮອນສະໜັບສະໜູນໃນການຄົ້ນຄ້ວາ ຜູ້ຄົ້ນຄວ້າຈຶ່ງໄດ້ເຮັດການທົດລອງໃນ 5 ເວັບ ແອັບພລິເຄຊັນ ທີ່ໃຊ້ພາຍໃນ ບໍລິສັດ ໄຟຟ້າ ນ້ຳເທີນ 2 ຈຳກັດ ໃນເວັບໄຊທ໌ (https://www.namtheun2.com) ເຊິ່ງໄດ້ຮັບອະນຸຍາດເປັນທີ່ຮຽບຮ້ອຍແລ້ວ.

ຕາຕະລາງ 1: 5 ເວັບ ແອັບພລິເຄຊັນ ທີ່ຈະໃຊ້ໃນການທົດລອງ

ລາຍລະອຽດ	ຊື່ເວັບໄຊທ໌	ພາສາ
1. IT Asset Management Control	https://lansweeper.namtheun2.com/	ASP.net (C#)
2. IT Accessory Controlling	https://accessory.namtheun2.com	PHP
3. Purchase Order Approval	https://sapwbp.namtheun2.com	ASP.net (C#)
4. Company Info. Screen Show	https://display.namtheun2.com/	HTML
5. IT Helpdesk	https://it-helpdesk:8080	JSP

3.2 ເຄື່ອງມືທີ່ໃຊ້ໃນການເກັບກຳຂໍ້ມູນ

ອຸປະກອນທີ່ນຳໃຊ້ໃນການທົດລອງ ແມ່ນປະກອບມີ 2 ພາກສ່ວນຫຼັກຄື: ຄອມພິວເຕີ ຮາດແວ (Computer Hardware) ແລະ ຊອບແວ (Software), ເຊິ່ງລວມມີ ລະບົບປະຕິບັດການ (OS-Operating System) ແລະ ໂປຣແກຣມເສີມ ຕ່າງໆ ດັ່ງທີ່ສະແດງໃນຕາຕະລາງ 2 ແລະ 3 ຕາມລຳດັບ.

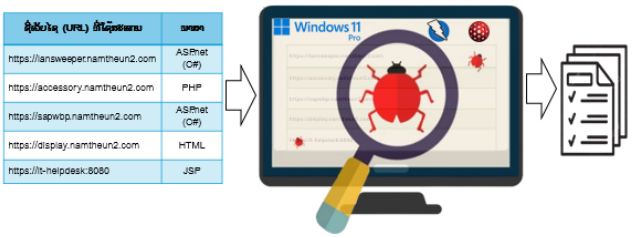
ຕາຕະລາງ **2:** ສະເປັກ ຂອງຄອມພິວເຕີ (Computer hardware specification)

Hardware	Specification
CPU	Intel(R) Core(TM) i7-7500U CPU @ 2.90GHz
RAM	16 GB
Hard disk	256 GB (SSD)

ຕາຕະລາງ 3: ຊອບແວ (Software)

Software	Version	Type
Operating system	Windows 11 Professional version 64-bit 22H2	OS
ZAP	ZAP_2_15_0_windows-x64	Tools
Vega	Vega v1.0 (Microsoft Windows 64-bit JRE (sig))	
JAVA Runtime	OpenJDK21U-jdk_x64_windows	Assembly
Web browsers	- Firefox 128.0.2 (64-bit) and Google Chrome - Version 126.0.6478.185 (Official Build) (64-bit)	

3.3. ວິທີເກັບກຳຂໍ້ມູນ



ຮູບທີ 3: ແຜນວາດການ ທົດລອງ ຈາກ 5 ເວັບໄຊ

ໃນການດຳເນີນການຄົ້ນຄວ້າຄັ້ງນີ້ເປັນການຄົ້ນຄວ້າທີ່ຕ້ອງ ການຊອກຫາ 1. ປະສິດທິພາບໃນການຄົ້ນຫາຈຳນວນຊ່ອງໂຫວ່, 2. ປະສິດທິພາບດ້ານຄວາມໄວ (ເວລາ) ໃນການກວດຈັບຫາຊ່ອງໂຫວ່ ແລະ 3. ປະສິດທິພາບໃນການແກ້ໄຂບັນຫາທີ່ພົບຂອງແຕ່ລະເຄື່ອງມື ສະນັ້ນການທົດລອງແມ່ນໄດ້ຕິດຕັ້ງ 2 ເຄື່ອງມື ແຊັບ ແລະ ເວກາ ລົງ ໃສ່ຄອມພິວເຕີ ວິນໂດ 11 ໂປຣເຟສຊັນນອລ ເວີຊັນ, ແລ້ວເຮັດການ ສະແກນຫາຊ່ອງໂຫວ່ທີ່ເປັນຈຸດອ່ອນຈາກການໂຈມຕີຂອງ ແອັກເກີ ຂອງ 5 ເວັບໄຊ ທີ່ຖືກພັດທັດນາຈາກຫຼາກຫຼາຍພາສາແຕ່ກ່າຕ່າງ ກັນເຊັ່ນ: ພາສາ ພີເຮັຈພີມ(PHP), ພາສາຊີຊາຟ (C#), ເຮັຈທີເອັມ ເອລ (HTML) ແລະ ຈາວາ (Java) ດັ່ງສະແດງໃນຕາຕະລາງ 1, ເຊິ່ງ ແຕ່ລະເວັບໄຊແມ່ນຈະເຮັດ ການສະແກນ 10 ຄັ້ງ ພ້ອມໆກັນທຸກຄັ້ງ ຂອງ 2 ເຄື່ອງມື. ຜົນລາຍລະອຽດການທົດລອງແມ່ນຖືກບັນທຶກເປັນ ແບບອັດຕະໂນມັດເປັນແຕ່ລະເຊັສຊັນ (Session) ຂອງການ ສະແກນໃນແຕ່ລະຄັ້ງຂອງແຕ່ລະເຄື່ອງມືເລີຍ. ຫຼັງຈາກນັ້ນ, ຜູ້ທົດ ລອງຈະໄດ້ຂັດເລືອກເອົາຂໍ້ມູນສະເພາະແຕ່ 3 ຫົວຂໍ້ທີ່ຈະເຮັດການ ວິໄຈໂດຍການບັນທຶກແຍກລົງໃສ່ໃນເອັກເຊວໄຟລ໌ (Excel files)) ເພື່ອເກັບເປັນກຳສະຖິຕິທີ່ຈະນຳໄປວິເຄາະໃນຫົວຂໍ້ຖັດໄປ.

