

UNIVERSIDAD BLAS PASCAL

Ingeniería en Telecomunicaciones



Trabajo Final de Carrera

**Implementación, evaluación y desarrollos funcionales
sobre Artemisa (Open Source back-end honeypot)**

Barirero, E.
Villarroel, M. T.

Director: Dr.-Ing. Do Carmo, R. D.
Asesor: Ing. Olocco, G.
Asesor Metodológico: Dr. Ing. Argüello, J.A.

– 2016 –

Agradecimientos

Al director y desarrollador de Artemisa, Dr.-Ing. Rodrigo D. Do Carmo por contagiarnos su espíritu colaborativo y fomentar la filosofía Open Source. Sin dejar de remarcar su aliento y apoyo para que se continúe el proyecto Artemisa *honeypot*.

Al que fue nuestro director de carrera, Ing. Raúl E. Echegaray, por incentivarnos y promover este proyecto y abrirnos el camino institucional para llevarlo adelante con mayor facilidad. Y por su puesto, al actual director de la carrera de Ingeniería en Telecomunicaciones, el Ing. Victor H. Frison por darle continuidad al mismo.

A los asesores, Ing. Gino Olocco y Dr. Ing. Juan A. Argüello, por sus aportes y valioso tiempo.

Al Ing. Gustavo Biasutto por su colaboración y contribuciones en el trabajo.

A la Lic. Cristina Ferreyra por haber sido una docente y guía fundamental durante los primeros años de nuestra carrera.

A la revisora voluntaria, Ing. Carolina Rey Pizza, por su esfuerzo y empeño porque la entrega final logre la mejor redacción técnica y calidad literaria posible.

A todos los profesores de la carrera de Ing. en Telecomunicaciones de la Universidad Blas Pascal, por habernos guiado y apoyado a lo largo de toda la carrera, y por habernos entregado las herramientas necesarias para finalmente lograr esta meta. Un gran aprendizaje que veo reflejado actualmente en mi vida profesional.

Al Centro de Relaciones Internacionales (CRI) de la Universidad Blas Pascal (UBP) por brindar la oportunidad de acceder a intercambios académicos internacionales estimulando el desarrollo humano, educativo y de la carrera profesional de los estudiantes de la UBP.

Dedicatorias

A mi familia, a quien le debo lo que soy y todo lo que construí con ello.

A mi futura esposa y madre de mi primer hijo Facundo, Carolina, por ser la razón que alegra mi corazón y en reconocimiento a su apoyo, aliento, contención y afecto incondicional. Siempre compañera y cariñosa.

A mi madre, Carmen, por ser siempre un pilar emocional, por su confianza y acompañamiento incondicional y quien me enseño a siempre sonreír y mirar hacia delante en la vida. Y por siempre creer en mí.

A mi padre, Domingo, por contagiarde su espíritu perseverante, a enseñarme que siempre podemos mejorar, y por ayudarme a encontrar la fortaleza necesaria durante todo este proceso.

A mi hermana, Jenifer, por haberme ayudado a través del ejemplo y de su palabra a entender casi “todo” con lo que tuve que convivir, desde las cosas más simples, hasta lo más complejo que uno podría imaginar. Siempre en el momento justo. Ejemplar por donde la mire, mi admiración y orgullo por ella son uno de los motores que me llevó y lleva a lograr mis objetivos en la vida.

A mi abuela, Nucha (Margarita), quien me lleno siempre de vida y energía con su amor sincero. Sin dejar de reconocer su apoyo escolar en los primeros años de mi educación. Y a mi abuela, Amanda, quien nos dejó hace algunos años, pero que siempre va a vivir en mí su nobleza, empatía y bondad. Enormes valores que agradezco haber recibido de ella.

A mis amigos de la vida, por su demostración de afecto y buena energía. Por siempre haber estado cerca a pesar de la distancia y ayudarme a superar todas las dificultades. Por dejarme saber que siempre están y sentirme querido por todos ellos.

A quienes fueron mis compañeros de carrera, en su gran mayoría hoy mis amigos y colegas, por su intercambio y cooperación a través de los años. Durante esta etapa pude aprender muchísimo de cada uno de ellos, y por llenarme de momentos inolvidables.

Exequiel Barrirero

A mis padres, Pedro y Elsa, por formar tan hermosa familia. Por brindarnos toda su vida. Porque sin contagiarme su ansiedad, vivieron atentos cada paso de este camino.

A los más grandes hermanos que alguien pueda anhelar, Germán, Liliana y Cecilia, de quienes siempre recibí todo su apoyo, guía y cariño, y estaré eternamente agradecido.

A Yamile, con quien nos elegimos para ser compañeros en toda la vida. Por acompañarme con infinito amor y paciencia.

Mauro T. Villarroel

Resumen

El objetivo general de este trabajo final de carrera fue el de implementar, evaluar y llevar a cabo desarrollos funcionales sobre el *honeypot* de código abierto Artemisa. A través de un programa de incentivos a proyectos de investigación del ministerio de ciencia y tecnología de la provincia de Córdoba en conjunto con la UBP, se evaluaron requerimientos de *hardware*, y como resultado se adquirieron servidores, dispositivos de *networking*, y teléfonos de Voz sobre IP (VoIP), los cuales nos encargamos de equipar y acondicionar en el laboratorio de redes de la universidad, para la implementación en laboratorio de un escenario de prueba y desarrollo. El proceso fue conformado por la implementación de Artemisa en entornos reales en mesa de prueba (*testbed*), llevando a cabo configuración, mejoras y riguroso *testing* de la plataforma. Luego se continuó con la ejecución de instancias del *honeypot* (Artemisa) accesibles y expuestas a Internet y capturando ataques reales. Paralelamente, se redactó y difundió una carta de participación en foros y comunidades de seguridad informática, adjunta en el Anexo A, para captar ataques auténticos y de fuentes internacionales, favoreciendo así el planteo de los parámetros óptimos de configuración para entornos reales. Este *honeypot* se hace fuerte en entornos de red privada, acusando en tiempo real y con registros detallados la presencia de un atacante interno. Una configuración en el *registrar* de Protocolo de Inicio de Sesión (SIP) del entorno que encamine hacia el *honeypot* todo intento de comunicación contra un agente SIP que no pertenezca al dominio, aumenta el grado de detección de atacantes, tanto internos como externos. En cuanto a la cantidad de llamadas VoIP SIP total (*registrars*) se validó un número importantísimo de llamadas rechazadas, lo que se ve directamente reflejado por las estadísticas del método REGISTER, como así también de los mensajes SIP *401 unauthorized* y *404 not found* presentes en gran número e involucrados en este proceso. Por su número exagerado y contrastando con los *logs* de Asterisk es que se acreditaron como intentos de registro mal intencionados. Integrando todo el tráfico con origen o destino a direcciones Protocolo de Internet (IP) públicas se destaca que Argentina y Alemania fueron los países de mayor protagonismo. En tercer lugar Estados Unidos, y por último Corea, Irlanda y Polonia. Se hizo notorio que permitiría proteger de forma más efectiva el dominio de comunicaciones SIP si se incrementa el alcance del *honeypot* para simular el rol de un servidor de registro SIP, es decir, agregar algunos métodos para que tenga funciones básicas de un *registrar* y así lograr desviar todos los ataques a los que la actual versión de Artemisa no se ve expuesto por su rol de *back-end user-agent*. La interfaz de Llamada de procedimiento remoto de Lenguaje de Marcas Extensibles (XML-RPC) añadida al código fuente agiliza y facilita la configuración, y permite la integración del *honeypot* con otros agentes de la red. Cabe mencionar que en base a la realización y los resultados del trabajo, hoy la UBP cuenta con los dispositivos y recursos necesarios para hacer viable la implementación del proyecto en materias como Redes II, Seguridad Informática, Protocolos y Gestión de Red, dentro de la carrera de Ingeniería en Telecomunicaciones, contribuyendo con experiencia práctica muy valiosa al perfil profesional del egresado.

Palabras claves: honeypot, seguridad informática, SIP, VoIP, voz sobre ip.

Abstract

The overall purpose of this engineering degree final work was to implement, evaluate and develop new functional features over the Open Source honeypot Artemisa. Through an incentive program for research projects from the Science and Technology Department from Cordoba province together with Blas Pascal University, hardware requirements were evaluated, and as a result, servers, networking devices and VoIP phones were acquired. Furthermore, we take care of the installation of all these new equipment and network elements in the university's computer networks lab, for the implementation of a development and experimentation testbed. The first stage of the project consisted in the implementation of Artemisa in testbeds, and during this process, carry out configurations, upgrades and rigorous testing of the platform. Next steps continued with the execution of Artemisa honeypot instances directly exposed and accessible via the Internet in order to capture real attacks. At the same time, a participation letter was drafted and shared in several computer security communities and forums, which is presented in the Annex A, so that authentic and international sources attacks were captured, thus enabling the proposition of optimal settings and configurations for real world scenarios. This honeypot is really strong in private network environments, exposing in real-time and with detailed logs the presence of an internal attacker. Additionally, a configuration in the SIP *registrar* that forwards all communication attempts destined to a user-agent that it's not registered in the domain to the honeypot, increases the level of attacks detection, both internal and external. Then considering the total number of VoIP SIP calls received by the registrars, a significant number of rejected calls were validated, which is directly reflected in the REGISTER method statistics, as well as SIP 401 *unauthorized* and 404 *not found* messages present in a substantial number and involved in this process. Directly related with the excessive amount of the before mentioned events and contrasted with Asterisk logs, is possible to certify that the mentioned traffic were malicious registration attempts. Moreover, when integrating all the network traffic, to or from public IP addresses, it stands out that Argentina and Germany were the countries with greater traffic presence. In third place the United States, and finally Korea, Ireland and Poland. In addition, it was clear that if the scope of the honeypot is increased to simulate the role of a SIP registrar server, that will protect in a more effective manner the SIP domain communications, that is to say, that adding some SIP methods to have basic registrar functions will achieve to evade all the attacks that the current version of Artemisa is not exposed because of its back-end user-agent role. The XML-RPC interface added to the source code speeds up and facilitates the honeypot configuration, it also allows Artemisa integration with other network agents. It should be noted that after the concretion of this project, today UBP has all the necessary resource and devices to make an implementation of the project in courses such as Computer Networks II, Computer Security, Network Protocols and Network Management that conforms the Telecommunications Engineering degree, contributing with valuable practical experience for the graduate student professional profile.

Keywords: honeypot, computer security, information security, SIP, VoIP, voice over ip.

Índice general

| | |
|---|----------|
| 1. Introducción | 1 |
| 2. Materiales y Métodos | 5 |
| 2.1 Estudio exploratorio de carácter bibliográfico de VoIP SIP conceptos básicos, entidades y su seguridad | 7 |
| 2.2 Estudio exploratorio de carácter bibliográfico de señalización SIP (llamada y registro) | 7 |
| 2.3 Estudio exploratorio de carácter bibliográfico de los riesgos y seguridad en VoIP | 7 |
| 2.4 Estudio exploratorio sobre <i>honeypots</i> VoIP específicos y Artemisa <i>honeypot</i> | 7 |
| 2.5 Evaluación de respuestas empíricas y enfoque en desarrollos funcionales del sistema de seguridad Artemisa a partir de su implementación y experimentación en los entornos propuestos | 7 |
| 2.6 Estudio descriptivo estadístico de las características de Artemisa <i>honeypot</i> en entornos reales en mesa de prueba (<i>testbed</i>) | 7 |
| 2.7 Desarrollos Funcionales sobre Artemisa | 7 |
| 3. Resultados y Discusiones | 8 |
| 3.1. Estudio exploratorio de carácter bibliográfico de VoIP SIP sus conceptos básicos, entidades y su seguridad | 8 |
| 3.1.1. Entidades de red SIP | 9 |
| 3.2. Estudio exploratorio de carácter bibliográfico de señalización SIP (llamada y registro) | 11 |
| 3.3. Estudio exploratorio de carácter bibliográfico de los riesgos y seguridad en VoIP | 14 |
| 3.3.1. Servicios de red | 14 |
| 3.3.2. Protocolos de <i>media</i> | 15 |
| 3.3.3. Protocolos de señalización | 15 |
| 3.4. Estudio exploratorio sobre <i>honeypots</i> VoIP específicos y Artemisa <i>honeypot</i> | 20 |
| 3.4.1. Introducción a Artemisa <i>honeypot</i> | 21 |
| 3.4.2. Artemisa <i>honeypot</i> y sus características funcionales | 22 |
| 3.4.3. Modulos e implementación del <i>honeypot</i> | 25 |
| 3.4.4. Experimentación y casos de estudio | 30 |
| 3.5. Evaluación de respuestas empíricas y enfoque en desarrollos funcionales del sistema de seguridad Artemisa a partir de su implementación y experimentación en los entornos propuestos | 35 |
| 3.5.1. Ataques, tipos de ataques de hackers y tipos de hackers | 35 |
| 3.5.2. Modelo de atacante: interno y externo | 37 |
| 3.5.3. Caso 1: dominio de colisión y <i>broadcast</i> compartidos | 38 |
| 3.5.4. Caso 2: dominio de <i>broadcast</i> compartido | 54 |

| | |
|--|------------|
| 3.5.5. Caso 3: dominio público. Colisión y <i>broadcast</i> no compratidos | 55 |
| 3.5.6. Resumen de resultados casos 1, 2 y 3 | 57 |
| 3.6. Estudio descriptivo estadístico de las características de Artemisa honeypot en entornos reales en mesa de prueba | 59 |
| 3.6.1. Implementación <i>testbed</i> UBP | 60 |
| 3.6.2. Implementación <i>testbed</i> Metropolitan | 63 |
| 3.6.3. Configuraciones | 64 |
| 3.6.4. Estudio descriptivo estadístico en entornos reales en mesa de prueba . . | 65 |
| 3.6.5. Estadísticas Artemisa | 68 |
| 3.6.6. Estadísticas <i>registrars</i> SIP | 75 |
| 3.6.7. Estadísticas ataques VoIP | 81 |
| 3.6.8. Estadística tráfico VoIP SIP | 88 |
| 3.6.9. Resumen de resultados | 97 |
| 3.7. Desarrollos Funcionales sobre Artemisa | 99 |
| 3.7.1. XML-RPC Interfaz de Programación de Aplicaciones (API) | 99 |
| 3.7.2. XML-RPC ejemplo de implementación | 103 |
| 3.7.3. Actualización de <i>fingerprints</i> | 115 |
| 3.7.4. Resumen de resultados | 117 |
| 3.8. Conclusiones | 118 |
| Bibliografía | 120 |
| Siglas y Acrónimos | 123 |
| Anexo A: carta de participación en foros y comunidades de seguridad informática | 126 |
| Anexo B: SIP - RFC 3665 | 129 |
| Anexo C: estudio descriptivo estadístico en entornos reales en mesa de prueba | 132 |
| Anexo D: configuraciones componentes <i>testbeds</i> (<i>firewall</i> Iptables, Artemisa, Asterisk, SIP Express Router (SER)) | 142 |
| Anexo E: Artemisa full output ataque SMAP | 161 |
| Anexo F: To-Do list | 165 |

Índice de figuras

| | |
|--|----|
| 3.1. Ejemplo de llamada SIP. | 11 |
| 3.2. Ejemplo de llamada SIP via <i>proxy</i> SIP. | 12 |
| 3.3. Ejemplo de registro y llamada SIP via <i>proxy</i> SIP. [Wei Chen, 2006] | 13 |
| 3.4. Diagrama de ataques VoIP específicos. [Nassar M., 2009] | 15 |
| 3.5. Arquitectura de red. [Ewald T., 2007] | 20 |
| 3.6. Arquitectura del <i>honeypot</i> SIP. [Ewald T., 2007] | 21 |
| 3.7. Implementación de Artemisa <i>honeypot</i> . [Do Carmo R., 2011a] | 22 |
| 3.8. Módulos de Artemisa <i>honeypot</i> . [Do Carmo R., 2011a] | 23 |
| 3.9. Módulos de Artemisa <i>honeypot</i> . [Do Carmo R., 2011a] | 26 |
| 3.10. Integración en tiempo real de los resultados del <i>honeypot</i> . [Do Carmo R., 2011a] | 29 |
| 3.11. Ejemplo de un mensaje de alerta. [Do Carmo R., 2011a] | 30 |
| 3.12. Mensaje INVITE de una traza de ataque. [Do Carmo R., 2011a] | 31 |
| 3.13. La cama de prueba (<i>testbed</i>). [Do Carmo R., 2011a] | 33 |
| 3.14. Diagrama topología <i>testbed testing UBP</i> , caso 1. | 38 |
| 3.15. Llamada regular entre dos agentes SIP finales. | 39 |
| 3.16. Llamada regular entre un <i>softphone</i> y Artemisa <i>honeypot</i> | 40 |
| 3.17. Mensajes SIP enviados a aquellas interfaces que han respondido al mensaje Protocolo de Mensajes de Control de Internet (ICMP). | 41 |
| 3.18. Mensajes SIP enviados cuando el ataque apunta a una dirección IP puntual. | 42 |
| 3.19. Escaneo a toda la red con la herramienta SIP-SCAN. | 44 |
| 3.20. Escaneo a toda la red con SVMAP. | 44 |
| 3.21. Escaneo a toda la red con módulo de escaneo de METAEXPLOIT. | 45 |
| 3.22. Mensajes intercambiados entre SVWAR y el servidor de registro para distintas extensiones. | 46 |
| 3.23. Dialogo completo provocado por el mensaje INVITE. | 47 |
| 3.24. Protección de Artemisa <i>honeypot</i> contra inundación de mensajes. | 49 |
| 3.25. Diálogo mínimo al utilizar Inviteflood. | 51 |
| 3.26. Mensajes generados cuando el objetivo del ataque es un SIP <i>registrar</i> | 51 |
| 3.27. Diagrama topología <i>testbed testing UBP</i> , caso 2. | 54 |
| 3.28. Diagrama topología <i>testbed testing UBP</i> , caso 3. | 55 |
| 3.29. Diagrama topología <i>testbed UBP</i> | 61 |
| 3.30. Diagrama rack <i>testbed UBP</i> | 62 |
| 3.31. Foto rack <i>testbed UBP</i> | 62 |
| 3.32. Diagrama topología <i>testbed Metropolitan</i> | 63 |
| 3.33. Cantidad de paquetes SIP total por tipo de mensaje SIP (instancias Artemisa). . | 69 |
| 3.34. Cantidad de paquetes SIP total por tipo de mensaje SIP (Artemisa <i>self-hosted</i>). . | 70 |
| 3.35. Cantidad de paquetes SIP total por tipo de mensaje SIP (Artemisa <i>hosted-services</i>). | 70 |
| 3.36. Cantidad de paquetes SIP total por tipo de mensaje SIP (instancias Artemisa). . | 71 |

| | |
|---|-----|
| 3.37. Cantidad de paquetes SIP total por tipo de método SIP (instancias Artemisa) | 73 |
| 3.38. Cantidad de paquetes SIP total por tipo de método SIP (Artemisa <i>self-hosted</i>) | 73 |
| 3.39. Cantidad de paquetes SIP total por tipo de método SIP (Artemisa <i>hosted-services</i>) | 74 |
| 3.40. Cantidad de paquetes SIP total por tipo de método SIP (instancias Artemisa) | 74 |
| 3.41. Cantidad de llamadas VoIP total (instancias Artemisa) | 75 |
| 3.42. Cantidad de llamadas VoIP total (instancias Artemisa) | 75 |
| 3.43. Cantidad de paquetes SIP total por tipo de mensaje SIP (<i>self-hosted registrars</i>) | 77 |
| 3.44. Cantidad de paquetes SIP total por tipo de mensaje SIP (<i>self-hosted registrars</i>) | 78 |
| 3.45. Cantidad de paquetes SIP total por tipo de método SIP (<i>self-hosted registrars</i>) | 80 |
| 3.46. Cantidad de paquetes SIP total por tipo de método SIP (<i>self-hosted registrars</i>) | 80 |
| 3.47. Cantidad de llamadas VoIP SIP total (<i>registrars</i>) | 81 |
| 3.48. Cantidad de paquetes SIP total por tipo de ataque (instancias Artemisa) | 82 |
| 3.49. Cantidad de paquetes SIP total por tipo de ataque (Artemisa <i>self-hosted registrars</i>) | 83 |
| 3.50. Cantidad de paquetes SIP total por tipo de ataque (Artemisa <i>hosted-services registrars</i>) | 83 |
| 3.51. Cantidad de paquetes SIP total por tipo de ataque (instancias Artemisa) | 83 |
| 3.52. Cantidad de paquetes SIP por tipo ataque (Asterisk Metropolitan <i>registrar</i>) | 85 |
| 3.53. Los Registros de Internet Regionales (RIR) alocan direcciones IP públicas. [Hundley, 2009] | 89 |
| 3.54. Estadísticas por dirección IP de origen pública/privada (Metropolitan <i>testbed</i>) | 90 |
| 3.55. Estadísticas por dirección IP de origen por país (Metropolitan <i>testbed</i>) | 90 |
| 3.56. Estadísticas por dirección IP de destino pública/privada (Metropolitan <i>testbed</i>) | 92 |
| 3.57. Estadísticas por dirección IP de destino por país (Metropolitan <i>testbed</i>) | 92 |
| 3.58. Estadísticas por dirección IP de origen pública/privada (UBP <i>testbed</i>) | 94 |
| 3.59. Estadísticas por dirección IP de origen por país (UBP <i>testbed</i>) | 94 |
| 3.60. Estadísticas por dirección IP de destino pública/privada (UBP <i>testbed</i>) | 96 |
| 3.61. Estadísticas por dirección IP de destino por país (UBP <i>testbed</i>) | 96 |
| 3.62. Diagrama topología XML-RPC cliente/servidor implementación | 104 |
| 3.63. Diagrama topología XML-RPC cliente/servidor <i>enumeration/Denegación de Servicio (DoS)</i> | 105 |
| 3.64. Diagrama topología XML-RPC cliente/servidor luego de su ejecución | 110 |
| 3.65. Wireshark I/O Graph Statistics w/ SIP filters (UBP SER <i>registrar</i>) | 138 |
| 3.66. Wireshark I/O Graph Statistics w/ SIP filters (UBP SER registered Artemisa) | 140 |
| 3.67. Wireshark I/O Graph Statistics w/ SIP filters (UBP SER registered Artemisa) | 141 |

Índice de tablas

| | | |
|-------|--|-----|
| 3.1. | Resultado de las pruebas. [Do Carmo R., 2011a] | 34 |
| 3.2. | Características y especificaciones topología <i>testbed testing</i> UBP, caso 1. | 38 |
| 3.3. | Características y especificaciones topología <i>testbed testing</i> UBP, caso 2. | 54 |
| 3.4. | Características y especificaciones topología <i>testbed testing</i> UBP, caso 3. | 56 |
| 3.5. | Tabla resumen resultados caso 1. | 57 |
| 3.6. | Tabla resumen resultados caso 2. | 57 |
| 3.7. | Tabla resumen resultados caso 3. | 58 |
| 3.8. | Características y especificaciones topología <i>testbed</i> UBP. | 60 |
| 3.9. | Características y especificaciones topología <i>testbed</i> West Metropolitan. | 64 |
| 3.10. | Estadísticas por dirección IP de origen (Metropolitan <i>testbed</i>). | 89 |
| 3.11. | Estadísticas por dirección IP de destino (Metropolitan <i>testbed</i>). | 91 |
| 3.12. | Estadísticas por dirección IP de origen (UBP <i>testbed</i>). | 93 |
| 3.13. | Estadísticas por dirección IP de destino (UBP <i>testbed</i>). | 95 |
| 3.14. | Cantidad de paquetes SIP total por tipo de mensaje SIP (Artemisa). | 133 |
| 3.15. | Cantidad de paquetes SIP total por tipo de método SIP (Artemisa). | 133 |
| 3.16. | Cantidad de llamadas VoIP total (Artemisa). | 133 |
| 3.17. | Cantidad de paquetes SIP total por tipo de mensaje SIP (<i>registrars</i>). | 134 |
| 3.18. | Cantidad de paquetes SIP total por tipo de método SIP (<i>registrars</i>). | 134 |
| 3.19. | Cantidad de llamadas VoIP SIP total (<i>registrars</i>). | 134 |
| 3.20. | Cantidad de paquetes SIP total por tipo de ataque. | 134 |
| 3.21. | Cantidad de paquetes SIP por tipo ataque (Asterisk Metropolitan). | 135 |
| 3.22. | Wireshark SIP Statistics with filter (UBP SER Registrar). | 137 |
| 3.23. | Wireshark VoIP calls (UBP SER <i>registrar</i>). | 138 |
| 3.24. | Wireshark SIP Statistics with filter (UBP SER registered Artemisa). | 139 |
| 3.25. | Wireshark VoIP Calls (UBP SER registered Artemisa). | 140 |

1

Introducción

En la actualidad, existen aproximadamente 650 millones de usuarios de Internet en el continente Americano, y alrededor de 3345 millones a nivel mundial. [Internet World Stats, 2015] Sin embargo, gran cantidad de usuarios, administradores o proveedores conectados a la red no tienen un conocimiento claro de las debilidades o vulnerabilidades de seguridad a la que su información está expuesta. La gran variedad de protocolos, tecnologías y servicios que sustentan Internet, hacen que la posibilidad de ejecutar un ataque sea muy elevada, debido a que el concepto de seguridad no fue un elemento primordial del diseño inicial de Internet a fines de la década del 60' y comienzos de la del 70'.

En las redes corporativas en estado de producción, es una tarea muy compleja llevar un registro oficial, específico y detallado sobre el número de incidentes o ataques de seguridad que se presentan. Saber identificar los eventuales riesgos a los que se pueda ver expuesta dicha información es imprescindible para asegurar la integridad y el correcto uso que a ésta se le puede dar. [Espinoza M. P., 2009] Principalmente, si la información en la que nos concentramos en proteger son las comunicaciones telefónicas. La seguridad es un requisito indispensable en el desarrollo confiable de una red de VoIP. Infortunadamente, asegurar VoIP es más complejo que asegurar una red de circuitos conmutados de telefonía tradicional. Con la voz como un servicio sobre la red IP (Internet), esta hereda ambas ventajas y desventajas de dicha red. A partir de su implementación a mediados de los noventa, han comenzado a desarrollarse diferentes tipos de ataques haciéndolo un servicio endeble a escuchas ilegales, falsificación de identidad, DoS, virus y gusanos, entre otros. [Collier M. D., 2006]

El *exploit* más común contra Ramales Privados de Comutación (PBX) que fueron comprometidas es *toll fraud* (fraude en tasas o tarifas telefónicas), utilizando el sistema telefónico de alguien más para realizar llamadas de larga distancia. En segundo lugar se encuentra forzar a la PBX para llamar a números *premium* controlados por el atacante para cobrarles por minuto, por lo que las empresas que han visto sus PBXs atacadas, serán cargadas con estas facturas. Un reporte de Sipera Viper Labs argumenta que frente a ambos tipos de fraude, las organizaciones frecuentemente no tienen la posibilidad de realizar el descargo de estos costos debido a que no son capaces de proveer evidencia suficiente de que los cargos facturados son un error.

Cisco también notó la prevalencia del *vishing*, *telephone-based phising* (*phising* basado en telefonía), donde los llamantes pretenden ser de bancos, del gobierno u otras instituciones y buscan hacer que las víctimas engañadas entreguen sus datos, como su número de seguridad social o números de tarjetas de crédito. Cisco en uno de sus reportes del 2011, acerca de seguridad en Tecnología de la Información (IT), expresó que el abuso en VoIP está en alza y listo para un mayor crecimiento. Dentro de la categorización de diferentes clases de ataques pone a VoIP entre aquellos con mayor potencial y dentro del grupo que Cisco llama como

“rising stars” (estrellas en ascenso) que incluyen *exploits* web y troyanos para robo de datos y lavado de dinero. Por su parte, Sipera Viper Labs notó que los porcentajes pico de ataques contra VoIP ascendieron rutinariamente a un nivel de 30 % con respecto a años anteriores, y que estos picos continúan. En investigaciones previas, habían encontrado que estas acciones mal intencionadas dirigidas a VoIP eran solo del 10 % con respecto al total de ataques analizados. [Greene T., 2011]

En cuanto a los ataques DoS, más específicamente los ataques de Denegación de Servicio Distribuida (DDoS), son una amenaza creciente a los servicios de VoIP. Tal fue el caso de un DDoS en 2011 que derribó totalmente el servicio de VoIP suministrado a miles de clientes por la compañía estadounidense TelePacific Communications. El ataque DDoS masivo provino fugazmente desde Internet en forma de una inundación de solicitudes de registro VoIP inválidas. El ataque resultó en una mayúscula degradación y pérdida de servicio por una serie de días y le costó a la compañía cientos de miles de dólares en crédito de clientes. El vicepresidente de ingeniería de red de TelePacific expresó que si bien existen multitud de escaneos diarios contra su red, y que ataques a bajo nivel pueden ocurrir hasta dos veces por día, este proveedor de servicios nunca antes había visto lo que ocurrió en este período del 2011, cuando el nivel normal de 34 millones de solicitudes de registro de tráfico SIP para conexiones VoIP repentinamente se disparó por encima de 69 millones e inundó su sistema, lo cual resultó en imposibilidad a la hora de realizar llamados. Otro dato muy relevante que detalló es que contactaron a la Oficina Federal de Investigación (FBI) para reportar el ataque, pero encontraron que TelePacific simplemente no disponía de toda la información necesaria que el FBI requería para ser capaz de analizar el evento y llevar adelante el caso. Consideraron que no se encontraban preparados defensivamente y que no pudieron capturar suficiente información, por lo cual rectificaron esta situación con nuevos *data-capture systems* (sistemas de captura de datos). [Messmer E., 2011]

De acuerdo a un informe presentado por el Computer Security Institute, se observa un incremento considerable de ataques año a año y una pérdida de cerca de 20 millones de dólares únicamente por infección de virus, sin considerar las pérdidas ocasionadas por otros tipos de ataques. Este incremento ha venido de la mano con una evolución de las técnicas y herramientas utilizadas por los atacantes. En la actualidad, una gran parte de las organizaciones poseen sus sistemas informatizados e íntimamente relacionados a Internet. Siendo esta última estructura fundamental para su diario funcionamiento, es por ello que la seguridad informática es sin duda de gran relevancia. Principalmente, ya que tanto la disponibilidad como la confiabilidad de la información son un tema crítico dentro de la organización. [Computer Security Institute, 2011] Con los datos de utilización de Internet anteriormente descritos, podríamos acotar que si tan sólo el 1 por ciento de la población realiza ataques mediante la red, se tiene que existen casi 33 millones de posibles atacantes, una cifra considerablemente significativa.

Cabe mencionar que desde sus comienzos y por un prolongado período, la seguridad informática ha sido meramente defensiva. Esta consistía principalmente en defender la infraestructura de la información, detectar fallos y reaccionar ante estos. Debido al cambio que está surgiendo en Internet, el cual se debe principalmente al gran avance tecnológico y a la evolución de las técnicas de los usuarios maliciosos, se requiere un nuevo modelo que permita ampliar el concepto de seguridad. [De Laet G., 2005]

Dentro de este nuevo modelo aparece el concepto de proactividad dentro de la seguridad informática, en el cual no se dejan de lado herramientas antes utilizadas, sino que junto a éstas aparecen nuevas técnicas que permiten detectar e incluso prevenir nuevos ataques. Una de estas nuevas técnicas es el *sinkhole* (sumidero), la cual consiste en eliminar el tráfico sospechoso de la red. Además, mediante el uso de señuelos se puede obtener información valiosa sobre las amenazas, las cuales pueden ser observadas, analizadas y monitorizadas previniendo nuevos

ataques. El *sinkhole* es implementado principalmente por los proveedores de Internet, lo que al sumarle señuelos produce una interesante aplicación, ya que el tráfico malintencionado, no sólo se elimina, sino que permite realizar estudios estadísticos, los que a su vez facilitan identificar cuáles son las técnicas más utilizadas en los ataques. En torno a esta nueva forma de ver la seguridad, aparecen los *honeypots* y *HoneyNets*. Estos, pretenden conocer al enemigo, conocer sus técnicas y formas de actuar. [Trapp Flores J. W., 2009]

Los *honeypots* aún son una táctica subutilizada. Todos los ataques, ya sean manuales o automatizados, tienen un componente exploratorio. Cuando los hackers o virus se encuentran probando un sistema o una red generalmente pasan desapercibidos. A menos que produzcan un *crash* (colapso) o recarga del sistema, las chances de detección son muy bajas. Un *honeypot* es un sistema que detecta actividad no esperada o sospechosa mediante la creación de *targets* (objetivos) falsos. En una red, por ejemplo, un *honeypot* simple puede reservar y hacer uso del espacio de direccionamiento IP no ocupado. Luego, si alguien intenta acceder a esta dirección IP que no debería estar en uso, se puede generar una alerta. De forma similar, un *honeypot* basado en puertos podría responder a solicitudes en puertos de Protocolo de Control de Transporte (TCP) no utilizados, creando la ilusión de servicios de red. Computadoras enteras, o incluso redes de computadoras, pueden ser creadas para atraer y tentar atacantes. Los *honeypots* pueden incluso automatizar la prevención de intrusiones al bloquear temporalmente direcciones IP, actuando así como trampas activas para los atacantes.

Lo que aún sorprende, es que solo unos pocos proveedores ofrecen soluciones *honeypot-like* (tipo *honeypot*) dentro de sus estándares como una característica de seguridad embebida en sus productos, lo que hace más valioso a Artemisa y a lo desarrollado en este Trabajo Final de Carrera (TFC). Equipos de red (*routers* y *switches*) podrían ofrecer redes *honeypot* fantasma que generan alertas preventivas. Softwares de virtualización podrían crear *data centers honeypot* fantasma. Un Proveedor de Servicios de Internet (ISP) podría implementar *honeypots* en espacios de direccionamiento no reservados. Más aún, *honeypots* sofisticados tendrían la capacidad de atraer y emboscar atacantes creando la ilusión de que han logrado violar la seguridad de la plataforma y permitiendo que la intrusión escale, aprovechando este proceso para obtener la mayor cantidad de información posible sobre el perfil del atacante.

Los *honeypots* nos dan una oportunidad para colocar una trampa es esos vacíos que poseen la mayoría de los Sistema Operativo (OS), aplicaciones, infraestructuras de red y entornos VoIP, haciendo más riesgosas para el atacante sus incursiones exploratorias y que sea más probable poder detectarlas. Un señuelo de cualquier Agente de Usuario (UA) o componente de la red puede retrasar al usuario mal intencionado dándonos la delantera para comenzar la detección, identificación y frustración de tal intrusión. Así es como se atrapan hackers con *honey* (miel). [Antonopolous A. M., 2011]

VoIP y SIP se están estableciendo como participantes protagónicos en el campo de las comunicaciones multimedia sobre IP, respaldados por su bajo costo y accesible manejo. Sin embargo, los aspectos de seguridad no han sido completamente logrados. Se presenta de forma específica a Artemisa, como la implementación de un *honeypot* de código abierto orientado a VoIP, basado en SIP. El *honeypot* esta diseñado para conectarse a un dominio corporativo de VoIP como un *back-end user-agent* o agente-usuario en plano implícito, con el fin de detectar actividad maliciosa en una etapa temprana. Asimismo, el *honeypot* puede jugar un rol fundamental en el ajuste de las políticas de seguridad del dominio corporativo en producción donde está actuando en tiempo real. Se apunta a motivar el desarrollo e implementación de este tipo de *honeypots* a gran escala, capturando y rastreando ataques. Se ha testeado la capacidad de Artemisa para sobrellevar una serie ataques SIP conocidos. [Do Carmo R., 2011a]

Este sistema de seguridad consiste en diseñar e implementar un prototipo de *honeypot*, en

nuestro caso el ya desarrollado Artemisa, el cual permitirá mejorar considerablemente el nivel de seguridad presente en la red de VoIP SIP. El antes mencionado, permite asegurar las comunicaciones telefónicas en una red corporativa en producción, la cual permitirá analizar, controlar y monitorear la red ante oportunos ataques, tanto internos como externos a la organización en cuestión.

Además, permitirá utilizar los datos obtenidos como un valioso recurso de investigación y facilitar estudios de *remote penetration* (penetración remota) y *vulnerability assessment* (evaluación de vulnerabilidades), aplicados a ámbitos corporativos en producción. En los cuales prima la disponibilidad y confiabilidad para un estable funcionamiento de la infraestructura; garantizando así, la seguridad y estabilidad de los sistemas en los que se basa una organización, enfocados a sus comunicaciones de voz.

Sin dejar de mencionar que el objetivo general de este trabajo final de carrera fue el de implementar, evaluar y llevar a cabo desarrollos funcionales sobre el *honeypot* de código abierto Artemisa.

2

Materiales y Métodos

1. Estudio exploratorio de carácter bibliográfico de VoIP SIP sus conceptos básicos, entidades y su seguridad:

Para este estudio se consultó:

- [Nassar M., 2009]

2. Estudio exploratorio de carácter bibliográfico de señalización SIP (llamada y registro):

Para este estudio se consultó:

- [Johnston A., 2003]
- [Wei Chen, 2006]

3. Estudio exploratorio de carácter bibliográfico de los riesgos y seguridad en VoIP:

Para este estudio se consultó:

- [Johnston A., 2003]
- [Nassar M., 2009]
- [Wheeler D., 2004]

4. Estudio exploratorio sobre *honeypots* VoIP específicos y Artemisa *honeypot*:

Para este estudio se consultó:

- [Acunetix®, 2015]
- [Digium®, 2015]
- [Do Carmo R., 2011a]
- [Do Carmo R., 2011b]
- [Ewald T., 2007]
- [Microsoft®, 2015]
- [Neira Ayuso P., 2014]
- [Reingold E., 2009]
- [Symantec TM® TechNet, 2011]

5. Evaluación de respuestas empíricas y enfoque en desarrollos funcionales del sistema de seguridad Artemisa a partir de su implementación y experimentación en los entornos propuestos:

- a) Se estudiaron los ataques, tipos de ataques de hackers y tipos de hackers.
- b) Se plantearon modelo de atacante: interno y externo. A partir de estos, se definieron tres casos de pruebas de penetración.
- c) Se definió la topología física y lógica de la red acorde a lo planteado para cada caso.
- d) Se instalaron dos servidores físicos en el laboratorio de redes de la UBP, en los cuales se virtualizaron los entornos en mesa de prueba mediante VMWare y Virtual-Box, corriendo los OS GNU/Linux Debian, Ubuntu y Centos como invitados.
- e) Se realizaron pruebas de uso ideal del sistema de telefonía para captar y lograr un mejor entendimiento del funcionamiento de Artemisa, SER y el *registrar* Asterisk frente a una situación libre de ataques.
- f) Se implementó y se realizó prueba de penetración sobre Artemisa en dominio de colisión y *broadcast* compartidos (con *hub*).
- g) Se implementó y se realizó prueba de penetración sobre Artemisa en dominio de *broadcast* compartidos (con *switch Ethernet*).
- h) Se implementó y se realizó prueba de penetración sobre Artemisa en dominio público. Colisión y *broadcast* no compartidos (con *router* y *switch Ethernet*).
- i) Se presentó en diagramas de flujo el tráfico de paquetes de los agentes involucrados en el ataque, y se capturaron las respuestas por consola y/o archivos *logs* de Artemisa, *proxy SIP*, *registrar* y de cada herramienta en cuestión.
- j) Se desarrolló analíticamente lo observado en cada ataque para ayudar a entender a lo que se enfrentan los dispositivos SIP por cada herramienta de penetración.
- k) Se resumió y tabuló el análisis de los tres casos de prueba de penetración.

6. Estudio descriptivo estadístico de las características de Artemisa *honeypot* en entornos reales en mesa de prueba (*testbed*):

Se realizó un extensivo relevamiento, de un período de aproximadamente seis meses (01/11/2012 - 01/06/2013), que permitió apreciar tendencias y lograr inferir conclusiones respecto a las variables analizadas. Se detallan los pasos de este proceso debajo:

- a) Se gestionó una conexión a Internet independiente de la proporcionada por la UBP para exponer Artemisa a una red pública y aumentar la precisión de los resultados.
- b) Se implementó *testbed* UBP.
- c) Se implementó el *testbed* Metropolitan fuera de las instalaciones de la UBP con un segundo proveedor de Internet.
- d) Se crearon cuentas en un SIP *registrar/proxy* externo, *iptel.org*, para plantear también un escenario de tipo *hosted-services*.
- e) Se presentaron las configuraciones de los *testbed*.
- f) Se configuró *Iptables* como *firewall* y *port filtering* en los *routers* de frontera hacia internet en ambos *testbeds*.

- g) Se programaron Bash scripts para automatizar la ejecución de todos los agentes de la red en ambos *testbeds*.
- h) Se almacenaron durante el período de prueba de aproximadamente seis meses en un Sistema de Archivos de Red (NFS) centralizado, *logs* y capturas de tráfico (Tcpdump) para Artemisa, SIP *proxies* y Asterisk. Y con herramientas propias de Bash se guardó la Salida de Error Estándar (STDERR) & Salida Estándar (STDOUT) de los componentes de la arquitectura SIP antes mencionados.
- i) Se utilizó Wireshark y scripts en Python para procesar capturas, *logs* y archivos generados durante el período de exposición. En base a sus resultados, se realizó estudio descriptivo estadístico de los entornos reales en mesa de prueba.

7. Desarrollos Funcionales sobre Artemisa:

Se desarrolló una API implementando el módulo *SimpleXMLRPCServer* de la librería estándar de Python. Permitiendo de esta manera crear un servidor XML-RPC. El cual utiliza en su código fuente el protocolo de Internet, como también el módulo de sistema *socket* embebido en la mayoría de los OS.

- a) Se desarrolló la API XML-RPC implementando el módulo *SimpleXMLRPCServer* de la librería estándar de Python.
- b) Se elaboró un ejemplo de implementación XML-RPC como caso de estudio. Consta de un cliente XML-RPC en Python en combinación de Bash *scripting*.
- c) Se actualizaron *fingerprints* de la base de firmas de Artemisa, con aparejado ejemplo de detección de las mismas.

3

Resultados y Discusiones

3.1. Estudio exploratorio de carácter bibliográfico de VoIP SIP sus conceptos básicos, entidades y su seguridad

VoIP es una reciente y muy prometedora tecnología de comunicaciones por voz. Su definición, claramente sobrepasa la simple transmisión de paquetes conmutados por redes IP para alcanzar otro tipo de aplicaciones y asegurar dinamismo, movilidad e innovar posibles aplicaciones. Debido a su demostrada flexibilidad y conveniencia económica por sobre la Red de Telefonía Pública Conmutada (PSTN), se ha visto aumentar su penetración tanto en el ámbito corporativo como en usuarios finales. Basada en un serie de estándares y protocolos propietarios, los productos VoIP no solo están limitados a equipos de usuario final, sino que también dispositivos de procesamiento de llamadas, señalización, *media gateways*, *proxies* y *firewalls*.