3.4 ການວິເຄາະຂໍ້ມູນ

ນຳຜົນລາຍລະອຽດການທົດລອງຈາກໃສ່ຕາຕະລາງ ເອັກ ເຊວໄຟລ໌ (Excel files)), ມາເຮັດການຄິດໄລ່ຫາຄ່າສະເລ່ຍລວມທັງ 10 ຄັ້ງ ຂອງແຕ່ລະປະສິດທິພາບຂອງທັງ 3 ຫົວຂໍ້ (ຈຳນວນຊ່ອງໂຫວ່, ເວລາໃນການຄົ້ນຫາ, ແລະ ຄຳແນະນຳສຳລັບການແກ້ໄຂບັນຫາເບື້ອງຕົ້ນ) ແລ້ວນຳມາວິເຄາະ ແລະ ສົມທຽບກັນຂອງ 2 ເຄື່ອງມື ຕາມແຕ່ ລະຕົວຊີ້ວັດ ໂດຍການຫາຄ່າສະເລ່ຍແມ່ນໃຊ້ສູດທີ່ 1, ສ່ວນການສົມ ທຽບແມ່ນອີງຕາມສູດການສົມທຽບສູດທີ່ 2 - 4.

1. ຄ່າສະເລ່ຍ

ອີງຕາມຂໍ້ມູນໃນຕາຕະລາງ 4 – 6 ແມ່ນຜົນຄ່າ ສະເລ່ຍລວມຂອງ 3 ຫົວຂໍ້ (ຈຳນວນຊ່ອງໂຫວ່, ເວລາໃນການຄົ້ນຫາ, ແລະ ຄຳແນະນຳສຳລັບການແກ້ໄຂບັນຫາເບື້ອງຕົ້ນ) ທີ່ໄດ້ຈາກການ ຄຳນວນໂດຍໃຊ້ສູດທີ່ 1 ທັງ 5 ເວັບໄຊທ໌ ທີ່ເຮັດການສະແກນ 10 ຄັ້ງ ຕໍ່ ເວັບໄຊທ໌

- ສູດທີ່ 1: Avg(No/Time/Sol)_Tool-x = $\frac{\sum X}{10}$

- $\sum X$: ຄ່າລວມຂອງຈຳນວນ ຫຼື ເວລາ ຫຼື ຄະແນນການແກ້ບັນຫາ ຊ່ອງໂຫວ່ຂອງແຕ່ລະເຄື່ອງມື ແຊັບ ຫຼື ເວກາ

- Avg(No/Time/Sol)_Tool-x: ຄ່າສະເລ່ຍລວມຂອງຈຳນວນ ຫຼື ເວລາ ຫຼື ຄະແນນການແກ້ບັນຫາ ຊ່ອງໂຫວ່ຂອງແຕ່ລະເຄື່ອງມື ແຊັບ ຫຼື ເວກາ

2. ການສົມທຽບ

ການສົມທຽບແມ່ນນຳຜົນລວມໃນຕາຕະລາງ 4 – 6 ມາ ເຮັດການສົມທຽບດ້ວຍສູດທີ່ 2 – 4 ດັ່ງລຸ່ມນີ້:

- ສູດທີ່ 2: ສູດສົມທຽບຄ່າຈຳນວນຊ່ອງໂຫວ່ Avg(No)_ZAP VS Avg(No)_Vega

- Avg(No)_ZAP: ຄ່າສະເລ່ຍຈຳນວນຊ່ອງໂຫວ່ລວມ ຂອງ ເຄື່ອງມື ແຊັບ

- Avg(No)_Vega: ຄ່າສະເລ່ຍຈຳນວນຊ່ອງໂຫວ່ລວມ ຂອງ ເຄື່ອງມື ເວກາ

ຕາຕະລາງ 4: ຄ່າສະເລ່ຍລວມຂອງຈຳນວນຊ່ອງໂຫວ່

ລ/ດ	ເວັບ ແອັບພລິເຄຊັນ	ຈຳນວນຊ່ອງໂຫວ່ທີ່ພົບ	
		ເຄື່ອງມື ແຊັບ (ZAP)	ເຄື່ອງມື ເວກາ
1	https://lansweeper.namtheun2.com	15	1.9
2	https://accessory.namtheun2.com	11.2	2.5
3	https://sapwbp.namtheun2.com	13.1	6.9
4	https://display.namtheun2.com	10	0
5	https://it-helpdesk:8080	11	0
ລວມ		60.3	11.3

- ສູດທີ 3: ສູດສົມທຽບຄ່າຄວາມໄວ (ເວລາ) ໃນການຄົ້ນຫາຊ່ອງໂຫວ່

Avg(Time)_ZAP VS Avg(Time)_Vega

- Avg(Time)_ZAP: ຄ່າສະເລ່ຍເວລາໃນການຄົ້ນຫາຊ່ອງໂຫວ່ລວມ ຂອງເຄື່ອງມື ແຊັບ
- Avg(Time)_Vega: ຄ່າສະເລ່ຍເວລາໃນການຄົ້ນຫາຊ່ອງໂຫວ່ລວມ ຂອງເຄື່ອງມື ເວກາ

ຕາຕະລາງ 5: ຄ່າສະເລ່ຍເວລາໃນການຄົ້ນຫາຊ່ອງໂຫວ່ລວມ

ເວັບ ແອັບພລິເຄຊັນ	**ເວລາທີ່ໃຊ້ (ວິນາທີ)**	
	ເຄື່ອງມື ແຊັບ (ZAP)	**ເຄື່ອງມື ເວກາ**
https://lansweeper.namtheun2.com	308.65	118.91
https://accessory.namtheun2.com	132.41	0.13
https://sapwbp.namtheun2.com	39,964.40	2.80
https://display.namtheun2.com	669.57	0.00
https://it-helpdesk:8080	288.94	3.95
ລວມ (ວິນາທີ)	**41,364.01**	**125.80**

- ສູດທີ 4: ສູດສົມທຽບຄ່າຄວາມໄວ (ເວລາ) ໃນການຄົ້ນຫາຊ່ອງໂຫວ່

Avg(Sol)_ZAP VS Avg(Sol)_Vega

- Avg(Sol)_ZAP: ຄ່າຄະແນນຄາແນະນຳການແກ້ບັນຫາຊ່ອງໂຫວ່ທີ່ພົບ ຂອງເຄື່ອງມື ແຊັບ

- Avg(Sol)_Vega: ຄ່າຄະແນນຄາແນະນຳການແກ້ບັນຫາຊ່ອງໂຫວ່ທີ່ພົບ ຂອງເຄື່ອງມື ເວກາ

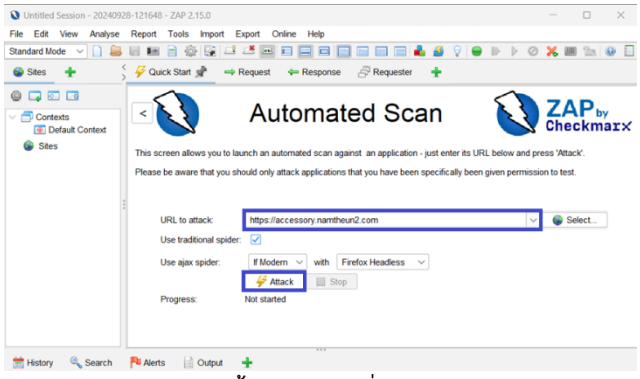
ຕາຕະລາງ 6: ຄ່າສົມທຽບຄຳແນະໃນການແກ້ບັນຫາຂອງຊ່ອງໂຫວ່ທີ່ພົບລວມ (The summary points of solution for initial troubleshooting)