Será primordial remarcar que voz sobre IP, más allá de heredar todas las fortalezas del protocolo de Internet, también lo ha hecho con los problemas de seguridad asociados a este protocolo, y aún más, otras desventajas específicas en relación a VoIP. Vulnerabilidades de señalización y de protocolos de *media* (*media protocols*) pueden ser explotados para escuchar secretamente (*eavesdropping*), uso fraudulento (falsificación de identidad) y ataques de DoS. Por ejemplo VoIP será atractivo para publicistas que pueden muy fácilmente implementar un centro de llamadas automáticas por Internet a un bajo coste. En ocasiones el usuario final no deseará recibir este tipo de llamadas considerándolas como *spam* o más precisamente Spam sobre Telefonía IP (SPIT).

Las consideraciones en cuanto a seguridad para voz sobre IP han sido un tema que atrae una gran audiencia en el ámbito industrial, gubernamental y académico. Es preciso considerar que la seguridad en VoIP se ve limitada por sus especiales características en torno a lo que concierne a calidad de servicio, traducción de direcciones y establecimiento dinámico de llamadas a través de *firewalls*. Por su parte, políticas de redes de datos convencionales como cifrado (*encryption*), autenticación e integridad de datos son altamente recomendables. Sin embargo, estas no se adecúan por completo a las implementaciones de las arquitecturas de voz sobre IP, las cuales son de gran escala, y de naturaleza dinámica y abierta. Dichas políticas son incompatibles con Traductor de Direcciones de Red (NAT), generan un incremento de retardos y necesitan una llave de infraestructura distribuida. Es por lo anterior, que debe ponerse mayor énfasis en políticas de defensa de segunda línea. Como por ejemplo defensa proactiva y preventiva, detección de intrusos y mecanismos de monitoreo para la mitigación o prevención de ataques.

Trabajos relevantes se han realizado para detección de intrusos tanto en dispositivos terminales (*host*) como redes de datos. Estos han sido llevados adelante por la comunidad corporativa industrial y por la académica. Estos desarrollos fueron enfocados para lograr la detección

de intrusos en protocolos de capa de transporte, red y aplicación. Sin embargo, podría considerarse que estas aproximaciones en cuanto a seguridad para VoIP aún se encuentran en una fase preliminar. Este trabajo final de carrera se concentra principalmente en presentar de forma teórica y práctica una solución de detección de intrusos para un entorno VoIP SIP. Este análisis consta de la implementación del *honeypot* Artemisa, el cual actúa dentro del entorno de voz sobre IP como *back-end user-agent* (agente-usuario en pleno implícito), con el fin de detectar actividad maliciosa en una etapa temprana. Esta puesta a prueba y en funcionamiento del aplicativo de seguridad Artemisa nos facilitará exponer este nuevo modelo y arquitectura para ejecutar una defensa proactiva, monitoreo y detección de intrusos en redes VoIP. [Nassar M., 2009]

3.1.1. Entidades de red SIP

En su tesis postdoctoral [Nassar M., 2009] refiere a los diferentes clientes y servidores SIP como se introducen a continuación:

- Agente de usuario (*user-agents*): un UA establece o cierra una sesión de *media* (audio) con otros UAs y debe mantener el estado de las llamadas. Un UA está compuesto por dos aplicaciones; el Agente de Usuario Cliente (UAC) para iniciar solicitudes a otros UAs y recibir sus respuestas, y el Servidor de Agente de Usuario (UAS) para procesar las solicitudes recibidas desde otros UAs y generar las respuestas asociadas. Un UA debería publicar sus capacidades y características de *media* en el cuerpo de la solicitud SIP permitiendo de esta manera a otros agentes de usuario aprender de él y establecer la comunicación en base a estas.
- Agente de presencia (*presence agents*): un agente de presencia notifica a sus suscriptores de cierta información de presencia. Fuentes de esta información pueden ser UAs SIP, los cuales se registran al agente de presencia y publican sus estados como también otras entidades no SIP.
- Agente de Usuario Espalda a Espalda (B2BUA): este oculta o enmascara el número de UAs como si el originador de la solicitud fuera de él. Cuando un B2BUA recibe una solicitud de uno de sus UAs, este reemplaza tanto la información de señalización como la de *media* y reenvía una solicitud reformulada. Por su parte, la operación inversa es llevada a cabo en la dirección opuesta cuando se reciben las respuestas a la solicitud previamente realizada. Así también, el B2BUA transmite la sesión de *media* para sus UAs. B2BUAs son utilizados para implementar servicios anónimos, PBX y emulación de servicios PSTN (PSTN Emulation Service).
- Puertas de enlace (*gateways*): una interfaz de *gateway* SIP hacia otra red, por ejemplo ISUP (Parte de Usuario ISDN), Señalización Asociada a Circuito (CAS) y H.225 puede ser definida como un agente de usuario que actúa en representación o como traductor de otro protocolo. Un *gateway* cierra el camino de señalización pero no necesariamente el camino o vínculo de *media* que fue establecido (por ejemplo, en el caso de un *gateway* SIP/H.323, el punto de finalización SIP (*SIP endpoint*) y el *endpoint* H.323 intercambian la sesión de *media* directamente o mediante un *gateway*). En un contexto MGCP, es decir un Controlador de Puerta de Enlace de Medios (MGC) controla uno o varios Puerta de Enlace de Medios (MG). El MGC controla la apertura y cierre de *media endpoints* en el MG mediante el protocolo MGCP. Los protocolos no SIP (*Non-SIP protocols*) entre los cuales Telefonía ruteada sobre IP (TRIP) y Protocolo de Registro de Puerta de Enlace de Telefonía (TGREP) proveen información de ruteo a los *proxy* SIP y a los UAs con el fin

de encontrar su ruta hacia los GWs cuando necesitan interconectarse con otras redes de telefonía.

- Servidores *proxy* (*proxy servers*): facilita el encaminamiento de mensajes SIP para los remitentes, o más precisamente los usuarios o agentes de usuario de la llamada telefónica. Está permitido para estos, realizar una pequeña modificación en sus solicitudes mientras se preserva la comunicación transparente de extremo a extremo (*end-to-end*) entre el remitente y el receptor. Un servidor *proxy* tienen acceso a una o varias base de datos con el fin de consultar información de presencia o ubicación. A su vez, un *proxy server* puede hacer uso de Sistema de Nombre de Dominio (DNS) para resolver nombres de las solicitudes SIP. Casos especiales de DNS son ENUM, el cual realiza la asociación de números telefónicos globales (E.164) e Identificadores de Recursos Uniformes (URI) SIP en conjunto con la Autoridad Puntero de Nombramiento (NAPTR) y registros DNS, con el fin de interconectar la PSTN. Complementariamente, DNS SRV (*service locator*) que ayuda a localizar SIP *proxies* dentro de un dominio. Se tienen dos tipos de servidores *proxy*: con monitoreo o seguimiento de estado (*stateful*) y sin monitoreo o seguimiento de estado (*stateless*). Una transacción *proxy stateful* debe mantener el estado de las transacciones que se están llevando adelante mientras que por ejemplo, un *proxy stateless* nunca debería retransmitir un mensaje perdido al no mantener un seguimiento de los mensajes. Un *proxy stateful* debe iniciar un reloj (*timer*) siempre que una solicitud es reenviada y esperar por su respuesta. Si no se recibe dicha respuesta dentro de un período de tiempo preestablecido, retransmite la solicitud.
- Servidores de redirección (*redirect servers*): no encamina o enruta ningún mensaje. En cambio, responde con un mensaje de redirección conteniendo información de ubicación.
- Servidor de registro (*registrar server*): es responsable de procesar mensajes de registro SIP (SIP REGISTER). Este crea asociaciones temporarias entre un número telefónico IP, también conocidos como Registros de Direcciones (AoR) y direcciones IP (o alias) del *registering device* (dispositivo de registro). Un servidor de registro, autentica, y registra un dispositivo utilizando un esquema de llave secreta compartida orientado a Protocolo de Transferencia de Hipertexto (HTTP) o HTTP-like. Un UA puede utilizar REGISTER messages (mensajes de registro) especiales para obtener una lista de los registros actuales, limpiar todos los registros o agregar un alias a sus registros.

3.2. Estudio exploratorio de carácter bibliográfico de señalización SIP (llamada y registro)

El lector podrá interpretar un ejemplo de establecimiento de sesión entre dos agentes de usuarios SIP en la Figura 3.1. La sesión comienza con una solicitud INVITE enviada desde el usuario que origina la llamada (*caller*) al destinatario de la misma (*callee*). Luego de esto, el receptor responde automáticamente con respuestas informativas, primero *TRYING* (intentando), inmediatamente después *RINGING* (timbrando). Una vez que la llamada fue atendida, el receptor envía una respuesta satisfactoria *200 OK*. El llamante confirma esta respuesta mediante el envío de un mensaje de confirmación (ACK), inmediatamente después comienza la sesión de datos (*media session*). Puede verse un comportamiento similar pero con la intervención de una entidad SIP intermedia como es el SIP proxy en la Figura 3.2.

El Protocolo de Descripción de Sesión (SDP) es el responsable por la caracterización de los datos (*media*) y de la negociación entre ambas partes. El usuario que comienza la llamada podría insertar un cuerpo SDP en el mensaje INVITE mientras que el receptor lo hace en el mensaje de respuesta *200 OK*. De esta manera estableciendo con anterioridad al comienzo del flujo de datos el tipo de codificación (ej.: *U-law* o *A-law encoding*) que será utilizado en esta llamada. Las partes se pondrán de acuerdo al aceptar el protocolo de codificación soportado por ambas, para luego comenzar la comunicación bilateral respetando la misma. Por su parte, el emisor elegirá el puerto Protocolo de Transporte en Tiempo Real (RTP), por ejemplo: puerto RTP de origen 10604, en el cual recibirá la sesión de datos y el usuario que fue llamado podría elegir el puerto 32706. Finalmente ambos interlocutores deberán enviar una solicitud de cierre BYE. En respuesta a la misma el extremo opuesto responderá con una confirmación *200 OK*.

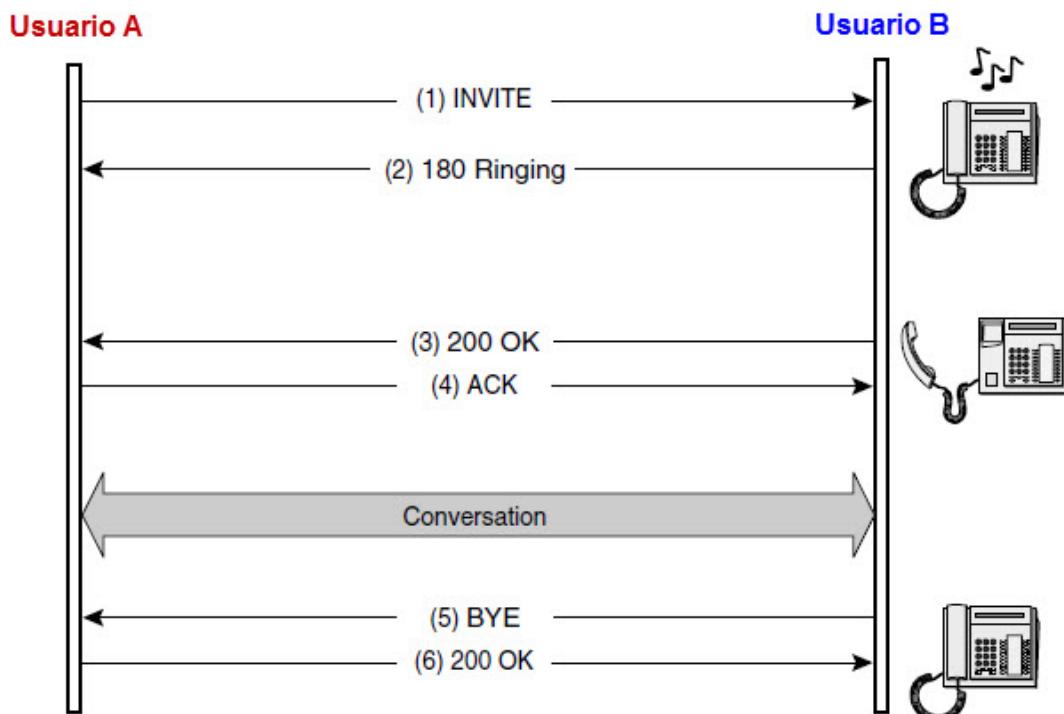


Figura 3.1: Ejemplo de llamada SIP.

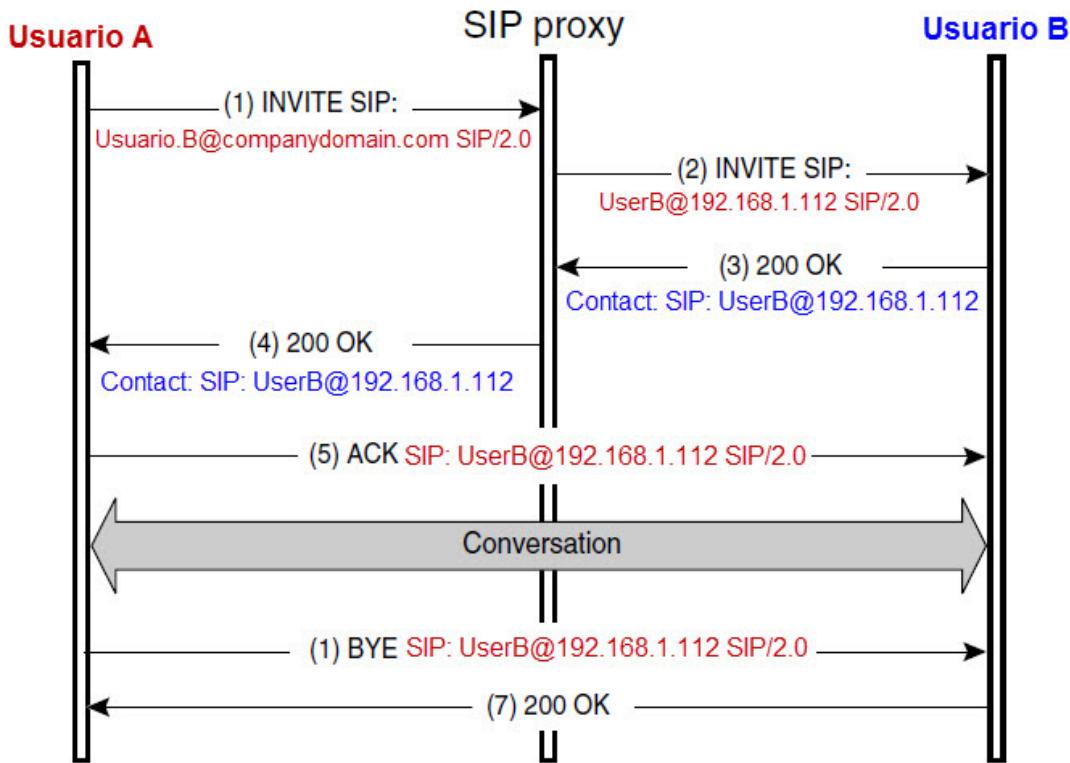


Figura 3.2: Ejemplo de llamada SIP via proxy SIP.

Se adjunta en Anexo B un apartado del documento *Establecimiento de sesión SIP* según RFC 3665 para tener un entendimiento más profundo en relación a la sintaxis del protocolo SIP, principalmente orientado al establecimiento de sesión y liberación de la misma.

Servicios SIP: los servicios basados en SIP (*SIP-based services*) son servidores SIP que ofrecen servicios relacionados, como por ejemplo encaminamiento de mensajes a terminales SIP, entre los cuales se incluyen los teléfonos IP. Por ejemplo, en la Figura 3.3, el SIP proxy y *registrar* ofrecen la posibilidad de realizar el registro SIP y el servicio proxy facilita a los terminales SIP a descubrirse y comunicarse entre ellos.

Puede resumirse lo presentado por el desarrollador de software [Wei Chen, 2006] en la Figura 3.3 mediante los pasos detallados debajo:

1. El usuario que recibe la llamada se registra en el servidor de registro al enviar una petición de registro (*REGISTER message*).
2. El *registrar server* acepta esta solicitud de registro mediante una respuesta satisfactoria con el código de estado *200 OK*. La solicitud contiene la dirección del nombre del usuario llamado.
3. El *caller* solicita establecer una sesión de comunicación con el destinatario de la misma enviando un mensaje o solicitud de invitación, INVITE al servidor *proxy*. El mensaje INVITE usualmente contiene la descripción de la sesión SIP que el usuario que realiza la llamada desea establecer, es decir, el protocolo de codificación de la voz, nivel de seguridad, o dirección IP. La descripción nombrada generalmente respeta el formato del protocolo SDP.

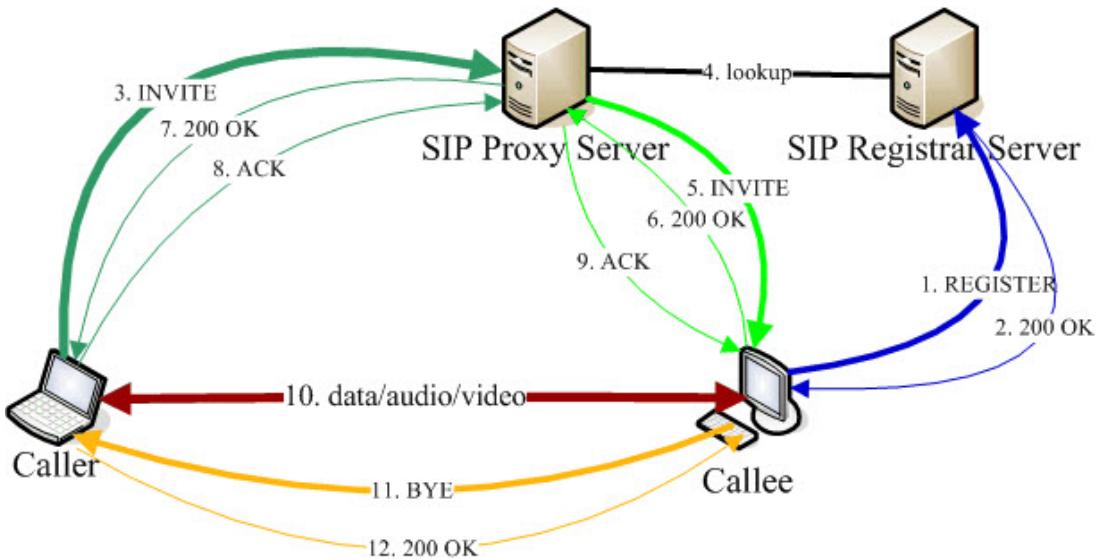


Figura 3.3: Ejemplo de registro y llamada SIP vía proxy SIP. [Wei Chen, 2006]

4. El servidor *proxy* apunta su búsqueda (*lookup*) al servidor de registro para encontrar la dirección actual del destinatario de la llamada. Se deberá notar que el *lookup* en el *registrar server* es un procedimiento externo a SIP en sí mismo.
5. El servidor *proxy* reenvía la solicitud INVITE desde el terminal que realiza la llamada al destinatario de la misma basándose en su dirección actual.
6. El usuario que fue llamado confirma la comunicación al responder con el código de estado *200 OK*. La respuesta *200 OK* a un INVITE usualmente contiene la descripción de la sesión SIP que el receptor puede establecer con el emisor de la comunicación.
7. El servidor *proxy* reenvía una respuesta *200 OK* desde el *callee* al *caller*.
8. El terminal llamante confirma el establecimiento de la sesión mediante el envío de un mensaje ACK al *proxy server*. El mensaje ACK puede contener el acuerdo final (*final agreement*) de la sesión.
9. Llegado su turno, el servidor *proxy* reenvía el ACK al *callee*. De esta forma, el intercambio de tres vías (*three-way handshake*) se completa mediante el servidor *proxy*, y la sesión SIP finalmente queda establecida.
10. A partir de este momento, la comunicación VoIP entre las dos partes se ha establecido. El protocolo utilizado para la señalización y establecimiento de la llamada puede o no ser SIP. Por ejemplo, pueden enviarse mensajes instantáneos (*instant messages*) a través de SIP. Por otro lado, las conversaciones de voz son típicamente transmitidas sobre RTP.
11. Finalmente, el usuario que recibió la comunicación termina la conversación y finaliza la sesión mediante el envío de un (*BYE message*).
12. El *caller* responde a el mensaje BYE con un mensaje de estado *200 OK* aceptando la terminación de la sesión SIP.

En el escenario descripto anteriormente, el servidor *proxy* SIP simplemente encamina los mensajes a la dirección actual del destinatario de la conversación. Existen implementaciones

más avanzadas de un servidor *proxy SIP*, como por ejemplo que el servidor *proxy* realice un seguimiento de donde se encuentra registrado el usuario, a título de ejemplo su celular, o computadora de escritorio en su trabajo. De esta manera la llamada será enrutada de manera inteligente a través del dominio SIP.

3.3. Estudio exploratorio de carácter bibliográfico de los riesgos y seguridad en VoIP

La aparición de nuevas tecnologías de telecomunicaciones está cambiando drásticamente nuestro estilo de vida. La integración de Internet, la telefonía conmutada tradicional y las redes de telefonía móvil en una sola red global, provee nuevos e innovadores medios para las aplicaciones de datos, voz y video. Sin embargo, como nunca antes, preocupaciones extremadamente relevantes en torno a la seguridad informática tienen que ser consideradas. Como nombramos anteriormente, VoIP hereda la carencia de seguridad de las diferentes capas de Internet (vulnerabilidades de red, OS/software y humanas). A diferencia de las redes de telefonía pública conmutada PSTN, Internet es abierta y accesible a hackers con basta experiencia a un bajo costo. Las llamadas de voz sobre IP cursadas a través de una Red de Área Local (LAN) se encuentran altamente expuestas a escuchas telefónicas no deseadas, llevadas a cabo por alguien dentro de esta misma LAN, violando la privacidad del usuario. Se deben también considerar las actualizaciones de software para VoIP, las cuales están expuestas a modificaciones maliciosas que no podrán ser detectadas sin revisiones de integridad de código estrictas y bien ejecutadas.

En este apartado se presenta de forma resumida los tipos de riesgos en torno a VoIP y su clasificación. Si basamos la clasificación considerando su objetivo, los diferentes ataques pueden clasificarse en tres categorías: ataques contra servicios de red, protocolos de datos (o *media*) y por último protocolos de señalización.

3.3.1. Servicios de red

El despliegue y el funcionamiento de un agente de usuario SIP es generalmente soportado por diferentes servicios de la infraestructura de red. Entre ellos podemos nombrar los protocolos DNS, Protocolo de Resolución de Direcciones (ARP), Protocolo de Configuración de Clientes Dinámico (DHCP) y Protocolo de Transferencia de Archivos Trivial (TFTP). En el cual ARP administra la capa dos (capa de enlace de datos) del *stack* de TCP/IP y DHCP permite obtener de manera dinámica los parámetros IP para nuestros terminales SIP en la red. Luego DNS resuelve los nombres de dominio SIP y TFTP obtiene actualizaciones de *firmware* para los dispositivos VoIP. Al comprometer cualquiera de estos protocolos dentro de la LAN se estaría impactando de forma directa sobre el dominio de telefonía IP. Desde obtener una actualización de un servidor TFTP no deseado, o recibir una resolución de dominio desde un servidor DNS envenenado (*DNS-cache poisoning*) el cual lo direccione a un *host* malicioso. Se puede decir, que la suplantación de identidad mediante la suplantación de direcciones MAC e IP se encuentran dentro de los principales ataques en Internet, los cuales causan muchos problemas en la infraestructura VoIP. [Nassar M., 2009]

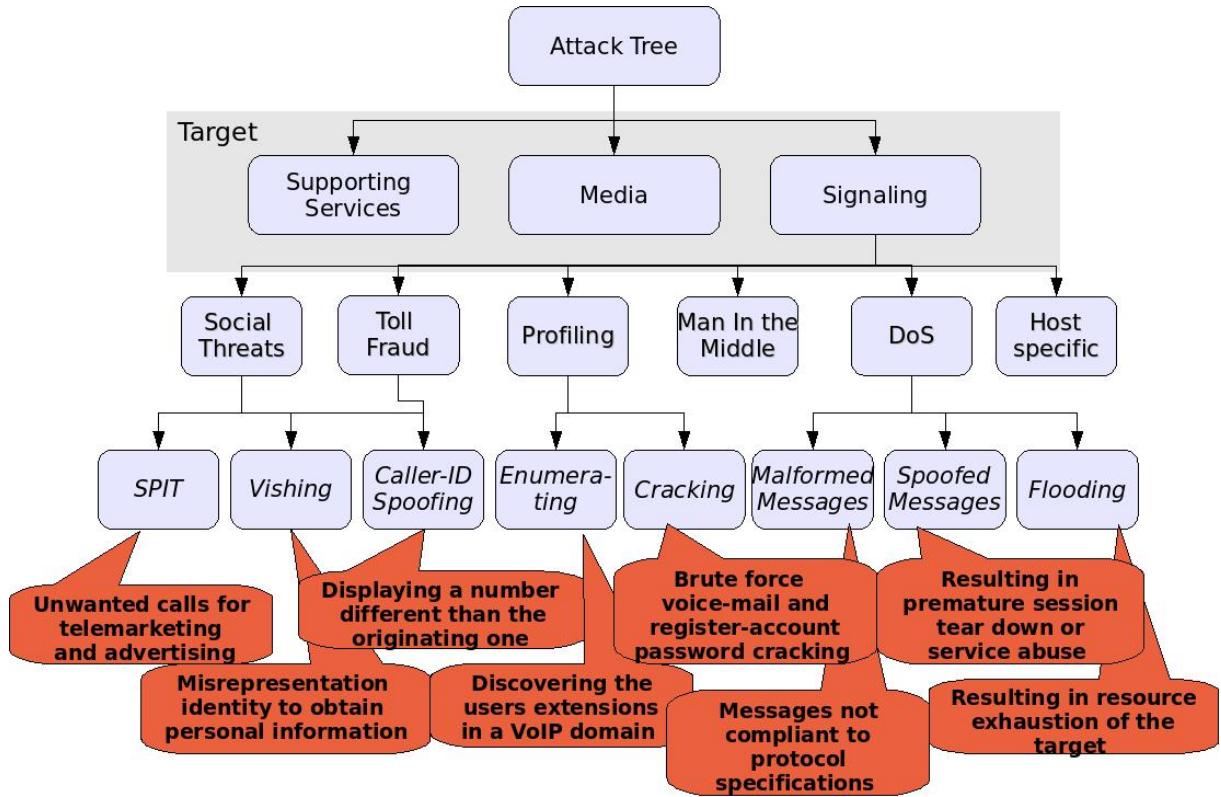


Figura 3.4: Diagrama de ataques VoIP específicos. [Nassar M., 2009]

3.3.2. Protocolos de media

Las aplicaciones VoIP son en tiempo real (*real-time*) por lo que alterar su Calidad de Servicio (QoS) podría tornar su uso en no conveniente o hasta imposible. Con un retardo (*delay*) de 150 milisegundos o un 5 % de paquetes perdidos, una llamada VoIP se transforma en incomprensible, incluso cuando *codecs* eficientes son implementados. Como para interrumpir la transmisión entre las dos partes comunicadas, un atacante solo deberá congestionar la red en algún punto del camino de transporte. Si el atacante es capaz de predecir el número de secuencia de un flujo RTP (ej.: mediante *sniffing* (espiando) cuando el flujo de datos no se encuentra cifrado, este podría inyectar datos extraños en el flujo de transmisión).

La señalización SIP es fuera de banda, por lo que no existe un mecanismo para monitorizar y controlar las sesiones de *media* establecidas por SIP. Por ejemplo, cualquier modificación en los *codecs* utilizados y consecuentemente las condiciones de QoS no son transparentes para SIP. De esta manera podría fácilmente existir abuso de facturación basada en QoS. [Nassar M., 2009]

3.3.3. Protocolos de señalización

De acuerdo a lo desarrollado por el investigador francés [Nassar M., 2009] el plano de señalización de cualquier arquitectura de telecomunicaciones incluyendo VoIP, es el *backbone* (red troncal o segmento de red principal), por lo que es objeto de análisis de una gran cantidad de riesgos de seguridad. Considerando la Figura 3.4 se detallan los ataques presentados en la misma:

- **Profiling:** *sniffing* e interceptación del tráfico SIP podrían revelar porciones de información sensible o valiosa. La interceptación o captura de un mensaje INVITE no cifrado brinda al llamante (en el encabezado *From*), el que recibe el llamado (en el encabezado

To) y la ruta de la llamada (en el encabezado *Via*). El cuerpo SDP contiene la descripción de *media* de la llamada lo que permitirá el filtrado de los paquetes correspondientes, reconstrucción de llamada y escuchar secretamente (*eavesdropping*). A veces, el cuerpo SDP contiene la llave de cifrado (*encryption key*) de la sesión de datos que hace que el *eavesdropping* a una sesión cifrada también sea posible. El *cracking* (quebrado) consiste en un algoritmo de fuerza bruta que intenta adivinar el *password* (clave) de una cuenta SIP mediante solicitudes recursivas contra el SIP *proxy* o *register servers*. Los ataques de *cracking* son usualmente precedidos por un escaneo de dispositivos SIP (puertos TCP/Protocolo de Datagrama de Usuario (UDP) 5060 abiertos) dentro de un rango de direcciones IPs y la enumeración y descubrimiento de los usuarios existentes. La enumeración (*enumeration*) consiste en realizar solicitudes para asociar números VoIP (SIP URI) y un usuario existente en la base de datos de ubicación. Dependiendo el caso (ya sea que el usuario existe, existe pero no está disponible, o no existe) la respuesta del servidor de ubicación (*location server*) podría ser diferente. Mediante el análisis de las respuestas, el usuario existente podría ser filtrado y utilizado en pasos futuros del ataque. Mientras que el tipo de enumeración antes descripto es del tipo activo, hay formas pasivas como el *web browsing* (ej.: los números VoIP de una empresa pueden encontrarse en su página de Internet) y *sniffing* (ej.: extraer una extensión valida de un mensaje SIP capturado).

- Hombre en el Medio (MITM): debido a la falta de buenas prácticas de autenticación, muchas situaciones permiten un “Hombre en el Medio” (Man In The Middle -MITM) a “estacionar” entre el caller y su callee. SIP implementa un fuerte esquema de autenticación similar al de HTTP. Un UAS, *proxy*, *redirect*, o *register* debería autenticar un UAC mediante una prueba de cifrado basado en un secreto compartido. Inversamente, un UAC puede autenticar el servidor. En la mayoría de las implementaciones, es aplicada la autenticación en una sola dirección. Este ataque es posible, se consideran los interlocutores Bob y Alice a los cuales se hace referencia en SIP - RFC 3665 [Johnston A., 2003] y una tercer parcipante Trudy (MITM usuario malicioso):

Bob está llamando a Alice. Bob envía un mensaje INVITE a su *proxy* de salida (*outbound proxy*). Trudy intercepta el mensaje y envía una redirección modificada personalmente a Bob para que apunte o alcance su dirección IP (IP de Trudy). Bob no necesita la autenticación de su *proxy*, acepta la respuesta y la redirecciona la llamada para que esté pase por Trudy. El *proxy* verdadero puede ser neutralizado mediante un DoS o explotando un condición de carrera (*race condition*)¹. Al mismo tiempo, Trudy remplaza la localización de Bob por su IP de contacto (*Contact IP*) en el encabezado y encamina el INVITE hacia Alice. A partir de este momento toda la señalización entre Bob y Alice va a pasar por Trudy. Secuestrar (*hijack*) el vínculo de señalización es un paso previo a secuestrar el de datos. Trudy puede hacerse pasar por Alice considerándolo desde la perspectiva de Bob y puede hacerse pasar por Bob desde el lugar de Alice. En otro escenario, Trudy explota la falta de autenticación de extremo-a-extremo (*end-to-end authentication*). Trudy intercepta el INVITE desde Bob hacia Alice, espera a que la sesión esté establecida, luego realiza un siguiente paso y envía (a Bob o Alice) un nuevo INVITE de Bob a Alice con los mismos encabezados *Call-ID*, *To* y *From* y un número de secuencia de comando incrementado (en *CSeq*). Por medio de este mensaje creado manualmente, Trudy puede cambiar

¹Una condición de carrera puede ser definida como ”Comportamiento anómalo debido a una dependencia crítica no esperada en el tiempo relativo de los eventos“ [FOLDOC]. Dichas condiciones generalmente involucran uno o más procesos de acceso a un recurso compartido (como ser un servicio de red, archivo o variable), donde este acceso múltiple no ha sido adecuadamente controlado. En general, los procesos no se ejecutan automáticamente; otro proceso podría interrumpirlo esencialmente entre cualquiera dos instrucciones. [Wheeler D., 2004]

las condiciones de *media* (datos), (ej.: remplazando *reception via route port* (puerto en tránsito de recepción), agregando o borrando cadenas de datos), o el vínculo de señalización (por medio de los encabezados *Via* y *Route*). Por un momento, la sesión habrá sido secuestrada por Trudy. Este caso teórico es el implementado en fraudes de tasas y facturación telefónica.

- Mensajes malformados (*malformed messages*): las vulnerabilidades de software VoIP son un objetivo de ataques remotos tales como sobre carga de memoria intermedia (*buffer overflow*) y mensajes manualmente modificados. Posibles daños son colapso del sistema (*system crash*), código de ejecución remota y acceso no autorizado.
- Mensajes falsificados (*spoofed messages*): los ataques de CANCEL y BYE son un tipo de ataque de denegación de servicio contra el procedimiento de establecimiento de una llamada SIP. Aquí damos cuatro ejemplos:
 - Si en cualquier momento alguien trata de llamar a Bob, Trudy envía un CANCEL falsificado (*spoofed CANCEL*) (como si sería originado desde el usuario que realiza la llamada) para Bob, luego podrá prevenirse a Bob de recibir cualquier llamada.
 - Si en cualquier momento Bob intenta hacer una llamada, Trudy envía un CANCEL falsificado (como si este fuese originado por Bob) hacia el destino, luego podrá prevenirse a Bob de realizar cualquier llamada. Además,
 - Trudy envía un BYE falsificado para terminar la sesión luego de un corto tiempo desde su comienzo.
 - Trudy toma el lugar del *proxy* de salida mediante el envío de respuestas de error falsificadas como 4xx (*client error*), 5xx (*server error*) o 6xx (*global error*) de manera de persuadir a Bob de que existe alguna situación de error previniendo la continuidad de sus llamadas.
- *Flooding attacks*: los ataques de inundación tienen como objetivo los elementos del plano de señalización (ej.: *proxy*, *gateway*, etc.) con el fin de tirarlo abajo y producir un colapso en la red de VoIP. Basados en la técnica de uso, estos ataques pueden ser clasificados como inundación de mensajes legales, mensajes mal formados aleatoriamente, o mensajes mal formados explotando vulnerabilidades específicas del objetivo. El autor [Nassar M., 2009], categoriza alguno de estos ataques basados en el URI del solicitante y lleva adelante un estudio comparativo de estos en contra de equipos VoIP de código abierto populares. Adoptaremos la misma categorización, la cual se presenta a continuación:
 - Inundación UDP: considerando que la gran mayoría de los sistemas SIP usan UDP como protocolo de transporte, un gran número de paquetes UDP aleatorios son enviados en un intento de congestionar el ancho de banda de red del objetivo. Dichos ataques producen una alta tasa de pérdida de paquetes, por lo qué el tráfico de señalización SIP de llamada legítima verá sus probabilidades de alcanzar su destino para ser procesada de manera muy reducida.
 - Inundación INVITE con un URI de solicitud válido: el atacante llama a un teléfono SIP registrado en el objetivo. El objetivo transmite la llamada al teléfono. Si el *proxy* es con monitoreo o seguimiento de estado (*stateful proxy*) manejará una máquina de estados para cada transacción. El teléfono será rápidamente sobrecargado por la alta tasa de llamadas y no le será posible terminar las llamadas. Como resultado, el servidor se encontrará reservando recursos por un largo período de tiempo y se quedará sin memoria disponible.

- Inundación INVITE con teléfono SIP inexistente: si el atacante no conoce ningún URI SIP válido registrado en el objetivo, puede enviar llamadas a una dirección no válida. El objetivo del ataque responde como "usuario no encontrado" ("user not found"). Cuando la tasa ataques es superior a la capacidad del objetivo, sus recursos quedarán exhaustos. Este tipo de inundaciones es menos perturbadora que la explicada en el ítem anterior, pero el CPU del dispositivo afectado será cargado con transacciones inútiles y de esa forma se rechazarán solicitudes válidas.
- Inundación INVITE con una dirección IP de dominio inválida: el usuario malicioso llama a un usuario con una dirección IP de origen (de destino para el usuario llamado) falsa. En este caso la víctima intentará varias veces comunicarse con esta red/host no alcanzable quedando atrapado por este tiempo al mantener el estado de la transacción SIP. Este ataque es eficiente contra algunos *proxies* como OpenSER.
- Inundación INVITE con un nombre de dominio inválido: también conocido como un ataque SIP-DNS. El usuario mal intencionado llama al usuario con un nombre de dominio de destino no posible de resolver. La víctima se ve forzada a preguntar al servidor DNS para resolver el nombre de dominio y esperar su respuesta. La respuesta del DNS podría tomar un largo lapso de tiempo, especialmente bajo ciertos errores de configuración. Algunos *proxies* son síncronos (ej.: estos bloquean el proceso hasta que una respuesta del DNS es recibida) mientras que otros son asíncronos (ej.: estos guardan el estado de la llamada y toman una nueva solicitud, una vez que la respuesta desde el DNS es recibida, estos serán notificados). Los *proxies* síncronos son fáciles de bloquear por este tipo de ataques, mientras que los asíncronos borrarán su memoria en cierto momento disminuyendo el impacto del mismo.
- Inundación *INVITE/REGISTER* con autenticación activada: el atacante envía un mensaje INVITE o REGISTER y luego detienen el proceso de *handshaking*. El *proxy/registrar* responde con un *challenge* y espera que la respuesta sea enviada nuevamente con las credenciales de autenticación. Este proceso es costoso para el *proxy/registrar* en cuanto a términos de procesamiento (generando muchas transacciones y *nonces* relacionadas a la autenticación) y uso de memoria (diálogos/transacciones de estado de los clientes).
- *Gateways*, PBX y Corta Fuego (FW): los MGC y servidores PBX son puntos críticos en las arquitecturas de voz sobre IP. Los MGC son responsables de realizar la intercomunicación entre diferentes tipos de redes. Tal es el caso, entre una red SIP-based y la PSTN ISUP/SS7-based. La traducción entre mensajes SIP y un mensaje ISUP consiste en el mapeo de los parámetros correspondientes. La administración maliciosa de este proceso de mapeo es una fuente de peligro o amenaza causada por la carencia de mecanismos de autenticación e integridad en la red SS7-based. PBX es un término usado en redes PSTN y adoptado por VoIP para representar el suministro de servicios VoIP (Comutación de llamadas, voice-mail, respuestas automáticas, etc). Un ejemplo de una PBX VoIP es Asterisk. Desde el punto de vista de SIP, una PBX es equivalente a un B2BUA. En PBX, los servidores de facturación y voice mail, los Registros de Detalle de Llamadas (CDR) y la información confidencial de usuario debe ser protegida contra intrusiones basadas en *hosts* (*host-based intrusions*) podemos nombrar entre otros el acceso no autorizado o destrucción de datos. VoIP requiere una nueva generación de *firewalls* con características dinámicas para abrir y cerrar puertos de *media* considerando los parámetros de la sesión negociados. Atacando los *firewalls* dinámicos o comprometiéndolos revela los dispositivos protegidos. Una forma es consumir todos los recursos del *firewall* mediante varias solicitudes de apertura y cierre de puertos. En otra dirección, los *malwares* (softwares

maliciosos) podrían ocultar su tráfico al transmitir el mismo mediante un puerto de *media* que se encuentre abierto. Si son atacadas, algunas aplicaciones VoIP pueden ser utilizadas como *backdoors*.

- Falsificación de identificador de llamante (*caller-id spoofing*): en la PSTN, Entrega de Numero Llamante (CDN) es un servicio telefónico que permite a la parte llamante proveer el Equipamiento de Permisos de Usuario (CPE) de la parte llamada con el número de directorio durante el ciclo de timbrado (*ringing*). Identificación de Número Automática (ANI) es un sistema utilizado por las compañías telefónicas y clientes para identificar los números de los usuarios llamantes. Tradicionalmente, la suplantación o falsificación del *caller-id*/ANI era un proceso complicado que requería acceso a los *switches* o PBXs de la compañía operadora. Con VoIP, puede fácilmente falsificarse el caller-id (ej.: campo *From* en el encabezado SIP) y engañar al usuario llamado con un numero falso. El reemplazo del caller-id promueve la suplantación de identidad (*identity theft*) y el *by-passing* (sobrepasso) de algunas verificaciones de sistema (ej.: obtener acceso al buzón de voz).
- Amenazas sociales (*social threats*): las amenazas sociales son ataques que van desde la generación de comunicaciones no solicitadas que son molestas y perturbadoras para los usuarios, hasta otras más peligrosas como ataques de robo de datos. La amenaza es clasificada como social debido a que el término “no solicitada” está estrictamente contenido a las preferencias específicas de usuario. Esto hace que este tipo de ataques sean difíciles de identificar. Un ejemplo de esto es una amenaza comúnmente referida como SPIT. SPIT es similar a *spam* en el sistema de e-mail pero entregado por medio de llamados de voz. Las llamadas SPIT pueden ser llamadas de *telemarketing* utilizadas para guiar a los llamados a un servicio de venta de productos. Son los más perturbadores para las víctimas considerando que requieren su reacción momentánea para detener la perturbación. La cantidad diaria de publicidad recibida en una casilla de correo postal es muchísimo menor al *spam* masivo recibido cada día en el correo electrónico. Esto es muy intuitivo ya que el *spam* prácticamente no aprieta costos para los publicadores. Ahora que pasaría si las llamadas telefónicas fueran de un costo mínimo o sin costo alguno? SPIT se vale del bajo costo que tienen las llamadas VoIP con respecto a los sistemas de voz telefónica tradicional. Actualmente está estimado que generar una llamada de voz sobre IP es tres órdenes de magnitud más económico que las llamadas generadas en PSTN. Una variante de SPIT es la comúnmente llamada ataque de *vishing* (VoIP *phishing*). En general, phising se trata de enmascarar una tercera parte confiable en el intento de adquirir información confidencial de las víctimas. Por ejemplo, el phisher (atacante) usa un link en un e-mail como señuelo, persuadiendo a la víctima mediante una página web con la misma apariencia que la de su banco y solicitando que complete algunos campos con información sensitiva. Con VoIP, *vishing* apunta tanto a realizar que el usuario de VoIP lleve a cabo llamadas a números excesivamente caros con el fin de obtener el premio prometido o para colectar información personal redirigiendo a los usuarios mediante un número DID (Direct Inward Dialing - Discado Interno Directo) hacia un IVR (Interactive Voice Responder) simulando ser confiable. La mayoría de estos ataques van a ser generados por máquinas (*bot-nets*) programadas para realizar dicho trabajo. Comunicaciones no solicitadas (como ser SPIT o *vishing*) son, desde un punto de vista de señalización, transacciones correctas técnicamente. No es posible distinguir de un mensaje INVITE si dicha transacción es SPIT o no. Desde un punto de vista técnico el reto es aun más complicado considerando que el contenido no está disponible para ayudar en la detección hasta que el teléfono timbre (interrumpiendo al usuario) y el usuario receptor de la llamada responde la misma. Por esta

razón, técnicas usadas con éxito contra e-mail *spam* por citar un caso alteración de texto, serán difícilmente reusables en la esfera de VoIP. Aún si la transacción es identificada como no solicitada, su procesamiento dependerá en gran medida del entorno legal en el país del usuario que realiza la llamada.

3.4. Estudio exploratorio sobre *honeypots* VoIP específicos y Artemisa *honeypot*

Encontraremos en la literatura sobre el tema, que un *honeypot* es una trampa instalada para detectar, desviar y monitorear ataques a sistemas informáticos. Generalmente consiste en una computadora, datos o un sitio de red que parecería ser parte de una red pero que en realidad está aislada y bajo monitoreo. Un *honeypot* tienen un valor específico en la detección y desvío de ataques. Los *honeypots* usualmente son un sistema particular que normalmente no deberían ver ningún tipo de tráfico o actividad legítima. Cualquiera sea la actividad vista en un *honeypot* puede ser interpretada como maliciosa o no autorizada.

El concepto de *honeypot* puede ser muy útil como componente específico de una arquitectura de prevención e intrusión de VoIP como la presentada en la Figura 3.5. Este concepto es traído al mundo de la voz sobre IP desarrollando una infraestructura de VoIP completamente paralela, totalmente separada tanto lógica como físicamente de la infraestructura real actividad de la cual estaremos monitoreando de manera continua. La separación física es necesaria de modo de evitar que el atacante quiebre la seguridad de nuestro *honeypot* y esto le permita atacar la infraestructura VoIP real desde allí.

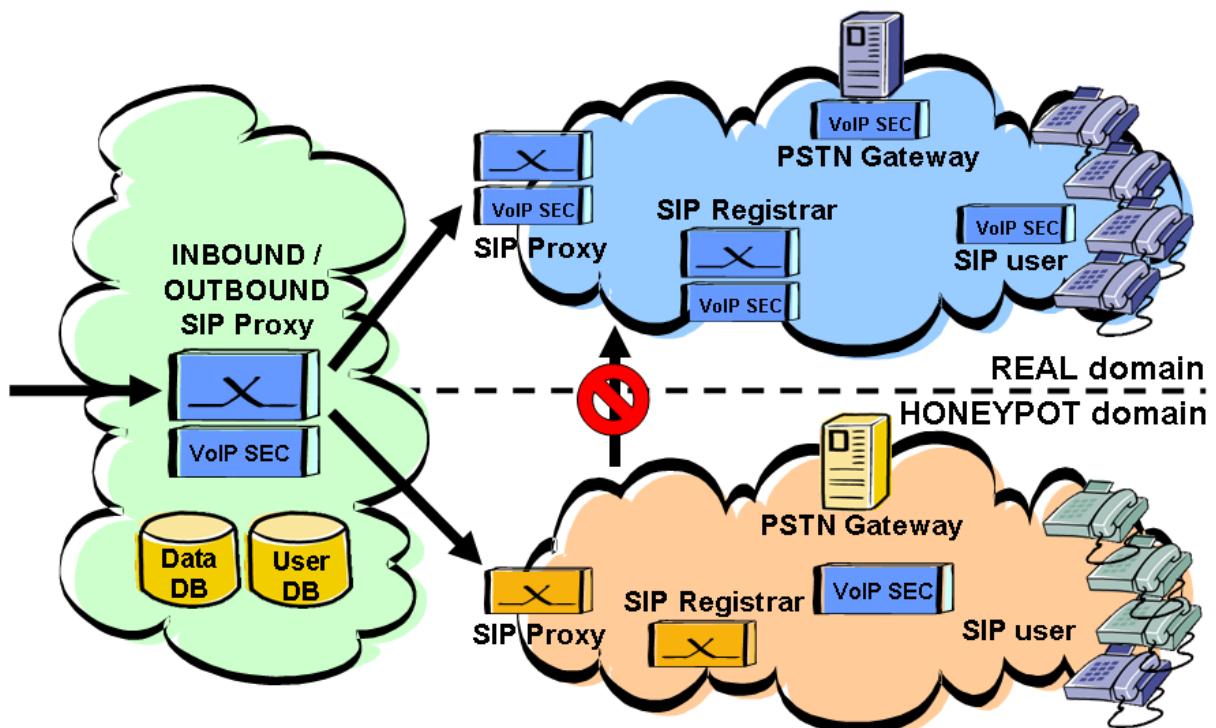


Figura 3.5: Arquitectura de red. [Ewald T., 2007]

Las características fundamentales de un *honeypot* específico de VoIP son la mitigación de la amenaza de SPIT con una infraestructura de bajo costo como puede verse en la Figura 3.6.

Además, el hecho de que dicha infraestructura complementa adecuadamente a todos los posibles sistemas de detección y prevención de intrusiones mediante la recopilación de información, muy útil para mejorar la tasa de error de otras metodologías.

El *honeypot* específico VoIP presentado en la Figura 3.6 en el cual está basado Artemisa está compuesto por componentes VoIP *Open Source* (código abierto) estándares basados en el protocolo SIP. El principal objetivo de un *honeypot* es atraer ataques hacia un entorno seguro y protegido para posteriormente analizar los esquemas de ataque y deducir nuevas maneras y métodos preventivos en contra de estos. Para alcanzar esta meta el *honeypot* tienen que ofrecer servicios atractivos que valgan la pena ser atacados. En nuestro estudio si bien estos servicios fueron expuestos a la Red de Area Extensa (WAN), los mismo son simulando un prestador de VoIP. Consideramos que sería de gran valor implementarlos en una red en producción. Un *honeypot* SIP debería o podría simular una red SIP completa que pueda ofrecer muchos entidades alcanzables. [Ewald T., 2007]

- SIP components.
- SIP services.
- SIP users.

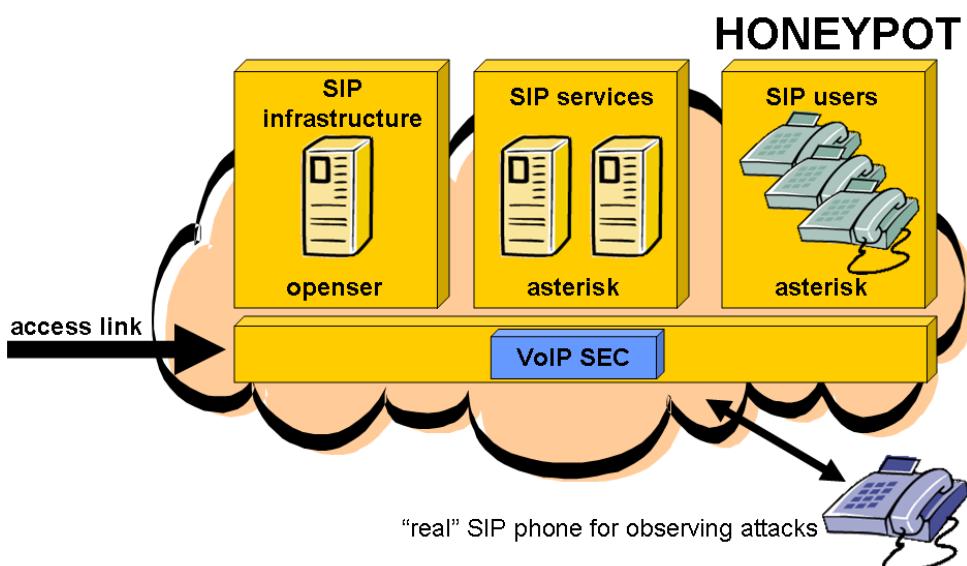


Figura 3.6: Arquitectura del *honeypot* SIP. [Ewald T., 2007]

3.4.1. Introducción a Artemisa honeypot

El *honeypot* es diseñado para conectarse al dominio VoIP de una compañía como un *back-end user-agent* con el fin de detectar actividad maliciosa en una etapa temprana. Asimismo, el *honeypot* puede jugar un rol en el ajuste en tiempo real de las políticas de seguridad del dominio corporativo en el que se encuentre implementado. Es bueno recordar que el esquema de direccionamiento SIP está basado en el URI con la siguiente sintaxis: `sip:user@host:port;parameters`.

Un dominio SIP corporativo está típicamente compuesto por las instalaciones de usuario (*hard* y *softphones*), una infraestructura SIP (ej.: *proxy*, *registrar*, *back-to-back UA*) y soportando servicios asociados (ej.: administración WEB (*web-based management*, TFTP, DNS)).

Para el desarrollo Artemisa se propuso su diseño de arquitectura implementada con herramientas *Open Source*, haciendo las veces de *honeypot* de SIP específico que puede ejecutarse como *back-end user-agent* en el dominio de la organización. Debido al hecho de que las extensiones del *honeypot* no representan extensiones reales (de usuario reales), toda actividad que lo alcance será percibida como sospechosa. La misma atiende las llamadas y las graba incluyendo la traza SIP. Artemisa es capaz de clasificar varios tipos de anomalías y reportarlos al administrador de la red o automáticamente controlar las políticas de seguridad del dominio bajo su protección. Las configuraciones de utilización típicas de un *honeypot* como se muestra en la Figura 3.7, el mismo se registra a uno o más SIP *registrars* y queda a la espera de llamadas y mensajes SIP. Dos opciones son posibles, la primer opción es conectar el *honeypot* en la Zona Desmilitarizada (DMZ). Consecuentemente, estará aislado de LAN y la VLAN de VoIP en caso que la máquina se vea comprometida. La segunda opción es conectarlo a través de Internet considerando que hoy en día muchos proveedores de voz sobre IP tienen abonados de todo el mundo a través de Internet.

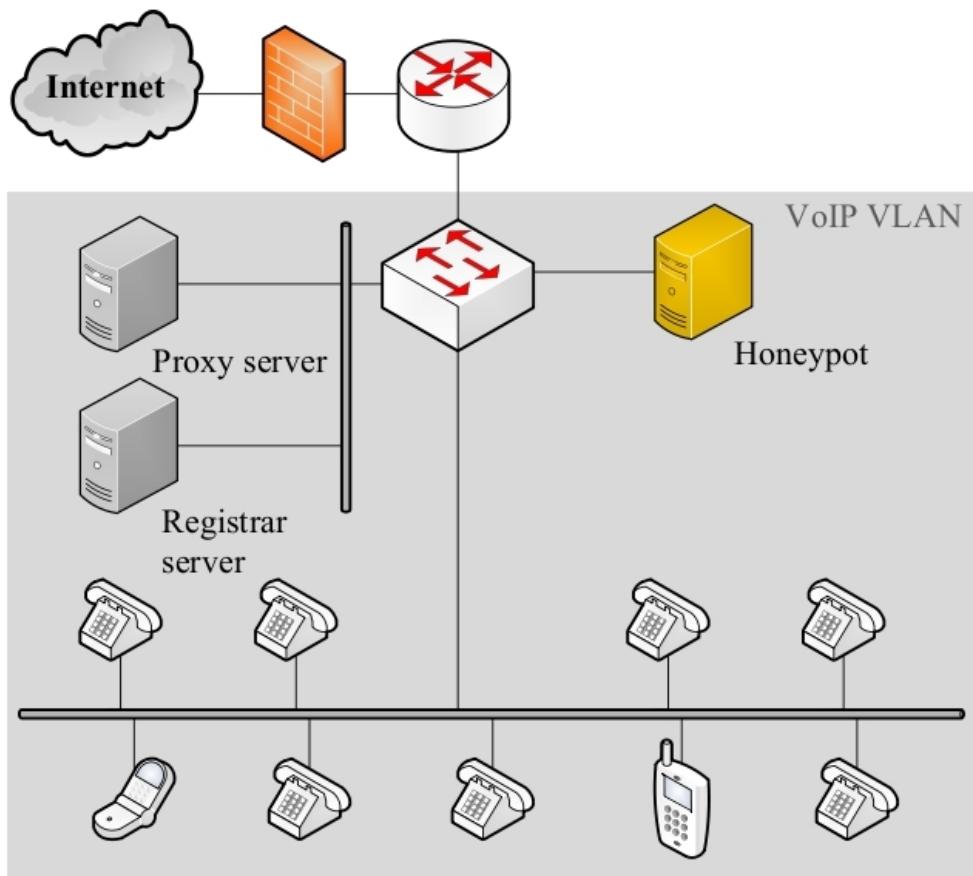


Figura 3.7: Implementación de Artemisa *honeypot*. [Do Carmo R., 2011a]

3.4.2. Artemisa honeypot y sus características funcionales

El *honeypot* registra una serie de extensiones virtuales contra uno o más SIP *registrars*. Las extensiones virtuales deben ser seleccionadas de modo de proteger a las reales. Por ejemplo, si las extensiones reales están todas compuestas por números de tres dígitos, sería recomendable que las extensiones virtuales cubran todos los números de 3 dígitos que son utilizados por los abonados reales. Alternativamente, el *proxy* de dominio o la PBX pueden ser configurados para

reenviar todos los mensajes que no tengan como destino extensiones reales hacia el *honeypot*. Las metas funcionales de dicha configuración van desde detección de enumeración y mitigación de *spam* sobre VoIP, hasta la colección de firmas y *fingerprints* (huellas digitales o identificador de dispositivos), como también el *blacklisting* (poner en lista negra o de bloqueados) a los atacantes. La política de seguridad del dominio VoIP puede ser controlado en tiempo real. Se detallaran las metas antes mencionadas dentro de esta unidad.

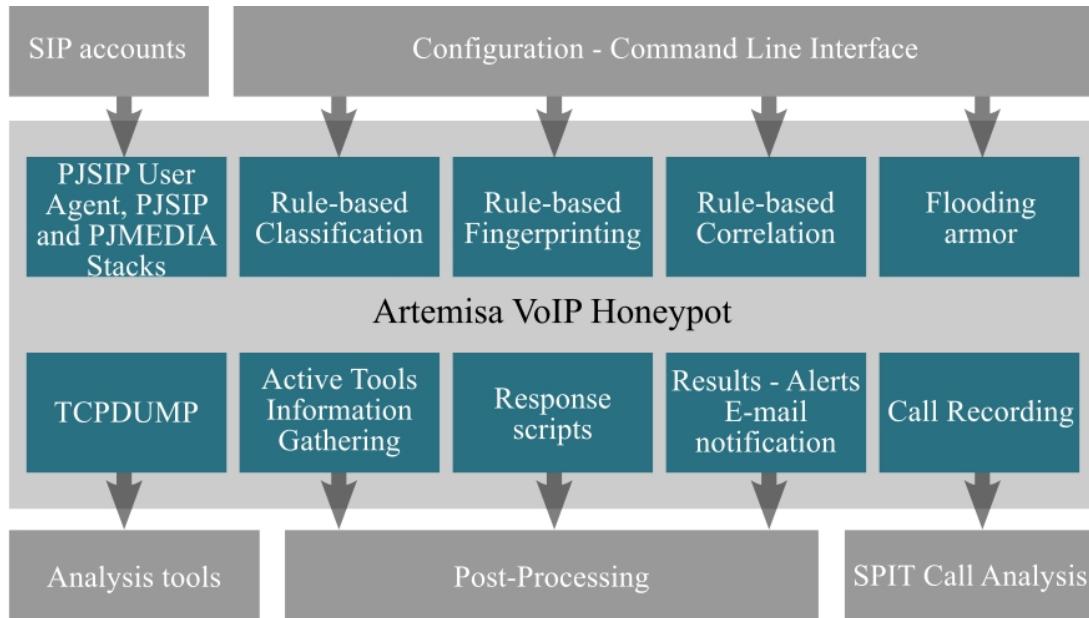


Figura 3.8: Módulos de Artemisa *honeypot*. [Do Carmo R., 2011a]

- Detección de enumeración (*Enumeration detection*): una enumeración es detectada como una serie de mensajes OPTIONS, REGISTER o INVITE destinados a una serie de extensiones virtuales. La fuente del ataque debe poder recibir las respuestas asociadas de manera de poder analizarlas. En modo activo, nuestro *honeypot* recolecta información de al fuente (ej.: la herramienta de ataque, el UA, la dirección IP y el dominio, la ubicación geográfica, el numero de sistema autónomo) y los reporta al administrador de dominio. La recolección de información esta basada en herramientas de *networking* tales como Nmap, Sipsaky y DNS Lookup. Esta información puede ser interpretada automáticamente con el objetivo de bloquear un ataque de enumeración en tiempo real.
- Mitigación de *spam* VoIP (*VoIP spam mitigation*): SPIT es detectada como una serie de mensajes INVITE dirigidos a una serie de extensiones de Artemisa seguidos del envío de trafico de *media* (voz). Los *spitters* (generadores de SPIT) esperan las respuestas de 200 OK cargando los SDP recibidos y envían su *spam* a la IP y puerto RTP anunciados. Nuestro *honeypot* responde a las llamadas de SPIT y las graba para un posterior análisis de contenido. El análisis de las llamadas de SPIT es muy útil para desarrollar filtros anti *spam* asociados a casillas de correo de voz (*voice mail-boxes*), o para ser aplicados a “Pruebas de Turin” o ”El Juego de la Imaginación“ (*Turing Test - The Imagination Game*)² para distinguir entre un humano de los patrones de SPIT automatizado. El *honeypot* recolecta

²Turing, en 1951, propuso un test, “The Imagination Game”, que resolvió la problemática de la máquina inteligente. El juego no involucraba IA computacional. Luego propuso un cambio, en el cual en lugar de que los participantes sean personas, había un humano y una computadora. El trabajo del juez sería decidir cual de los contendientes es humano, y cual la máquina. Propuso que si bajo estas condiciones, un juez tenía una efectivi-

piezas de información sobre la fuente de SPIT y las reporta al administrador del dominio. Estas piezas pueden ser interpretadas automáticamente y ser usadas para bloquear a los *spammers*.

- Colección de firmas y *fingerprints* (*signatures and fingerprint collection*): diferentes equipos VoIP y Firmwares (entre estos teléfonos, *call managers*, servidores) han sido identificados como vulnerables a ataques dentro de los cuales se encuentran *remote crash*, *SQL injection*³, *XSS cross-site scripting*⁴ y *remote eavesdropping* (escuchas a escondidas remotas). Estas vulnerabilidades pueden ser explotadas mediante el envío de mensajes SIP especialmente trabajados. El *fingerprint* del dispositivo de la víctima debería anteceder el ataque con el fin de seleccionar un *exploit* (algoritmo malicioso) adecuado al mismo. Nuestro *honeypot* anuncia *fingerprints* falsos (ej.: fabricante, nombre del producto, versión *firmware*) en el UA o encabezado del servidor cuando el *fingerprinting* se realiza de forma activa (por ejemplo mediante un mensaje OPTIONS). Por ejemplo, un administrador de dominio VoIP puede tener interés en anunciar *fingerprints* similares a los utilizados en de forma activa en el dominio en producción. Asimismo, los mensajes SIP modificados que son recibidos serán recolectados y reportados al administrador de la red. Estos mensajes son importantes al objeto de extraer *signatures* (firmas) y *fingerprints* e incorporarlas en los *SIP-aware firewalls* (firewalls que interpretan SIP). Se reportan los *zero day exploits* (*exploits* de día cero)⁵ al fabricante a fin de corregir sus sistemas. Se reporta y se crean listas negras o de bloqueo (*black list*) basados en la fuente de los ataques si estás no son identificadas como falsificadas (*spoofed*).
- Control en tiempo real y en circuito cerrado de la política de seguridad del dominio (*real-time closed-loop control of the domain security policy*): como fue mencionado anteriormente en las funcionalidades previas, se apuntó a la automatización de la interpretación y el reconocimiento de las actividades percibidas en tiempo real. El resultado de este análisis es reportado al administrador, pero también puede disparar un script de respuesta e intercambiar datos con otros componentes en la arquitectura VoIP aplicado de manera

dad menor al 50 %, esto refleja que es tan probable seleccionar tanto al humano como la computadora. Luego, la computadora sería una simulación pasable de un ser humano, en consecuencia, inteligente. Actualmente, el juego ha sido modificado de modo tal que hay solo un contendiente, y el rol del juez no es seleccionar entre dos participantes, sino simplemente decidir si el participante es un humano o una máquina. [Reingold E., 2009]

³SQL injection (inyección SQL) es un ataque en el cual el código malicioso es insertado dentro de una cadena de caracteres que luego son pasados a una instancia de servidor SQL para *parsing* (análisis sintáctico) y ejecución. Cualquier proceso que construye una instrucción SQL sera revisada para vulnerabilidades de inyección (*injection vulnerabilities*) debido que el servidor SQL ejecutará todas las consultas (*queries*) válidas que este reciba. Incluso los datos parametrizados pueden ser manipulados por un atacante especializado y determinado. [Microsoft®, 2015]

⁴Cross-Site Scripting (también conocido como XSS) es uno de los ataques de capa de aplicación web (*application layer*) más comunes. Las vulnerabilidades XSS tienen como blanco scripts embebidos en una página, los cuales son ejecutados en el extremo del cliente (*client side*, es decir en el navegador web del usuario) en lugar de ser ejecutado del lado del servidor (*server side*). XSS en si mismo es una amenaza que es traída por las debilidades de la seguridad en Internet de los lenguajes de *scripting* del lado del cliente como HTML y JavaScript. El concepto de XSS es manipular los *client side* scripts de una aplicación web para ser ejecutado de la manera deseada por el usuario malicioso. Este tipo de manipulación puede embeber un script en una página la cual puede ejecutar el mismo cada vez que es cargada, o cuando quiera un evento asociado a esta sea ejecutado. [Acunetix®, 2015]

⁵Vulnerabilidades *zero day* (*zero day vulnerabilities*) son vulnerabilidades de las cuales los fabricantes no han lanzado un parche (*patch*) para corregirlo. La ausencia de un parche para una *zero day vulnerability* presenta una amenaza para las organizaciones y los usuarios, ya que en muchos casos este tipo de amenazas pueden evadir la detección basada en firmas (*signature based*) hasta que un parche sea lanzado. La naturaleza inesperada de una amenaza de día cero es una preocupación muy seria, especialmente porque podrían ser usadas en ataques específicos y en la propagación de código malicioso. [Symantec TM® TechNet, 2011]

progresiva y continúa, especialmente cuando el ataque persiste. El control de circuito cerrado (*closed-loop*) de toda la amplitud del dominio permite establecer un compromiso entre la seguridad y el alcance del servicio. Por ejemplo cuando debemos tratar con SPIT, podemos comenzar por una política de “intente más tarde” (“*try later*”). Si el ataque persiste, terminaremos por bloquear totalmente las IP de origen. Desde otro enfoque, la reacción en tiempo real a las amenazas de seguridad es mandatorio para servicios sensibles y de alto costo como VoIP.

- Configurando el *honeypot*: las funcionalidades del *honeypot* son definidas a través de un modo de comportamiento representado por una máquina de estados. La máquina de estados es descripta mediante reglas y operaciones cuidadosas. Modos personalizados pueden ser definidos por el administrador. Complementariamente, definimos dos modos primitivos: pasivo y activo. En el modo pasivo, Artemisa no utiliza ninguna herramienta de *networking*, por ende, no envía ningún tipo de solicitud que pueda revelar su presencia. En modo activo y cuando el *honeypot* revela que el ataque proviene de una fuente real, el mismo intenta recolectar información acerca de este, entre los cuales se encuentran: el *fingerprint SIP*, el *fingerprint* del sistema operativo de la fuente de ataque, la ruta IP, los puertos y servicios abiertos (varios puertos abiertos pueden indicar que una serie de sesiones están siendo mantenidas en paralelo).

3.4.3. Modulos e implementación del honeypot

Artemisa está compuesto por una serie de módulos integrados utilizando el lenguaje de *scripting Python*. Estos módulos son representados en la Figura 3.8 que fue presentada anteriormente. El módulo Tcpdump controla la colección de tráfico de red en bruto (*raw network traffic*) en la máquina donde se está ejecutando el *honeypot* empleando la herramienta Tcpdump. Los módulos de grabación de llamada trabajan con la librería PJMedia para salvaguardar los flujos de audios recibidos en un formato apropiado. Se describirán brevemente los módulos restantes a continuación.

- PJSIP *user-agent, stacks* (pilas) PJSIP y PJMEDIA: el SIP UA es el responsable de registrar la extensión virtual en uno o varios servidores de registro, respuesta de llamadas, respuesta a solicitudes SIP y anunciar los *fingerprints* virtuales. Complementariamente, controla el resto de los componentes basados en el modo de comportamiento. El aplicativo utiliza la vinculación de Python (*Python binding*) de la librería PJSIP6 *user-agent* (PJSUA) de C/C++. Esta librería hace uso de dos pilas (*stacks*): PJSIP y PJMedia para el manejo de los protocolos SIP y RTP. PJSUA permite emular un agente de usuario SIP de una manera simple. En contraste, los binding de librería no ofrecen acceso por completo al *stack* del protocolo SIP (ej.: para el manejo de mensajes OPTIONS). PJSUA recibe como parámetro la cuentas SIP virtuales y configurada a través de los archivos de configuración en el momento de *bootstrap* (arranque) y vía una Interfaz de Línea de Comandos (*Interfaz de linea de comandos (CLI)*) en ejecución.
- Herramientas activas. Recopilación de información: este componente es responsable de correr herramientas de reconocimiento con el fin de colectar información complementaria acerca de las fuentes de los mensajes recibidos. Aquí se investigan varios items en los mensajes SIP y SDP: la IP de origen del mensaje, el URI del llamante, el encabezado Contact (contacto), el encabezado(s) Via, los encabezados de Ruta (Route) y Record-Route si los hubiere, la IP y puerto de media (en los atributos SDP: Connection(c), Owner(o) and Media(m)). Estos componentes invocan las siguientes herramientas:

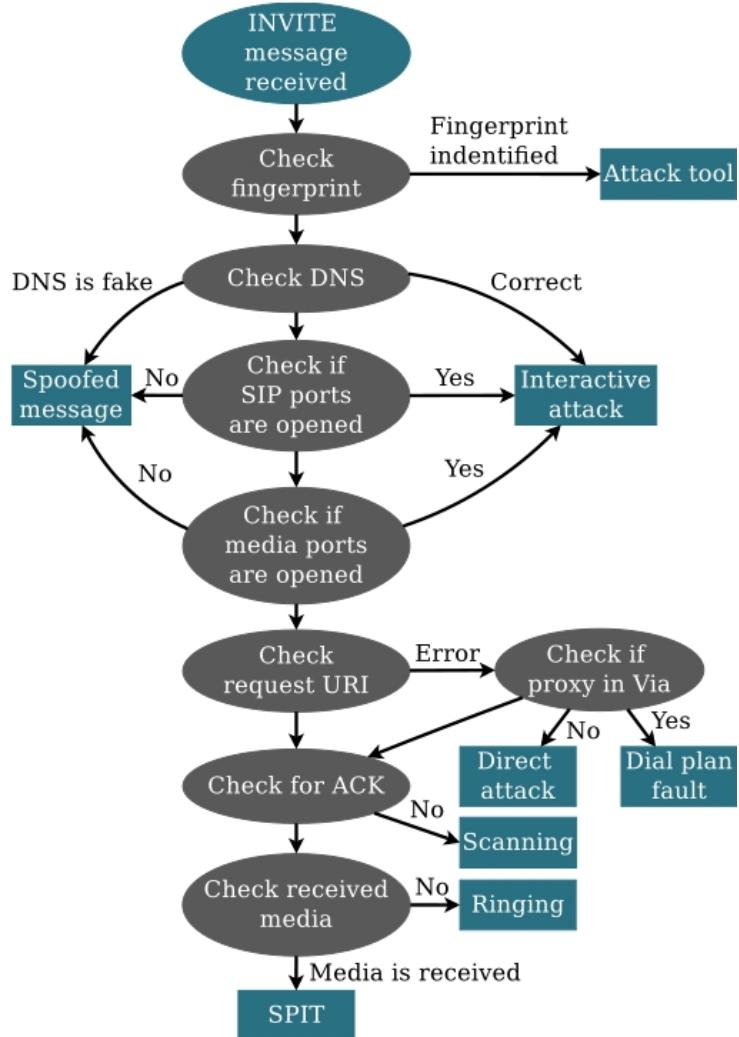


Figura 3.9: Módulos de Artemisa honeypot. [Do Carmo R., 2011a]

- Dig: siendo la alternativa del precursor Nslookup, esta herramienta se aprovecha para consultar los registro de DNS. Lleva a cabo búsquedas DNS y DNS reversas (*Reverse DNS lookups*)
- Jwhois: siendo este un *whois* a nivel de Internet (*Internet whois*), esta herramienta ayuda a obtener información disponible públicamente acerca de los nombres de dominio relacionados a los mensajes SIP.
- Sipsak: esta es una herramienta SIP que puede realizar diferentes pruebas SIP pudiendo nombrar el envío de un mensaje OPTIONS a una URI de destino.
- Nmap: esta es una herramienta de escaneo que podemos usar para escanear el terminal (*host*) del llamante y determinar si SIP, *media*, o ambos puertos se encuentran abiertos.
- Traceroute: ayuda a trazar y rastrear las rutas IP hacia los diferentes *proxies SIP* y UAs envueltos en el mensaje SIP.
- P0f: es una herramienta de *fingerprinting* pasivo que nos facilita recuperar información adicional acerca de la fuente, tal como el sistema operativo.

La información colectada es formateada y provista al conjunto de clasificaciones y reglas de correlación.

- Clasificación basada en reglas (*rule-based classification*): el clasificador interpreta los datos obtenidos por las herramientas de reconocimiento y genera conclusiones cualitativas basadas en reglas formando un árbol de decisión (*decision tree*). Se presenta un ejemplo en la Figura 3.9. Este árbol de decisión recibe e interpreta un mensaje INVITE y genera un o más conclusiones acerca de la naturaleza del mensaje. Los óvalos representan acciones llevadas a cabo y los rectángulos (los nodos "hoja" o "*leaf*" en inglés) representan las conclusiones.

El primer y más importante paso es chequear si el mensaje posee algún *fingerprint* conocido de alguna herramienta de ataque, ya que es fácil tratar con ataque si sabemos como ha sido generado el mismo.

El segundo paso es determinar si el mensaje ha sido falsificado (alguno de sus campos) o no (*spoofed message*). Los mensajes que no han sido falsificados o modificados serán señal de un ataques interactivos (*interactive attacks*) donde el atacante necesita capturar las respuestas y analizarlas. Esto puede ser llevado a cabo verificando que los nombres de dominios usados sean reales, y revisando si el puerto anuncio SIP (puerto UDP:5060 y/o de *media* (RTP) están abiertos.

El tercer paso es verificar si el URI solicitado transporta una de las extensiones virtuales. En caso negativo, revisamos si el mensaje está viniendo a través del *proxy* en uso (*Via header*), o PBX (IP de origen). Esto puede diferenciar un ataque directo (probablemente desde dentro de la red LAN) de un error en el plan de discado (*dial plan error*). Podríamos preguntarnos. ¿Por qué este mensaje es enrutado hacia el *honeypot* considerando que no apunta a una extensión virtual?.

Otras verificaciones aplican al diálogo completo iniciado por dicho mensaje INVITE. Artemisa responde con un *200 OK* y espera por un ACK y *media*. Si es recibido, este diálogo es considerado como SPIT. De lo contrario, el INVITE necesita ser correlacionado con otros eventos porque podría ser parte de un escaneo (*scanning* intentando alcanzar un *gateway* hacia la PSTN o extensiones válidas) o un intento de *ringing* (haciendo timbrar todos los teléfonos en el dominio comprometido).

Se da al administrador la posibilidad de definir el proceso de operaciones de investigación, y las reglas a ser aplicadas en los resultados de investigación. Y de esta forma, controlar el proceso de decisión del *honeypot*. Las reglas de clasificación son ubicadas bajo un contexto (por ejemplo el mensaje INVITE recibido) y están compuestos por una pre-acción (*pre-action*), una condición y una acción.

Se usa una sintaxis simple *attribute = value* para describir las reglas. Por ejemplo, el primer nodo del árbol de decisiones previamente mencionado puede ser representado por la siguiente regla:

```
[rule_1]
type = Classification
context = INVITE
pre-action = assign %a Check_Fingerprint()
condition= %a in FINGERPRINTS
action= add \
"Tool identified as: %a" CONCLUSIONS
```

Donde *assign*, *in* y *add* son operadores específicos, FINGERPRINTS y CONCLUSIONS

son arreglos (*arrays*) de datos predefinidos y `Check_Fingerprint` es una operación predefinida.

- *Rule-based fingerprinting*: las reglas de *fingerprinting* son aplicadas cuando la operación `Check_Fingerprint` es ejecutada. Una regla de *fingerprinting* busca una expresión regular (*regular expression*) en un encabezado o atributo SIP específico, o en el mensaje SIP completo. Por ejemplo la siguiente regla localiza la cadena de caracteres `friendly-scanner` en el encabezado de agente de usuario del el mensaje SIP en cuestión:

```
[rule_2]
type = Fingerprinting
context = INVITE
re = 'friendly-scanner'
where = SipMessage.UserAgent action is executed at the end.
action = return "SIPVicious"
```

- Correlación basada en reglas (*rule-based correlation*): las reglas de correlación son aplicadas cuando varios mensajes SIP son considerados para inferir una conclusión. Se definen diversos tipos de reglas de correlación. Estos tipos son particularmente necesarios para detectar series de eventos como inundaciones (*flooding*), escaneo (*scanning*) y SPIT. Este tipo de reglas soporta atributos de temporizador y umbrales (*timer and threshold attributes*). Como referencia, la siguiente regla es usada para verificar si un ACK es recibido luego de un INVITE en una ventana de tiempo de 5 segundos.

```
[rule_3]
type = Correlation.EventInWindow
context = INVITE
scope = DIALOG
timer = 5
condition = context == ACK
action-if-false = add \
"no ACK is received" CONCLUSIONS
```

El atributo `scope` define a cual mensaje SIP esta regla puede ser aplicada. `scope = DIALOG` significa que la regla solo podrá ser aplicada al mensaje que posea el mismo call-ID que el INVITE que lo antecede. El atributo `action-if-false` significa que la acción será ejecutada en caso de que la ventana de tiempo haya expirado y la condición antes mencionada todavía sea falsa (en contraste con el atributo `action`).

- Defensa o blindaje contra inundaciones (*flooding armor*): este componente protege al *honeypot* de ser inundado (*flooded*) por solicitudes SIP (*SIP requests*). Esto se lleva a cabo asegurándose de que el *honeypot* procese un número limitado de solicitudes en un período de tiempo determinado. Las reglas de correlación son utilizadas. Por ejemplo, la siguiente regla detecta si 3 mensajes INVITE son recibidos desde la misma IP de origen en un ventana de tiempo de un segundo. La acción requerido es ejecutada al final.

```
[rule_4]
type = Correlation.ThresholdInWindow
```

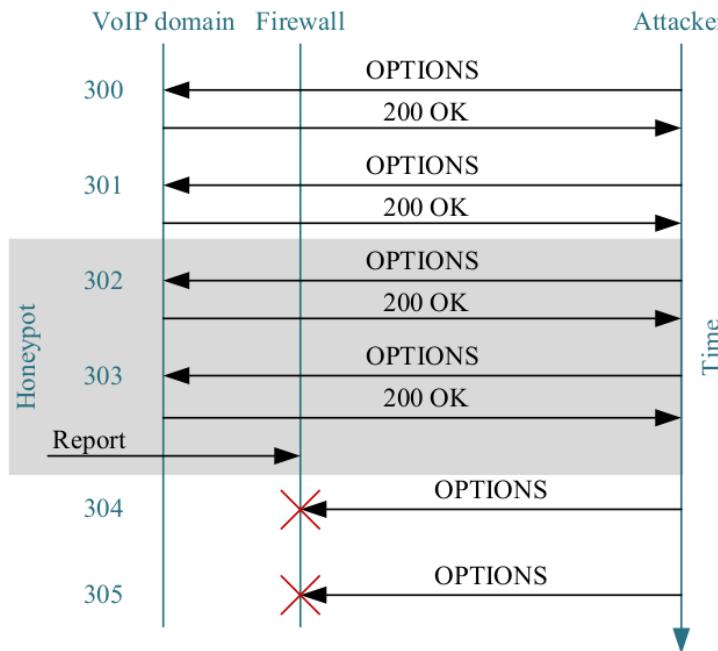


Figura 3.10: Integración en tiempo real de los resultados del *honeypot*. [Do Carmo R., 2011a]

```

context = INVITE
scope = IPSrc
timer = 1
threshold = 3
action = system "bash ./on_flood.sh %IPSrc"

```

- Scripts de respuesta (*response scripts*): los scripts de respuesta (*response scripts*) son ejecutados por el *honeypot* con el fin de reaccionar a un ataque detectado a través del bloqueo o mitigación del mismo. Podemos visualizar esto en la Figura 3.10, la cual muestra un intento de enumeración que es bloqueado en tiempo real luego de ser detectado por Artemisa. Los scripts de respuesta son llamados dentro de las definiciones de las reglas mediante el empleo de la palabra clave (*keyword*) `system` y pasando los argumentos necesarios de la fuente del ataque (ej., IP de origen, SIP *From URI*). La reacción puede ser efectuada en diferente niveles: *firewall IP*, *firewall SIP-aware*, plan de llamado (*dial plan*), entre otros. El administrador definirá los scripts apropiados con respecto a las configuraciones del dominio.
- Resultados, alertas, y notificaciones via e-mail. Existen dos tipos de alerta:
 - Un alerta de mensaje: Contiene todas las conclusiones que son inferidas acerca de un mensaje SIP.
 - Un alerta compuesta: Contiene información acerca de una lista de mensajes SIP correlacionados.

Las alertas son provistas a la interfaz de linea de comandos y registradas (*logged*) en formato de texto y HTML. Artemisa puede ser configurado para enviar alertas al administrador via e-mail. Un ejemplo de mensaje de alerta se muestra en la Figura 3.11.

El código fuente del *honeypot* es distribuido bajo la licencia GPLv310. El último release del proyecto se encuentra disponible en su *home page*: <http://artemisa.sourceforge.net/>.

```

***** Information about the call *****
From: in 192.168.0.103:9/udp
To: 500 in 192.168.0.104
Contact: in 192.168.0.103:9/udp
Connection: 192.168.0.103
Owner: 192.168.0.103
Via 0: 192.168.0.103:9/udp
User-Agent: Elite 1.0 Brcom Callctrl/1.5.1.0 MxSF/v.3.2.6.26

.

***** Conclusions *****
* Tool identified as: Inviteflood
* Spoofed Message
* no ACK is received
* no Media is received
* Scanning or Ringing or flooding attack
* waiting for correlation results

Alert is saved on file 2010-07-17_2.txt
Alert is saved on file 2010-07-17_5.html
E-mail notification is disabled.

```

Figura 3.11: Ejemplo de un mensaje de alerta. [Do Carmo R., 2011a]

3.4.4. Experimentación y casos de estudio

El investigador y desarrollador de Artemisa [Do Carmo R., 2011a] llevó a cabo un caso de estudio en el cual nos basaremos para desarrollar esta sección. Se ha probado la performance en términos de robustez y precisión de la herramienta Artemisa. El fin de la prueba de robustez es asegurar que el *honeypot* no se quiebre (*crash*) fácilmente cuando es alcanzado por ataques o cuando es inundado de forma anormal. El fin de la prueba de precisión es verificar que efectúa una interpretación acertada de los mensajes SIP recibidos.

A continuación, se comenzará mostrando la interpretación fuera de linea (*offline*) de un mensaje SIP detectado por un sensor desarrollado. Luego de eso, se presenta un *testbed* de referencia y se resumen los resultados de 4 escenarios que involucran varias herramientas de ataque. Por su parte, se expone cómo las respuestas pueden ser disparados en niveles de protección paralelos.