No	Solution for each web	Solution Points	
		ZAP	Vega
1	https://lansweeper.namtheun2.com	27	12
2	https://accessory.namtheun2.com	24	12
3	https://sapwbp.namtheun2.com	27	9
4	https://display.namtheun2.com	21	0
5	https://it-helpdesk:8080	24	0
ລວມ		**123**	**33**

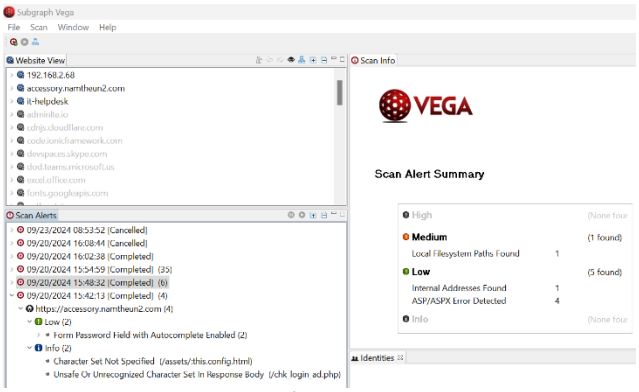
ລດ	**ເວັບ ແອັບພລິເຄຊັນ**	**ຄະແນນ ຄຳແນະໃນການແກ້ບັນຫາ**	
		ເຄື່ອງມື ແຊັບ (ZAP)	**ເຄື່ອງມື ເວກາ**

4. ຜົນການຄົ້ນຄວ້າ

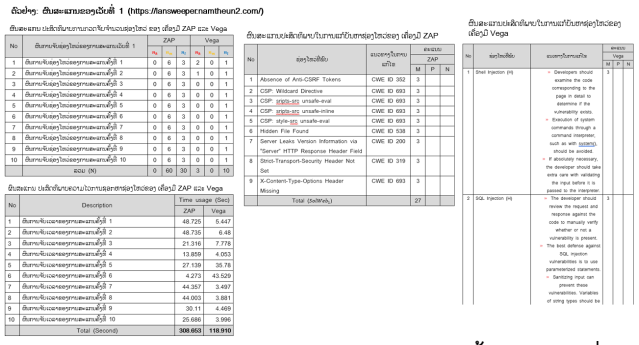
ຜົນການທົດລອງແມ່ນຈະໄດ້ຖືກບັນທຶກແຍກລົງໃສ່ຕາຕະລາງ ເອັກເຊວໄຟລ໌ (Excel files)) ຕ່າງຫາກ, ເພື່ອເກັບກຳສະຖິຕິກ່ຽວກັບແຕ່ລະຊ່ອງໂຫວ່ເຊັ່ນ: ຈຳນວນຊ່ອງໂຫວ່, ເວລາໃນການຄົ້ນຫາ, ແລະ ຄຳແນະນຳສຳລັບການແກ້ໄຂບັນຫາເບື້ອງຕົ້ນ. ຫຼັງຈາກນັ້ນ, ທຳການຄິດໄລ່ຫາຄ່າສະເລ່ຍລວມທັງ 10 ຄັ້ງ ຂອງແຕ່ລະປະສິດທິພາບຂອງທັງ 3 ຫົວຂໍ້ ແລ້ວນຳມາວິເຄາະ ແລະ ສົມທຽບກັນ ຂອງ 2 ເຄື່ອງມື ຕາມແຕ່ລະຕົວຊີ້ວັດ.



ຮູບ 4: ໜ້າຕາ ແລະ ໃສ່ຂໍ້ມູນເວັບໄຊ ເພື່ອເຮັດການສະແກນ ຂອງເຄື່ອງມືແຊັບ



ຮູບ 5: ໜ້າຕາ ແລະ ໃສ່ຂໍ້ມູນເວັບໄຊ ເພື່ອເຣັດການສະແກນ ຂອງໂປຣແກຣມເຄື່ອງມື ເວກາ



ຮູບ 6: ຜົນສະແກນ ຫາ 3 ປະສິດທິພາບ 10 ຄັ້ງ ຂອງເວັບທີ1 ເຄື່ອງມື ແຊັບແລະ ເວກາ

ຕາຕະລາງຜົນສະຫຼຸບລວມຂອງ 3 ຫົວຂໍ້ຂອງ 2 ເຄື່ອງມື ຕາມແຕ່ລະ ຕົວຊີ້ວັດດັ່ງນີ້:

ຫົວຂໍ້	ເຄື່ອງມື ZAP	ເຄື່ອງມື Vega
1 ປະສິດທິພາບການກວດຈັບຈຳນວນ ຊ່ອງໂຫວ່	60.3 Points	11.3 Points
2 ປະສິດທິພາບຄວາມໄວການຊອກຫາ ຊ່ອງໂຫວ່	41,364.01 Sec	125.80 Sec
3 ປະສິດທິພາບໃນການແກ້ບັນຫາຊ່ອງ ໂຫວ່	123 Points	33 Points

4.1 ສົມທຽບຈຳນວນຊ່ອງໂຫວ່

ຜົນການທົດສອບທັ້ງ 10 ຄັ້ງ ຂອງຈຳນວນຊ່ອງໂຫວ່ທີ່ໄດ້ບັນທຶກແຍກ ໄວ້ໃນຕາຕະລາງແມ່ນໄດ້ຖືກຄຳນວນຫາຄ່າສະເລ່ຍລວມ ໂດຍ ປະຕິບັດຕາມລຳດັບແຕ່ ເວັບທີ່ 1 ຫາ ເວັບທີ່ 5 ຂອງແຕ່ລະເຄື່ອງມື ແຊັບ ແລະ ເວກາ ດັ່ງທີ່ໄດ້ສະແດງໃນ ຕາຕະລາງ 4.

ສຸດທ້າຍຈຶ່ງນຳຜົນຂອງທັງ 2 ເຄື່ອງມື ມາສົມທຽບກັນດ້ວຍ ສູດທີ່ 2, ຈາກຜົນໄດ້ຮັບທີ່ສະແດງໃນຕາຕະລາງ 4 ແລະ ການສົມ ທຽບຄ່າຈຳນວນຊ່ອງໂຫວ່ລວມດ້ວຍ ສູດທີ່ 2 ຊີ້ໃຫ້ເຫັນວ່າ: ປະສິດທິພາບໃນການກວດຈັບຊອກຫາຈຳນວນຊ່ອງໂຫວ່ຂອງເຄື່ອງມື ແຊັບ ແມ່ນພົບເຫັນຈຳນວນຂອງຊ່ອງໂຫວ່ໄດ້ຫຼາຍກ່ວາ ເຄື່ອງມື ເວ ກາ.