- Interpretación de un mensaje SIP: se supone el mensaje INVITE detallado en la Figura 3.12 recibido por el *honeypot* Artemisa e interpretado por el árbol de decisión clasificatoria. Del cual se obtienen las siguientes observaciones:
 - Falta el encabezado de Agente de Usuario (*user-agent header*): no se puede estimar que herramienta fue usada.
 - La verificación de DNS (*check DNS*) devuelve negativo. La IP en el SDP (1.1.1.1) presenta que el mensaje es falsificado (*spoofed*).
 - Verifica si el *host* esta arriba y si SIP, *media*, o ambos puertos están abiertos: el *host* y los puertos ciertamente no están disponibles.
 - Verificar el ACK y los datos (*media*) recibidos: no se notificó la entrega de *media* o ACK. En este nivel, el mensaje recibido es reportado como *spoofed*: del lado del atacante se puede decir que no hay involucrada una máquina de estado real SIP. La IP utilizada está falsificada o pertenece a una máquina comprometida.

```

INVITE sip:00442075005000@x SIP/2.0
Via: SIP/2.0/UDP IP hidden:58585;branch=z9hG4bKaergjerugroijrg
To: <sip:x>
From: <sip:IP hidden:58585>;tag=Zerogij34
Call-ID: 213948958-34384780214-384748@IP hidden
CSeq: 1 INVITE
Max-Forwards: 69
Contact: <sip:sip@IP hidden:58585;transport=udp>
Allow: INVITE,ACK,OPTIONS,BYE,CANCEL,NOTIFY,REFER,MESSAGE
Content-Type: application/sdp
Content-Length: 520
Session-Expires: 3600;
Allow-Events: refer..
v=0
o=sip 2147483647 1 IN IP4 1.1.1.1
s=sip
c=IN IP4 1.1.1.1
t=0 0
m=audio 29784 RTP/AVP 8 0 4 18 18 18 18 96 3 98
a=rtpmap:96 telephone-event/8000
a=sendrecv=a=ptime:20
a=rtpmap:18 G729AB/8000
a=rtpmap:18 G729B/8000
a=rtpmap:18 G729A/8000
a=rtpmap:18 G729/8000
a=rtpmap:4 G723

```

Figura 3.12: Mensaje INVITE de una traza de ataque. [Do Carmo R., 2011a]

Para disponer de un panorama general, el otro mensaje reportado debe ser considerado y correlacionado. Si todos los mensajes apuntan a la misma extensión, el correlacionador reporta un *flooding*. Si varias extensiones son alcanzadas en un período de tiempo corto, existen tres posibles conclusiones:

1. El objetivo del atacante puede ser generar un disturbio máximo al hacer sonar todos los teléfonos del dominio. Esto es probable si el ataque persiste en el tiempo.
 2. El objetivo del atacante es enumerar todas las extensiones posibles en el dominio. Esto es de esperar si una gran cantidad de extensiones son alcanzadas (parte de usuario del URI). Sin embargo, uno podría preguntarse, porque el atacante no utiliza el escaneo OPTIONS, el cual es más sigiloso. Al mismo tiempo, no se puede asumir el comportamiento lógico del atacante.
 3. La meta es identificar los *gateways* SIP/PSTN. Esto será cierto si algunos pocos mensajes tienen como objetivo varios *gateways* (parte de dominio de la solicitud URI - *request URI*).
- *Testing* contra herramientas de ataque: físicamente nuestro *testbed* está compuesto por dos máquinas: la primer máquina contiene al *honeypot* y un servidor PBX Asterisk⁶.
- La segunda máquina contiene dos máquinas virtuales vinculadas en modo puente o *bridge* una representando las herramientas de ataque y una disponiendo de un *softphone* para e-

⁶Asterisk es un framework de código abierto (*Open Source*) para el armado de aplicaciones de comunicaciones. Asterisk convierte un computadora ordinaria en un servidor de comunicaciones. Asterisk impulsa sistemas PBX IP, *gateways* VoIP, servidores de conferencia y otras soluciones personalizadas o a medida. Es implementado en pequeños y grandes negocios, call centers, *carriers* (operadores) y agencias de gobierno, alrededor del mundo. Asterisk es gratuito y *Open Source*. Asterisk está esponsoreado por Digium. [Digium©, 2015]

mular un llamante externo legal. Se utiliza un *hardphone* para representar un usuario del servidor Asterisk que debe ser protegido. El teléfono SIP (*hardphone*) registra la extensión 305 en el servidor Asterisk. El *honeypot* registra las extensiones 300-304 y 306-310 con el fin de proteger la extensión atacada por ambos lados. Otra opción es configurar Asterisk para reenviar todas las llamadas a extensiones no registradas hacia el *honeypot*.

Deabajo se desarrollan cuatro casos de estudio y se muestra como Artemisa puede reaccionar y ayudar a soportar la seguridad de la cama de prueba.

1. Detectar e interrumpir a SIPVicious: en este escenario, disparamos la herramienta SIPVicious en contra de nuestro dominio. Esta tiene tres scripts principales:

- SVMAP que identifica *hosts* con puertos SIP abiertos
- SVWAR que identifica extensiones válidas en el dominio
- SVCRACK que quiebra claves de autenticación débiles (*weak authentication passwords*) mediante la utilización de ataques de diccionario.

Así también, SVCRACK es un script provisto en la *suite* SIPVicious que puede quebrar (inhibir) la herramienta atacante. Se hace uso de este script del lado del *honeypot* cuando se detecta la herramienta SIPVicious. Este puede mostrarse en la siguiente regla:

```
[rule_sip_vicious]
type = classification
context = INVITE
pre-action = assign %a Check_Fingerprint()
condition= %a == "SIPVicious"
action = system 'python /sipvicious/svcrash.py \
-d SipMsg.Contact.Ip \
-p SipMsg.Contact.Port'
```

2. Atrapar y bloquear SPIT a nivel de *dial plan*: en este escenario, como se muestra en la Figura 3.13, se lanza la herramienta de SPIT o *Spitter tool* contra nuestro dominio. Cuando el *honeypot* detecta el arribo de las llamadas de SPIT, agrega la fuente de estas *calls* a un lista negra (*backlist*) en la Base Datos de Asterisk (AstDb). La extensión del *hardphone* es protegida por un script Interfaz de Puerta de Enlace de Asterisk (AGI) dentro de el plan de discado (*dial plan*) de Asterisk. En otras palabras, cada vez que esta extensión sea llamada, el script AGI es ejecutado primero. El script AGI verifica si la fuente de la llamada se encuentra en la *blacklisted* antes de reenviar la llamada al destino. De esta forma, las llamadas de SPIT son bloqueadas, mientras que las llamadas “buenas” se continúan encaminando. La lista negra esta basada en la dirección IP de una fuente externa y el identificador SIP URI para una fuente interna registrada.
3. Atrapar y bloquear un escaneo a nivel IP: en este escenario, se dispara un intento de escaneo en contra de nuestro dominio. Cuando el *honeypot* clasifica los mensajes recibidos como escaneo, el mismo agrega una regla al *firewall* al nivel de *firewall IP (IPTables)*⁷ para bloquear el ataque en una etapa temprana.

⁷IPTables es el programa de linea de comandos de espacio de usuario usado para configurar el conjunto de reglas de filtrado de paquetes (*packet filtering ruleset*) de Linux 2.4.x y posteriores. Está apuntado a administradores de sistemas. Considerando que el NAT también es configurado desde el conjunto de reglas de filtrado de paquetes,

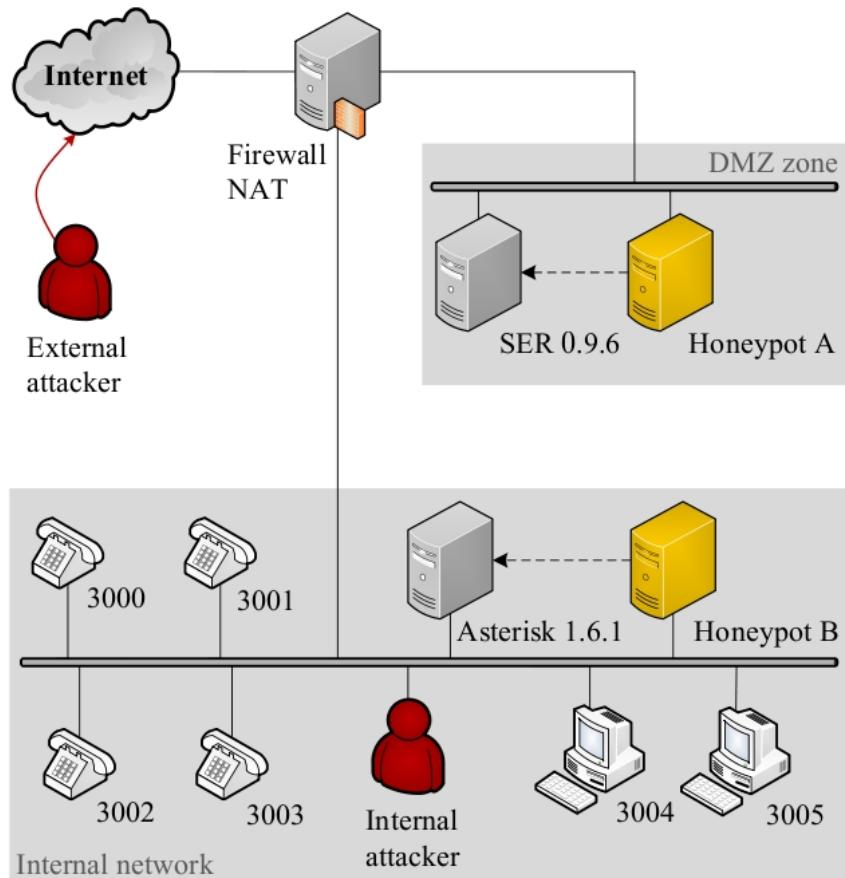


Figura 3.13: La cama de prueba (*testbed*). [Do Carmo R., 2011a]

4. *Fingerprint* de un grupo de herramientas de ataque SIP: lanzamos una serie de herramientas de hack SIP en contra del *honeypot* con el fin de probar su robustez. Particularmente, la robustez del análisis sintáctico es heredado del *stack* PJSIP. Al mismo tiempo mostramos la capacidad de *fingerprinting* del *honeypot*. Las siguientes herramientas fueron utilizadas:
 - PROTOS Test-Suit (c07-SIP): una “Prueba de Tortura” SIP que permite enviar una mayor cantidad de mensajes mal formados.
 - Sipscan: soporta escaneo REGISTER, OPTIONS e INVITE.
 - SIPVicious: enumera los servidores SIP en un rango de IPs dado a través del envío de mensajes OPTIONS o INVITE.
 - Inviteflood: herramienta simple que permite varios tipos de ataques basados en inundación.
 - Sipp: posibilita probar la performance de los servidores SIP bajo condiciones de stress.
 - Sipsak: herramienta de prueba SIP de línea de comando, útil para tests de inundación y robustez.

IPTables también se usa para esto. El paquete de IPTables también incluye ip6tables. ip6tables es usado para configurar el filtro de paquetes de IPv6. Sus principales funcionalidades son: listado del contenido del conjunto de reglas del filtrado de paquetes. Y agregado/borrado/modificación de reglas en el paquete de filtrado del conjunto de reglas. Por último, listado/puesta a cero los contadores de paquetes de filtrado del *ruleset* (conjunto de reglas). [Neira Ayuso P., 2014]

Tabla 3.1: Resultado de las pruebas. [Do Carmo R., 2011a]

| Tool | Fingerprint | Field | Real IP? | Field |
|--------------|---------------------------------------|-------------|----------|---------|
| Sipp | “Sipp” | From | Yes | From |
| Inviteflood | “Elite 1.0” | User-Agent | Yes | From |
| Spitter | “Asterisk” | From | Yes | From |
| SIPSCAN | “X-Lite” | User-Agent | Yes | Contact |
| SIPVicious | “friendly-scanner” | User-Agent | Yes | Via |
| Sipbot | “Twinkle” | User-Agent | Yes | From |
| SIP Send Fun | “Bad Guy Scripting Host” | User-Agent | No | - |
| PROTOS | ‘\d \d \d IN IP4’ where \d increments | Owner (SDP) | No | - |
| Sipsak | “sipsak” | From | Yes | Contact |
| VoIPER | “VoIPER” | From | No | - |

- Spitter: funciona sobre Asterisk y genera automáticamente llamadas con un mensaje de audio a ser entregado.
- SIP Send Fun: soporta un variedad de mensajes INVITE confusos (modificados) para *testing* de rebustez.
- SipBot: herramienta de ataque SIP controlada remotamente la cual soporta varios comandos de ataque.
- VoIPER: set de herramientas de ataque, entre ellos *fuzzing* (envío de mensajes SIP malformados) y *torturing* (para ataques de *brute force*).

El software *honeypot* presenta una buena performance en términos de robustez: no se evidenció un *crash* (colapso) incluso contra el *fuzzing* y las herramientas de inundación. También ha demostrado buen rendimiento en términos de la precisión en cuanto a la interpretación. Esta es la ventaja de la aproximación basada en reglas (*rule-based approach*) cuando tratamos con ataques bien conocidos (*well-known attacks*). Seremos capaces de identificar y sacar las huellas de todas las herramientas de ataque conocido como *fingerprinting*. Los resultados son resumidos en la Tabla 3.1. Las dos primeras columnas representan la expresión regular y los campos usados en las reglas de *fingerprinting* respectivamente. La 4ta columna indica si existe una IP real en el mensaje enviado por la herramienta o si todas las IPs están falsificadas. La última columna indica en qué encabezado o campos las IPs reales pueden ser encontradas.

Para mayor información y un ejemplo de implementación con los parámetros básicos de Artemisa *honeypot* referirse a [Do Carmo R., 2011b].

3.5. Evaluación de respuestas empíricas y enfoque en desarrollos funcionales del sistema de seguridad Artemisa a partir de su implementación y experimentación en los entornos propuestos

Se debe tener en cuenta que en los tres casos desarrollados en esta sección se considera lo estudiado en las anteriores secciones 3.4.1, 3.4.2, 3.4.3 y 3.4.4.

3.5.1. Ataques, tipos de ataques de hackers y tipos de hackers

3.5.1.1. Ataques

El recurso o activo de información que está siendo protegido de los ataques es usualmente referido como el destino u objetivo de evaluación. Este puede ser definido como un sistema, producto o componente de IT, que es identificado de requerir una evaluación de seguridad. Un ataque es un asalto deliberado en este sistema de seguridad. Los ataques pueden ser clasificados de forma general en pasivos y activos.

- Ataques activos: modifican el sistema de destino. Por ejemplo DoS ataca los recursos del objetivo disponibles en la red. Los ataques activos pueden afectar la disponibilidad, integridad, confidencialidad y autenticidad del sistema.
- Ataques pasivos: violan la confidencialidad de los datos del sistema sin afectar el estado de tal sistema, tales como las escuchas electrónicas ilegales (colectando información confidencial enviada de manera no cifrada). La palabra clave aquí es la confidencialidad.

La diferencia entre estas categorías es que mientras un ataque activo intenta alterar los recursos del sistema o afectar su operación, un ataque pasivo intenta aprender o hacer uso de la información del sistema, pero no afecta los recursos del mismo.

Los ataques también pueden categorizarse como ataques internos o externos.

- Ataque interno: es iniciado desde dentro de una red por un usuario autorizado. Esto puede ser de alguien con intenciones maliciosas, sin embargo sería apresurado darlo por asumido; un incidente involuntario también podría derivar en un daño no intencional al recurso de red.
- Ataque externo: es causado por un intruso externo que no posee autorización para acceder a la red.

3.5.1.2. Tipos de ataques de hackers

La seguridad es una preocupación crítica de cara a los brotes de intentos de intrusión, *phising*, hackeo, gusanos y virus.

Hay varias maneras mediante la cual el atacante puede obtener acceso a una red a través de la explotación de vulnerabilidades. Los ataques de hackers pueden categorizarse como:

- Ataques de OS:

- Los OS de hoy en día contienen varias características, haciéndolos cada vez más complejos. Estas características o prestaciones utilizan servicios y procesos adicionales, lo que significa una mayor cantidad de vulnerabilidades a ser explotadas por los hackers.
 - Mantener nuestro sistema con los últimos parches y actualizaciones (*patches & hotfixes*) puede ser muy desafiante con las complejas redes que se manejan actualmente. La mayoría de los parches y correcciones tienden a resolver una falla o problema inmediato, pero no proveen soluciones permanentes.
 - Los atacantes están constantemente buscando vulnerabilidades de OS para explotar. Los administradores de sistemas deben mantenerse informados de la gran variedad de nuevos *exploits*, y monitorear sus redes continuamente.
- Ataques a nivel aplicación: los desarrolladores de software generalmente se encuentran bajo una intensa presión para alcanzar los plazos de entrega, y esto puede significar que ellos no disponen de suficiente tiempo para probar completamente sus productos antes de entregarlos, dejando agujeros de seguridad no descubiertos. Esto es especialmente problemático en nuevas aplicaciones de software que vienen con una multitud de características y funcionalidades, haciéndolas cada vez más complejas de mantener. Como con los OS, mayor complejidad significa más oportunidades para vulnerabilidades.
- La seguridad no siempre es una prioridad para los desarrolladores de software, y es frecuentemente entregada como un componente agregado o algo complementario luego de lanzada la *app* (aplicativo de software). Esto significa que no todas las instancias del software van a presentar el mismo nivel de seguridad. El chequeo de errores en estas aplicaciones puede ser muy pobre o incluso inexistente, resultando en ataques de saturación de *buffer* (*buffer overflow attacks*).
- Ataques de código *shrink-wrap*: los desarrolladores de software implementarán usualmente librerías gratuitas y licencias de código de otras fuentes en sus programas. Esto significa que grandes porciones de varias partes del software van a ser exactamente iguales, y si se descubren vulnerabilidades en esas porciones de código, muchas partes del software estarán en riesgo.
- El problema es que los *developers* (desarrolladores de software) dejan las librerías y el código sin cambios. Los *developers* necesitan personalizar y afinar su código de modo no solo de hacerlo más seguro, pero lo suficientemente diferente para que el mismo código malicioso (*exploit*) no funcione.
- Ataques por malas configuraciones: incluso sistemas que son diseñados con mucha seguridad pueden ser hackeados si no son correctamente configurados. Los administradores de sistemas deben tener cuidado al momento de configurar un sistemas, y siempre saber lo que se está ejecutando. Es muy importante crear una configuración simple pero utilizable, removiendo todas las configuraciones y software innecesario.

3.5.1.3. *Hacktivism & clases de hackers*

El hacktivismo (*hacktivism*) es cuando un hacker quiebra y gana acceso a sistema de computadoras gubernamentales o corporativas en un acto de protesta. Los hactivistas (*hacktivists*) utilizan el *hacking* para acentuar su preocupación en relación a la agenda política y social, tanto en el plano *online* como *offline*. Objetivos comunes de los hactivistas incluyen agencias de

gobierno, corporaciones multinacionales, o cualquier otra entidad que perciban como una amenaza. Sin embargo, es un hecho, que ganar acceso no autorizado sigue siendo un crimen sin importar las razones o intenciones para tal acción.

Clases de hackers

Los hackers usualmente caen dentro de las siguientes categorías basados en sus actividades:

- *Black hats* (sombrero negro): usan sus habilidades para fines ilegales y maliciosos. Esta categoría de hacker está generalmente involucrada en actividades criminales y es buscado por agencias de orden público.
- *White hats* (sombrero blanco): usan sus habilidades de *hacking* para fines defensivos. Dentro de los *white hat* hackers se incluyen analistas de seguridad que tienen conocimiento acerca de contra ataques o tácticas defensivas de hackeo.
- *Gray hats* (sombrero gris): creen en la divulgación o revelación total. Ellos creen que la información es mejor mantenerla abierta que mantenerla en secreto y la persona promedio hará buen uso de esa información en lugar de abusar de la misma.
- *Suicide hackers* (hackers suicidas): son hacktivistas que quieren convertirse en mártires para sus causas. Estos intentan sabotear infraestructuras de gran escala y están totalmente dispuestos a afrontar consecuencias por sus acciones.

[EC-Council, 2010]

3.5.2. Modelo de atacante: interno y externo

Nos basamos en un modelo de atacante que posee la *suite* de herramientas *BackTrack 5 R2* con conocimientos avanzados sobre Ethernet, TCP/IP, VoIP; sin realizar modificación del código fuente de las herramientas disponibles, pero con utilización de todas sus variantes. El atacante tendrá como objetivo en la mayoría de los casos, un usuario final UA, llevando adelante intentos de detección de dispositivos y enumeración de extensiones, DoS, suplantación de identidad y derivados de los anteriores. Resultando en un perfil tanto de hacker activo como pasivo. Valiéndose principalmente de ataques por malas configuraciones del entorno. El tipo de hacker podría considerarse un *black hat* hacker intentando sabotear o inmiscuirse ilegalmente en una infraestructura SIP privada.

Plantearemos este mismo atacante en tres escenarios de configuración de red distintos.

3.5.2.1. Atacante interno

En el primer caso (caso 1) el atacante se encuentra dentro de la red LAN con medio compartido (capa 2 compartida, *hub*, WiFi). En el caso 2 la capa de enlace ya no es un medio compartido, forma parte de una red LAN con concentradores *switch* sin uso de 802.1q (VLAN) para diferenciar el tipo de tráfico. Solo fue utilizado para segmentar los escenarios en cama de prueba dentro del laboratorio.

3.5.2.2. Atacante externo

Finalmente, el caso 3, es un escenario más cotidiano, el atacante se encuentra detrás de los *routers* de frontera de la red, es un agente más de Internet.

En todo momento se utilizó tráfico SIP sin cifrar. Se presupone que el atacante no posee información de la topología lógica de la red ni numeración telefónica y llevará adelante tareas de escaneo y enumeración.

3.5.3. Caso 1: dominio de colisión y *broadcast* compartidos

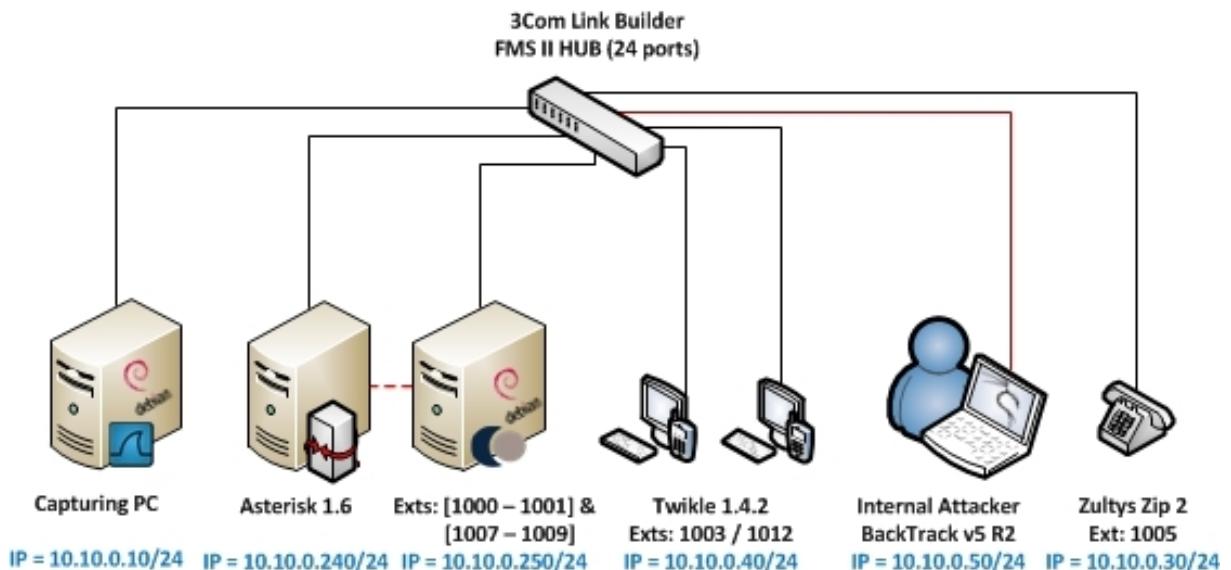


Figura 3.14: Diagrama topología *testbed testing* UBP, caso 1.

Tabla 3.2: Características y especificaciones topología *testbed testing* UBP, caso 1.

| UBP EXPERIMENTATION & TESTING 1 - TEST-BED Infrastructure | | | | | |
|---|---|---|---|-----------------------|-------------|
| DEVICE | HW specifications | OS / Firmware | SERVICE | IPv4 Network Address | GW / Domain |
| 3Com Link Builder FMS II HUB | | v1.2.3C16671 | IEEE 802.3 100BASE-TX HUB | | |
| Zultys Zip 2 IP Tel. | | ZUTS 3.5.2 | IP SIP Hard Phone - Reg. Ext: 1005 | eth - 10.10.0.30/24 | 10.10.0.240 |
| Twinkle 1.4.2 Tel. | CPU: Intel(R) Core(TM)2 Duo CPU E7400 2.80GHz RAM: 2Gb DIMM DDR2 Sync 800 MHz NetCard: RTL8101E/RTL8102E PCI Express Fast Ethernet controller (rev 02) | Linux Ubuntu 11.04 2.6.38-8-generic i686 i386 GNU/LINUX | IP SIP Soft Phone Reg. Ext: 1003 / 1012 | eth0 - 10.10.0.40/24 | 10.10.0.240 |
| Asterisk PBX | CPU: Intel(R) Pentium(R) Dual CPU E2160 1.80GHz RAM: 1Gb DIMM DDR2 Sync 533 MHz NetCard: Realtek Semiconductor Co., Ltd. RTL-8139/8139C/8139C+ (rev 10) | Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/LINUX | Asterisk v1.6.2.9-2+squeeze5 SIP Protocol (RFC 3261) IP PBX (Registrar) - Voicemail | eth0 - 10.10.0.240/24 | |
| ARTEMISA Honeypot | CPU: Intel(R) Core(TM)2 Duo CPU E8200 2.66GHz RAM: 2Gb DIMM SDRAM Sync NetCard: RTL8101E/RTL8102E PCI Express Fast Ethernet controller (rev 01) | Linux Debian 6.0.2 2.6.32-5-686 i686 GNU/LINUX | Firewall (IPTables v1.4.8-3) ARTEMISA Honeypot v1.0.91 Registered to Asterisk (10.10.0.240/24) | eth0 - 10.10.0.250 | 10.10.0.240 |
| Capaturing Terminal | CPU: Intel(R) Core(TM)2 Duo CPU E8200 2.66GHz RAM: 2Gb DIMM SDRAM Sync NetCard: RTL8101E/RTL8102E PCI Express Fast Ethernet controller (rev 01) | Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/LINUX | Wireshark 1.2.11-6+squeeze6 network traffic analyzer tcpdump 4.1.1-1 | eth1 - 10.10.0.10/24 | 10.10.0.240 |
| Attacker's Terminal | CPU: Intel(R) Atom(TM) CPU N455 1.66GHz RAM: 2Gb SODIMM Sync 667 MHz NetCard: RTL8101E/RTL8102E PCI Express Fast Ethernet controller (rev 02) | Linux Back Track 5 R2 i686 GNU/Linux | Inviteflood 2.0-bt1 RTPflood 1.0-bt0 Sipsak 0.9.6-bt0 Sipscan 0.1-bt1 Sipvicious 0.2.7-bt0 Smap 0.6.0-bt0 Metasploit v4.2.0-release | eth0 - 10.10.0.50/24 | 10.10.0.240 |

En este primer caso (caso 1) según puede verse en la Figura 3.14 el atacante se encuentra dentro de la red LAN, la misma cuenta con una capa física con medio compartido. Se utilizó concentrador de red de tipo *hub* con el fin de poder capturar el tráfico de red con más facilidad. Otro motivo muy importante para el uso de un medio compartido, es la realidad de este escenario en la actualidad con el uso de tecnología 802.11 (WiFi), ya que existe una amplia utilización de *softphones* VoIP corriendo sobre dispositivos móviles (*notebooks*, *smartphones*, etc.), expuestos a las vulnerabilidades que mostraremos a continuación, para mayor detalle del entorno referirse a la Tabla 3.2. De forma complementaria puede consultarse en el Anexo D las configuraciones de los componentes y agentes SIP para el caso 1.

En este primer caso además de mostrar la respuesta empírica de Artemisa frente a las herramientas, nos explayaremos en mostrar el modo de trabajo de las mismas para aportar entendimiento respecto de lo que enfrenta el *honeypot*.

Antes de comenzar con las herramientas de seguridad, en la Figura 3.15 se muestra el diagrama temporal de una llamada regular desde un teléfono IP a un *softphone*, para esta implementación se usó el *softphone Open Source Twinkle*.

| Time | 10.10.0.30 | 10.10.0.240 | 10.10.0.40 | Comment |
|--------|------------|-------------|------------|--|
| 0.000 | (5060) | (5060) | (5060) | SIP/SDP: Request: INVITE sip:1002@10.10.0.240:5060, with session description |
| 0.001 | (5060) | (5060) | (5060) | SIP: Status: 100 Trying |
| 0.001 | (5060) | (5060) | (5060) | SIP/SDP: Status: 200 OK, with session description |
| 0.218 | (5060) | (5060) | (5060) | SIP: Request: ACK sip:1002@10.10.0.240 |
| 0.225 | (8002) | (12430) | (12430) | RTP: PT=ITU-T G.711 PCMU, SSRC=0x88E4954, Seq=11824, Time=123041920, Mark |
| 0.226 | (8002) | (12430) | (12430) | RTP: PT=ITU-T G.711 PCMU, SSRC=0x5DF7B3EE, Seq=52804, Time=160, Mark |
| 1.827 | (5060) | (5060) | (5060) | SIP/SDP: Request: INVITE sip:phone_2@10.10.0.40, with session description |
| 1.828 | (5060) | (5060) | (5060) | SIP: Status: 100 Trying |
| 1.828 | (5060) | (5060) | (5060) | SIP: Status: 180 Ringing |
| 16.295 | (5060) | (5060) | (5060) | SIP/SDP: Status: 200 OK, with session description |
| 16.296 | (5060) | (5060) | (5060) | SIP: Request: ACK sip:phone_2@10.10.0.40 |
| 16.300 | (18314) | (8000) | (8000) | RTP: PT=ITU-T G.711 PCMU, SSRC=0x1D5C1628, Seq=51501, Time=123170560, Mark |
| 24.491 | (5060) | (5060) | (5060) | SIP: Request: BYE sip:1005@10.10.0.240 |
| 24.492 | (5060) | (5060) | (5060) | SIP: Status: 200 OK |
| 24.584 | (5060) | (5060) | (5060) | SIP: Request: BYE sip:phone_5@10.10.0.30:5060 |
| 24.645 | (5060) | (5060) | (5060) | SIP: Status: 200 OK |

Figura 3.15: Llamada regular entre dos agentes SIP finales.

La Figura 3.16 corresponde al diagrama temporal de un intento de llamada desde el *softphone* a una extensión logueada por Artemisa.

La intención de estos diagramas previos es refrescar los mensajes utilizados en una sesión VoIP-SIP normal, a fin de poder ser contrastados con los enviados por las herramientas de seguridad.

| Time | 10.10.0.30 | 10.10.0.240 | 10.10.0.250 | Comment |
|--------|-----------------------------------|-------------|-------------|--|
| 0.000 | (5060) Request: INVITE sip:(5060) | | | SIP/SDP: Request: INVITE sip:1009@10.10.0.240:5060, with session description |
| 0.001 | (5060) Status: 100 Trying | | | SIP: Status: 100 Trying |
| 0.001 | (5060) Status: 200 OK, wit | | | SIP/SDP: Status: 200 OK, with session description |
| 0.218 | (5060) Request: ACK sip:10 | | | SIP: Request: ACK sip:1009@10.10.0.240 |
| 0.225 | (5060) PT=ITU-T G.711 PCMU | | | RTP: PT=ITU-T G.711 PCMU, SSRC=0x77188B05, Seq=12899, Time=124987040, Mark |
| 0.226 | (8002) PT=ITU-T G.711 PCMU | | | RTP: PT=ITU-T G.711 PCMU, SSRC=0xE23DA8B, Seq=37611, Time=160, Mark |
| 1.827 | (8002) Request: INVITE sip:(5060) | | | SIP/SDP: Request: INVITE sip:phone_9@10.10.0.250:5060;ob, with session description |
| 1.854 | (5060) Status: 100 Trying | | | SIP: Status: 100 Trying |
| 1.855 | (5060) Status: 180 Ringing | | | SIP/SDP: Status: 180 Ringing |
| 1.858 | (5060) Status: 200 OK, wit | | | SIP/SDP: Status: 200 OK, with session description |
| 1.858 | (5060) Request: ACK sip:ph | | | SIP: Request: ACK sip:phone_9@10.10.0.250:5060;ob |
| 1.864 | (5060) PT=ITU-T G.711 PCMU | | | RTP: PT=ITU-T G.711 PCMU, SSRC=0x4BB3F3F9, Seq=8542, Time=125000160, Mark |
| 25.774 | (8002) PT=ITU-T G.711 PCMU | | | RTP: PT=ITU-T G.711 PCMU, SSRC=0x77188B05, Seq=14168, Time=125191440, Mark |
| 25.834 | (8002) PT=ITU-T G.711 PCMU | | | RTP: PT=ITU-T G.711 PCMU, SSRC=0x77188B05, Seq=14173, Time=125191920, Mark |
| 26.014 | (8002) PT=ITU-T G.711 PCMU | | | RTP: PT=ITU-T G.711 PCMU, SSRC=0x77188B05, Seq=14184, Time=125193360, Mark |
| 27.136 | (8002) PT=ITU-T G.711 PCMU | | | RTP: PT=ITU-T G.711 PCMU, SSRC=0x4BB3F3F9, Seq=9742, Time=125202320, Mark |
| 28.153 | (5060) Request: BYE sip:10 | | | SIP: Request: BYE sip:1009@10.10.0.240 |
| 28.154 | (5060) Status: 200 OK | | | SIP: Status: 200 OK |
| 28.220 | (5060) Request: BYE sip:ph | | | SIP: Request: BYE sip:phone_9@10.10.0.250:5060;ob |
| 28.246 | (5060) Status: 200 OK | | | SIP: Status: 200 OK |

Figura 3.16: Llamada regular entre un softphone y Artemisa honeypot.

3.5.3.1. Information gathering (recopilación de información)

Esta fase cuenta de dos etapas, escaneo y enumeración:

Scanning (escaneo)

En esta etapa se busca recopilar información sobre la topología, servidores y clientes. Existe el interés en encontrar *live hosts* (*hosts* activos); tipos y versiones de PBXs, servidores/pasarelas (*servers/gateways*) y clientes (*hardphones* y *softphones*). Por lo tanto en esta etapa se destinan los ataques a la dirección IP de cada *host*, en nuestro caso la dirección IP de Artemisa (10.10.0.250 utilizada en este escenario).

A continuación se realizará un desarrollo de las herramientas utilizadas: SMAP, SIP-SCAN, SVMAP, METASPLOIT.

SMAP⁸: herramienta que combina las funcionalidades de Nmap y SIPSACK. La misma tiene por objetivo detectar elementos de red con protocolo SIP habilitado. Para tal fin envía distintos mensajes de métodos de solicitud SIP (*SIP request method*) a una dirección IP puntual, o a un rango de ellas, a la espera de mensajes SIP-response. Cuenta también con la opción heredada de Nmap para identificación de tipo de agente (*user-agent fingerprinting*).

Primero realizamos un escaneo a la red completa en busca de dispositivos con protocolo SIP habilitado (sobre puerto UDP 5060 por defecto). SMAP indicará qué dispositivos han sido alcanzados a nivel de red (ICMP) y cuales de ellos cuentan con protocolo SIP habilitado. En caso de mostrar *host* activos a nivel de red pero sin SIP habilitado puede repetirse el escaneo indagando en otros puertos y sobre el protocolo TCP.

```
root@bt# ./smap 10.10.0.0/24
smap 0.6.0 <hs@123.org> http://www.wormulon.net/
10.10.0.240: ICMP reachable, SIP enabled
```

⁸SMAP: http://www.backtrack-linux.org/wiki/index.php/Pentesting_VOIP#SMAP

```

10.10.0.250: ICMP reachable, SIP enabled
10.10.0.1: ICMP unreachable, SIP disabled
10.10.0.2: ICMP unreachable, SIP disabled
10.10.0.3: ICMP unreachable, SIP disabled
...
10.10.0.99: ICMP unreachable, SIP disabled
10.10.0.100: ICMP reachable, SIP disabled
10.10.0.253: ICMP unreachable, SIP disabled
10.10.0.254: ICMP unreachable, SIP disabled
10.10.0.255: ICMP unreachable, SIP disabled
256 hosts scanned, 3 ICMP reachable, 2 SIP enabled (0.8%)

```

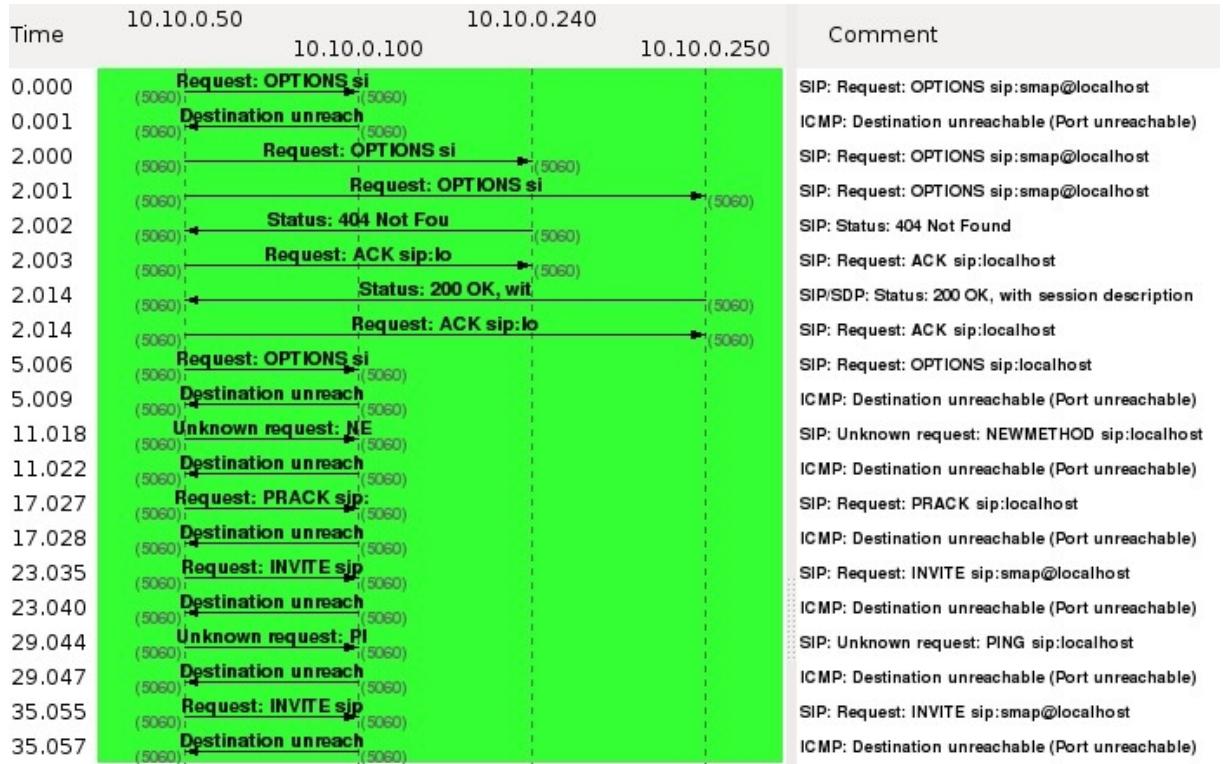


Figura 3.17: Mensajes SIP enviados a aquellas interfaces que han respondido al mensaje ICMP.

Se puede apreciar en la Figura 3.17, como SMAP envía mensajes OPTIONS a los *host* que han respondido previamente a un paquete PING-ICMP (no presente en la Figura 3.17). Aquellos *hosts* que escuchan mensajes SIP (en puerto UDP 5060) responden al método OPTIONS con una descripción de sus capacidades de sesión (*SIP session description*), indicando de este modo a SMAP que son *hosts* atractivos para continuar con un escaneo puntual. En el caso de la dirección IP 10.10.0.100, la cual no tiene SIP activo, SMAP indaga sobre ésta enviando otros tipos de mensajes SIP (INVITE, PRACK, etc.) a la espera de un SIP-RESPONSE. Artemisa, para esta configuración de la herramienta, sólo muestra el arriba un un mensaje OPTIONS, ya que por sí sólo no aparenta ser un ataque. Si lo sería, si llegan varios mensajes OPTIONS consecutivos, lo cual veremos más adelante con otras herramientas.

Salida de Artemisa por consola:

```
2012-07-11 17:29:48,877 OPTIONS message detected in extension smap from 201.252.57.89:12345
```

Para aquellos *host* que indicaron tener SIP-enabled continuaremos enviándoles un ataque puntual. En este caso apuntamos el ataque a la dirección IP de Artemisa y agregamos el modificador “-o” para obtener más información acerca del objetivo.

```

root@bt# ./smap -o 10.10.0.250
smap 0.6.0 <hs@123.org> http://www.wormulon.net/
10.10.0.250: ICMP reachable, SIP enabled
FP match: options
FP match: accept_class
FP match: accept_class
....
FP match: accept_class
FP match: options
....
FP match: accept_class
FP match: options
FP match: options
Guess: LANCOM 1[78]xx Series (VPN/VoIP/Wireless) / 6.xx.xxxx
User-Agent: D-Twinkle/1.4.2
1 host scanned, 1 ICMP reachable, 1 SIP enabled (100.0%)
1 host scanned, 1 ICMP reachable, 1 SIP enabled (100.0%)

```

El *user-agent* que detecta SMAP (D-Twinkle/1.4.2) es, justamente, el configurado para ser mostrado por Artemisa. En la Figura 3.18 se aprecian los distintos mensajes utilizados para obtener un *fingerprint* del dispositivo apuntado.

| Time | 10.10.0.50 | Comment |
|-------|--|---|
| | 10.10.0.250 | |
| 0.000 | Request: OPTIONS sip (5060) → (5060) | SIP: Request: OPTIONS sip:smap@localhost |
| 0.042 | Status: 200 OK, wit (5060) ← (5060) | SIP/SDP: Status: 200 OK, with session description |
| 0.043 | Request: ACK sip:lo (5060) → (5060) | SIP: Request: ACK sip:localhost |
| 1.000 | Request: OPTIONS sip (5060) → (5060) | SIP: Request: OPTIONS sip:localhost |
| 2.001 | Unknown request: NE (5060) → (5060) | SIP: Unknown request: NEWMETHOD sip:localhost |
| 3.001 | Request: PRACK sip (5060) → (5060) | SIP: Request: PRACK sip:localhost |
| 4.002 | Request: INVITE sip (5060) → (5060) | SIP: Request: INVITE sip:smap@localhost |
| 4.029 | Status: 100 Trying (5060) ← (5060) | SIP: Status: 100 Trying |
| 4.030 | Status: 180 Ringing (5060) ← (5060) | SIP: Status: 180 Ringing |
| 4.032 | Status: 200 OK, wit (5060) ← (5060) | SIP/SDP: Status: 200 OK, with session description |
| 4.032 | Request: ACK sip:lo (5060) → (5060) | SIP: Request: ACK sip:localhost |
| 5.002 | Unknown request: PI (5060) → (5060) | SIP: Unknown request: PING sip:localhost |
| 6.003 | Request: INVITE sip (5060) → (5060) | SIP: Request: INVITE sip:smap@localhost |
| 6.029 | Status: 100 Trying (5060) ← (5060) | SIP: Status: 100 Trying |
| 6.030 | Status: 180 Ringing (5060) ← (5060) | SIP: Status: 180 Ringing |
| 6.031 | Status: 200 OK, wit (5060) ← (5060) | SIP/SDP: Status: 200 OK, with session description |
| 6.033 | Request: ACK sip:lo (5060) → (5060) | SIP: Request: ACK sip:localhost |

Figura 3.18: Mensajes SIP enviados cuando el ataque apunta a una dirección IP puntual.

Ante la presencia de mensajes consecutivos y fundamentalmente por el mensaje INVITE, Artemisa lanza el algoritmo de detección de ataques. A continuación se puede observar un resumen del informe creado, el cual es enviado al administrador de la red y guardado como archivo de texto.

```

2012-07-11 18:02:37,658 Extension "phone_9" <sip:phone_9@10.10.0.240:5060> registered, status=200
(OK)
2012-07-11 18:02:43,186 OPTIONS message detected in extension smap from 201.252.57.89:12345
2012-07-11 18:02:44,157 OPTIONS message detected in extension from 201.252.57.89:12345
2012-07-11 18:02:47,172 INVITE message detected.