(ZAP)60.3 > 11.3(Vega)

4.2 ສົມທຽບຄວາມໄວໃນການຄົ້ນຫາຊ່ອງໂຫວ່

ເຊັ່ນດຽວກັນ, ຜົນການທົດສອບທັ້ງ 10 ຄັ້ງ ຂອງຄວາມໄວ (ເວລາ) ໃນການຄົ້ນຫາຊ່ອງໂຫວ່ທີ່ໄດ້ບັນທຶກແຍກໄວ້ໃນຕາຕະລາງ ແມ່ນໄດ້ຖືກຄຳນວນຫາຄ່າສະເລ່ຍລວມ ໂດຍປະຕິບັດຕາມລຳດັບ ເວັບທີ່ແຕ່ 1 ຫາ ເວັບທີ່ 5 ຂອງແຕ່ລະເຄື່ອງມື ແຊັບ ແລະ ເວກາ ດັ່ງທີ່ ໄດ້ສະແດງໃນຕາຕະລາງ 5.

ສຸດທ້າຍຈຶ່ງນຳຜົນຂອງທັງ 2 ເຄື່ອງມື ມາສົມທຽບກັນດ້ວຍ ສູດທີ່ 3 ຈາກຜົນໃນຕາຕະລາງ 5 ແລະ ການສົມທຽບຄ່າເວລາໃນການ ຄົ້ນຫາຊ່ອງໂຫວ່ລວມ ດ້ວຍສູດທີ່ 3 ຊີ້ໃຫ້ເຫັນວ່າ: ປະສິດທິພາບ ຄວາມໄວໃນການກວດຈັບຊອກຫາ ຊ່ອງໂຫວ່ ຂອງເຄື່ອງມື ແຊັບ ແມ່ນໃຊ້ເວລາຫຼາຍກວ່າ (ຊ້າກ່ວາ) ເຄື່ອງມື ເວກາ.

41,364.01 Sec > 125.80 Sec

4.3 ສົມທຽບຄຳແນະໃນການແກ້ບັນຫາຂອງຊ່ອງໂຫວ່ທີ່ພົບ

ປະຕິບັດຄ້າຍຄືກັນ, ຜົນຮັບຂອງການທົດສອບ 10 ຄັ້ງ ຂອງ ການຊອກຫາຊ່ອງໂຫວ່ໃນ 5 ເວັບ ຈະມີຄຳແນະໃນການແກ້ບັນຫາ ຂອງຊ່ອງໂຫວ່ທີ່ພົບແມ່ນໄດ້ບັນທຶກແຍກໄວ້ໃນຕາຕະລາງເຊັ່ນກັນ. ເຖິງຢ່າງໃດກໍ່ຕາມ, ຄະແນນການຄິດໄລ່ຫາຄ່າຄຳແນະໃນການແກ້ ບັນຫາຂອງຊ່ອງໂຫວ່ທີ່ພົບແມ່ນແຕກຕ່າງກັບຈຳນວນ ແລະ ຄວາມ ໄວຂອງການຊອກຫາຊ່ອງໂຫວ່ ເພາະວ່າຄະແນນທີ່ໄດ້ໃນແຕ່ລະ ເຄື່ອງມືແມ່ນອີງໃສ່ການຄຳນວນຈາກຂໍ້ມູນທີ່ໄດ້ຈາກເຄື່ອງມືຕົນເອງ ພຽງຝ່າຍດຽວເທົ່ານັ້ນ, ສະນັ້ນ ຜູ້ຄົ້ນຄວ້າຈະຕ້ອງໄດ້ວິເຄາະ, ພິຈາລະນາ ແລະ ປຽບທຽບຄຳແນະໃນການແກ້ບັນຫາຂອງແຕ່ລະ ເຄື່ອງມືອີກ່ອນ, ໂດຍການໄປຫາຂໍ້ມູນໃນການແກ້ບັນຫາສົມທຽບ ເພີ່ມຕື່ມ ຈາກຫຼາຍແຫຼ່ງທີ່ໜ້າເຊື່ອຖືອື່ນໆເຊັ່ນ: https://owasp.org, https://cwe.mitre.org ແລະ https:/ /nvd.nist.gov/ ວ່າ: ໝາະ ສົມ ແລະ ຖືກຕ້ອງ ຫຼື ບໍ່, ແລ້ວຈຶ່ງມາທຳນິດການໃຫ້ຄະແນນຄືນໃໝ່ ໂດຍການໃຫ້ຄະແນນຕາມຮູບແບບຕໍ່ໄປນີ້.

- ຖ້າຫາກວ່າວິທີແກ້ໄຂບັນຫາຂອງເຄື່ອງມືນັ້ນໆ ແມ່ນກົງກັບແຫຼ່ງອື່ນເປັນສ່ວນຫຼາຍ, ແມ່ນກຳນົດໃຫ້ຄະແນນເທົ່າ 3
- ຖ້າຫາກວ່າວິທີແກ້ໄຂບັນຫາຂອງເຄື່ອງມືນັ້ນໆ ແມ່ນກົງກັບແຫຼ່ງອື່ນເປັນບາງສ່ວນ, ແມ່ນກຳນົດໃຫ້ຄະແນນເທົ່າ 2
- ຖ້າຫາກວ່າວິທີແກ້ໄຂບັນຫາຂອງເຄື່ອງມືນັ້ນໆ ບໍ່ກົງກັບແຫຼ່ງອື່ນເລີຍ, ແມ່ນກຳນົດໃຫ້ຄະແນນເທົ່າ 0

ສຸດທ້າຍ, ຈຶ່ງມາຄຳນວນຫາຄ່າສະເລ່ຍລວມ, ໂດຍປະຕິບັດຕາມລຳດັບ ເວັບທີ່ 1 ຫາ ເວັບທີ່ 5 ຂອງແຕ່ລະເຄື່ອງມື ແຊັບ ແລເວກາ ດັ່ງທີ່ໄດ້ສະແດງໃນຕາຕະລາງ 6.

ສຸດທ້າຍຈຶ່ງນຳຜົນຂອງທັງ 2 ເຄື່ອງມື ມາສົມທຽບກັນດ້ວຍສຸດທີ່ 4 ຈາກຜົນ ໃນຕາຕະລາງ 5 ແລະ ການສົມທຽບຄ່າເວລາໃນການຄົ້ນຫາຊ່ອງໂຫວ່ລວມ ດ້ວຍ ສຸດທີ່ 4 ຊີ້ໃຫ້ເຫັນວ່າ: ປະສິດທິພາບຂອງຄຳແນະນຳໃນການແກ້ບັນຫາ ຕ່ຳ ກັບຊ່ອງໂຫວ່ທີ່ພົບ ຂອງເຄື່ອງມື ແຊັບ ແມ່ນດີກ່ວາ ເຄື່ອງມື ເວກາ.

(ZAP) 123 > 33 (Vega)

5. ອະພິປາຍຜົນ

ອີງຕາມຜົນໄດ້ຮັບຂອງການທົດລອງສຳລັບ 3 ຫົວຂໍ້ ສຳລັບ 2 ເຄື່ອງມື ແຊັບ ແລະ ເວກາ ຂ້າງເທິງຊີ້ໃຫ້ເຫັນວ່າ: ໃນແກ້ບັນຫາຕົວຈິງນັກພັດທະນາ ເວັບ ແອັບພລິເຄຊັນ ຈຳເປັນຕ້ອງໄດ້ເຮັດການປະເມີນກ່ອນທີ່ຈະເລືອກເອົາເຄື່ອງມືທີ່ເໝາະສົມຕໍ່ການແກ້ໄຂບັນຫານັ້ນໆມານຳໃຊ້ ເພື່ອໃຫ້ໄດ້ຜົນຮັບທີ່ມີປະສິດທິພາບທີ່ດີທີ່ສຸດ, ເຊິ່ງມັນສອດຄ່ອງກັບບົດຄົ້ນຄວ້າ (Hilmi, 2020) ເຮັດການປະເມີນເຄື່ອງມືຊອກຫາຊ່ອງໂຫວ່ແບບແຫຼ່ງເປີດທີ່ໃຫ້ໃຊ້ລ້ຳ (Open source vulnerability scanners) ໂດຍໃຊ້ເຄື່ອງມື Paros ແລະ ZAP ເຊິ່ງຜົນປະກົດວ່າເຄື່ອງມື ZAP ແມ່ນເຮັດໄດ້ດີກວ່າ ແຕ່ວ່າການທົດລອງດັ່ງກ່າວແມ່ນເນັ້ນໃສ່ແຕ່ສົມທຽບໃນຫົວຂໍ້ດຽວຄື: ປະສິດທິພາບໃນການກວາຈັບຫາຊ່ອງໂຫວ່ເທົ່ານັ້ນ, ຖ້າສົມທຽບກັບການຄົ້ນຄວ້າໃນຄັ້ງນີ້ແມ່ນຍັງບໍ່ພຽງພໍ ແລະ ດ້ອຍກວ່າ 2 ຫົວຂໍ້. ເຖິງກໍ່ຕາມ, (Matti, 2021) ເພື່ອແທດເໝາະກັບສະພາບການໃນໄລຍະນັ້ນໆ ແລະ ໃຫ້ຮັບມືກັບເຫດການຢ່າງທັນເວລາຈຳເປັນຕ້ອງໄດ້ໃຊ້ເທັກນິກພິເສດແບບສະເພາະຂອງເຄື່ອງ ມືນັ້ນໆ ສຸມໃສ່ໃນສະຖານະການການແຜ່ລະບາດແບບກວ້າງຂວາງຂອງໄພການໂຈມຕີດັ່ງກ່າວ, ແຕ່ຄຳຖາມມີຢູ່ວ່າຖ້າບໍ່ມີການປະເມີນ ຫຼື ເຮັດການຄົ້ນຄວ້າແບບຮອບດ້ານມາກ່ອນແລ້ວເຄື່ອງມືອັນເໝາະສົມທີ່ຈິງຄວນຈະໃຊ້ອັນໃດ. ສະນັ້ນ, ນັກພັດທະນາເອງກໍ່ຄວນຈະຕ້ອງໄດ້ມີການສຶກສາ, ຄົ້ນຄວ້າ ແລະ ທົດລອງດ້ວຍວິທີໃໝ່ໆຢູ່ເລື້ອຍໆ, ເຊິ່ງມັນສອດຄ່ອງກັບບົດຄົ້ນຄວ້າ-ວິໄຈ Abdulghaffar et al. (2023) ໄດ້ເຮັດການວິໄຈ ການປັບປຸງຄວາມປອດໄພຂອງ ເວັບ ແອັບພລິເຄຊັນ ຜ່ານການທົດສອບການເຈາะລະບົບແບບອັດຕະໂນມັດດ້ວຍເຄື່ອງສະແກນຊ່ອງໂຫວ່ຫຼາຍອັນ ໂດຍການອອກແບບເພື່ອເຮັດໃຫ້ເປັນອັດຕະໂນມັດໃນການດຳເນີນງານຂອງຫຼາຍຕົວສະແກນຫາຊ່ອງໂຫວ່ໃນ ເວັບ ແອັບພິເຄຊັນ (Web Application Vulnerability Scanners (WAVS)) ພາຍໃຕ້ແພລັດຟອມ (Platform) ດຽວ, ເຊິ່ງໄດ້ນຳເອົາຄຸນສົມບັດຂອງ 2 ເຄື່ອງມືສະແກນຄື: ອາຣັກນີ (Arachni) ແລະ ແຊັບ ມາຮວມເຂົ້າກັນ, ຜົນການສຶກສາສະແດງໃຫ້ເຫັນວ່າ: ແມ່ນກວດຈັບຈຳນວນຊ່ອງໂຫວ່ໄດ້ຫຼາຍກວ່າເກົ່າເມື່ອປຽບທຽບກັບການສະແກນແບບດ່ຽວຂອງແຕ່ລະເຄື່ອງມື. ສອດຄ່ອງກັບຜົນການຄົ້ນຄວ້າຂອງ (Khemprakhon et al. 2023) ເພື່ອເປັນການເສີມສ້າງຄວາມໝັ້ນຄົງໃຫ້ກັບເວັບໄຊພື້ນຖານທີ່ສ້າງມາຈາກເວີດເພຣສ (Wordpress) ໄດ້ໃຊ້ເຄື່ອງມື ເບີສຊູສ (Burpsuite) ເຮັດການທົດສອບຊ່ອງໂຫວ່ ແລະ ບັນຫາດ້ານຄວາມປອດໄພຂອງ ເວັບ ແອັບພລິເຄຊັນ ດ້ວຍວິທີການ ບຣັສຟອສ ແອັດແທັກ (Brute-force attack) ແລະ ເອັສເອັສແອລ ສຕຣິບ ແອັດແທັກ (SSL Strip attack) ຜົນການທົດສອບແມ່ນສະທ້ອນຄວາມເປັນຈິງຂອງ ປະສິດທິພາບຂອງ ເວັບ ແອັບພລິເຄຊັນ ໄດ້ຢ່າງຖືກຕ້ອງ. (Pasomsup, 2020) ໄດ້ເຮັດການຄົ້ນຄ້ວາຫົວຂໍ້: ການປະເມີນເຄື່ອງສະແກນຊ່ອງໂຫວ່ຂອງເວັບແຫຼ່ງເປີດ ແລະ ເຕັກນິກ ທີ່ໃຊ້ເພື່ອຊອກຫາຊ່ອງໂຫວ່ ເອສຄິວແອວ ອິນເຈັກຊັນ (SQL injection(SQLi)) ແລະ ຄຣອສໄຊສະຄຣິບຕິ້ງ (cross-site scripting (XSS)) (Evaluation of open-source web vulnerability scanners and their techniques used to find SQL injection (SQLi) and cross-site scripting vulnerabilities (XSS)) ໂດຍການຈັດຫາເຄື່ອງສະແກນຫາຊ່ອງໂຫວ່ຂອງເວັບແອັບພິເຄຊັນແບບແຫຼ່ງເປີດ (Open-source) ມາ 7 ເຄື່ອງມື, ເຊິ່ງລ້ວນແລ້ວແຕ່ຖືກພັດທະນາຈາກຫຼາກຫຼາຍພາສາເຊັ່ນ: ພາສາ Java, Perl, Python, C ແລະ Ruby. ນຳມາວິເຄາะ ແລະ ຄັດເລືອກ ເຄື່ອງມືທີ່ເໝາะສົມທີ່ສຸດໃນການທົດລອງ, ໃນທີ່ສຸດກໍ່ເລືອກໄດ້ 3 ເຄື່ອງມືຄື: ແຊັບ, ເວກາ ແລະ ອາປິຕິ (Wapiti) ທີ່ຈະໃຊ້ໃນການສະແກນຊອກຫາຊ່ອງໂຫວ່ທັງ 2 ແບບ ທີ່ກ່າວມາຂ້າງເທິງ, ຜົນສະຫຼຸບສຸດທ້າຍໄດ້ສະແດງໃຫ້ເຫັນວິທີການ ແລະ ຄວາມສາມາດເດັ່ນທີ່ແຕກຕ່າງກັນຂອງໝູນໃຊ້ແຕ່ລະ