```

```

.....
2012-07-11 18:02:47,175 Call from <sip:smap@localhost> is EARLY, last code = 180 (Ringing)
2012-07-11 18:02:47,176 Call from <sip:smap@localhost> is CONNECTING, last code = 200 (OK)
.....
2012-07-11 18:02:49,174 Incoming call from <sip:smap@localhost>
***** Information about the call *****
2012-07-11 18:02:52,179
2012-07-11 18:02:52,179 From: smap in localhost:12345/udp
2012-07-11 18:02:52,179 To: smap in localhost
2012-07-11 18:02:52,179 Contact: smap in 201.252.57.89:12345/udp
2012-07-11 18:02:52,180 Connection:
2012-07-11 18:02:52,180 Owner:
2012-07-11 18:02:52,180 Via 0: 201.252.57.89:12345/udp
2012-07-11 18:02:52,180 User-Agent: smap 0.6.0
2012-07-11 18:02:52,180
***** Classification *****
.....
2012-07-11 18:02:52,705 + The message is classified as:
2012-07-11 18:02:52,705 | Interactive attack
2012-07-11 18:02:52,706 | Spoofed message
2012-07-11 18:02:52,706 | Dial plan fault
2012-07-11 18:02:52,706 | Scanning
2012-07-11 18:02:52,706 | Ringing
2012-07-11 18:02:52,706
***** Correlation *****
2012-07-11 18:02:52,706
2012-07-11 18:02:52,706 Artemisa concludes that the arrived message is likely to be:
2012-07-11 18:02:52,707
2012-07-11 18:02:52,708 * A scanning attempt.
.....
2012-07-11 18:02:52,711 This report has been saved on file ./results/2012-07-11_5.txt
2012-07-11 18:02:52,711 NOTICE This report has been saved on file ./results/2012-07-11_5.html
2012-07-11 18:02:52,711 Sending this report by e-mail...

```

Una copia completa del informe generado para este ataque se encuentra en Anexo E. Para un estudio del comportamiento de Artemisa como el que proponemos en esta sección, debemos prestar atención al ítem *correlation* donde se encuentra plasmada la conclusión a la que arribó el software. En este caso, indica correctamente que se trata de un intento de escaneo.

SIP-SCAN⁹: una herramienta simple para detectar *hosts* con SIP habilitado mediante el envío de mensajes *OPTIONS request*. No debe confundirse con *sipscan*¹⁰

Los resultados obtenidos con esta herramienta no fueron útiles desde el punto de vista del atacante, solamente detectó la presencia de Artemisa, y esto debido a los intentos del *honeypot* por recabar más datos del origen del mensaje recibido. Así mismo, no se logra concluir que es un ataque por la falta de mensajes consecutivos o la ejecución de otro tipo de método por parte de la herramienta atacante.

```

root@bt# ./sipscan -v -i eth0 10.10.0.1-254
Using interface eth0
Using own IP 10.10.0.50
Scan 10.10.0.1
.....
Scan 10.10.0.39
Scan 10.10.0.40
Scan 10.10.0.41
.....
Scan 10.10.0.239
Scan 10.10.0.240
.....
Scan 10.10.0.249

```

⁹SIP-SCAN: <http://skora.net/voice-over-ip-security>

¹⁰sipscan: herramienta de escaneo desarrollada por los autores de *Hacking VoIP exposed* (www.hackingvoip.com)

```

Scan 10.10.0.250
10.10.0.250: D-Twinkle/1.4.2
...
Scan 10.10.0.254
Done...waiting for remaining answers!

```

En el diagrama de flujo de la Figura 3.19 de la comunicación SIP se observa que el único dispositivo SIP que responde con un mensaje distinto a *404 Not Found* es Artemisa.

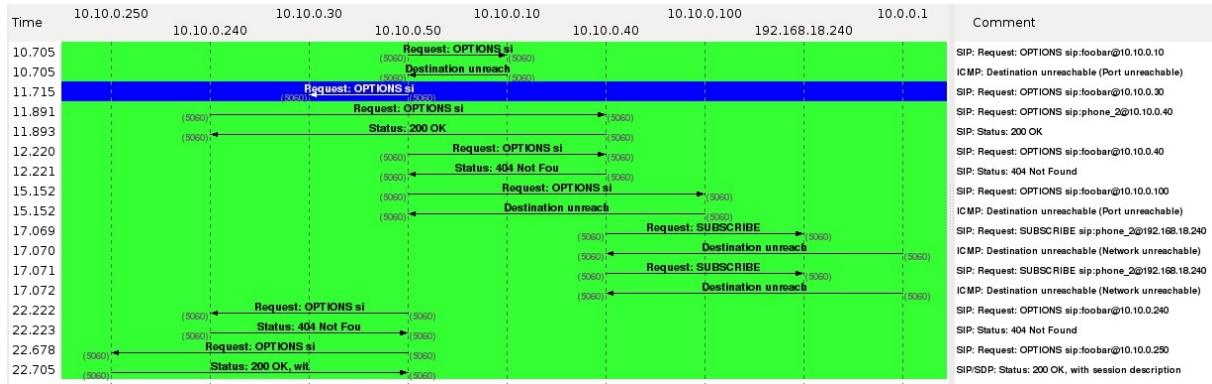


Figura 3.19: Escaneo a toda la red con la herramienta SIP-SCAN.

SVMAP¹¹: es parte de la *suite* de herramientas SIPVicious y permite escanear direcciones IP puntuales o un grupo de ellas, detectar el tipo de agente y seleccionar el método enviado (OPTIONS por defecto, REGISTER, INVITE).

La respuesta de esta herramienta es muy efectiva y precisa, enviando un mensaje OPTIONS únicamente, motivo por el cual Artemisa no pudo detectarla como puede deducirse del diagrama de flujo que corresponde a la Figura 3.20.

```
root@bt#./svmap -v 10.10.0.1-254 --fp
```

| SIP Device | User Agent | Fingerprint |
|------------------|----------------------------------|---|
| | | |
| 10.10.0.40:5060 | Twinkle/1.4.2 | T-Com Speedport W500V / Firmware v1.37 MxSF/v3.2.6.26 |
| 10.10.0.250:5060 | D-Twinkle/1.4.2 | T-Com Speedport W500V / Firmware v1.37 MxSF/v3.2.6.26 |
| 10.10.0.240:5060 | Asterisk PBX 1.6.2.9-2+squeezee6 | SJphone/1.60.289a (SJ Labs) |

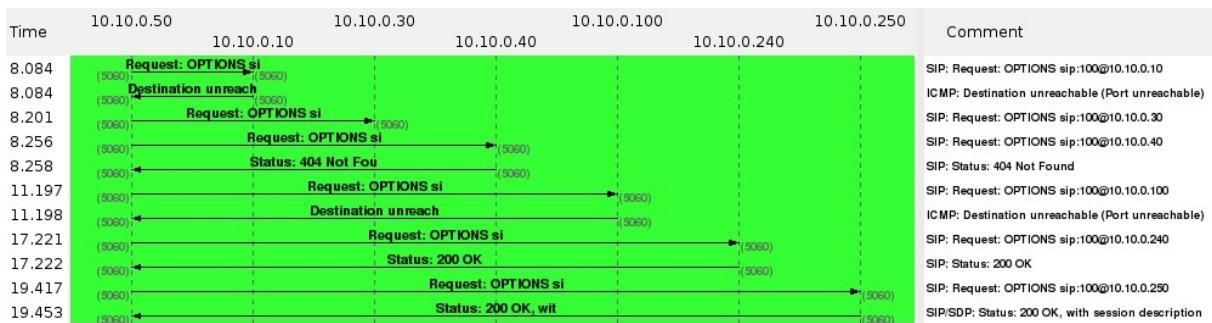


Figura 3.20: Escaneo a toda la red con SVMAP.

METASPLOIT: “Metasploit es un proyecto *Open Source* de seguridad informática que proporciona información acerca de vulnerabilidades de seguridad y ayuda en tests de penetración

¹¹SVMAP: <http://code.google.com/p/sipvicious/wiki/SvmapUsage>

y en el desarrollo de firmas para sistemas de detección de intrusos. Su subproyecto más conocido es el Metasploit Framework, una herramienta para desarrollar y ejecutar *exploits* contra una máquina remota.” [Rapid7©, 2015]

De esta *suite* de seguridad se utilizaron tres módulos explicados en cada categoría. Para esta etapa del ataque se usó el módulo auxiliar para escaneo de agentes con SIP habilitado, el cual utiliza TCP ó UDP como protocolo de transporte y permite descubrir dispositivos con SIP habilitado utilizando el método OPTIONS, lo que se refleja en la Figura 3.21.

```
msf > use auxiliary/scanner/sip/options
msf auxiliary(options) > set RHOSTS 10.10.0.0/24
RHOSTS => 10.10.0.0/24
msf auxiliary(options) > run

[*] 10.10.0.50 sip:nobody@10.10.0.0 agent='CgHua4'
[*] 10.10.0.240 404 server='Asterisk PBX 1.6.2.9-2+squeezed' verbs='INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO'
[*] 10.10.0.240 404 server='Asterisk PBX 1.6.2.9-2+squeezed' verbs='INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO'
[*] 10.10.0.50 sip:nobody@10.10.0.50 agent='EPjc8TUn'
[*] 10.10.0.240 404 server='Asterisk PBX 1.6.2.9-2+squeezed' verbs='INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO'
[*] 10.10.0.250 200 agent='D-Twinkle/1.4.2' verbs='PRACK, INVITE, ACK, BYE, CANCEL, UPDATE, SUBSCRIBE, NOTIFY, REFER, MESSAGE, OPTIONS'
[*] 10.10.0.50 sip:nobody@10.10.0.255 agent='6AsY'
[*] 10.10.0.240 404 server='Asterisk PBX 1.6.2.9-2+squeezed' verbs='INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO'
[*] 10.10.0.240 404 server='Asterisk PBX 1.6.2.9-2+squeezed' verbs='INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO'
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
```

Del mismo modo que la herramienta anterior, METASPLOIT con el envío de mensajes OPTIONS únicamente, logró obtener la información deseada para la etapa de escaneo.

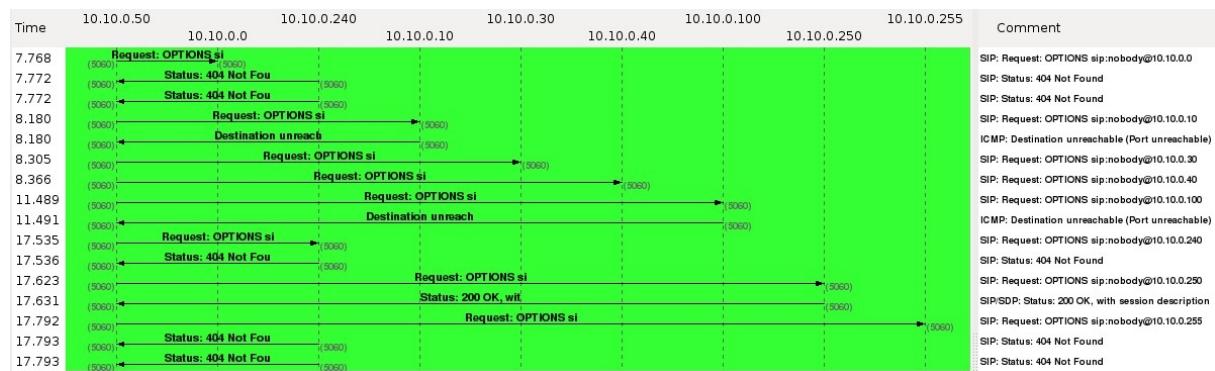


Figura 3.21: Escaneo a toda la red con módulo de escaneo de METAEXPLOIT.

Artemisa detecta este mensaje pero no dispara el algoritmo de detección por no tratarse de más de tres mensajes consecutivos.

```
2012-07-12 20:32:57,202 OPTIONS message detected in extension nobody from 10.10.0.50:5060
```

3.5.3.2. *Enumerating* (enumeración)

Consiste en descubrir extensiones SIP válidas para el entorno, lo que permite luego apuntar los ataques de *brute force* (fuerza bruta) y demás sobre tales cuentas. Trabaja analizando

los mensajes de error ante las solicitudes de métodos OPTIONS, INVITE y REGISTER enviadas apuntando a distintas extensiones para determinar cuales son válidas en el entorno SIP en cuestión. Herramientas utilizadas para este tipo de ataque fueron: SVWAR y METASPLOIT.

Para un resultado útil, desde el punto de vista del atacante, los mensajes deben ser enviados al servidor de registro Asterisk para este escenario de pruebas. Así mismo, también pusimos a prueba al *honeypot* como objetivo directo del ataque, ya que puede configurarse a Artemisa que devuelva como su *fingerprint*, el de un Asterisk PBX por ejemplo, y convertirse de este modo en objeto de una enumeración. Debe tenerse en cuenta que un factor importante del éxito del ataque, es el diccionario o rango de extensiones utilizadas con la herramienta, en nuestro diccionario agregamos intencionalmente, valores de extensiones utilizados en el entorno.

SVWAR¹²: es una herramienta que permite enumerar extensiones probando con valores dentro de un rango o tomados de un archivo de diccionario. Puede enviar mensajes de métodos OPTIONS, INVITE y REGISTER.

Con la dirección IP de Asterisk como objetivo, se obtuvo la enumeración deseada utilizando los métodos INVITE y REGISTER, no así con el método OPTIONS. Esto, ya que con los dos primeros, Asterisk devuelve un mensaje (*401 Unauthorized*) con los parámetros necesarios para la autenticación cuando se trata de una extensión registrable en este servidor ó de un nombre de usuario de extensión registrable en este servidor. Si no se trata de extensión propia de este servidor, Asterisk devuelve un mensaje *100 Trying* y luego un *200 OK* indicando que trata de cursar la llamada.

```
root@bt#./svwar -d dict 10.10.0.240 -m INVITE
| Extension      | Authentication |
-----
| phone_4        | reqauth          |
| phone_5        | reqauth          |
| phone_2        | reqauth          |
| phone_3        | reqauth          |
| phone_0        | reqauth          |
| phone_1        | reqauth          |
```

| Time | 10.10.0.50 | Comment |
|--------|-----------------------------------|---|
| | 10.10.0.240 | |
| 0.000 | Request: INVITE sip:(5060) | SIP: Request: INVITE sip:2370167112@10.10.0.240 |
| 0.001 | Status: 100 Trying | SIP: Status: 100 Trying |
| 0.002 | Status: 200 OK, wit | SIP/SDP: Status: 200 OK, with session description |
| 1.001 | Status: 200 OK, wit | SIP/SDP: Status: 200 OK, with session description |
| 2.001 | Status: 200 OK, wit | SIP/SDP: Status: 200 OK, with session description |
| 4.001 | Status: 200 OK, wit | SIP/SDP: Status: 200 OK, with session description |
| 8.001 | Status: 200 OK, wit | SIP/SDP: Status: 200 OK, with session description |
| 12.000 | Status: 200 OK, wit | SIP/SDP: Status: 200 OK, with session description |
| 15.999 | Status: 200 OK, wit | SIP/SDP: Status: 200 OK, with session description |

(a) Extensión NO perteneciente al dominio SIP.

| Time | 10.10.0.50 | Comment |
|--------|-----------------------------------|--|
| | 10.10.0.240 | |
| 0.000 | Request: INVITE sip:(5060) | SIP: Request: INVITE sip:phone_0@10.10.0.240 |
| 0.001 | Status: 401 Unautho | SIP: Status: 401 Unauthorized |
| 1.001 | Status: 401 Unautho | SIP: Status: 401 Unauthorized |
| 2.001 | Status: 401 Unautho | SIP: Status: 401 Unauthorized |
| 4.000 | Status: 401 Unautho | SIP: Status: 401 Unauthorized |
| 8.000 | Status: 401 Unautho | SIP: Status: 401 Unauthorized |
| 11.999 | Status: 401 Unautho | SIP: Status: 401 Unauthorized |
| 15.999 | Status: 401 Unautho | SIP: Status: 401 Unauthorized |

(b) Extensión o nombre de usuario de extensión válido para el servidor de registro.

Figura 3.22: Mensajes intercambiados entre SVWAR y el servidor de registro para distintas extensiones.

En las Figuras 3.22a y 3.22b se pueden observar los mensajes explicados anteriormente, los cuales se repiten modificando el *status code* del campo *status line*, seis reenvíos sin tener respuesta. La diferencia en utilizar el ataque con método REGISTER o INVITE radica en el tratamiento que hará Asterisk con el mensaje. Para el primero solamente genera un registro de

¹²SVWAR: <http://code.google.com/p/SIPVicious/wiki/Svwarusage>

error que veremos a continuación, cuando se intenta registrar una extensión no perteneciente al dominio.

```
[Jul 11 20:49:09] NOTICE[2825] chan_sip.c: Registration from ->
'2348559351"<sip:2348559351@10.10.0.240>' failed for '10.10.0.50' ->
No matching peer found
```

En cambio, si se trata de un mensaje INVITE, y de una extensión que se encuentra registrada, el proxy SIP generará un nuevo mensaje INVITE con destino al UA propietario de la extensión en cuestión tal puede verificarse en la Figura 3.23.

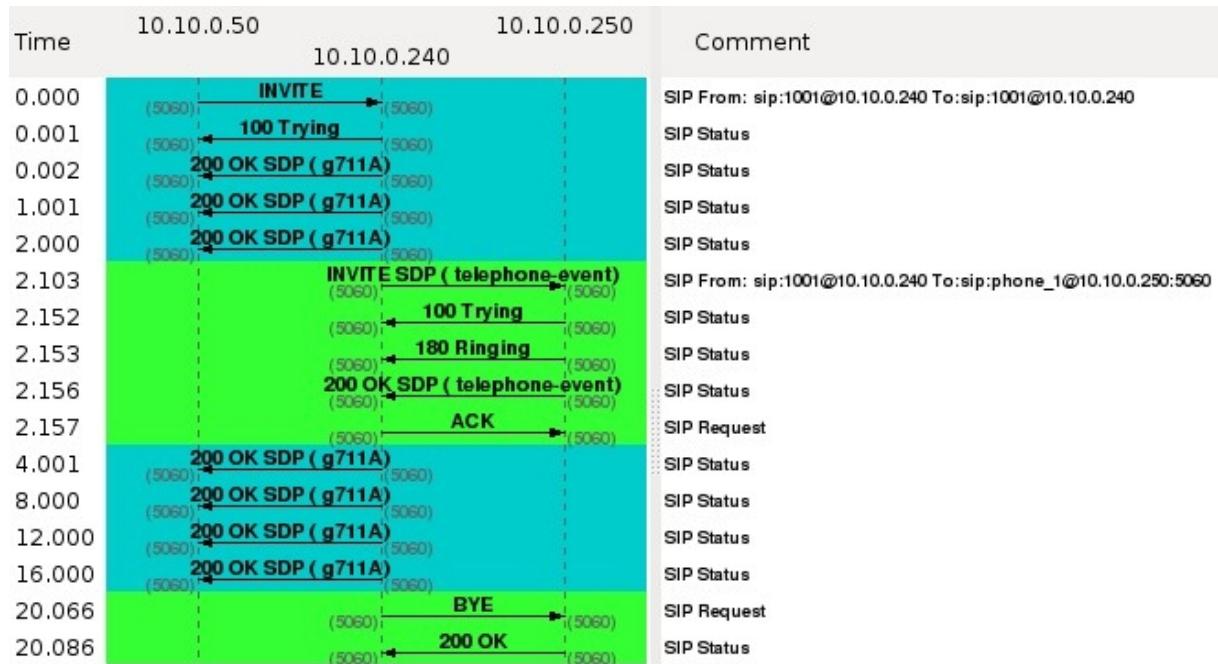


Figura 3.23: Dialogo completo provocado por el mensaje INVITE.

Es en esta situación donde Artemisa, al recibir un mensaje INVITE apuntado a una de sus extensiones logueadas, detecta el mensaje y comienza a recopilar información.

```
2012-07-11 20:51:59,338 INVITE message detected.
2012-07-11 20:51:59,341 Incoming call from "1001" <sip:1001@10.10.0.240>
2012-07-11 20:51:59,342 Call from "1001" <sip:1001@10.10.0.240> is EARLY, last code = 180 (Ringing)
2012-07-11 20:51:59,345 Call from "1001" <sip:1001@10.10.0.240> is CONNECTING, last code = 200 (OK)
...
2012-07-11 20:51:59,357 Call from "1001" <sip:1001@10.10.0.240> is CONFIRMED, last code = 200 (OK)
2012-07-11 20:52:04,345 ***** Information about the call *****
2012-07-11 20:52:04,345 From: 1001 in 10.10.0.240:5060/udp
2012-07-11 20:52:04,346 To: phone_1 in 10.10.0.250
2012-07-11 20:52:04,346 Contact: 1001 in 10.10.0.240:5060/udp
2012-07-11 20:52:04,346 Connection: 10.10.0.240
2012-07-11 20:52:04,346 Owner: 10.10.0.240
2012-07-11 20:52:04,346 Via 0: 10.10.0.240:5060/udp
2012-07-11 20:52:04,346 User-Agent: Asterisk PBX 1.6.2.9-2+squeeze6
2012-07-11 20:52:04,346 Classification *****
```

Como podemos apreciar en el fragmento de la salida de Artemisa, la información que llega tiene como fuente al *proxy server*, de modo que no podemos obtener datos sobre el verdadero origen del ataque. Igualmente, dependiendo de la cantidad de mensajes INVITE que llegan se puede inferir sobre el tipo de ataque (*scanning*, *flooding*, etc). Es importante destacar que si se

utiliza SER como *register server*, el campo *user-agent* de la cabecera SIP no es enmascarado, motivo por el cual Artemisa podría determinar e indicar la presencia de un ataque utilizando alguna de las herramientas de la *suite* SIPVicious. También hay que destacar que aunque la herramienta no logró su cometido de detectar las extensiones válidas del dominio, por la frecuencia de arriba de los mensajes, Artemisa detecta la presencia de un ataque cuando utilizamos el método OPTIONS.

Cuando apuntamos el ataque a la dirección IP de Artemisa, sin importar el método utilizado, SVWAR no logra enumerar las extensiones por no recibir por parte de Artemisa ningún mensaje solicitando autenticación. Sin embargo el *honeypot* si es capaz de recopilar datos acerca del atacante. Ya sea con método INVITE u OPTIONS, Artemisa indica intento de escaneo o inundación dependiendo de la frecuencia de arriba de los mensajes, si son mensajes esporádicos o continuos respectivamente. Lo importante aquí es que detecta ciertamente que el ataque fue realizado con la herramienta SIPVicious, esto gracias al (*fingerprinting*), proceso mediante el cual llevamos a cabo la correlación de huella que nos facilita dicho resultado.

```

2012-07-11 20:46:18,122 INVITE message detected.
2012-07-11 20:46:18,122 Waiting for SIP messages (5)...
2012-07-11 20:46:18,124 Incoming call from "4184408506" <sip:4184408506@10.10.0.250>
2012-07-11 20:46:18,125 Call from "4184408506" <sip:4184408506@10.10.0.250> is EARLY, last code =
180 (Ringing)
2012-07-11 20:46:18,126 Call from "4184408506" <sip:4184408506@10.10.0.250> is CONNECTING, last code
= 200 (OK)
2012-07-11 20:46:18,177 INVITE message detected.
...
2012-07-11 20:46:18,186 The maximum number of calls to simultaneously analyze has been reached.
2012-07-11 20:46:18,186 INVITE message detected.
2012-07-11 20:46:18,187 The maximum number of calls to simultaneously analyze has been reached.
...
2012-07-11 20:46:18,419 INVITE message detected.
2012-07-11 20:46:18,420 The maximum number of calls to simultaneously analyze has been reached.
2012-07-11 20:46:19,124 Waiting for SIP messages (4)...
...
2012-07-11 20:46:23,129 ***** Information about the call *****
2012-07-11 20:46:23,129
2012-07-11 20:46:23,129 From: 4184408506 in 10.10.0.250:5060/udp
2012-07-11 20:46:23,129 To: 4184408506 in 10.10.0.250
2012-07-11 20:46:23,129 Contact: 4184408506 in 10.10.0.250:5060/udp
2012-07-11 20:46:23,129 Connection:
2012-07-11 20:46:23,129 Owner:
2012-07-11 20:46:23,130 Via 0: 127.0.1.1:5060/udp
2012-07-11 20:46:23,130 User-Agent: friendly-scanner
...
2012-07-11 20:46:25,447 ***** Correlation *****
2012-07-11 20:46:25,447
2012-07-11 20:46:25,447 Artemisa concludes that the arrived message is likely to be:
2012-07-11 20:46:25,448
2012-07-11 20:46:25,448 * The attack was created employing the tool SIPVicious.
2012-07-11 20:46:25,448 * A flooding attack.
2012-07-11 20:46:25,448

```

Puede verse en la salida de consola de Artemisa, que luego de un determinado número de mensajes consecutivos (tres configurados en este caso), se comienzan a ignorar mensajes posteriores por un cierto lapso de tiempo. Esto mismo se muestra a continuación en el diagrama de flujo en la Figura 3.24.

Con el método REGISTER (apuntando a un *user-agent client*), SVWAR devuelve un error y termina su ejecución sin generar salida por consola.

METASPLOIT: el módulo auxiliar scanner/SIP/enumerator nos permite enumerar extensiones utilizando los métodos OPTIONS ó REGISTER también sobre TCP ó UDP.

| Time | 10.10.0.250 | 10.10.0.240 | 10.10.0.50 | 10.10.0.40 | 192.168.18.240 | Comment |
|-------|-------------|---------------------|------------|------------|----------------|---|
| 0.000 | (5060) | Request: INVITE sip | | (5060) | | SIP: Request: INVITE sip:4184408506@10.10.0.250 |
| 0.034 | (5060) | Status: 100 Trying | | (5060) | | SIP: Status: 100 Trying |
| 0.035 | (5060) | Status: 180 Ringing | | (5060) | | SIP: Status: 180 Ringing |
| 0.037 | (5060) | Status: 200 OK, wit | | (5060) | | SIP/SDP: Status: 200 OK, with session description |
| 0.044 | (5060) | Request: INVITE sip | | (5060) | | SIP: Request: INVITE sip:1001@10.10.0.250 |
| 0.050 | (5060) | Request: INVITE sip | | (5060) | | SIP: Request: INVITE sip:1002@10.10.0.250 |
| 0.056 | (5060) | Request: INVITE sip | | (5060) | | SIP: Request: INVITE sip:1003@10.10.0.250 |
| 0.063 | (5060) | Request: INVITE sip | | (5060) | | SIP: Request: INVITE sip:1004@10.10.0.250 |
| 0.069 | (5060) | Request: INVITE sip | | (5060) | | SIP: Request: INVITE sip:1005@10.10.0.250 |
| 0.075 | (5060) | Request: INVITE sip | | (5060) | | SIP: Request: INVITE sip:1006@10.10.0.250 |
| 0.081 | (5060) | Request: INVITE sip | | (5060) | | SIP: Request: INVITE sip:1007@10.10.0.250 |
| 0.086 | (5060) | Request: INVITE sip | | (5060) | | SIP: Request: INVITE sip:1008@10.10.0.250 |
| 0.089 | (5060) | Status: 100 Trying | | (5060) | | SIP: Status: 100 Trying |
| 0.090 | (5060) | Status: 180 Ringing | | (5060) | | SIP: Status: 180 Ringing |
| 0.091 | (5060) | Status: 200 OK, wit | | (5060) | | SIP/SDP: Status: 200 OK, with session description |
| 0.094 | (5060) | Status: 100 Trying | | (5060) | | SIP: Status: 100 Trying |
| 0.095 | (5060) | Status: 180 Ringing | | (5060) | | SIP: Status: 180 Ringing |
| 0.096 | (5060) | Status: 200 OK, wit | | (5060) | | SIP/SDP: Status: 200 OK, with session description |
| 0.096 | (5060) | Status: 486 Busy He | | (5060) | | SIP: Status: 486 Busy Here |
| 0.097 | (5060) | Status: 486 Busy He | | (5060) | | SIP: Status: 486 Busy Here |
| 0.098 | (5060) | Status: 486 Busy He | | (5060) | | SIP: Status: 486 Busy Here |
| 0.099 | (5060) | Status: 486 Busy He | | (5060) | | SIP: Status: 486 Busy Here |
| 0.100 | (5060) | Status: 486 Busy He | | (5060) | | SIP: Status: 486 Busy Here |
| 0.101 | (5060) | Status: 486 Busy He | | (5060) | | SIP: Status: 486 Busy Here |
| 0.109 | (5060) | Request: INVITE sip | | (5060) | | SIP: Request: INVITE sip:1009@10.10.0.250 |
| 0.115 | (5060) | Request: INVITE sip | | (5060) | | SIP: Request: INVITE sip:1010@10.10.0.250 |
| 0.121 | (5060) | Request: INVITE sip | | (5060) | | SIP: Request: INVITE sip:1015@10.10.0.250 |
| 0.127 | (5060) | Request: INVITE sip | | (5060) | | SIP: Request: INVITE sip:1016@10.10.0.250 |

Figura 3.24: Protección de Artemisa *honeypot* contra inundación de mensajes.

El comportamiento de este módulo de METASPLOIT es similar al mismo tipo de ataque utilizando SVWAR, pero existen dos diferencias importantes que trataremos a continuación. La primera de ellas es que no permite hacer uso de un diccionario como fuente de nombres de extensiones para probar, sólo puede configurarse un rango de extensiones numéricas, lo que en nuestro caso, impidió a la herramienta enumerar exitosamente. La segunda diferencia con SVWAR es que no finaliza su ejecución al apuntar el ataque a la dirección IP de un *user-agent client*, de modo que podremos observar el comportamiento de Artemisa ante mensajes REGISTER apuntados a su dirección IP. Artemisa recibe los mensajes REGISTER sin emitir respuesta alguna a estos, ya que no es menester de un cliente SIP de extremo final el proceso de desafío de autenticación que debería seguir a mensajes de este tipo. Más allá de no haber respuesta, igualmente puede detectarse la existencia de un intento de denegación de servicio mediante un (*flooding attack*). También puede obtenerse la IP de la fuente, aunque no debe confiarse en este dato ya que puede ser falsificada con la misma herramienta.

```

2012-07-12 20:48:46,912 **** REGISTER message ****
2012-07-12 20:48:46,912 To:
2012-07-12 20:48:46,913
2012-07-12 20:48:46,913 From: 10.10.0.50:5060
2012-07-12 20:48:46,913 To:
2012-07-12 20:48:46,913
2012-07-12 20:48:46,913 From: 10.10.0.50:5060
2012-07-12 20:48:46,914 **** REGISTER analysis*****
2012-07-12 20:48:46,914 The REGISTER message is detected as a flood attack!
2012-07-12 20:48:46,914
2012-07-12 20:48:46,914 The REGISTER message is detected as a flood attack!
2012-07-12 20:48:46,914 * A flooding attack.

```

2012-07-12 20:48:46,914

Los mensajes generados con este módulo de METASPLOIT son mal formados, motivo por el cual pueden ser descartados según el agente de usuario, y en nuestro caso, lleva a que Artemisa no muestre los nombres de las extensiones que se utilizan en el ataque.

3.5.3.3. Monitoreo de tráfico y espionaje de llamadas.

BackTrack cuenta con varias herramientas pensadas para monitoreo de tráfico y espionaje utilizando técnicas como ARP-spoofing, las cuales aprovechan vulnerabilidades subyacentes a SIP/VoIP propias de las capas de protocolos de red inferiores para las cuales Artemisa se encuentra totalmente indefenso y escapa a sus capacidades y objetivos.

3.5.3.4. Denegación de Servicio.

Un ataque de denegación de servicio DoS apunta a afectar la disponibilidad del servicio de VoIP enviando grandes cantidades de datos en general o solicitudes específicas de un protocolo, para consumir los recursos de la red o sobrecargar un servidor específico respectivamente. Herramientas utilizadas: Inviteflood y Rtpflood.

Inviteflood: es una herramienta que realiza una inundación de mensajes SIP-INVITE sobre TCP/IP. Inviteflood realiza envío de mensajes únicamente y no responde cuando es consultada desde el *host* objetivo.

```
root@bt#./inviteflood eth0 1001 10.10.0.250 10.10.0.40 100
inviteflood - Version 2.0
                June 09, 2006
source IPv4 addr:port      = 10.10.0.50:9
dest   IPv4 addr:port      = 10.10.0.250:5060
targeted UA                 = 1000@10.10.0.240
Flooding destination with 100 packets
sent: 1
sent: 2
sent: 3
....
sent: 98
sent: 99
sent: 100
```

Vale aclarar que, si bien la herramienta no responde a los mensajes SIP recibidos debido a los ataques generados, si se crea, un paquete de respuesta a nivel TCP(UDP)/IP. La mínima cantidad de mensajes que se generan apuntando el ataque a un cliente SIP es de tres mensajes, dos SIP y uno ICMP por cada uno generado por la herramienta, tal se aprecia en la Figura 3.25.

Cuando el ataque es apuntado a un servidor SIP, el número de paquetes intercambiados aumenta a QUINCE paquetes por cada uno enviado por la herramienta, esto último debido a que el servidor envía siete veces el mensaje 200-OK en caso de no recibir el ACK desde su par, como se evidencia en la Figura 3.26.

Es entonces notable el tráfico que genera en la red la herramienta INVITEFLOOD con su consecuente procesamiento. En caso de ser apuntado el ataque a Artemisa, el intento de obtener más información del posible atacante, genera un tráfico de 19 paquetes por cada uno enviado

| Time | 10.10.0.50 | Comment |
|-------|--|---|
| | 10.10.0.40 | |
| 4.342 | Request: INVITE sip (9) (5059) | SIP/SDP: Request: INVITE sip:1001@10.10.0.245, with session description |
| 4.346 | Status: 100 Trying (9) (5059) | SIP: Status: 100 Trying |
| 4.346 | Destination unreachable (5059) (9) | ICMP: Destination unreachable (Port unreachable) |

Figura 3.25: Diálogo mínimo al utilizar Inviteflood.

| Time | 10.10.0.50 | Comment |
|--------|--|---|
| | 10.10.0.245 | |
| 2.641 | Request: INVITE sip (9) (5060) | SIP/SDP: Request: INVITE sip:1001@10.10.0.245, with session description |
| 2.642 | Status: 100 Trying (9) (5060) | SIP: Status: 100 Trying |
| 2.643 | Status: 200 OK, wit (9) (5060) | SIP/SDP: Status: 200 OK, with session description |
| 2.645 | Destination unreachable (5060) (9) | ICMP: Destination unreachable (Port unreachable) |
| 2.645 | Destination unreachable (5060) (9) | ICMP: Destination unreachable (Port unreachable) |
| 3.643 | Status: 200 OK, wit (9) (5060) | SIP/SDP: Status: 200 OK, with session description |
| 4.643 | Status: 200 OK, wit (9) (5060) | SIP/SDP: Status: 200 OK, with session description |
| 6.643 | Status: 200 OK, wit (9) (5060) | SIP/SDP: Status: 200 OK, with session description |
| 6.644 | Destination unreachable (5060) (9) | ICMP: Destination unreachable (Port unreachable) |
| 10.643 | Status: 200 OK, wit (9) (5060) | SIP/SDP: Status: 200 OK, with session description |
| 10.643 | Destination unreachable (5060) (9) | ICMP: Destination unreachable (Port unreachable) |
| 14.643 | Status: 200 OK, wit (9) (5060) | SIP/SDP: Status: 200 OK, with session description |
| 14.643 | Destination unreachable (5060) (9) | ICMP: Destination unreachable (Port unreachable) |
| 18.643 | Status: 200 OK, wit (9) (5060) | SIP/SDP: Status: 200 OK, with session description |
| 18.643 | Destination unreachable (5060) (9) | ICMP: Destination unreachable (Port unreachable) |

Figura 3.26: Mensajes generados cuando el objetivo del ataque es un SIP registrar.

por INVITEFOOD. Debido a que lo indicado puede perjudicar la estabilidad de la red y del *host* donde reside Artemisa, es que el *honeypot* atenderá solo el numero de mensajes configurados mediante MAX_CALLS en *artemisa.conf*. Superado este número responderá al ataque con un mensaje “486 BUSY HERE”, puede observarse gráficamente en la Figura 3.24.

```

2012-07-12 13:12:02,269 INVITE message detected.
2012-07-12 13:12:02,269 Waiting for SIP messages (5)...
2012-07-12 13:12:02,271 Incoming call from <sip:10.10.0.50>
2012-07-12 13:12:02,272 Call from <sip:10.10.0.50> is EARLY, last code = 180 (Ringing)
...
2012-07-12 13:12:02,276 Call from <sip:10.10.0.50> is CONNECTING, last code = 200 (OK)
...
2012-07-12 13:12:02,291 The maximum number of calls to simultaneously analyze has been reached.
...
2012-07-12 13:12:07,277 ***** Information about the call *****
2012-07-12 13:12:07,277
2012-07-12 13:12:07,277 From: in 10.10.0.50:9/udp
...
2012-07-12 13:12:07,278 User-Agent: Elite 1.0 Brcm Callctrl/1.5.1.0 MxSF/v.3.2.6.26
2012-07-12 13:12:07,278
2012-07-12 13:12:07,278 ***** Classification *****
2012-07-12 13:12:07,278
2012-07-12 13:12:07,278 + Checking fingerprint...
2012-07-12 13:12:07,278 |
2012-07-12 13:12:07,278 | User-Agent: Elite 1.0 Brcm Callctrl/1.5.1.0 MxSF/v.3.2.6.26
2012-07-12 13:12:07,278 |
2012-07-12 13:12:07,279 | Fingerprint found. The following attack tool was employed: inviteflood
2012-07-12 13:12:07,279 |

```

```

2012-07-12 13:12:07,279 | Category: Attack tool
2012-07-12 13:12:07,279
...
2012-07-12 13:12:08,401 ***** Correlation *****
2012-07-12 13:12:08,401
2012-07-12 13:12:08,401 Artemisa concludes that the arrived message is likely to be:
2012-07-12 13:12:08,401
2012-07-12 13:12:08,401 * The attack was created employing the tool inviteflood.
2012-07-12 13:12:08,402 * A flooding attack.

```

RTP Flood: crea paquetes RTP válidos (*well-formed packet*) que son enviados en multitud hacia un teléfono IP (*softphone* o *hardphone*) ó un proxy RTP. Por tratarse de un agente SIP exclusivamente, Artemisa no atiende este tipo de ataque.

3.5.3.5. Suplantación de identificación de llamadas

En entornos de VoIP la suplantación de identidad es tan simple como modificar el campo *From* de la cabecera del mensaje INVITE. El destino de su utilización puede obedecer a necesidades de privacidad o intenciones de estafa. Herramientas utilizadas: Inviteflood y METASPLOIT.

Inviteflood y METASPLOIT: en este caso utilizaremos estas herramientas para modificar el campo *From* de mensajes INVITE adulterados. Las analizaremos juntas debido a que el mensaje resultado que generan tiene una gran similitud. Cuando utilizamos estas herramientas apuntando a Asterisk pero con extensiones pertenecientes a Artemisa, esta última no puede detectar la falsedad del mensaje porque este pertenece al UA indicado en el campo *From*, el cual indica a Asterisk. Un caso distinto es cuando el ataque es apuntado a la dirección IP de Artemisa sin importar la extensión que se coloque, aquí el *honeypot* sí podrá evaluar los datos presentados en el campo *From* y detectar la presencia de un mensaje adulterado. Más aún, cuando se utiliza la herramienta Inviteflood, Artemisa acusa su uso.

```

...
2012-07-12 21:03:41,266 ***** Classification *****
2012-07-12 21:03:41,266
2012-07-12 21:03:41,266 + Checking fingerprint...
2012-07-12 21:03:41,266 |
2012-07-12 21:03:41,266 |
2012-07-12 21:03:41,280 |
2012-07-12 21:03:41,280 | No fingerprint found.
2012-07-12 21:03:41,280
2012-07-12 21:03:41,280 + Checking DNS...
2012-07-12 21:03:41,280 |
2012-07-12 21:03:41,280 | + Checking 192.168.1.1...
2012-07-12 21:03:41,281 |
2012-07-12 21:03:41,281 | | This is already an IP address. Nothing done.
2012-07-12 21:03:41,281 |
2012-07-12 21:03:41,281 | + Checking 127.0.0.1...
2012-07-12 21:03:41,281 |
2012-07-12 21:03:41,281 | | This is already an IP address. Nothing done.
2012-07-12 21:03:41,281 |
2012-07-12 21:03:41,281 | + Checking ...
2012-07-12 21:03:41,282 |
2012-07-12 21:03:41,282 | | IP cannot be resolved.
2012-07-12 21:03:41,282 |
2012-07-12 21:03:41,282 | | Category: Spoofed message
2012-07-12 21:03:41,282 |
2012-07-12 21:03:41,282 | + Checking 10.10.0.250...
2012-07-12 21:03:41,282 |
2012-07-12 21:03:41,282 | | This is already an IP address. Nothing done.

```

```

2012-07-12 21:03:41,282
2012-07-12 21:03:41,283 + Checking if SIP port is opened...
2012-07-12 21:03:41,283 |
2012-07-12 21:03:41,283 | + Checking 127.0.0.1:5060/udp...
2012-07-12 21:03:41,283 | |
2012-07-12 21:03:41,391 | | Port state: closed
2012-07-12 21:03:41,391 | |
2012-07-12 21:03:41,391 | | Category: Spoofed message
2012-07-12 21:03:41,391
2012-07-12 21:03:41,391 + Checking if media port is opened...
2012-07-12 21:03:41,392 |
2012-07-12 21:03:41,392 | No RTP info delivered.
2012-07-12 21:03:41,392 |
2012-07-12 21:03:41,392 | Category: Spoofed message
2012-07-12 21:03:41,392
2012-07-12 21:03:41,392 + Checking request URI...
2012-07-12 21:03:41,392 |
2012-07-12 21:03:41,392 | Extension in field To:
2012-07-12 21:03:41,392 |
2012-07-12 21:03:41,393 | Request addressed to the honeypot? No
2012-07-12 21:03:41,393
2012-07-12 21:03:41,393 + Checking if proxy in Via...
2012-07-12 21:03:41,393 |
2012-07-12 21:03:41,393 | + Checking 10.10.0.250:/udp...
2012-07-12 21:03:41,393 | |
2012-07-12 21:03:41,393 | | Error while scanning.
2012-07-12 21:03:41,393 | |
2012-07-12 21:03:41,394 | | Category: -
2012-07-12 21:03:41,394
2012-07-12 21:03:41,394 + Checking for ACK...
2012-07-12 21:03:41,394 |
2012-07-12 21:03:41,394 | ACK received: No
2012-07-12 21:03:41,394 |
2012-07-12 21:03:41,394 | Category: Scanning
2012-07-12 21:03:41,394
2012-07-12 21:03:41,394 + Checking for received media...
2012-07-12 21:03:41,395 |
2012-07-12 21:03:41,395 | Media received: No
2012-07-12 21:03:41,395 |
2012-07-12 21:03:41,395 | Category: Ringing
2012-07-12 21:03:41,395
2012-07-12 21:03:41,395 + The message is classified as:
2012-07-12 21:03:41,395 | Spoofed message
2012-07-12 21:03:41,395 | Scanning
2012-07-12 21:03:41,396 | Ringing
2012-07-12 21:03:41,396

```

3.5.3.6. Ringing attack

Si bien este ataque no está contemplado como tal por la *suite* BackTrack, su existencia es consecuencia del envío de mensajes INVITE a los clientes SIP. Ya sea con la intención de una inundación ó en el intento de enumerar extensiones, cuando se enviaban mensajes de este tipo los *hardphone* y *softphone* comenzaban a sonar.