ເຄື່ອງມືສະແກນແຕ່ລະອັນແມ່ນສາມາດນຳໃຊ້ໄດ້ດີເດັ່ນແຕກຕ່າງກັນ ກັນ, ແຕ່ວ່າແຕ່ລະເຄື່ອງມືກໍ່ມີຂໍ້ດີ ແລະ ຂໍ້ຈຳກັດຂອງໃຜມັນ. ດັ່ງທີ່ ໄດ້ກ່າວມາແລ້ວການຄົ້ນຄວ້າສ່ວນຫຼາຍແມ່ນມຸ່ງເນັ້ນໄປໃສ່ແຕ່ການ ພັດທະນະປະສິດທິພາບການກວດຈັບຫາຈຳນວນຊ່ອງໂຫວ່ ເຊິ່ງບໍ່ໄດ້ ຄຳນຶງເຖິງຄວາມໄວ ແລະ ການແກ້ບັນຫາຢ່າງວ່ອງໄວ ແລະ ມີ ປະສິດທິພາບ. ດັ່ງການ, ການຄົ້ນຄວ້າຄັ້ງນີ້ຈຶ່ງໄດ້ ບໍຈຳກັດຢູ່ພຽງຫົວຂໍ້ ດຽວຈຶ່ງໄດ້ສຸມໃສ່ 3 ຫົວຂໍ້ປະສິດທິພາບ ເພື່ອເຮັດໃຫ້ການພັດທະນາ ເວັບ ແອັບພລິເຄຊັນ ໃຊ້ເຄື່ອງມືທີ່ເໝາະສົມ, ມີປະສິດທິພາບ ແລະ ຫຼຸດຜ່ອນຊ່ອງໂຫວ່ໃນ ເວັບ ແອັບພລິເຄຊັນ ໃຫ້ໜ້ອຍທີ່ສຸດເທົ່າທີ ເປັນໄປໄດ້ ຫຼື ບໍ່ມີເລີຍກໍ່ຍິ່ງດີ.

ເຖິງຢ່າງໃດກໍ່ຕາມ, ການຄົ້ນຄວ້າຄັ້ງນີ້ອາດຈະເປັນແບບ ສະເພາະໃນແຫຼ່ງຂໍ້ມູນບ່ອນດຽວໃນວົງປິດ, ສະນັ້ນ ຖ້າໃນອະນາຄົດ ທີ່ມີຄວາມພ້ອມທາງດ້ານງົບປະມານ, ເວລາ ແລະ ບຸກຄະລາກອນ ທີ່ ພຽງພໍ ໃນຄວາມຄິດເຫັນຂອງຜູ້ຄົ້ນຄວ້າແມ່ນຈະເຮັດການທົດລອງ ຢ່າງໜ້ອຍນຳໃຊ້ 10 ເວັບໄຊທ໌ແບບເປີດ (Public Websites) ທີ່ມີ ແຫຼ່ງທີ່ມາຫຼາຍແຫ່ງແຕກຕ່າງກັນ ໂດຍການນຳໃຊ້ 6 ເຄື່ອງມືໃນ 3 ປະເພດຄື: 2 ເຄື່ອງມືຈາກແບບປະເພດໃຫ້ໃຊ້ລ້າ (Free), 2 ເຄື່ອງມື ຈາກແບບປະເພດໃຫ້ໃຊ້ທົດລອງ (Trial version) ແລະ 2 ເຄື່ອງມື ຈາກແບບປະເພດທາງການຄ້າ (ແບບມີຄ່າໃຊ້ຈ່າຍ - Commercial) ເພື່ອໃຫ້ໄດ້ຜົນທີ່ຫຼາກຫຼາຍຂຶ້ນ ແລະ ກວ້າງຂວາງ ເຮັດໃຫ້ການວິໄຈ ຊັດເຈນຂຶ້ນກວ່າເກົ່າ ແລະ ໃຫ້ໄດ້ຮັບປະສິດທິພາບສູງກວ່າເດີມ.

6. ສະຫຼຸບຜົນ

ບົດຄົ້ນຄວ້ານີ້ ແມ່ນການສຶກສາປະສິດທິພາບຂອງ 2 ເຄື່ອງມື ແບບໂອເພັນຊອສຄື: ແຊັບ ແລະ ເວກາ ໃນການຊອກຫາຊ່ອງໂຫວ່ໃນ ເວັບ ແອັບພລິເຄຊັນ ເຊິ່ງຈາກຜົນການຄົ້ນຄວ້ານີ້ໃຫ້ເຫັນວ່າ : ເຄື່ອງມື ແຊັບ ແມ່ນເຮັດໄດ້ກວ່າທາງດ້ານ ປະສິດທິພາບໃນການກວດຈັບຫາ ຈຳນວນຊ່ອງໂຫວ່ (ດ້ວຍຜົນຄະແນນສະເລ່ຍ (ZAP) 60.3 ຕໍ່ 11.3 (Vega)) ແລະ ຄຳແນະນຳໃນການແກ້ໄຂບັນຫາ (ດ້ວຍຜົນຄະແນນ ສະເລ່ຍ (ZAP) 123 ຕໍ່ 33 (Vega)). ແຕ່ເຖິງຢ່າງໃດກໍ່ຕາມ, ເຄື່ອງມື ເວກາ ກໍ່ສາມາດສະແກນຫາຊ່ອງໂຫວ່ໄດ້ໄວກວ່າ (ດ້ວຍຜົນເວລາ ສະເລ່ຍ (ZAP) 41,364.01 Sec ຕໍ່ 125.80 Sec (Vega)).

ແນ່ນອນວ່າ, ນີ້ເປັນການສຶກສາທົດລອງສະເພາະ-ເຈາະຈົງ ຂອງ5 ເວັບ ແອັບພລິເຄຊັນ ທີ່ນຳໃຊ້ພາຍໃນ ບໍລິສັດ ໄຟຟ້າ ນ້ຳເທີນ 2 ຈຳກັດ ແລະ ເພື່ອການສຶກສາເທົ່ານັ້ນ, ເຊິ່ງການສຶກສາເພີ່ມເຕີມຍັງ ຕ້ອງໄດ້ຮັບການສືບຕໍ່. ແຕ່ເຖິງຢ່າງໃດກໍ່ຕາມ, ໃນກໍລະນີທີ່ພົບບັນຫາ ໃນການປະຕິບັດງານຕົວຈິງ, ຫວັງເປັນຢ່າງຫຍິ່ງວ່າ: ນັກພັດທະນາ ເວັບ ແອັບພລິເຄຊັນ ຈະເລືອກເອົາ ເຄື່ອງມືທີ່ເຫມາະສົມ ແລະ ແກ້ ໄຂບັນຫາໄດ້ຖືກຕ້ອງ, ເໝາະສົມ ແລະ ໃຫ້ປະສິດທິພາບທີ່ດີທີ່ສຸດ ມານຳໃຊ້ໃນສະຖານະການນັ້ນໆໂດຍບໍ່ຍຶດຕິດກັບເຄື່ອງໃດໜຶ່ງພຽງ ອັນດຽວ.

7. ຂໍ້ຈຳກັດ

ເນື່ອງຈາກວ່າການຄົ້ນຄວ້ານີ້ມີຂໍ້ຈຳກັດຫຼາຍດ້ານເຊັ່ນ: ບໍ່ ມີທຶນງົບປະມານໃນການເຮັດວິໄຈແບບວົງກວ້າງ, ບຸກຄະລາກອນ ຈຳກັດ, ເວລາສັ້ນ ແລະ ຂໍ້ຈຳກັດທາງດ້ານລິຂະສິດຂອງແຕ່ລະເວັບໄຊ , ສະນັ້ນ ການປະຕິບັດການທົດລອງຈຶ່ງໄດ້ດຳເນີນແບບວົງແຄບ ສະເພາະ-ເຈາະຈົງ ໃນບາງ ເວັບ ແອັບພລິເຄຊັນ ທີ່ນຳໃຊ້ພາຍໃນ ບໍລິສັດ ໄຟຟ້າ ນ້ຳເທີນ 2 ຈຳກັດ ເພື່ອໃຫ້ການຄົ້ນຄວ້າສາມາດດຳເນີນ ການໃຫ້ສຳເລັດ-ລຸລ່ວງໄປດ້ວຍດີ, ອີກປະການໜຶ່ງ ເພື່ອໃຫ້ການ ຄົ້ນຄວ້ານີ້ເປັນແນວທາງໃຫ້ນັກຄົ້ນຄວ້ານຸ້່ນໃໝ່ໄດ້ສຶກສາ-ຮຽນຮູ້ ແລະ ນຳໄປຕໍ່ຍອດໃນອະນາຄົດ.

8. ຂໍ້ສະເໜີແນະ

ແນ່ນອນວ່າ, ການສຶກສາເພີ່ມເຕີມຍັງຕ້ອງໄດ້ຮັບການສືບຕໍ່ ເນື່ອງຈາກວ່າ ການຄົ້ນຄວ້າຄັ້ງນີ້ແມ່ນດຳເນີນແບບວົງແຄບສະເພາະຢູ່ ພາຍໃນ ບໍລິສັດ ໄຟຟ້າ ນ້ຳເທີນ 2 ຈຳກັດ ເທົ່ານັ້ນ, ເພາະສະນັ້ນໃນ ການຄົ້ນຄວ້າຄັ້ງໜ້າແນະນຳໃຫ້ທົດລອງນຳໃຊ້ 10 ເວັບໄຊທ໌ທີ່ມີແຫຼ່ງ ທີ່ມາແຕກຕ່າງກັນເຊັ່ນ: ອົງກອນ, ເອກະຊົນ, ບໍລິສັດ, ລັດຖະບານ ,..... ດ້ວຍເຫດຜົນທີ່ຮັບຮູ້ນຳກັນແລ້ວວ່າ: ເຕັກໂນໂລຊີ ນັ້ນມີ ການພັດທະນາແບບກ້າວກະໂດດ ແລະ ບໍ່ຢຸດໜຶ່ງ, ເຊິ່ງໄພຄຸກຄາມ ທາງໄຊເບີ ກໍ່ເປັນໄພຮ້າຍອັນໜຶ່ງທີ່ເກາະຕິດກັບການຈະເລີນເຕີບໂຕ ຂອງເຕັກໂນໂລຊີ, ສະນັ້ນ ເທັກນິກ ແລະ ກົນລະຢຸດອື່ນໆທີ່ຄາດວ່າ ຈະໃຫ້ຜົນດີກໍ່ແນະນຳໃຫ້ໝູນໃຊ້ເຂົ້າໃນການຄົ້ນຄວ້າໃນຄັ້ງຕໍ່ໆໄປ ອີກຕື່ມ.