3.5.4. Caso 2: dominio de broadcast compartido

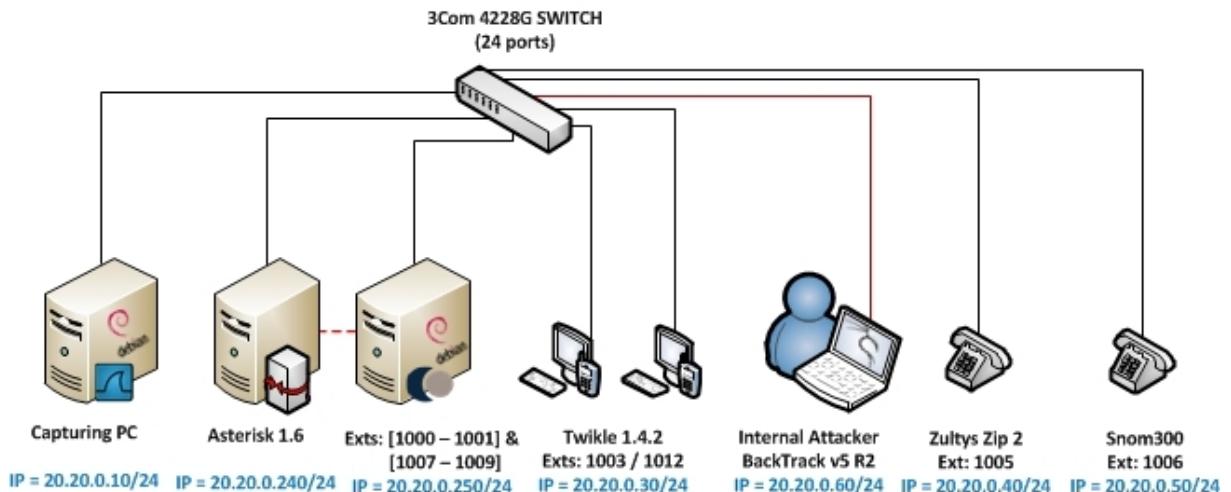


Figura 3.27: Diagrama topología *testbed testing* UBP, caso 2.

Tabla 3.3: Características y especificaciones topología *testbed testing* UBP, caso 2.

| UBP EXPERIMENTATION & TESTING 2 - TEST-BED Infrastructure | | | | | |
|---|---|--|--|-----------------------|-------------|
| DEVICE | HW specifications | OS / Firmware | SERVICE | IPv4 Network Address | GW / Domain |
| 3Com Switch 4228G 3C17304 | | v3.05 3C16671 | IEEE 802.3/802.1Q 100BASE-TX Switch | | |
| Zultys Zip 2 IP Tel. | | ZUTS 3.5.2 | IP SIP Hard Phone - Reg. Ext: 1005 | eth - 20.20.0.40/24 | 20.20.0.240 |
| Snom300 IP Tel. | | v8.4.32 | IP SIP Hard Phone - Reg. Ext: 1006 | eth - 20.20.0.50/24 | 20.20.0.240 |
| Host Asterisk PBX | CPU: Intel(R) Pentium(R) Dual CPU E2160 1.80GHz RAM: 1Gb DIMM DDR2 Sync 533 MHz NetCard: Realtek Semiconductor Co., Ltd. RTL-8139/8139C/8139C+ (rev 10) | Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/LINUX | Asterisk v1.6.2-9-2+squeeze5 SIP Protocol (RFC 3261) IP PBX (Registrar) - Voicemail | eth0 - 20.20.0.240/24 | |
| Vmware Guest 1 | CPU (shared) RAM: 512Mb (shared) Virtual NetCard: Ethernet controller Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE] (rev 10) | Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/Linux | ARTEMISA Honeypot v1.0.91 Registered to Asterisk (20.20.0.240/24) | eth0 - 20.20.0.250/24 | 20.20.0.240 |
| Vmware Guest 2 | CPU (shared) RAM: 512Mb (shared) Virtual NetCard: Ethernet controller Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE] (rev 10) | Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/Linux | Twinkle 1.4.2 Softphones Reg. Ext: [2000-2009] & [2020-2030] domain: ubpsipserver.mvnc.com | eth0 - 20.20.0.30/24 | 20.20.0.240 |
| Capaturing Terminal | CPU: Intel(R) Core(TM)2 Duo CPU E8200 2.66GHz RAM: 2Gb SODIMM Sync 667 MHZ NetCard: RTL8101E/RTL8102E PCI Express Fast Ethernet controller (rev 01) | Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/LINUX | Wireshark 1.2.11-6+squeeze6 network traffic analyzer tcpdump 4.1.1-1 | eth1 - 20.20.0.10/24 | 20.20.0.240 |
| Attacker's Terminal | CPU: Intel(R) Atom(TM) CPU N455 1.66GHz RAM: 2Gb SODIMM Sync 667 MHZ NetCard: RTL8101E/RTL8102E PCI Express Fast Ethernet controller (rev 02) | Linux Back Track 5 R2 i686 GNU/Linux | Inviteflood 2.0-bt1 RTPflood 1.0-bt0 Sipsak 0.9.6-bt0 Sipscan 0.1-bt1 Sip vicious 0.2.7-bt0 Smap 0.6.0-bt0 Metasploit v4.2.0-release | eth0 - 20.20.0.60/24 | 20.20.0.240 |

Aquí se presenta un típico escenario de red LAN, en el cual no se utiliza IEEE 802.3Q (VLAN) según puede verse en la Figura 3.27 con sus respectivos componentes en la Tabla 3.3. Se denomina atacante interno a quien hará uso en este escenario de las herramientas vistas en el caso anterior. Cabe aclarar que en el caso 1 sólo se utilizaron configuraciones de red en modo promiscuo para la interfaz de red donde se realizaba la captura de tráfico. De modo que desde el punto de vista de SIP, como protocolo de capa de aplicación, no existe diferencia entre el uso de un dispositivo *hub* o *switch* como concentrador de red. Sí existen diferencias desde la perspectiva del atacante, ya que pudiendo lograr capturas de tráfico tan completas, logrará obtener ciertos datos sin necesidad de hacer uso de las herramientas utilizadas en los primeros ataques, con la ventaja de mantenerse más oculto. Se puede dar como ejemplo el caso donde el atacante capture tráfico entre un *UA* y su *registrar server*, obteniendo así la dirección IP de servidor y el posible software que obra de *proxy* SIP en el mismo, obteniendo también el nombre

de usuario y valor de una extensión utilizada en el dominio; comenzando de este modo con un ataque de fuerza bruta con el método REGISTER contra el servidor SIP, pasando de este modo totalmente desapercibido para el *honeypot*. Otra diferencia importante radica en el deterioro de desempeño de la red cuando el atacante realice un intento de denegación de servicio mediante una inundación de mensajes SIP. En este punto, la degradación de la calidad del audio se hacía notable al utilizar algunos miles de mensajes con la herramienta Inviteflood, no siendo así con el uso de un *switch* y su consecuente segmentación de dominio de colisión.

3.5.5. Caso 3: dominio público. Colisión y *broadcast* no compratidos

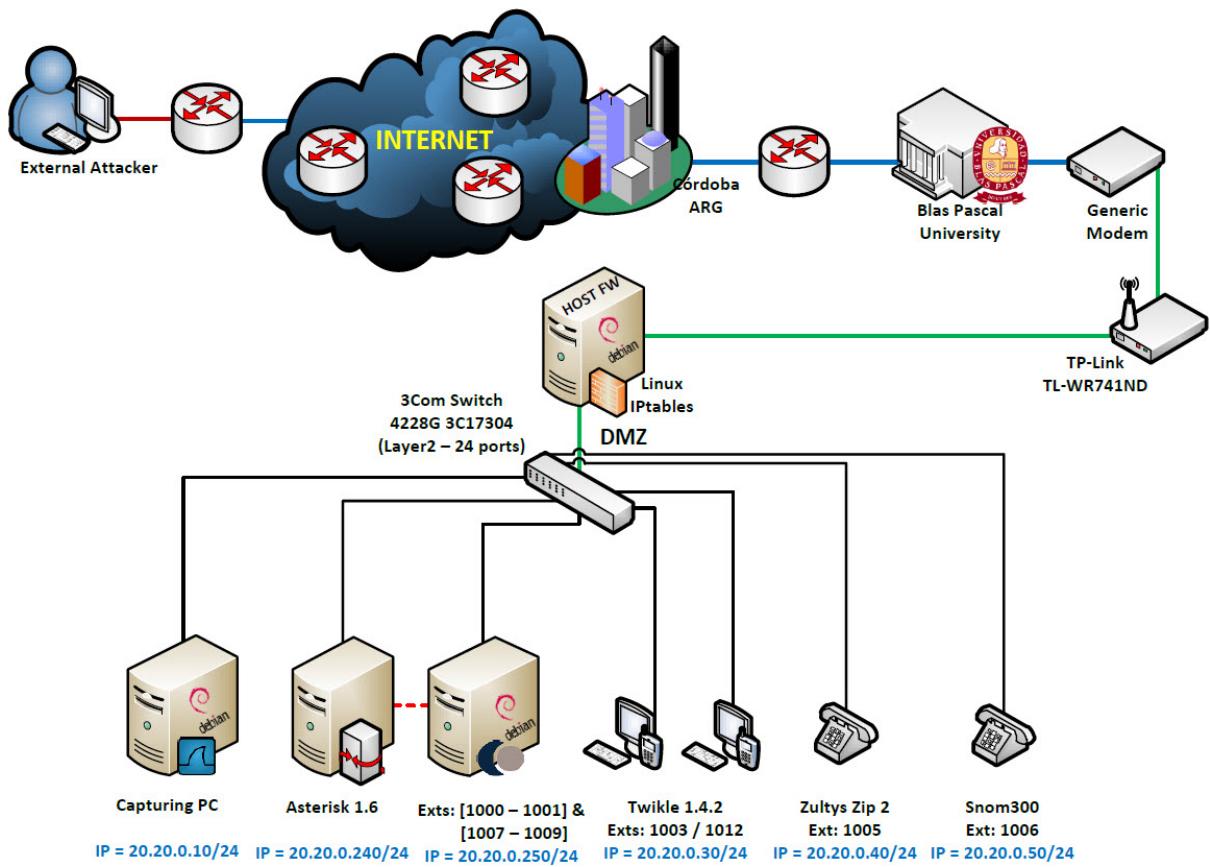


Figura 3.28: Diagrama topología testbed testing UBP, caso 3.

Tabla 3.4: Características y especificaciones topología *testbed testing* UBP, caso 3.

| UBP EXPERIMENTATION & TESTING 3 - TEST-BED Infrastructure | | | | | |
|---|--|--|---|---|-----------------|
| DEVICE | HW specifications | OS / Firmware | SERVICE | IPv4 Network Address | GW |
| Motorola SB5101 | | SB5101-2.8.3.0-GA-01-455-NOSH | Cable-Modem | ISP dynamic (eg: 190.215.119.150) ubpsipserver.myvnc.com | |
| TPLink TL-WR741N | v3.12.11 build 11091 | DHCP/NAT/FW/GW | | eth - 10.10.0.100/24 | 190.215.119.150 |
| Host System 1 CX Server | CPU: Intel(R) Pentium(R) CPU G630T 2.30GHz RAM: 4Gb DDR3 DIMM Synchronous 1333 MHz NetCard eth0: Intel 82574L Gigabit Net Cx NetCard eth1: Intel 82579LM Gigabit Net Cx (rev 0) | Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/Linux | Firewall (IPTables v1.4.8-3) VMware Player v4.0.4 build-744019 Oracle VM VirtualBox v4.1.18 r7836 | eth0 - 10.10.0.254/24 | 10.10.0.100 |
| 3Com Switch 4228G 3C17304 | | v3.05 3C16671 | IEEE 802.3/802.1Q 100BASE-TX Switch | | |
| Zultys Zip 2 IP Tel. | ZUTS 3.5.2 | IP SIP Hard Phone - Reg. Ext: 1005 | | eth - 10.10.0.40/24 | 10.10.0.240 |
| Snom300 IP Tel. | v8.4.32 | IP SIP Hard Phone - Reg. Ext: 1006 | | eth - 10.10.0.50/24 | 10.10.0.240 |
| Host Asterisk PBX | CPU: Intel(R) Pentium(R) Dual CPU E2160 1.80GHz RAM: 1Gb DIMM DDR2 Sync 533 MHz NetCard: Realtek Semiconductor Co., Ltd. RTL-8139/8139C/8139C+ (rev 10) | Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/LINUX | Asterisk v1.6.2.9-2+squeeze5 SIP Protocol (RFC 3261) IP PBX (Registrar) - Voicemail | eth0 - 10.10.0.240/24 | |
| Vmware Guest 1 | CPU (shared) RAM: 512Mb (shared) Virtual NetCard: Ethernet controller Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE] (rev 10) | Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/Linux | ARTEMISA Honeypot v1.0.91 Registered to Asterisk (20.20.0.240/24) | eth0 - 10.10.0.250/24 | 10.10.0.240 |
| Vmware Guest 2 | CPU (shared) RAM: 512Mb (shared) Virtual NetCard: Ethernet controller Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE] | Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/Linux | Twinkle 1.4.2 Softphones Reg. Ext: [2000-2009] & [2020-2030] domain: ubpsipserver.myvnc.com | eth0 - 10.10.0.30/24 | 10.10.0.240 |
| Capaturing Terminal | CPU: Intel(R) Core(TM)2 Duo CPU E8200 2.66GHz RAM: 2Gb DIMM SDRAM Sync NetCard: RTL8101E/RTL8102E PCI Express Fast Ethernet controller (rev 01) | Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/LINUX | Wireshark 1.2.11-6+squeeze6 network traffic analyzer tcpdump 4.1.1-1 | eth1 - 10.10.0.10/24 | 10.10.0.240 |
| Attacker's Terminal | CPU: Intel(R) Atom(TM) CPU N455 1.66GHz RAM: 2Gb SODIMM Sync 667 MHz NetCard: RTL8101E/RTL8102E PCI Express Fast Ethernet controller (rev 02) | Linux Back Track 5 R2 i686 GNU/Linux | Inviteflood 2.0-bt1 RTPFlood 1.0-bt0 Sipsak 0.9.6-bt0 Sipscan 0.1-bt1 Sipvicious 0.2.7-bt0 Smap 0.6-bt0 Metasploit v4.2.0-release | Public IP Address | |

Un escenario al que nos enfrentamos diariamente, y en el que una simple conexión a Internet contratada a casi cualquier ISP del mundo entero por un atacante, lo coloca a este último en posición para intentar vulnerar nuestro sistema VoIP. Se realizaron aquí las pruebas como atacantes externos utilizando los servicios de acceso a Internet provistos por dos diferentes ISP dentro de las facilidades de la UBP, tal como se describe en la Figura 3.28 y Tabla 3.4. La división del dominio de *broadcast* que este escenario plantea, no permitirá al atacante acceder a interfaces con direcciones IP que se encuentren dentro de la LAN, excepto a aquellas a las que el *router/firewall* de la frontera LAN/WAN les reenvíe los paquetes que cumplan con las reglas configuradas por los administradores. Si bien ciertos paquetes son reenviados desde la WAN hacia interfaces de la LAN, el atacante externo observará una única dirección IP del dominio, aquella que es pública. Esta situación no permitirá a las herramientas tener acceso a todos los agentes SIP de nuestra red, sino sólo a aquellos que están expuestos públicamente con sus debidas reglas configuradas en el *firewall*. De forma complementaria puede consultarse en el Anexo D las configuraciones de los componentes y agentes SIP para el caso 3.

Utilizando las herramientas de escaneo, se logra, por lo expuesto en el párrafo anterior, obtener únicamente información sobre el SIP *registrar*. Con respecto a los ataques de enumeración, estos pueden ser dirigidos, nuevamente, sólo al SIP *registrar*, y los ataques realizados utilizando los métodos OPTIONS e INVITE, serán los que lleguen a Artemisa, acusando su presencia, pero sin poder determinar más datos que el UA (en caso de utilizar SER como *proxy SIP*). En este escenario un ataque de inundación podría afectar los recursos de red y procesamiento del servidor SIP, pero un ataque de este tipo es fácilmente evitable con técnicas como TARPIT en Iptables.

3.5.6. Resumen de resultados casos 1, 2 y 3

A continuación se encuentran las Tablas 3.5, 3.6 y 3.7 donde se resumieron los resultados obtenidos frente a los diferentes tipos de ataques y herramientas en los tres escenarios propuestos en la unidad 3.5.

Tabla 3.5: Tabla resumen resultados caso 1.

| CASO 1 | | | | | |
|--|--------------|-----------------------------|---------------|--|-------------------|
| Etapa | Herramienta | Método Utilizado | Configuración | Resultado | Artemisa |
| ESCANEO | SMAP | OPTIONS | RED | Alcanzado | Arribo de OPTIONS |
| | | OPTIONS, INVITE, PRACK, ACK | IP ARTEMISA | Alcanzado | Detecta Ataque |
| | SIP-SCAN | OPTIONS | RED | Solo Artemisa | Arribo de OPTIONS |
| | SVMAP | OPTIONS | RED | Alcanzado | Arribo de OPTIONS |
| ENUMERACION | SVWAR | OPTIONS | RED | Alcanzado | Arribo de OPTIONS |
| | | REGISTER | IP PROXY SIP | No Alcanzado | No Detecta Ataque |
| | | INVITE | IP PROXY SIP | Alcanzado | No Detecta Ataque |
| | | INVITE | IP ARTEMISA | No Alcanzado | Detecta Ataque |
| | | OPTIONS | IP ARTEMISA | No Alcanzado | Detecta Ataque |
| | METASPLOIT | REGISTER | IP ARTEMISA | Se Cierra la Herramienta | No Aplica |
| | | OPTIONS | IP PROXY SIP | No Alcanzado por NO Permitir extensiones Alfanuméricas | No Aplica |
| | | REGISTER | IP PROXY SIP | No Alcanzado por NO Permitir extensiones Alfanuméricas | No Aplica |
| | | OPTIONS | IP ARTEMISA | No Alcanzado | Detecta Ataque |
| | | REGISTER | IP ARTEMISA | No Alcanzado | Detecta Ataque |
| INUNDACION | INVITE FLOOD | INVITE | IP PROXY SIP | Alcanzado | Detecta Ataque |
| | RTP FLOOD | INVITE | IP ARTEMISA | Alcanzado | Detecta Ataque |
| SUPLANTACION DE IDENTIFICACION DE LLAMADAS | INVITE FLOOD | No Aplica | IP ARTEMISA | Alcanzado | No Detecta Ataque |
| | METASPLOIT | INVITE FLOOD | IP PROXY SIP | Alcanzado | Detecta Ataque |
| | | INVITE | IP ARTEMISA | Alcanzado | No Detecta Ataque |

Tabla 3.6: Tabla resumen resultados caso 2.

| CASO 2 | | | | | |
|--|--------------|-----------------------------|---------------|--|-------------------|
| Etapa | Herramienta | Método Utilizado | Configuración | Resultado | Artemisa |
| ESCANEO | SMAP | OPTIONS | RED | Alcanzado | Arribo de OPTIONS |
| | | OPTIONS, INVITE, PRACK, ACK | IP ARTEMISA | Alcanzado | Detecta Ataque |
| | SIP-SCAN | OPTIONS | RED | Solo Artemisa | Arribo de OPTIONS |
| | SVMAP | OPTIONS | RED | Alcanzado | Arribo de OPTIONS |
| ENUMERACION | SVWAR | OPTIONS | RED | Alcanzado | Arribo de OPTIONS |
| | | REGISTER | IP PROXY SIP | No Alcanzado | No Detecta Ataque |
| | | INVITE | IP PROXY SIP | Alcanzado | Detecta Ataque |
| | | INVITE | IP ARTEMISA | No Alcanzado | Detecta Ataque |
| | | OPTIONS | IP ARTEMISA | No Alcanzado | Detecta Ataque |
| | METASPLOIT | REGISTER | IP ARTEMISA | Se Cierra la Herramienta | No Aplica |
| | | OPTIONS | IP PROXY SIP | No Alcanzado por NO Permitir extensiones Alfanuméricas | No Aplica |
| | | REGISTER | IP PROXY SIP | No Alcanzado por NO Permitir extensiones Alfanuméricas | No Aplica |
| | | OPTIONS | IP ARTEMISA | No Alcanzado | Detecta Ataque |
| | | REGISTER | IP ARTEMISA | No Alcanzado | Detecta Ataque |
| INUNDACION | INVITE FLOOD | INVITE | IP PROXY SIP | Alcanzado con Menor Impacto | Detecta Ataque |
| | RTP FLOOD | INVITE | IP ARTEMISA | Alcanzado con Menor Impacto | Detecta Ataque |
| SUPLANTACION DE IDENTIFICACION DE LLAMADAS | No Aplica | INVITE | IP ARTEMISA | Alcanzado con Menor Impacto | No Detecta Ataque |
| | INVITE FLOOD | INVITE | IP PROXY SIP | Alcanzado | No Detecta Ataque |
| | METASPLOIT | INVITE | IP ARTEMISA | Alcanzado | Detecta Ataque |

Tabla 3.7: Tabla resumen resultados caso 3.

| CASO 3 | | | | | |
|--|--------------|-----------------------------|-------------------------------|--|-------------------|
| Etapa | Herramienta | Método Utilizado | Configuración | Resultado | Artemisa |
| ESCANEO | SMAP | OPTIONS, INVITE, PRACK, ACK | IP PUBLICA : PUERTO PROXY SIP | Alcancado | No Detecta Ataque |
| | SIP-SCAN | OPTIONS | IP PUBLICA : PUERTO PROXY SIP | No Alcancado | No Detecta Ataque |
| | SVMAP | OPTIONS | IP PUBLICA : PUERTO PROXY SIP | Alcancado | No Detecta Ataque |
| | METASPLOIT | OPTIONS | IP PUBLICA : PUERTO PROXY SIP | Alcancado | No Detecta Ataque |
| ENUMERACION | SVWAR | OPTIONS | IP PUBLICA : PUERTO PROXY SIP | No Alcancado | No Detecta Ataque |
| | | REGISTER | IP PUBLICA : PUERTO PROXY SIP | Alcancado | No Detecta Ataque |
| | METASPLOIT | INVITE | IP PUBLICA : PUERTO PROXY SIP | Alcancado | Detecta Ataque |
| | | OPTIONS | IP PUBLICA : PUERTO PROXY SIP | No Alcancado por NO Permitir extensiones Alfanuméricas | No Aplica |
| INUNDACION | METASPLOIT | REGISTER | IP PUBLICA : PUERTO PROXY SIP | No Alcancado por NO Permitir extensiones Alfanuméricas | No Aplica |
| | | INVITE FLOOD | INVITE | Alcancado | Detecta Ataque |
| SUPLANTACION DE IDENTIFICACION DE LLAMADAS | INVITE FLOOD | No Aplica | IP PUBLICA : PUERTO PROXY SIP | No Alcancado | No Detecta Ataque |
| | METASPLOIT | INVITE | IP PUBLICA : PUERTO PROXY SIP | Alcancado | No Detecta Ataque |
| | | | | | |

Sobre la evaluación de respuestas empíricas a partir de su implementación y experimentación:

- Artemisa se presenta como una solución proactiva, estable y de fácil integración al entorno. Entrega al administrador información procesada y en tiempo real, de la presencia de entes que intenten vulnerar la seguridad del dominio SIP.
- Este *honeypot* se hace fuerte en entornos de red privada, acusando en tiempo real y con registros detallados la presencia de un atacante interno.
- Quedó demostrada la robustez de Artemisa ante intentos de DoS ya que frente a un gran número de ataques simultáneos, el *honeypot* pudo valerse de su mecanismo configurable de limitación de atención de llamadas para no verse comprometido.
- Una configuración en el SIP *registrar* del entorno que encamine hacia el *honeypot* todo intento de comunicación contra un agente SIP que no pertenezca al dominio, aumenta el grado de detección de atacantes, tanto internos como externos.
- Considerar en el caso de un atacante interno, que disponer de dos instancias de Artemisa, una con *fingerprint* de un *softphone* y otra de *registrar* será muy ventajoso. Ya que de esta manera podremos detectar a través de nuestros *honeypots* tanto las IP de origen y de destino de los ataques dentro de la LAN. Esto se debe a que con una sola instancia de Artemisa esta información quedaría enmascarada por el SIP *proxy/registrar* del dominio, ej.: Asterisk, agente SIP que recibe la comunicación y encamina la llamada modificando el campo IP de origen por su propia IP. Más aun teniendo en cuenta que la mayoría de los ataques tienen como *target* el servidor de registro del entorno VoIP.
- El uso del *honeypot* demostró que al colocar una trampa en la infraestructura de red VoIP, hace claramente más riesgosas a las incursiones exploratorias para un atacante y que sea más probable poder detectarlo. El señuelo Artemisa pudo retrasarnos cuando asumimos el rol del usuario mal intencionado, lo cual nos daría la delantera para comenzar la detección, identificación y frustración de tal intrusión en el rol de administrador de la red.

3.6. Estudio descriptivo estadístico de las características de Artemisa honeypot en entornos reales en mesa de prueba

Se realizó un extenso relevamiento de un período de seis meses (1/11/2012 - 1/06/2013), que permitió apreciar tendencias e inferir conclusiones respecto a las variables analizadas. Según se presentan debajo:

Variables:

- Aplicativo Artemisa y su código fuente: se toma como una constante la última versión estable del código disponible en el repositorio oficial de SourceForge¹³.
- Topología de la red: se adjuntan los diagramas topológicos de los entornos en las Figuras 3.29 y 3.32 . Pueden diferenciarse dos configuraciones lógicas en cuanto a la implementación del SIP *registrar*. Por un lado *self-hosted* (entorno privado) y por el otro *hosted-services* (entorno rentado). Se usó iptel.org para la implementación de *hosted-services*.
- Sistemas de seguridad existente: considerando las camas de prueba (entornos *self-hosted*) se optó por la misma solución de seguridad para ambas implementaciones. En las cuales se configuró *IPtables Linux firewall* y *Port Filtering* a través de los Puerta de Enlace (GW) *routers*. En cuanto a los *hosted-services* de iptel.org no se dispone de esta información.
- Componentes y características particulares de la red: se plantearon dos entornos de características muy próximas entre sí. Estos fueron instalados con componentes muy similares de hardware y software. Entre ellos, número de servidores (SIP *registrar*, SIP *proxy server* y *gateway*), equipos VoIP (IP *phones*, *softphones*) y equipamiento de *networking* necesario (*firewall*, *routers* y *switches*) recordando que gran parte de estos componentes fueron virtualizados con VMWare y VirtualBox. Como única diferencia considerable los *testbeds* accedían a Internet a través de dos ISPs diferentes dentro de la misma ciudad. Pero en base a los resultados obtenidos puede considerarse como una variable individual o constante. Cabe aclarar que se estudio el aplicativo Artemisa bajo diferentes topologías de red en la unidad anterior “Evaluación de respuestas empíricas y enfoque en desarrollos funcionales del sistema de seg Artemisa” - casos 1, 2 y 3 (sección 3.5). De forma complementaria puede consultarse en el Anexo D las configuraciones de los componentes y agentes SIP de las 2 implementaciones estudiadas en esta sección.
- Análisis del tráfico de voz, posibles vulnerabilidades y ataques: se analizó tráfico de señalización de voz (VoIP), posibles vulnerabilidades y ataques. Considerando todos los gráficos de esta sección, se presenta un análisis comparativo entre tráfico procesado por Artemisa y *registrars*. Esto será por método SIP y tipo de mensajes SIP, cantidad de llamadas, tráfico sospechoso y ataques, como así también geolocalización por IP *address*.
- *Threat model or adversary model* (modelo de atacante): Durante el análisis que se realizó en esta sección se mantuvo el perfil de atacante que se modelizó en la sección 3.5.2.

En el transcurso de esta etapa, se pusieron en funcionamiento cinco instancias de Artemisa *honeypot*. Dos de ellas en la implementación *testbed UBP* dentro de los laboratorios de la UBP.

¹³<http://sourceforge.net/projects/artemisa/files/> de la versión artemisa_1.0.91.tar.gz (2011-02-22 1.7 MB - autor: rodrigodocarmo)

Las restantes en la implementación *testbed Metropolitan*. En todas estas instancias de Artemisa expuestas a Internet durante el plazo detallado, se llevaron adelante capturas de tráfico y *logs* a nivel de aplicación, con el claro objetivo de obtener información relevante de ataques reales. Deberá considerarse que se dispuso de números telefónicos SIP gratuitos (iptel.org), los cuales son con frecuencia un buen objetivo de ataque además de los configurados localmente en nuestras arquitecturas SIP de prueba. Podrán apreciarse sus configuraciones y componentes en las Tablas 3.8, 3.9 y diagramas que se presentan en las Figuras 3.29, 3.30 y 3.32 de las siguientes subsecciones 3.6.1 y 3.6.2.

A partir de lo anterior, se procedió al estudio de las óptimas condiciones de instalación y distribución de las instancias de Artemisa. Y tuvo lugar el estudio técnico, y posteriores ajustes funcionales en base a la experimentación en entornos de prueba como se verá en sección 3.7. Durante este trabajo se asentaron y categorizaron las mediciones realizadas.

Luego de la recopilación de extensivos resultados, apropiadamente categorizados y tratados en base a métricas claramente definidas, se plasmará a continuación un análisis estadístico de los mismos en este TFC, para nuestro caso de estudio, desde la perspectiva del administrador de seguridad del sistema.

3.6.1. Implementación *testbed UBP*

Tabla 3.8: Características y especificaciones topología *testbed UBP*.

| UBP CAPTURING & ANALYSIS - TEST-BED Infrastructure | | | | | |
|--|---|--|--|--|-----------------|
| DEVICE | HW specifications | OS / FwWare | SERVICE | IPv4 Network Address | GW |
| Motorola SB5101 | | S85101-2.8.3.0-GA-01-455-NOSH | Cable-Modem | ISP dynamic (eg: 190.215.119.150) ubpsipserver.myvnc.com | |
| TPLink TL-WR741N | | v3.12.11 build 110914 | DHCP/NAT/FW/GW | eth - 10.10.0.100/24 | 190.215.119.150 |
| Host System 1 CX Server | CPU: Intel(R) Pentium(R) CPU G630T 2.30GHz RAM: 4Gb DDR3 DIMM Synchronous 1333 MHz NetCard eth0: Intel 82574L Gigabit Net Cx NetCard eth1: Intel 82579LM Gigabit Net Cx (rev 05) | Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/Linux | Firewall (IPTables v1.4.8-3) VMware Player v4.0.4 build-744019 Oracle VM VirtualBox v4.1.18 r78361 | eth0 - 10.10.0.254/24 | 10.10.0.100 |
| VirtualBox Guest 1 | CPU (shared) RAM: 512Mb (shared) Virtual NetCard: Intel Corporation 82543GC Gigabit Ethernet Controller (Copper) (rev 02) | Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/Linux | SER v0.9.8_i386 SIP Protocol (RFC 3261) SIP Proxy Server | eth0 - 10.10.0.240/24 | 10.10.0.254 |
| Vmware Guest 2 | CPU (shared) RAM: 512Mb (shared) Virtual NetCard: Ethernet controller Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE] (rev 10) | Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/Linux | ARTEMISA Honeypot v1.0.91 Registered to SER (10.10.0.240/24) | eth0 - 10.10.0.250/24 | 10.10.0.254 |
| Vmware Guest 3 | CPU (shared) RAM: 512Mb (shared) Virtual NetCard: Ethernet controller Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE] (rev 10) | Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/Linux | Twinkle 1.4.2 Softphones Reg. Ext: [2000-2009] & [2020-2030] domain: ubpsipserver.myvnc.com | eth0 - 10.10.0.30/24 | 10.10.0.254 |
| Vmware Guest 4 | CPU (shared) RAM: 768Mb (shared) Virtual NetCard: Ethernet controller Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE] (rev 10) | Linux CentOS release 6.2 2.6.32-220.17.1.el6.i686 i686 i386 GNU/Linux | ARTEMISA Honeypot v1.0.91 Registered to: ubp1@iptel.org / 229049@iptel.org ubp2@iptel.org / 229056@iptel.org | eth0 - 10.10.0.230/24 | 10.10.0.254 |

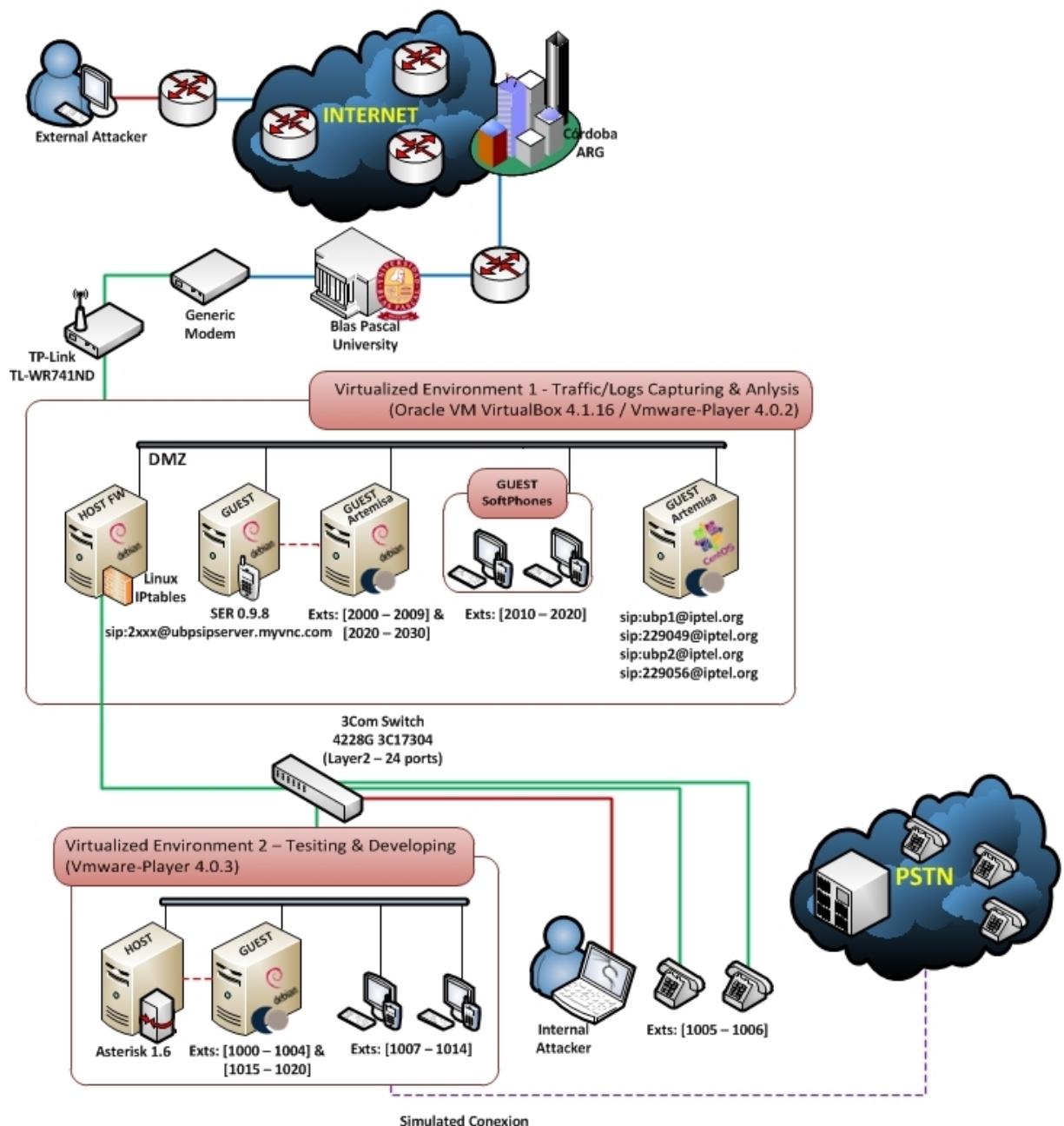


Figura 3.29: Diagrama topología *testbed UBP*.

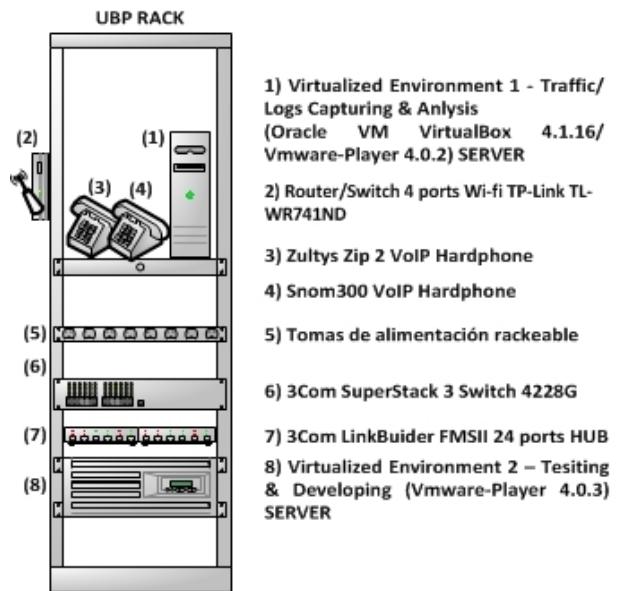


Figura 3.30: Diagrama rack *testbed* UBP.

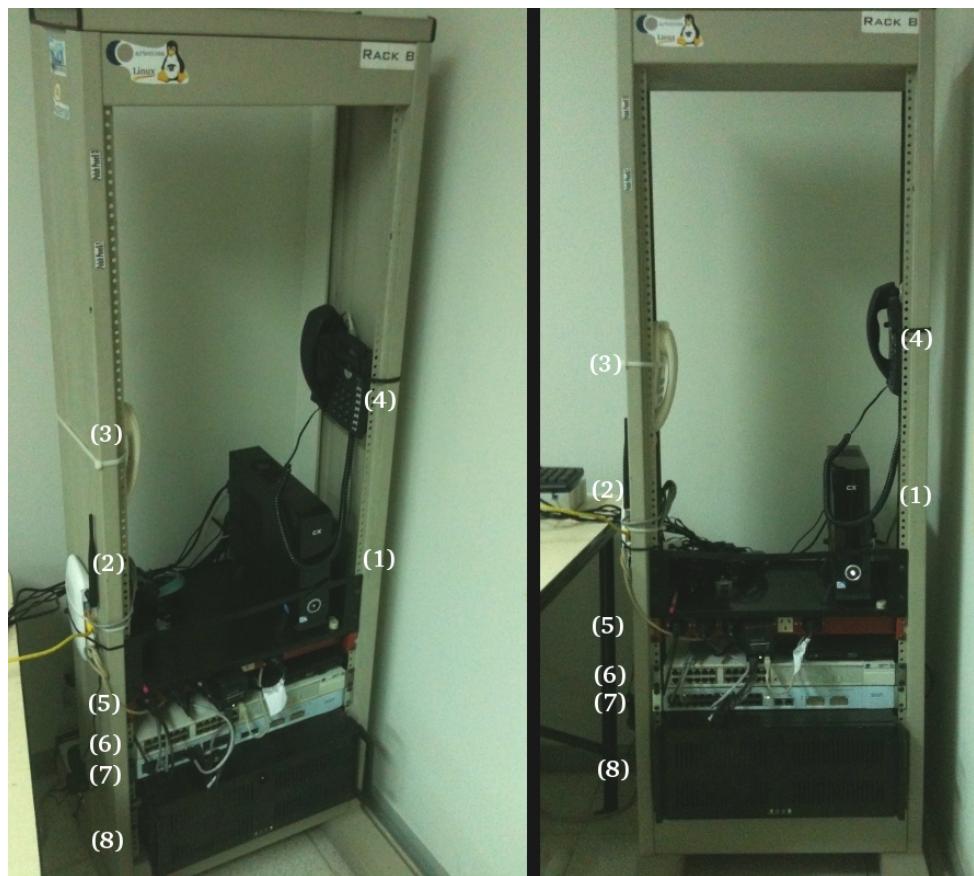


Figura 3.31: Foto rack *testbed* UBP.

3.6.2. Implementación *testbed* Metropolitan

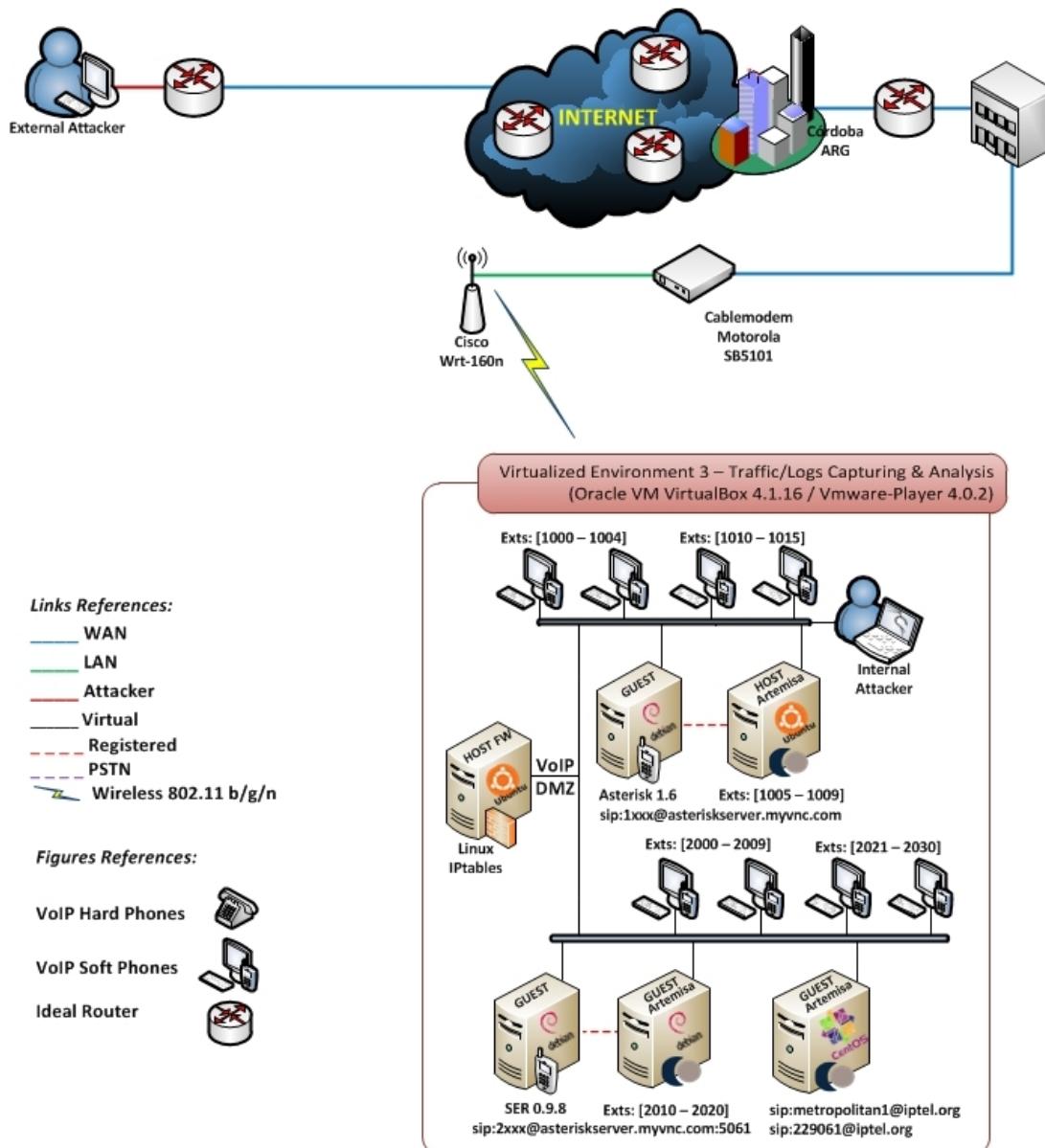


Figura 3.32: Diagrama topología *testbed* Metropolitan.

Tabla 3.9: Características y especificaciones topología *testbed* West Metropolitan.

| METROPOLITAN 1 TEST-BED Infrastructure | | | | | |
|--|--|---|--|---|-----------------|
| DEVICE | HW specifications | OS / Fwrmware | SERVICE | IPv4 Network Address | GW |
| Motorola SB5101 | | SB5101-2.8.3.0-GA-01-455-NOSH | Cable-Modem | ISP dynamic (e.g: 190.246.129.142) asteriskserver.mynic.com | |
| Cisco Wrt-160nv3 | | v3.0.02 build 004 | DHCP/NAT/fw/GW | eth - 192.168.0.1/24 | 190.246.129.142 |
| Host System 1 Generic Server | CPU: AMD Athlon(tm) 64 X2 Dual Core 2.6Ghz RAM: 4Gb DDR2 800Mhz NetCard: RTL-8185 IEEE 802.11a/b/g WLAN | Linux Ubuntu 12.04 LTS 3.2.0-24-generic-pae i686 athlon i386 GNU/Linux | Firewall (IPTables v1.4.8-3) VMware Player v4.0.2 build-591240 Oracle VM VirtualBox v4.1.16 r78094 ARTEMISA Honeypot v1.0.91 Twinkle 1.4.2 Softphones | wlan0 - 192.168.0.101/24 | 192.168.0.1 |
| VirtualBox Guest 1 | CPU (shared) RAM: 512Mb (shared) Virtual NetCard: Intel Corporation 82543GC Gigabit Ethernet Controller (Copper) (rev 02) | Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/Linux | Asterisk v1.6.2.9-2+squeeze5 SIP Protocol (RFC 3261) IP PBX (Registrar) - Voicemail VoIP-PSTN Gateway (simulated) | eth0 - 192.168.0.103/24 | 192.168.0.101 |
| VirtualBox Guest 2 | CPU (shared) RAM: 512Mb (shared) Virtual NetCard: Intel Corporation 82543GC Gigabit Ethernet Controller (Copper) (rev 02) | Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/Linux | SER v0.9.8_i386 SIP Protocol (RFC 3261) SIP Proxy Server | eth0 - 192.168.0.106/24 | 192.168.0.101 |
| Vmware Guest 3 | CPU (shared) RAM: 512Mb (shared) Virtual NetCard: Ethernet controller Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE] (rev 10) | Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/Linux | ARTEMISA Honeypot v1.0.91 Registered to SER (192.168.0.106/24) | eth0 - 192.168.0.104/24 | 192.168.0.101 |
| Vmware Guest 4 | CPU (shared) RAM: 768Mb (shared) Virtual NetCard: Ethernet controller Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE] (rev 10) | Linux CentOS release 6.2 2.6.32-220.17.1.el6.i686 i686 i386 GNU/Linux | ARTEMISA Honeypot v1.0.91 Registered to: sip:metropolitan1@iptel.org sip:2290612@iptel.org | eth0 - 192.168.0.105/24 | 192.168.0.101 |

3.6.3. Configuraciones

En las secciones subsiguientes se presentan las configuraciones de los componentes fundamentales implementados en los dos *testbeds* en cuestión. Las cuales son equivalentes, esto se debe a que los servicios fueron implementados en máquinas virtuales como se detalla a continuación:

Virtual machines:

- *Oracle VM VirtualBox v4.1.x machines*

- *VirtualBox Asterisk guest:*
 - ⇒ CPU = Shared
 - ⇒ RAM = 512Mb (shared)
 - ⇒ NetCard = Intel Corp. 82543GC Gig-Eth Controller (rev 02)
 - ⇒ OS = Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/Linux
 - ⇒ Service 1 = Asterisk v1.6.2.9-2+squeeze5
 - ⇒ Service 2 = SIP Protocol (RFC 3261)
 - ⇒ Service 3 = IP PBX (registrar) - Voicemail
 - ⇒ Service 4 = VoIP-PSTN Gateway (simulated)

- *VirtualBox SER guest:*

- ⇒ CPU = Shared
- ⇒ RAM = 512Mb (shared)
- ⇒ NetCard = Intel Corp. 82543GC Gig-Eth Controller (rev 02)
- ⇒ OS = Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/Linux
- ⇒ Service 1 = SER v0.9.8 i386
- ⇒ Service 2 = SIP Protocol (RFC 3261)
- ⇒ Service 3 = SIP Express Router

- *VMware Player v4.0.x Machines*

- *VMware Debian Artemisa guest*
 - ⇒ CPU = Shared
 - ⇒ RAM = 512Mb (shared)
 - ⇒ NetCard = Eth controller Micro Dev [AMD] 79c970 (rev 10)
 - ⇒ OS = Linux Debian 6.0.5 2.6.32-5-686 i686 GNU/Linux
 - ⇒ Service 1 = ARTEMISA honeypot v1.0.91
 - ⇒ Service 2 = Registered to SER or Asterisk
- *VMware Centos Artemisa guest*
 - ⇒ CPU = Shared
 - ⇒ RAM = 512Mb (shared)
 - ⇒ NetCard = Eth controller Micro Dev [AMD] 79c970 (rev 10)
 - ⇒ OS = Linux CentOS 6.2 2.6.32-220.17.1.el6.i686 i386
 - ⇒ Service 1 = ARTEMISA honeypot v1.0.91
 - ⇒ Service 2 = Registered to iptel.org

3.6.4. Estudio descriptivo estadístico en entornos reales en mesa de prueba

Mediante la utilización de herramientas de captura de tráfico y eventos. Entre estas tcpdump, Wireshark, *logs* de aplicación (Artemisa/Asterisk) se recopiló valiosa información. Tal cual fue presentado al comienzo de esta unidad, se realizó un extensivo relevamiento, de un período de aproximadamente seis meses (1/11/2012 - 1/06/2013). Este permitió apreciar tendencias e inferir conclusiones respecto a las variables analizadas.

Partiendo del análisis de las tablas y figuras de esta sección donde se presenta de forma detallada los entornos UBP (Tabla 3.8 y Figura 3.29) y Metropolitan *testbed* (Tabla 3.9 y Figura 3.32) se detallan las herramientas utilizadas para realizar las capturas y el almacenamiento de los *logs*. Cabe aclarar que en ambos entornos se montó un NFS *server* permitiendo archivar todos los datos de forma centralizada.

Metropolitan testbed

Artemisa *instance* STDOOUT & STDERR responses (Ubuntu host *instance* #1):

```
delivery@localhost artemisa_1.0.91]$ sudo ./artemisa 2>&1 | tee
/media/Storage/nfs\_server/DUHUBuArtemisa/artemisa\_stdout\_\_stderr\_\_dd\_\_mm\_\_aa
```

Tcpdump captures (Ubuntu host):

```
delivery@deliveryUbuntuHOME:~$ sudo tcpdump -s0 -n -vvv -i wlan0 udp and \ (port
5060 or 5063\)\ or icmp -w
/media/Storage/nfs_server/DUHUBuArtemisa/Ubuntu_Artemisa_asteriskserver_dd_mm_aa
```

Artemisa STDOOUT & STDERR responses (Debian Artemisa VMWare guest *instance* #2):

```
root@debianDUH:/home/delivery/Artemisa/artemisa_1.0.91
2010-2015# ./artemisa 2>&1 | tee
/media/Storage/nfs_server/DUHDebArtemisa/artemisa_SER_2010-2015_stdout_stderr_06
_07_12
```

Artemisa STDOUT & STDERR responses (Debian Artemisa VMWare guest instance #3):

```
root@debianDUH:/home/delivery/Artemisa/artemisa_1.0.91
2016-2020# ./artemisa 2>&1 | tee
/media/Storage/nfs_server/DUHDebArtemisa/artemisa_SER_2016-2020_stdout_stderr_dd
_mm_aa
```

Tcpdump captures (Debian Artemisa VMWare guest):

```
delivery@deliveryUbuntuHOME:~$ sudo tcpdump -s0 -p -n -vvv -i eth0 udp and
\ (port 5060 or 5061 or 5062\ ) or ICMP -w
/media/nfs/DUHDebArtemisa/Debian_Artemisa_SER_asteriskserver_dd_mm_aa
```

Artemisa instance STDOUT & STDERR responses (Centos Artemisa VMWare guest instance #4):

```
[delivery@localhost artemisa_1.0.91]$ sudo ./artemisa 2>&1 | tee
/media/Storage/nfs_client/DUHCentosArtemisa/artemisa_stdout_stderr_dd_mm_aa
```

Tcpdump captures (Centos Artemisa VMWare guest):

```
[delivery@localhost DUHCentosArtemisa]$ sudo tcpdump -s0 -n -p -vvv -i eth0 udp
and \ (port 5060 or 5062\ ) or icmp -w
/media/Storage/nfs_client/DUHCentosArtemisa/Centos_Artemisa_metropolitan1_dd_mm_
aa
```

Instance logs (Debian Asteriks VirtualBox guest instance):

```
delivery@debiannv:~$ sudo tail -f /var/log/asterisk/messages | tee
/media/Storage/nfs_server/DUHDebAsterisk/var_log_asterisk_messages_dd_mm_aa
```

Tcpdump captures (Debian Asterisk VirtualBox guest):

```
delivery@debiannv:~$ sudo tcpdump -s0 -n -p -vvv -i eth1 udp and \ (port 5060 or
5063\ ) or icmp -w
/media/Storage/nfs_server/DUHDebAsterisk/Debian_Asterisk_1xxx_asteriskserver_dd_
mm_aa
```

Tcpdump captures (Debian SER VirtualBox guest):

```
delivery@debiannv:~$ sudo tcpdump -s0 -n -p -vvv -i eth1 udp and \ (port 5060 or
5063\ ) or icmp -w
/media/Storage/nfs_server/DUHDebAsterisk/Debian_Asterisk_1xxx_asteriskserver_dd_
mm_aa
```

UBP testbed

Artemisa STDOUT & STDERR responses (Debian Artemisa VMWare guest instance #1):

```
$ sudo ./artemisa 2>&1 | tee
/media/nfs/captures/UBPDebArtemisa_2000-2007_stdout_stderr_dd_mm_aa
```

Artemisa STDOUT & STDERR responses (Debian Artemisa VMWare guest instance #2):

```
$ sudo ./artemisa 2>&1 | tee
/media/nfs/captures/UBPDebArtemisa_2008-2009_2020-2024_stdout_stderr_dd_mm_aa
```

Artemisa STDOUT & STDERR responses (Debian Artemisa VMWare guest instance #3):

```
$ sudo ./artemisa 2>&1 | tee
/media/nfs/captures/UBPDebArtemisa_2025-2030_stdout_stderr_dd_mm_aa
```

Tcpdump captures (Debian Artemisa VMWare guest):

```
$ sudo ./artemisa 2>&1 | tee
/media/nfs/captures/UBPDebArtemisa_2025-2030_stdout_stderr_dd_mm_aa
```

Tcpdump captures (Debian SER host):

```
$ sudo tcpdump -s0 -n -p -vvv -i eth0 udp and port 5060 or icmp -w
/media/nfs/captures/UBPDebSER_ubpsipserver_dd_mm_aa
```

Tcpdump captures (Centos Artemisa VMWare guest instance #4):

```
[delivery@localhost DUHCentosArtemisa]$ sudo tcpdump -s0 -n -p -vvv -i eth0 udp
and \(\port 5060 or 5062\)\) or icmp -w
/media/Storage/nfs_client/DUHCentosArtemisa/Centos_Artemisa_metropolitani_dd_mm_
aa
```

Tcpdump captures (Centos Artemisa VMWare guest instance #5):

```
[delivery@localhost DUHCentosArtemisa]$ sudo tcpdump -s0 -n -p -vvv -i eth0 udp
and \(\port 5060 or 5062\)\) or icmp -w
/media/Storage/nfs_client/DUHCentosArtemisa/Centos_Artemisa_metropolitani_dd_mm_
aa
```

Tcpdump captures (Centos Artemisa VMWare guest instance #6):

```
[delivery@localhost DUHCentosArtemisa]$ sudo tcpdump -s0 -n -p -vvv -i eth0 udp
and \(\port 5060 or 5062\)\) or icmp -w
/media/Storage/nfs_client/DUHCentosArtemisa/Centos_Artemisa_metropolitani_dd_mm_
aa
```

A fines prácticos se automatizó mediante Bash *scripting* al momento de iniciar las VMs tal como se muestra a continuación:

Tcpdump captures (VMs Init Bash script):

```
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local
/etc/init.d/network restart

# 1 - mount NFS
mount 10.10.0.254:/media/nfs_server/UBPCentosArtemisa /media/nfs_client/

# 2 - kill gnome
#telinit 3

# 3 - tcpdump
/usr/sbin/tcpdump -s0 -n -p -i eth1 udp and \(\port 5060 and 5061\)\) or icmp -w
/media/nfs_client/captures/Centos_Artemisa_SER_\$(date
+%-m'_'%d'_'%Y'_'%H'_'%M'_'%S).cap

# 4 - Artemisa UBP1 stdout & stderr responses
python /home/delivery/Artemisa\
iptel.org/artemisa_1.0.91_ubp1@iptel.org/artemisa 2>&1 | tee
/media/nfs_client/logs/artemisa_ubp1_stdout_stderr_\$(date
```

```

+%m'_'%d''_%Y'_'%H'_'%M'_'%S)

# 5 - Artemisa UBP2 stdout & stderr responses
python /home/delivery/Artemisa\
iptel.org/artemisa_1.0.91_ubp2\@iptel.org/artemisa 2>&1 | tee
/media/nfs_client/logs/artemisa_ubp2_stdout_stderr_$(date
+%m'_'%d''_%Y'_'%H'_'%M'_'%S)

```

3.6.5. Estadísticas Artemisa

En ambos entornos los *logs* de las instancias de Artemisa se obtuvieron de los archivos de *logging* generados en tiempo real por la propia aplicación, accesibles desde el árbol de directorio en el que se encontraba el mismo, debajo ruta de directorio a modo de referencia:

```
root_directory/.../ artemisa_1.0.91/logs/artemisa_yyyy-dd-mm.log
```

3.6.5.1. Cantidad de paquetes SIP total por tipo de mensaje SIP instancias Artemisa

En las figuras que contienen los gráficos axiales debajo Figuras 3.33, 3.34, 3.35 se presentan el número de paquetes SIP total por cada tipo de mensaje SIP de todas las instancias de Artemisa configuradas.

Se especifican a continuación los mensajes SIP para ambos *testbeds* UBP y Metropolitan para todas las instancias de Artemisa mencionadas tanto *self-hosted* como *hosted-services*:

- 100 *Trying*: esta respuesta indica que la solicitud ha sido recibida por el servidor de siguiente salto (*next-hop*) y que alguna acción no especificada está siendo tomada en representación de esta llamada (por ejemplo, una base de datos está siendo consultada). Esta respuesta, como toda otra respuesta provisoria, detiene la retransmisión de un INVITE mediante un UAC. La respuesta SIP 100 (*Trying*) es diferente de otras respuestas provisionales, en que nunca será reenviada a contramano por un *stateful proxy*. Recordar lo analizado en la subsección 3.2.
- 180 *Ringing*: el UA recibiendo el INVITE está tratando de alertar al usuario. Esta respuesta puede ser utilizada para iniciar el timbrado local del dispositivo SIP.
- 200 *OK*: la solicitud ha sido exitosa. La información retornó con la respuesta. Esto depende del método que haya sido utilizado en la solicitud.
- 401 *Unauthorized*: la solicitud requiere de autenticación. Esta respuesta es emitida por los UAs y *registrars*, mientras que 407 (Requerimiento de autenticación de *proxy* - *Proxy Authentication Required*) es usada por *proxy servers*.
- 405 *Method Not Allowed*: el método especificado en la *Request-Line* (línea de solicitud) es entendido, pero no permitido para la dirección identificada por el *Request-URI* (URI-solicitado). La respuesta tiene que incluir un campo de encabezado permitido que contenga la lista de métodos válidos para la dirección indicada.
- 408 *Request Timeout*: el servidor no puede producir una respuesta dentro de un período de tiempo apropiado, por ejemplo, si no puede determinar la ubicación del usuario a tiempo. El cliente podría repetir la solicitud sin ningún tipo de modificaciones en cualquier momento más adelante.

- 486 *Busy Here*: el sistema final llamado, fue contactado satisfactoriamente, pero el receptor de la llamada no se encuentra actualmente disponible para tomar llamadas adicionales en este sistema final. La respuesta podría indicar un mejor momento para llamar en el campo del encabezado *Retry-After*. El usuario también podría estar disponible en cualquier forma, por ejemplo a través de un servicio de mensajes de voz (*voice mail service*). El mensaje *Status 600 (Busy Everywhere)* podría ser usado si el cliente sabe que ningún otro sistema final podrá aceptar esta llamada.
- 500 *Server Internal Error*: el servidor encontró una condición inesperada que impidió al mismo, completar la solicitud. El cliente podría presentar la condición de error específica y puede reiterar la solicitud luego de algunos segundos. Si la condición es temporal, el servidor podría indicar cuando el cliente puede intentar reiterar la solicitud, utilizando el campo del encabezado *Retry-After*.

[Rosenberg J., 2002].

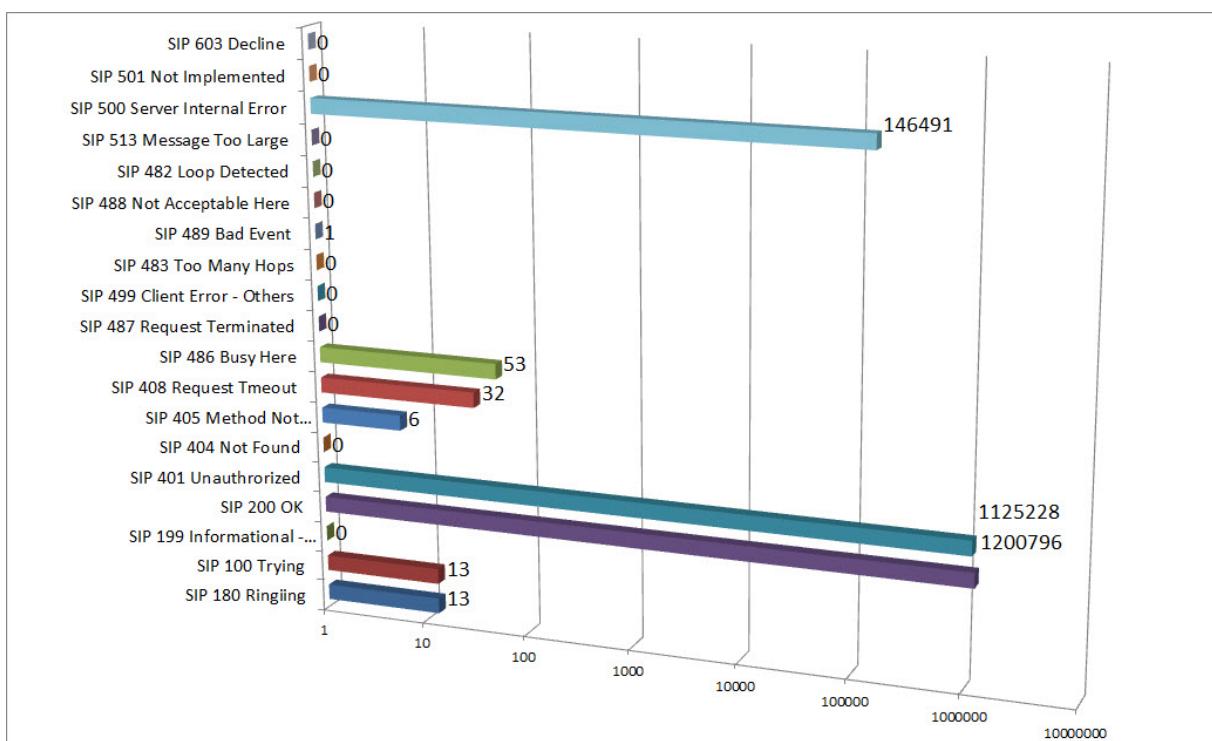


Figura 3.33: Cantidad de paquetes SIP total por tipo de mensaje SIP (instancias Artemisa).

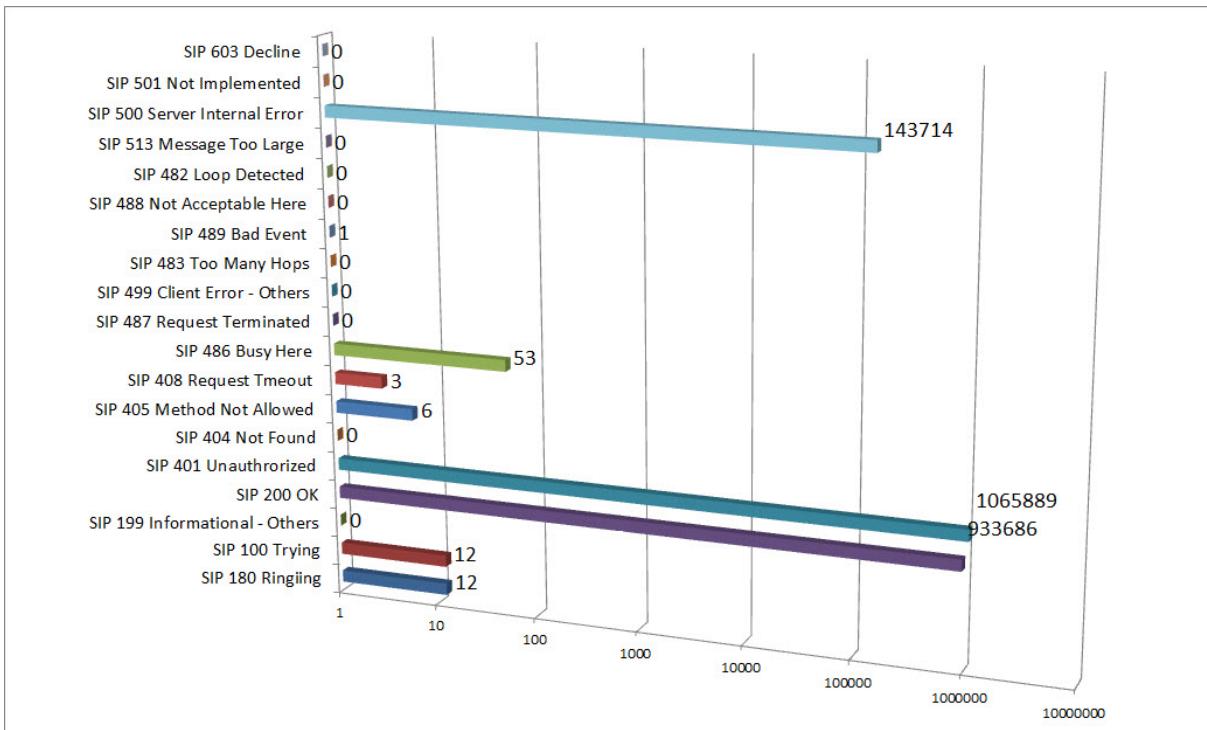


Figura 3.34: Cantidad de paquetes SIP total por tipo de mensaje SIP (Artemisa *self-hosted*).

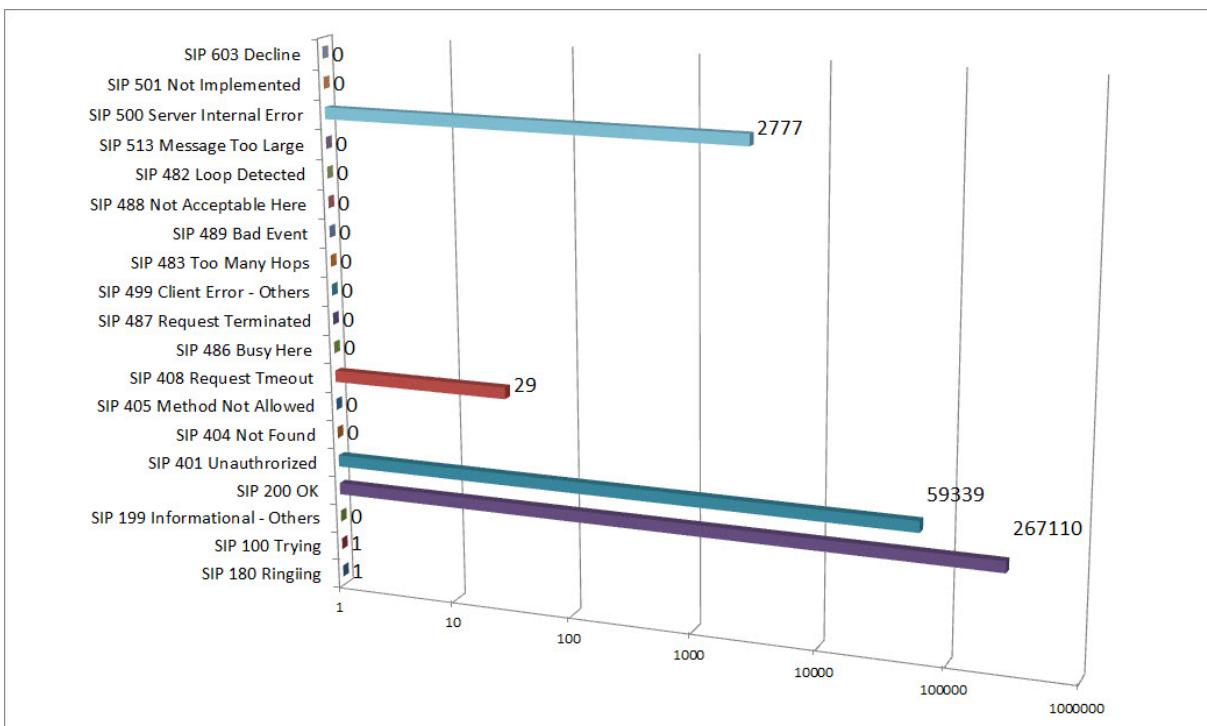


Figura 3.35: Cantidad de paquetes SIP total por tipo de mensaje SIP (Artemisa *hosted-services*).

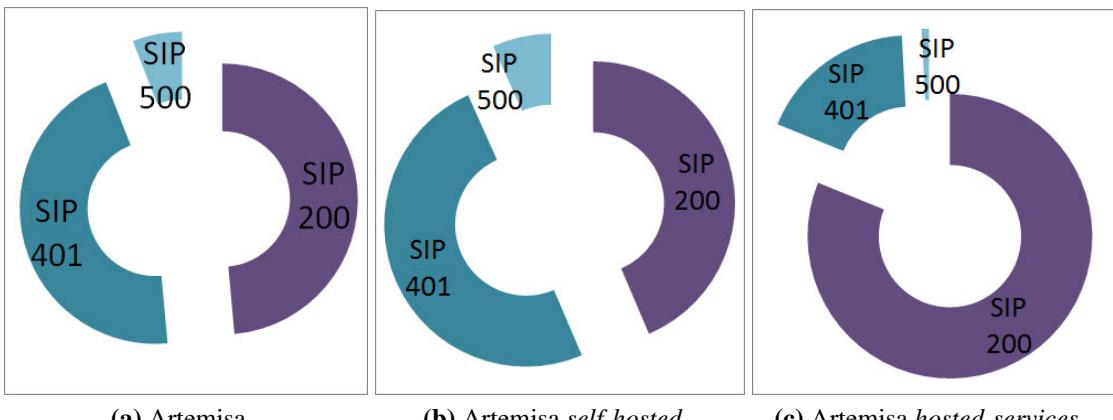


Figura 3.36: Cantidad de paquetes SIP total por tipo de mensaje SIP (instancias Artemisa).

Se aprecian en los gráficos de las Figuras 3.33, 3.34, 3.35 y 3.36 que los tipos de mensajes que predominan son:

- 200 *OK*: en respuesta a las solicitudes exitosas procesadas por Artemisa, tanto recibidas como realizadas.
- 401 *Unauthorized*: error en la autenticación contra Artemisa, podrían ser asociados a intentos de registro fallidos. Posiblemente no esperados, es decir generados por un atacante.
- 500 *Server Internal Error*: esta respuesta está íntimamente relacionada a que el servidor, en nuestro caso Artemisa, se enfrentó con una condición inesperada que impidió que al mismo completar la solicitud. Esto está asociado a que Artemisa solo tienen la capacidad de responder a ciertos mensajes SIP considerando que el mismo actua como un *user-agent* y no tiene la capacidad de un *proxy* SIP o *registrar*.

3.6.5.2. Cantidad de paquetes SIP total por tipo de método SIP instancias Artemisa

En cuanto a los gráficos de cantidad de paquetes SIP total por tipo de método SIP *Sip Request*¹⁴ de las instancias de Artemisa de las Figuras 3.37, 3.38, 3.39 y 3.40 puede verse que los métodos con mayor recurrencia son:

- INVITE: esta solicitud invita a los usuarios a participar en la sesión. El cuerpo de la solicitud contiene la descripción de la sesión. Por ejemplo, cuando un usuario A llama a otro B, su UA envía un INVITE con la descripción de sesión al UA de B. En este caso con la descripción SDP.
- REGISTER: los usuarios envían una solicitud de REGISTER para informar al servidor acerca de su ubicación actual. El usuario A puede enviar un REGISTER al *registrar* a su dominio “company.com” especificando que todas las solicitudes entrantes para SIP:usuarioA@company.com deberán utilizar *proxy*, o redirigidas a una cierta dirección SIP:usuarioA@IP

¹⁴SIP Request: La especificación del *core SIP* define seis tipos de solicitudes SIP, cada una de ellas con un propósito diferente. Cada solicitud SIP contiene un campo, llamado “método”, que denota su propósito. La lista muestra los seis métodos: INVITE, ACK, OPTIONS, BYE, CANCEL, REGISTER. [Camarillo G., 2002]

Los servidores SIP son usualmente co-ubicados con un SIP *registrar*. Un SIP *registrar* puede enviar toda la información recibida en varias solicitudes REGISTER a un servidor de locación única, haciendo que éste se encuentre disponible para cualquier servidor SIP intentando encontrar un usuario.

Los mensajes REGISTER también contienen los momentos en los que se dio el registro. Por ejemplo, el usuario A puede registrar su ubicación actual hasta las cuatro de la tarde, ya que sabe que luego dejará su oficina. Un usuario también podría estar registrado en varias locaciones al mismo tiempo, indicando al servidor que puede buscar al usuario en todas las ubicaciones registradas hasta que éste sea alcanzado.

- ACK: las *ACK requests* (solicitudes ACK) son utilizadas para acusar el recibo de una respuesta final a un INVITE. De esta forma, un cliente originando una solicitud INVITE genera un *ACK-response* (respuesta de ACK) cuando recibe el *final-INVITE* (repuesta final para el INVITE), proveyendo un intercambio de tres vías (*three-way handshake*).
- BYE: las solicitudes BYE (*BYE requests*) se usan para abandonar una sesión. En una sesión de dos partes, el abandono de una de las partes implica la terminación de la sesión. Por ejemplo, cuando un usuario A envía un BYE al usuario B, su sesión es terminada automáticamente. En escenarios *multicast*, sin embargo, una solicitud BYE de uno de los participantes solo significa que un participante particular deja la conferencia. La sesión en sí misma no se ve afectada. De hecho, es una práctica común en grandes sesiones *multicast* no enviar un BYE cuando se deja la sesión.

[Camarillo G., 2002]

Deabajo un establecimiento de sesión exitoso donde se puede ver lo descripto en los párrafos anteriores.



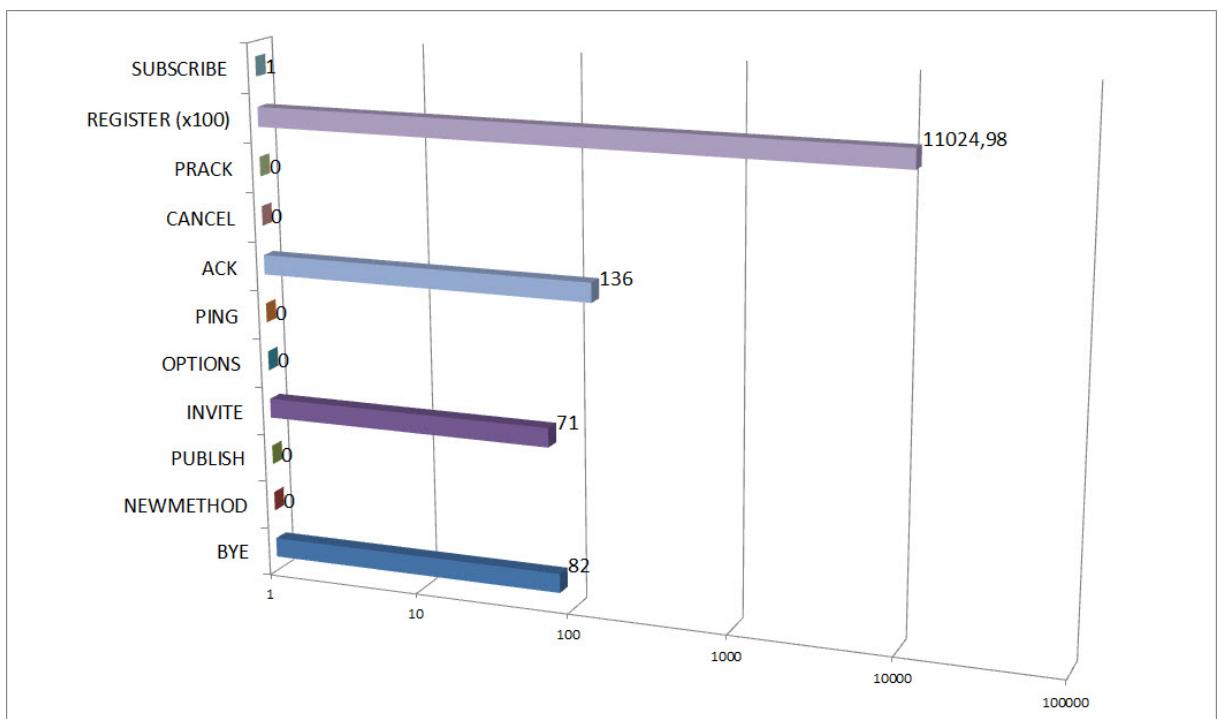


Figura 3.37: Cantidad de paquetes SIP total por tipo de método SIP (instancias Artemisa).

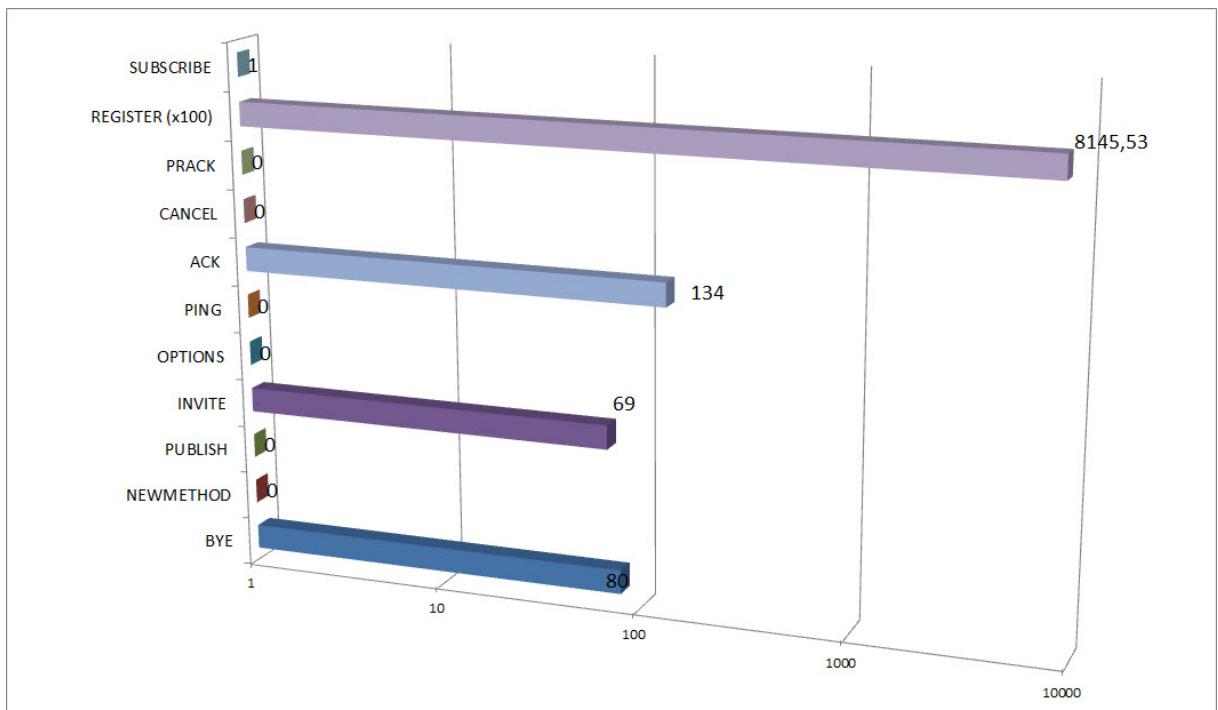


Figura 3.38: Cantidad de paquetes SIP total por tipo de método SIP (Artemisa self-hosted).

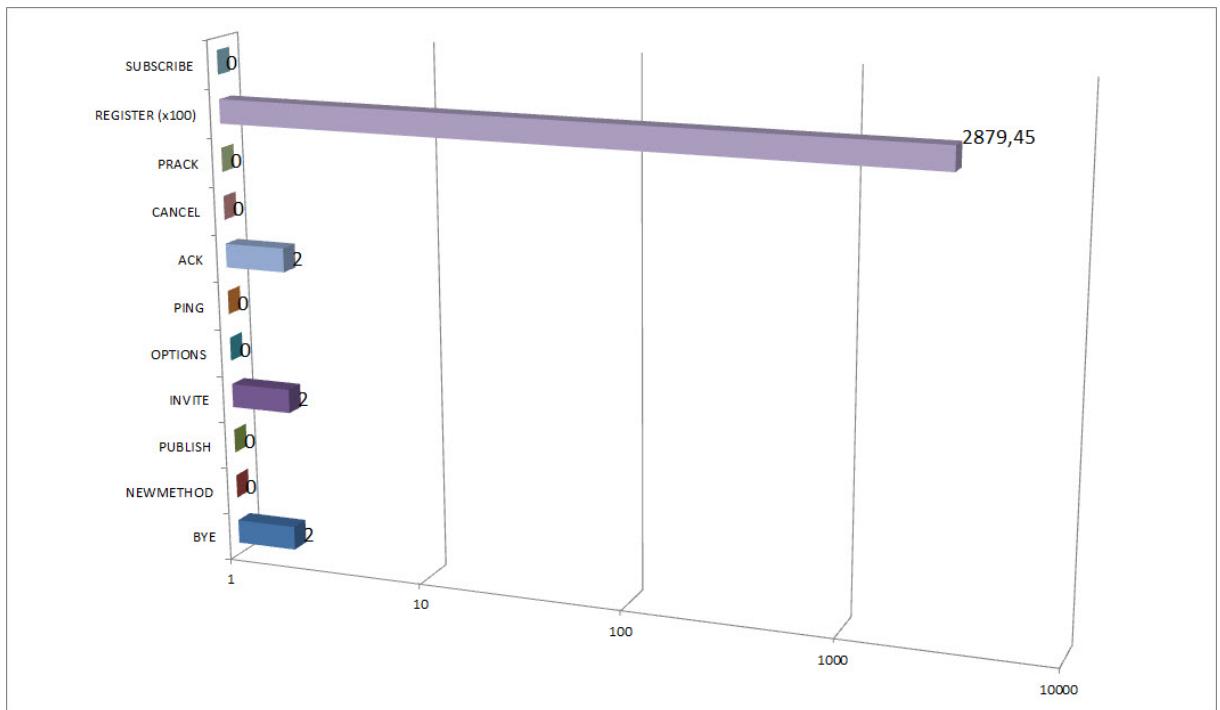


Figura 3.39: Cantidad de paquetes SIP total por tipo de método SIP (Artemisa *hosted-services*).

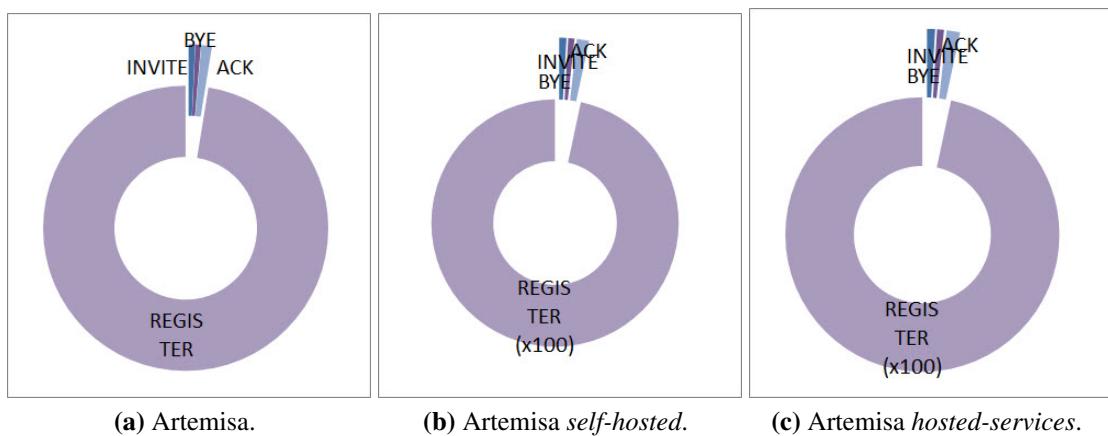


Figura 3.40: Cantidad de paquetes SIP total por tipo de método SIP (instancias Artemisa).