ໃນຄວາມຄິດເຫັນຂອງຜູ້ຄົ້ນຄວ້າ, ວິທີການຄົ້ນຄວ້ານີ້ແມ່ນ ສາມາດເອົາໄປນຳໃຊ້ໄດ້ຈິງແບບປະຢັດງົບປະມານໂດຍການນຳໃຊ້ຈິງ ນັ້ນແມ່ນແນະນຳໃຫ້ໃຊ້ທັ້ງ 2 ເຄື່ອງມືເລີຍ ເພາະວ່າການກວດຈັບຫາ ຊ່ອງໂຫວ່ຂອງແຕ່ລະເຄື່ອງມືນັ້ນມີລັກສະນະເດັ່ນທີ່ແຕກຕ່າງກັນ, ເຊິ່ງຈະເຮັດໃຫ້ສາມາດກວດພົບຊ່ອງໂຫວ່ທີ່ແຕກຕ່າງກັນຫຼາຍກວ່າ. ແຕ່ເຖິງຢ່າງໃດກໍ່ຕາມ, ຖ້າມີເວລາຈຳກັດ ແລະ ເລັ່ງດ່ວນແມ່ນແນະນຳ ໃຫ້ໃຊ້ເຄື່ອງມື ເວກາ ເພາະໃຊ້ເວລາສັ້ນກວ່າ ແຕ່ວ່າຄວາມລະອຽດ ອາດຈະດ້ອຍກວ່າເລັກນ້ອຍ. ແຕ່ເຖິງຢ່າງໃດກໍ່ຕາມ, ຖ້າຫາກມີ ງົບປະມານພຽງພໍ ເຄື່ອງມືການສະແກນຫາຊ່ອງໂຫວ່ແບບ ທາງການ ຄ້າ (ແບບມີຄ່າໃຊ້ຈ່າຍ - Commercial Vulnerability Scanning Tools) ແລະ ອົງກອນມີບຸກຄະລາກອນວິຊາການສະເພາະດ້ານ ທີ່ພຽບພ້ອມ ແມ່ນແນະນຳໃຫ້ຄວນພິຈາລະນາໃຫ້ນຳໃຊ້, ໂດຍອາໄສ ການຄົ້ນຄວ້ານີ້ປະກອບເປັນແນວທາງໃນການຈັດຕັ້ງປະຕິບັດ ເພື່ອ ໃຫ້ໄດ້ຜົນທີ່ດີ ແລະ ປອດໄພທີ່ສຸດຕໍ່ກັບລະບົບ ເວັບ ແອັບພິເຄຊັນ.

ຕ່ໜ້ານຄົ້ນຄວ້າ-ວິໃຈ ຂ້ານເທິ່ງທັ້ງໝົດ, ຂ້າພະເຈົ້າໃນນາມຜູ້ຄົ້ນຄວ້າວິທະຍາສາດ ຂໍປະຕິຍານວ່າ: ຂໍ້ມູນທັ້ງໝົດທີ່ມີ ໃນບົດຄວາມວິຊາການດັ່ງກ່າວນີ້ ແມ່ນບໍ່ມີຂໍ້ຂັດແຍ່ງທາງຜົນປະໂຫຍດກັບ ພາກສ່ວນໃດ ແລະ ບໍ່ໄດ້ເອື້ອປະໂຫຍດໃຫ້ກັບ ພາກສ່ວນໃດໜຶ່ງ ຫຼື ຜົນປະໂຫຍດຂອງບຸກຄົນໃດໜຶ່ງ. ໃນກໍລະນີມີການລະເມີດໃນຮູບການໃດໜຶ່ງ ຂ້າພະເຈົ້າມີຄວາມ ຍິນດີຮັບຜິດຊອບແຕ່ພຽງຜູ້ດຽວ

๑. ເອກະສານອ້າງອີງ

Abdulghaffar, K., Elmrabit, N., & Yousefi, M. (2023). Enhancing Web Application Security through Automated Penetration Testing with Multiple Vulnerability Scanners.

ADPT (adapt, adept, and adopt). (2022). 10 Security matters that you need to know about OWASP Top 10 Web Application Security. https://www.adpt.news/2022/03/15/10-security-topics-you-should-know-with-owasp-top-10-web-application-security/

Anas, A., Elgamal, S. & Youssef, B. **(**2023). Survey on detecting and preventing web application broken access control attacks. Department of Computer Science, Faculty of Computers and Artificial Intelligence, Cairo University, Cairo, Egypt.

BorntoDev. (2022). 6+1 Solutions for Application Security. https://www.borntodev.com/

Common Weakness Enumeration Organization. (2024). Top 25 Most Dangerous Software Weaknesses. https://cwe.mitre.org/

Douangphachanh, V. (2018). Study of Prevention DDoS (Distributed Denial of Service) Attacks to impact of Internet Server. Department of Computer Engineering and Information, Faculty of Engineering, National University of Laos, Lao PDR.

Hilmi, S. (2020). Evaluation of Open Source Web Application Vulnerability Scanners

Hoffman, A. (2020). Exploittation and Countermeasures for Modern Web Applications. Publisher: O'Reilly MediaInc., 1005 Gravenstein Highway North, Sebastopol, City: California. Page: 117 – 130.

Khemprakhon, Y., & Janawa, S. (2023). Security Enhancement and Performance Optimization for WordPress–based Websites, A Case Study of WorayuthIT Website. Computer Science Department, Faculty of Information, Mahasarakham University, Thailand.

Khiaovongphachanh, V., Chanthavong, K., Mekthanavanh, V., Joundaly S., Luangxasana, K., & Vilathsadavong, L. (2024). Study on Web Application using PHP, JAVA and Python. *Souphanouvong University Journal of Multidisciplinary Research and Development, 10*(1), 142-151. https://doi.org/10.69692/SUJMRD1001241.

Matti, E. (2021). Evaluation of open-source web vulnerability scanners and their techniques used to find SQL injections and cross-site scripting vulnerabilities.

Mongkolsawat, W. (2016). Cyberrange Simulation System Penetration Testing. Department of Technology Information, Thai-Nichi Institute of Technology (TNI).

Morimoto, P. (2022). Vulnerability Assessment (VA) Testing in the Organization by Yourself Easily with OWASP ZAP. https://www.cyfence.com/article/how-to-va-pentest-in-your-organization/

Pasomsup, C. (2020). Role-Based Access Control Framework for Microservice Security. Chulalongkorn University Theses and Dissertations (Chula ETD). 3682. Resource: https://digital.car. chula.ac.th/chulaetd/3682.

Pentest Tools. (2024). A comprehensive benchmark of web application vulnerability scanners 2024. https://pentest-tools.com/benchmarks/web-app-vulnerability-scanners-benchmark-2024.pdf?utm_medium=website&utm_source=blog&utm_campaign=ns-benchmark&utm_content=call-to-action&utm_term=download-the-benchmark.

Program OWASP ZAP System. (2016). chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://csperson.kku.ac.th/ chakchai/Tools/322222_Spring2016/nettool_2_owasp.pdf

Sengudone, P. (2017). Study on behavior of Attacker using Honeypot. Department of Computer Engineering and Information, Faculty of Engineering, National University of Laos, Lao PDR.

Stuttard, D., & Pinto, M. (2008). The Web Application Hacker's Handbook. Publisher: Wiley Publishing, Inc., City: Indianapolis. Page: 217 – 235.

The Open Worldwide Application Security Project (OWASP). (2021). OWASP Top 10. https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/