Del análisis realizado a los primeros gráficos incluidos en las Figuras 3.33, 3.34, 3.35, 3.36, 3.37, 3.38, 3.39 y 3.40 en el comienzo de esta sección, se puede inferir que debido a un posible intento de intrusión existe un muy elevado número de REGISTERS.

3.6.5.3. Cantidad de llamadas VoIP SIP total instancias Artemisa

Se continúa con el análisis de la cantidad de llamadas VoIP total en las instancias de Artemisa presentadas en los gráficos incluidos en las figuras 3.41 y 3.42. Por un lado se observa un número importante de llamadas completadas y rechazadas. Aunque no son significativas con respecto a la cantidad de intentos de REGISTER. Por su parte, más acorde a la cantidad de INVITES capturados. En cuanto al entorno *hosted-services* se puede apreciar que solo hubo una llamada exitosa evidenciada con los 2 INVITES presentes.

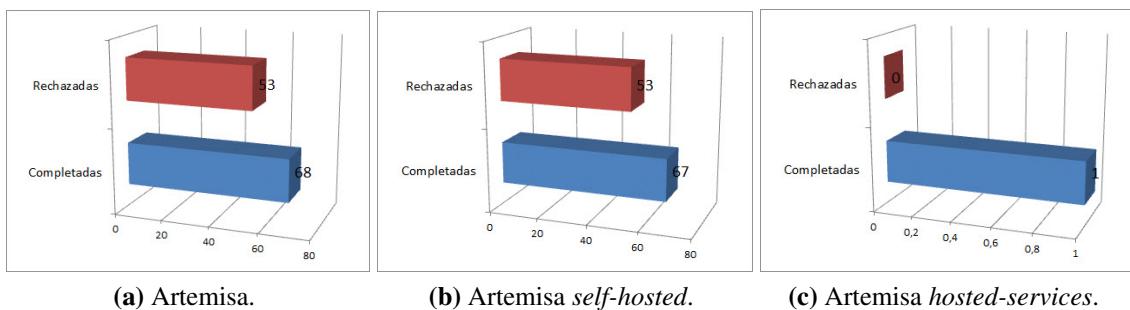


Figura 3.41: Cantidad de llamadas VoIP total (instancias Artemisa).

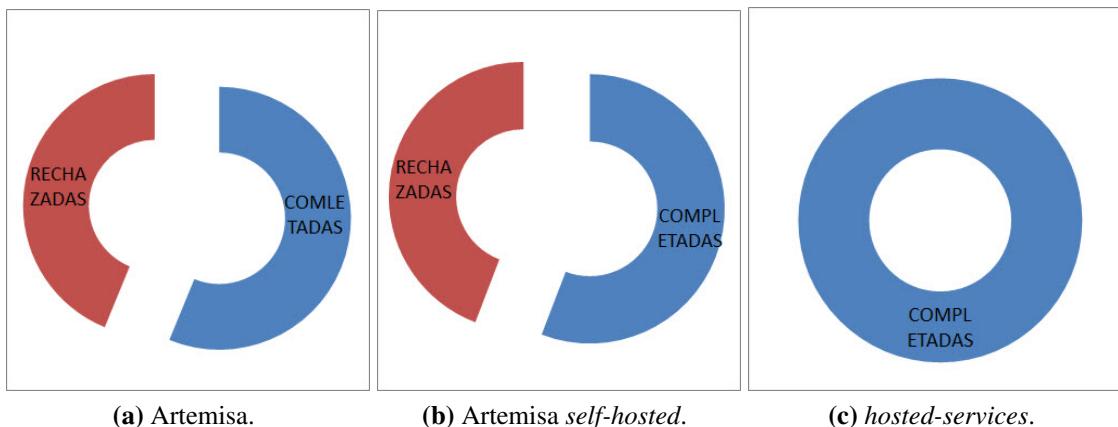


Figura 3.42: Cantidad de llamadas VoIP total (instancias Artemisa).

3.6.6. Estadísticas registrars SIP

3.6.6.1. Cantidad de paquetes SIP total por tipo de mensaje SIP registrars

En cuanto a la cantidad de paquetes SIP total por tipo de mensaje SIP de las Figuras 3.43 y 3.43 se encontraron otros tipos de paquetes SIP no considerados anteriormente los cuales se detallan debajo:

- 404 *Not Found*: el servidor posee información definitoria de que el usuario no existe en el dominio especificado en el *Request URI*. Este estado es también devuelto si el dominio

en el *Request URI* no coincide con ninguno de los dominios manejados por el receptor de la solicitud.

- 405 *Method Not Allowed*: el método especificado en la *Request-Line* es entendible, pero no permitido por la dirección identificada mediante el *Request URI*. La respuesta debe incluir un campo de encabezado *Allow header field* que contenga una lista de los métodos válidos para la dirección indicada.
- 408 *Request Timeout*: el servidor no ha podido producir una respuesta dentro de un tiempo apropiado, por ejemplo, si este no ha podido determinar la localización del usuario a tiempo. El cliente podría repetir la solicitud sin realizar modificaciones en cualquier momento posterior.
- 483 *Too Many Hops*: el servidor recibió una solicitud que contiene el campo *Max-Forwards header field* (encabezado Máximos-Reenvíos) con el valor en cero.
- 486 *Busy Here*: el sistema final del usuario que recibe el llamado, fue contactado de manera satisfactoria, pero este usuario actualmente no tiene intenciones o no está disponible para tomar llamadas adicionales en este sistema final. El usuario podría también estar disponible.
- 488 *Not Acceptable Here*: la respuesta tiene el mismo significado que un 606 (No aceptable - *Not Acceptable*), pero solo aplica a una dirección de recurso específico mediante el *Request URI* y la solicitud puede ser exitosa en cualquier lugar. Un cuerpo de mensaje que contenga una descripción de las capacidades puede estar presente en la respuesta, la cual está formateada de acuerdo al campo *Accept* del encabezado en el mensaje INVITE (o *application/SDP* si no se encuentra presente), de igual forma que en el cuerpo de un mensaje en una respuesta 200 (OK) a una solicitud OPTIONS.
- 501 *Not Implemented*: el servidor no soporta la funcionalidad requerida para completar la solicitud. Esta es la respuesta apropiada cuando un UAS no reconoce el método de solicitud y no es capaz de soportarlo para ningún usuario. (Los *proxies* reenvían todas las solicitudes a pesar del método). Notar que un 405 *Method Not Allowed* (Método no Permitido) es enviado cuando un servidor reconoce el método de solicitud, pero ese método no está permitido o soportado.
- 513 *Message Too Large*: el servidor no fue capaz de procesar la solicitud ya que la longitud del mensaje excede sus capacidades.
- 603 *Decline*: la máquina del usuario que recibe la llamada fue contactada satisfactoriamente, pero el usuario explícitamente no desea o no puede participar. La respuesta puede indicar un mejor horario para llamar en el campo *Retry-After* del encabezado. Este estado de respuesta es solo devuelto si el cliente sabe que ningún otro *endpoint* responderá la solicitud.

Más allá de estos nuevos tipos de mensajes SIP es posible apreciar que el comportamiento del *registrar* se encuentra directamente relacionado con las capturas analizadas de las instancias de Artemisa. Es decir que también se aprecian en los gráficos, que los tipos de mensajes que predominan son:

- 100 *Trying*: es un mensaje SIP muy común al intentar establecer una llamada, podría o no estar relacionada con una llamada mal intencionada.
- 200 *OK*: en respuesta a las solicitudes exitosas procesadas por Artemisa. Tanto recibidas como realizadas.
- 401 *Unauthorized*, error en la autenticación contra Artemisa, podrían ser asociados a intentos de registro fallidos. Posiblemente no esperados, es decir generados por un atacante.
- 404 *Not Found*: el servidor posee información definitoria de que el usuario no existe en el dominio especificado en el *Request URI*. Este estado es también devuelto si el dominio en el *Request URI* no coincide con ninguno de los dominios manejados por el receptor de la solicitud. Claramente pueden estar asociados a solicitudes de un atacante al intentar alcanzar usuarios inexistentes en un *scanning attack*.
- 483 *Too Many Hops* y 513 *Message Too Large*: aunque no necesariamente, estos podrían estar relacionados a mensajes mal formados de un usuario malicioso.
- 500 *Server Internal Error*: esta respuesta está íntimamente relacionada a que el servidor, en nuestro caso Asterisk, se enfrentó con una condición inesperada que impidió al mismo completar la solicitud. Esto está asociado a que Artemisa solo tienen la capacidad de responder a ciertos mensajes SIP considerando que el mismo actua como un UA y no tiene la capacidad de un proxy SIP o registrar.

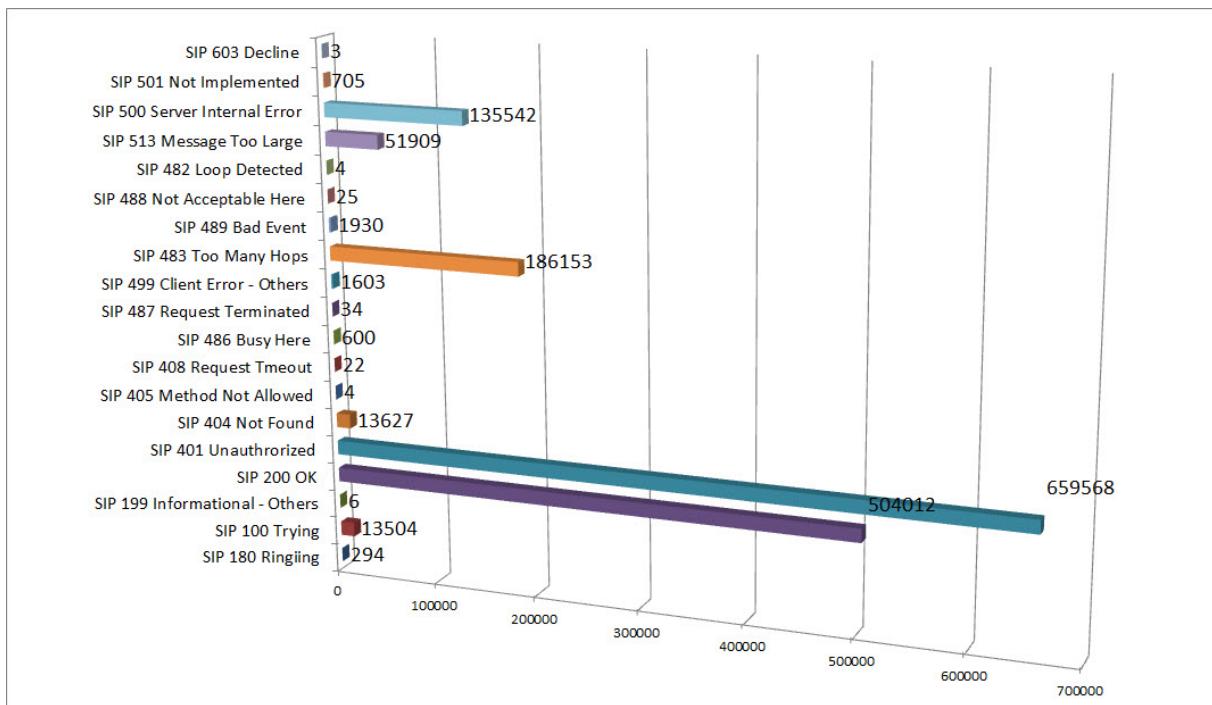


Figura 3.43: Cantidad de paquetes SIP total por tipo de mensaje SIP (*self-hosted registrars*).

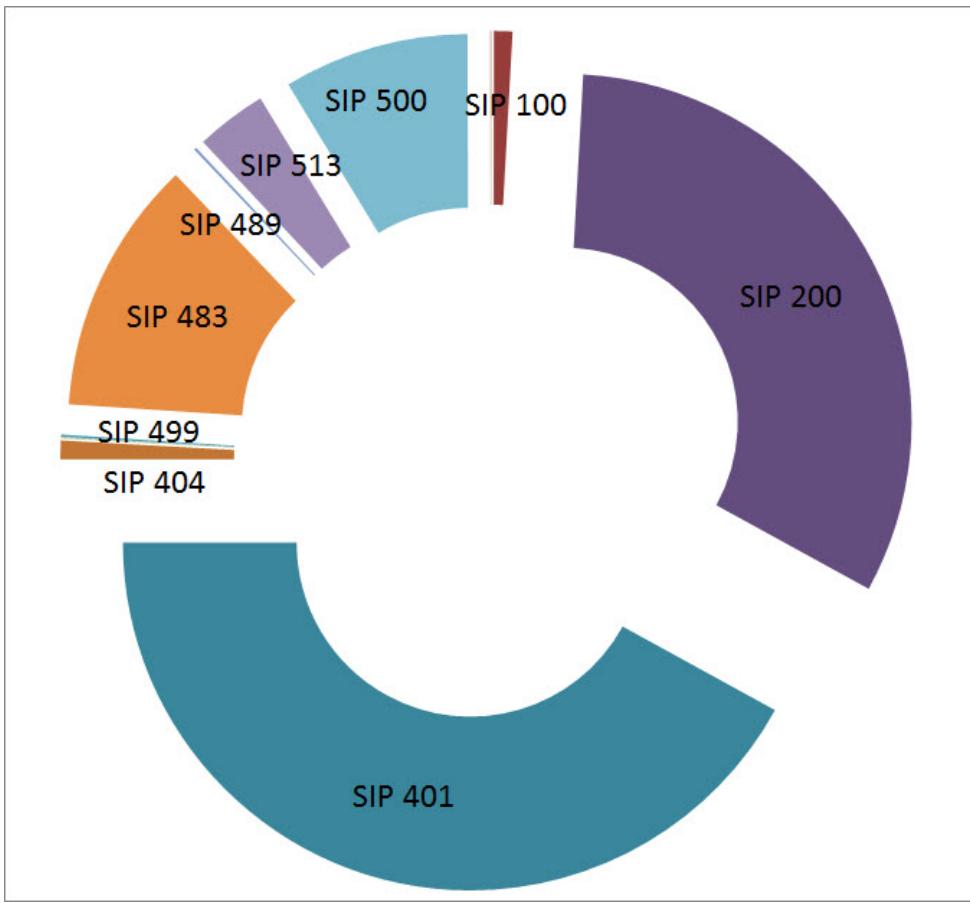


Figura 3.44: Cantidad de paquetes SIP total por tipo de mensaje SIP (*self-hosted registrars*).

3.6.6.2. Cantidad de paquetes SIP total por tipo de método SIP *registrars*

Analizando ahora las Figuras 3.45 y 3.46 de la cantidad de paquetes por tipo de método SIP para los *registrar* Asterisk y SER de las camas de prueba.

- SUBSCRIBE: el *framework*¹⁵ de notificación de eventos SIP permite a SIP informar a los usuarios acerca de una variedad de eventos en los cuales han indicado un previo interés mediante señalización. Como por ejemplo, solicitando el estado de un usuario en particular para saber en qué momento poder comunicarse en caso de que al momento de llamar este ocupado, evitando así llamar reiteradas veces. Como es esto realizado, dos métodos fueron definidos para proveer notificación de eventos asíncronos: SUBSCRIBE y NOTIFY. SUBSCRIBE es usado por una entidad SIP para declarar su interés en un evento en particular. Una entidad SIP se suscribe a un cierto evento de una clase de eventos. Cuando el evento al que el usuario se ha suscrito ocurre, las solicitudes NOTIFY son enviadas contenido información acerca de la sesión.
- CANCEL: son solicitudes que cancelan las transacciones pendientes. Si un servidor SIP

¹⁵**Framework:** en el desarrollo de software es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio y provee una estructura y una especial metodología de trabajo, la cual extiende o utiliza las aplicaciones del dominio. [Riehle D., 2000]

ha recibido un INVITE pero aun no ha retornado una respuesta final, va a parar de procesar el INVITE en cuanto reciba un CANCEL. Pero, sin embargo, ha respondido con una respuesta final para este INVITE, la solicitud CANCEL no tendrá efecto en la transacción.

- OPTIONS: estas solicitudes consultan al servidor acerca de sus capacidades y funciones, incluyendo qué métodos y que tipos de sesión soportan. Un servidor SIP podría responder a una solicitud OPTIONS que soporta SDP como protocolo de descripción de sesión y cinco métodos: INVITE, ACK, CANCEL, BYE, y OPTIONS. Debido que el servidor no soporta el método REGISTER, luego se podría deducir que este dispositivo no es un *registrar*. El método OPTIONS, podría no verse útil en este momento, pero a medida que nuevas extensiones agregan nuevos métodos a SIP, el método OPTIONS es una excelente manera de descubrir qué métodos específicos son soportados por un servidor en particular.

[Camarillo G., 2002]

- PUBLISH: se declaró un nuevo método SIP, PUBLISH, para publicar estado de eventos. PUBLISH es similar a REGISTER permitiendo a un usuario crear, modificar, y remover estados en otra entidad que maneja este estado en nombre del usuario (ej. un *register server*). Direccionar una solicitud PUBLISH es idéntico a direccionar una solicitud SUBSCRIBE. El *Request URI* de una solicitud PUBLISH, es completada con la dirección del recurso para el cual el usuario desea publicar el estado del evento. El usuario podría tener múltiples *endpoints* o *user-agents* que publican estado de eventos. Cada *endpoint* podría publicar su propio estado único, a partir del cual el compositor de estado de eventos genera el estado de evento compuesto del recurso. Adicionalmente, a un recurso en particular, todo evento de estado publicado es asociado con un paquete de eventos específico. A través de una suscripción a este paquete de eventos, el usuario es capaz de descubrir el estado de eventos compuesto de todas las publicaciones activas.

[Niemi, 2004]

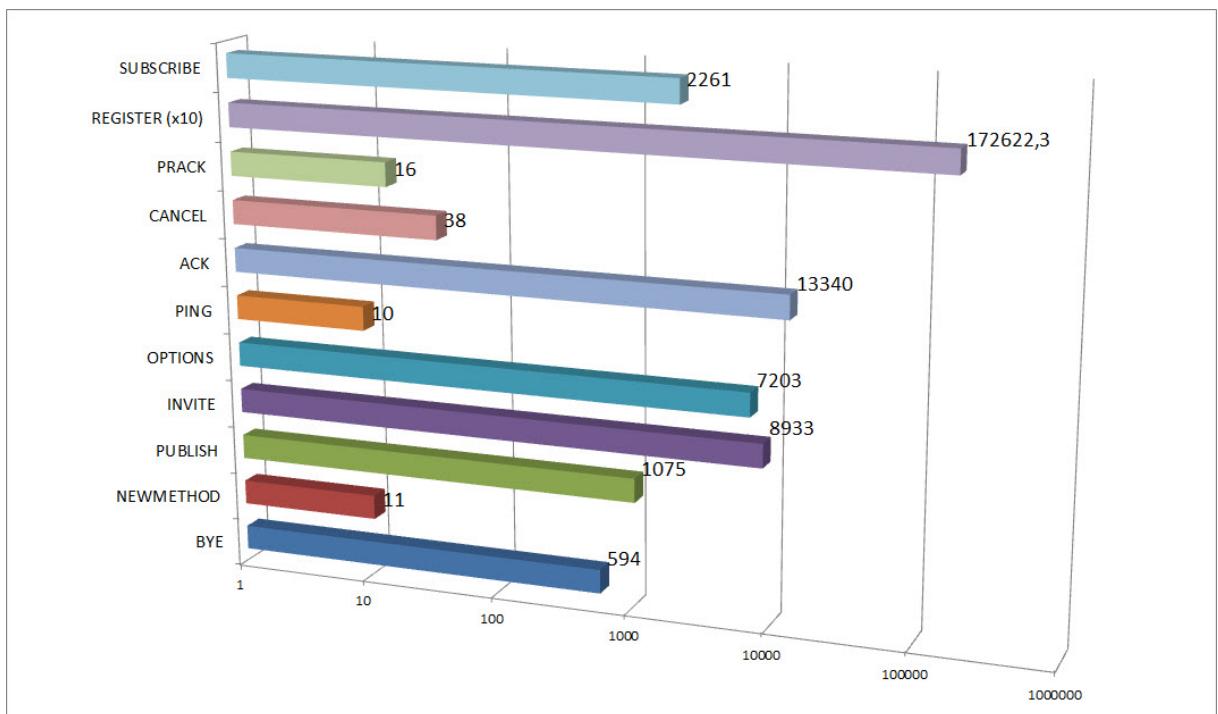


Figura 3.45: Cantidad de paquetes SIP total por tipo de método SIP (*self-hosted registrars*).

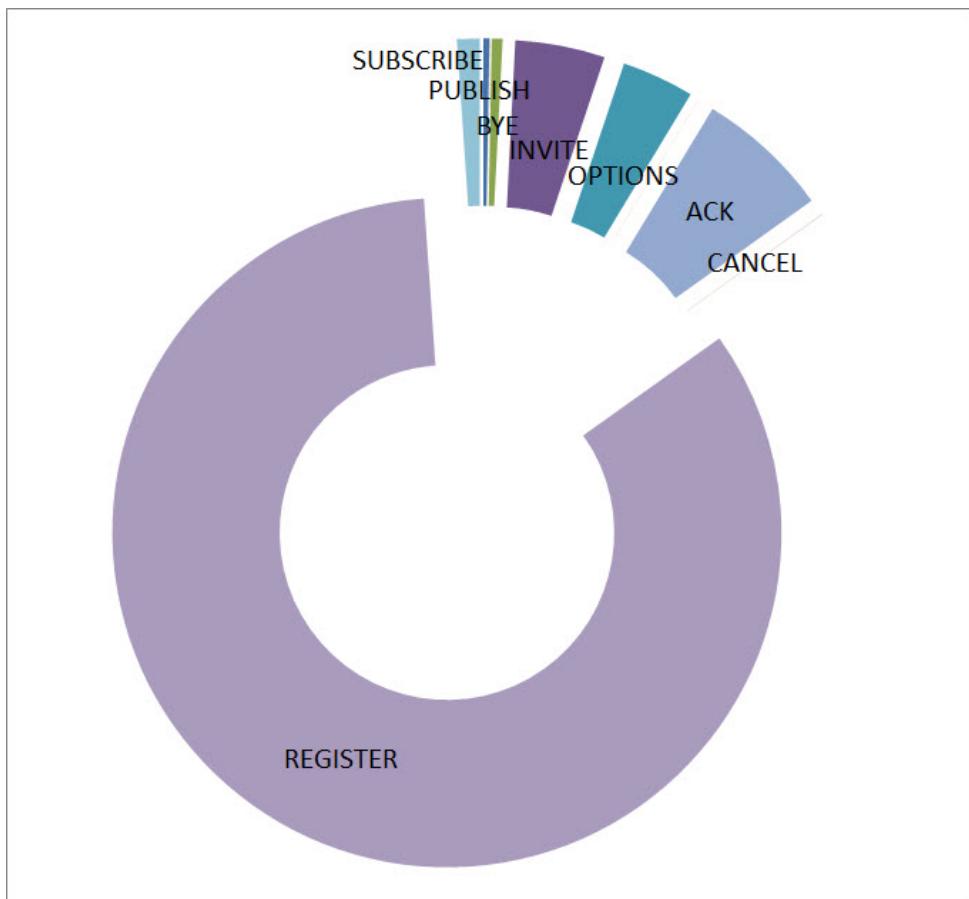


Figura 3.46: Cantidad de paquetes SIP total por tipo de método SIP (*self-hosted registrars*).

3.6.6.3. Cantidad de llamadas VoIP SIP total registrars

Tal se muestra en los gráficos de la Figura 3.47, por un lado se ve un número importantísimo de llamadas rechazadas, lo que se ve directamente reflejado por las estadísticas del método REGISTER, como así también de los mensajes SIP que 401 Unauthorized y 404 Not Found involucrados en este proceso. Los cuales resultan de una gran cantidad de registros fallidos. Por su número exagerado, podría ser asociado a intentos de registro mal intencionados. Se puede confirmar la anterior suposición al verificar los *logs* de Asterisk. Por su parte, se aprecian un número importante de SIP 200 OK, SIP 100 trying y SIP 180 ringing relacionados a las 591 llamadas realizadas. Las cuales podrían haber sido generadas por atacantes considerando que no se esperaban llamadas dentro del dominio. Cabe aclarar que un porcentaje de las mismas se encuentran asociadas a llamadas intencionadas que se realizaron para probar el SIP *register*.

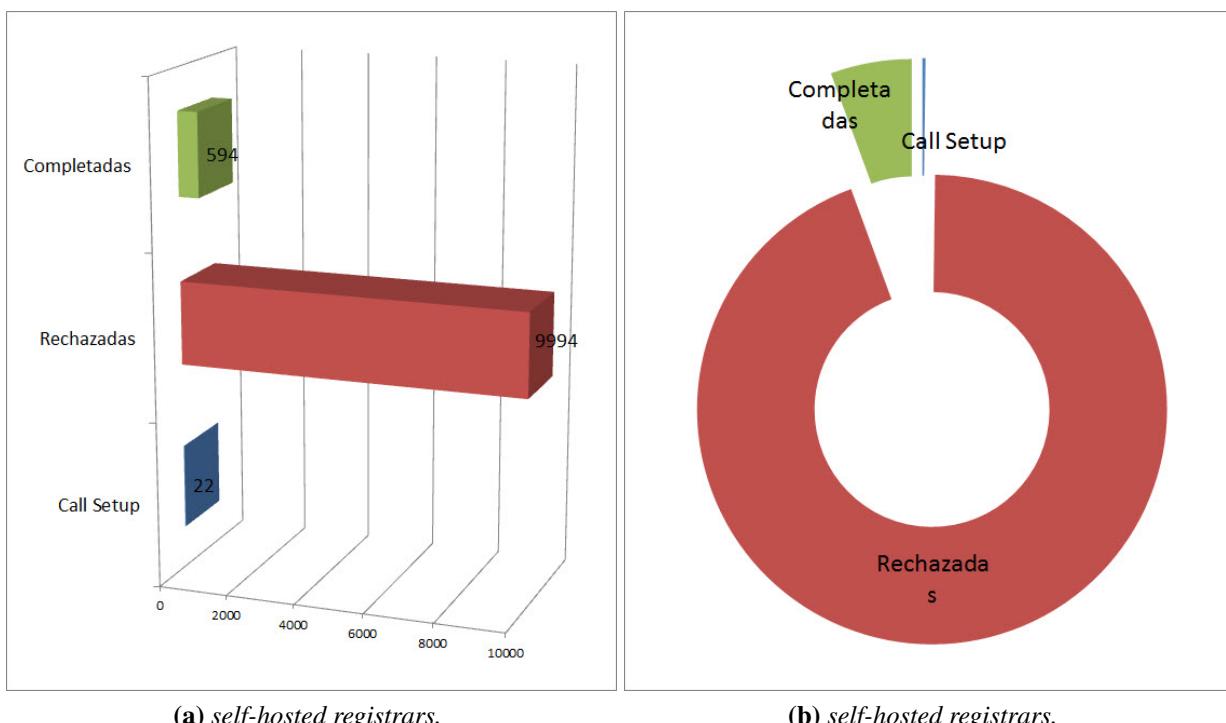


Figura 3.47: Cantidad de llamadas VoIP SIP total (*registrars*).

3.6.7. Estadísticas ataques VoIP

3.6.7.1. Cantidad de paquetes SIP total por tipo de ataque

Recordando lo estudiado en el marco teórico de la sección 3.3 de riesgos y seguridad en VoIP, como así también la descripción y respuesta a cada tipo de ataque detectado por Artemisa en la unidad 3.5.3, a través del caso 1. Se resumen a continuación los tipos de ataques y resultados obtenidos de los *logs* de Artemisa de las Figuras 3.48, 3.49, 3.50 y 3.51. Cabe destacar la efectividad que tuvo la plataforma en los tres entornos propuestos para detectar los ataques cubiertos por la misma. Principalmente los ataques de inundación en un número mayor a 180.000 durante el período de exposición de seis meses. Sin descontar los 102 ataques de ringing y scanning attempts. Luego se podría decir que los entornos no recibieron ataques de SPIT.

- DoS - *flooding attacks*. Los ataques DoS son probablemente el tipo de ataques más comunes en las comunicaciones VoIP. Un DoS attack puede ocurrir cuando un gran número de solicitudes INVITE inválidas son enviadas a un *proxy server* en un intento de congestionar el sistema. Los ataques son relativamente simples de implementar, y su efecto en los usuarios del sistema son inmediatos. SIP tiene varios métodos para minimizar los efectos de los ataques DoS, pero finalmente estos son imposibles de prevenir. [L. Madsen, 2011] Directamente relacionado con un DoS, se encuentran los ataques de inundación que tienen como objetivo los elementos del plano de señalización (ej.: *proxy*, *gateway*, etc.) con el fin de tirarlo abajo y producir un colapso en la red de VoIP. [Nassar M., 2009] En otras palabras, un ataque de denegación de servicio (DoS) apunta a afectar la disponibilidad del servicio de VoIP enviando grandes cantidades de datos en general o solicitudes específicas de un protocolo, para consumir los recursos de la red o sobrecargar un servidor específico respectivamente. Las posibles herramientas a ser utilizadas serían: Iviteflood y Rtplood.
- Scanning attemp (escaneo). A través de este tipo de ataques se busca recopilar información sobre la topología, servidores y clientes del entorno VoIP. El usuario malicioso estaría interesado en encontrar *hosts* activos; tipos y versiones de PBXs, *servers/gateways* y clientes (*hardphones* y *softphones*). Por lo tanto se destinan los ataques a la dirección IP de cada *host*, por ejemplo la dirección IP de Artemisa. Las herramientas utilizadas podrían ser: SMAP, SIP-SCAN, SVMAP y METASPLOIT, entre otras.
- Enumeración (*enumerating*). Podría considerarse como *scanning*. Consiste en descubrir extensiones SIP válidas para el entorno, lo que permite luego apuntar los ataques de fuerza bruta y demás sobre tales cuentas. Trabaja analizando los mensajes de error ante las solicitudes de métodos OPTIONS, INVITE y REGISTER, enviadas apuntando a distintas extensiones para determinar cuáles son válidas en el entorno SIP en cuestión. Herramientas utilizadas: SVWAR y METASPLOIT. Para un resultado útil, desde el punto de vista del atacante, los mensajes deben ser enviados al servidor de registro, Asterisk para este escenario de pruebas. Debe tenerse en cuenta que un factor importante del éxito del ataque, es el diccionario o rango de extensiones utilizadas con la herramienta.

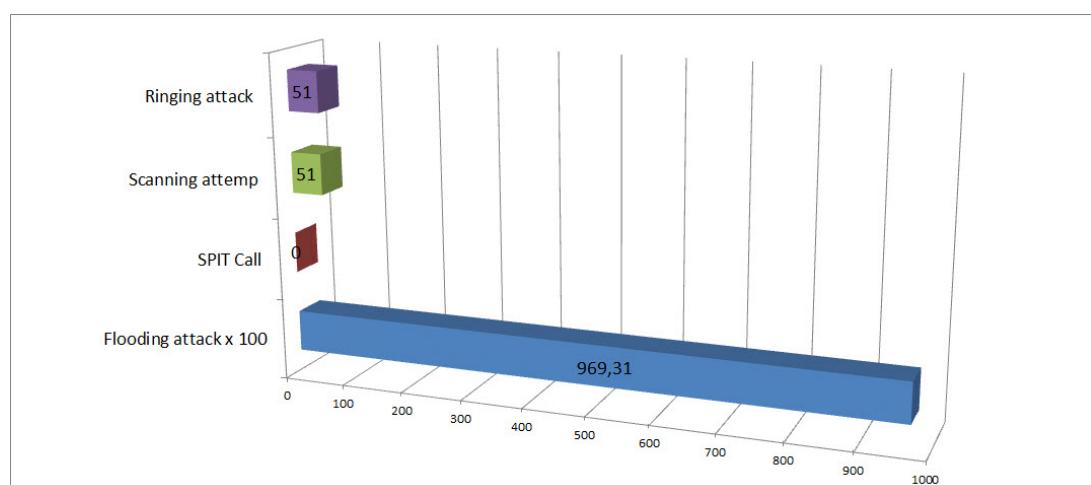


Figura 3.48: Cantidad de paquetes SIP total por tipo de ataque (instancias Artemisa).

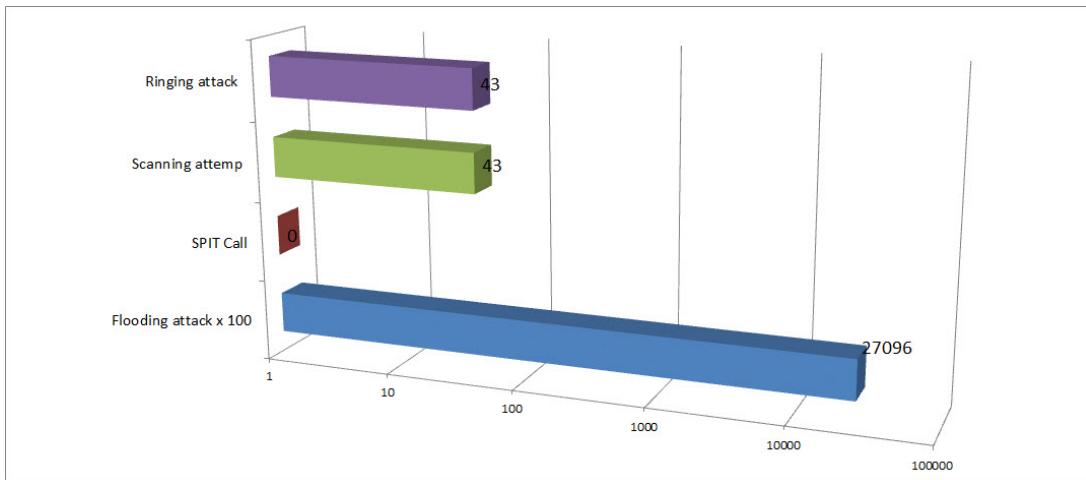


Figura 3.49: Cantidad de paquetes SIP total por tipo de ataque (Artemisa *self-hosted registrars*).

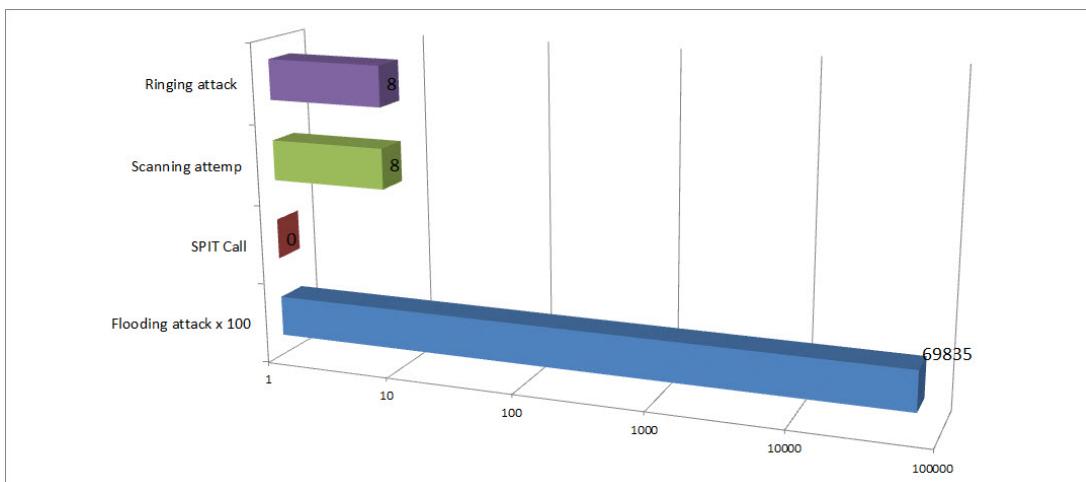


Figura 3.50: Cantidad de paquetes SIP total por tipo de ataque (Artemisa *hosted-services registrars*).

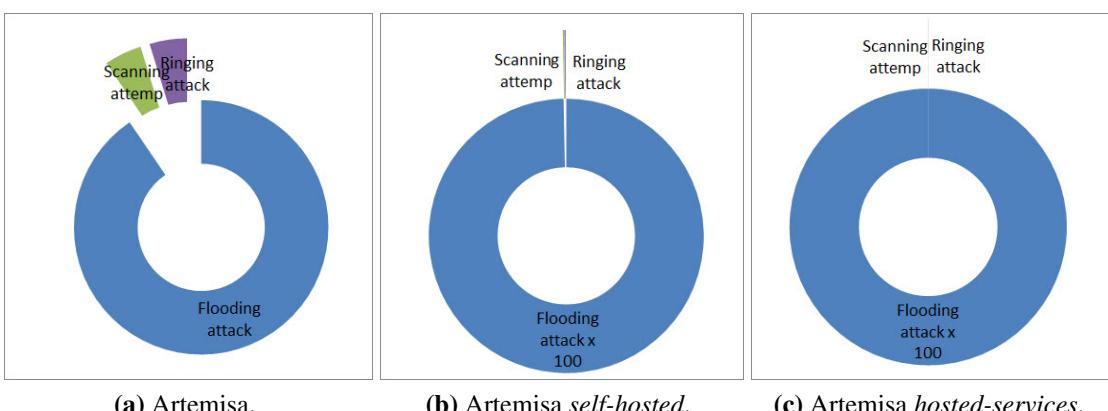


Figura 3.51: Cantidad de paquetes SIP total por tipo de ataque (instancias Artemisa).

3.6.7.2. Cantidad de paquetes SIP por tipo ataque (Asterisk Metropolitan)

Recordando que para este caso de estudio se expuso una instancia de Asterisk la cual reportó en sus *logs* como se aprecia en la Figura 3.52 un números muy significativos de eventos, *No matching peer found* (no se encuentra coincidencia con un par). SIP dispone de un sistema de

autenticación de usuarios *challenge/response* (desafío/respuesta). Un INVITE inicial es enviado al *proxy* con el cual el dispositivo final desea comunicarse. El *proxy* responde luego con un mensaje *407 Proxy Authorization Request*, que contiene un set random de caracteres al cual nos referimos como “*nonce*”. Este es usado conjuntamente con el *password* para generar el *hash MD5*¹⁶, el cual luego es enviado como respuesta en el INVITE subsecuente. Asumiendo que el hash MD5 coincide con el que fue generado por el *proxy*, el cliente luego se encuentra autenticado.

Si se expone al sistema Asterisk a la Internet pública, una de las cosas que con seguridad se ve es un *scan* para descubrir cuentas válidas (lo cual claramente podrá validarse en los *logs* presentados debajo). Este tipo de escaneos comienzan chequeando varios nombres de usuarios comunes, y a continuación de estos se comienza a escanear por cuentas numeradas. Es común dentro de los usuarios configurar el nombre para sus cuentas SIP de la misma forma que su extensión en la PBX. Ej.: extensión PBX = “1001” y usuario SIP = “1001”. Este *scan* saca ventaja de esta realidad. Esto nos lleva a un primer consejo relacionado a la seguridad de Asterisk:

Tip #1: se deberían seleccionar nombres de usuario no-numéricos para las cuentas VoIP para que así se dificulte la posibilidad de adivinarlas. Por ejemplo, se podría usar la dirección MAC de un teléfono SIP como su nombre de cuenta en Asterisk. Estos escaneos de cuenta sacan ventaja del hecho de que las respuestas que provienen del servidor por un intento de registro van a diferir dependiendo de si la cuenta existe o no. Si esta existe, el servidor solicitará autenticación. Si la cuenta no existe, el servidor inmediatamente denegará el intento de registro. Este comportamiento es simplemente como se define el protocolo. Esto nos lleva a un segundo consejo de seguridad para Asterisk:

Tip #2: setear “*alwaysauthreject*” a “*yes*” en la sección [general] de */etc/asterisk/sip.conf*. Esta opción indica a Asterisk a responder como si cualquier cuenta fuera válida, lo que hace inútiles a los escaneos por nombres de usuarios. Incluso si tenemos nombres de usuarios que son difíciles de adivinar, es de vital importancia que a su vez dispongamos de contraseñas fuertes. Si un atacante es capaz de obtener un nombre de usuario válido, este va a intentar quebrarla mediante fuerza bruta. *Passwords* fuertes van a hacer que esto sea mucho más difícil de llevar a cabo. El esquema de autenticación por defecto para el protocolo SIP es débil. La autenticación es realizada a través de un desafío MD5 (*MD5 challenge*) y mecanismo de respuesta a este desafío. Si el atacante es capaz de capturar cualquier tráfico de llamada, como una llamada SIP realizada desde una *laptop* en una red *wireless* abierta, sería mucho más fácil trabajar en quebrar el *password* mediante fuerza bruta. Esto se debe a que ya que no requeriría la solicitud de autenticación al servidor porque se trabajaría sobre el *challenge* capturado.

Tip #3: hacer uso de *passwords* fuertes. Existen innumerables recursos disponibles en Internet que pueden ayudar a definir lo que constituye un *password* fuerte. Por su parte también hay disponibles muchos generadores de *passwords* fuertes los cuales podrían ser usados.

¹⁶**hash MD5:** El algoritmo de mensaje-resumen (*message-digest*) es una función de resumen criptográfico (*cryptographic hash function*) que produce un valor de hash (o resumen criptográfico) de 128-bits (16-bytes), típicamente expresado en formato de texto como un número de 32 dígitos hexadecimales. MD5 ha sido utilizado en una amplia variedad de aplicaciones criptográficas, y también es comúnmente usado para verificar la integridad de datos. Los MD5 *digests* han sido ampliamente empleados en el mundo del software para proveer alguna garantía de que un archivo ha arribado intacto. Por ejemplo, los servidores de archivos generalmente proveen un *checksum* MD5 pre-computado (conocido como un MD5sum) para los archivos, de forma tal que los usuarios sean capaces de contrastar el *checksum* del archivo descargado con éste. La mayoría de los OS basados en Unix incluyen utilidades MD5sum en sus paquetes de distribución; los usuarios de Windows deberían instalar una utilidad de Microsoft o correr alguna aplicación de tercero para poder realizarlo. [Pachghare, 2008].

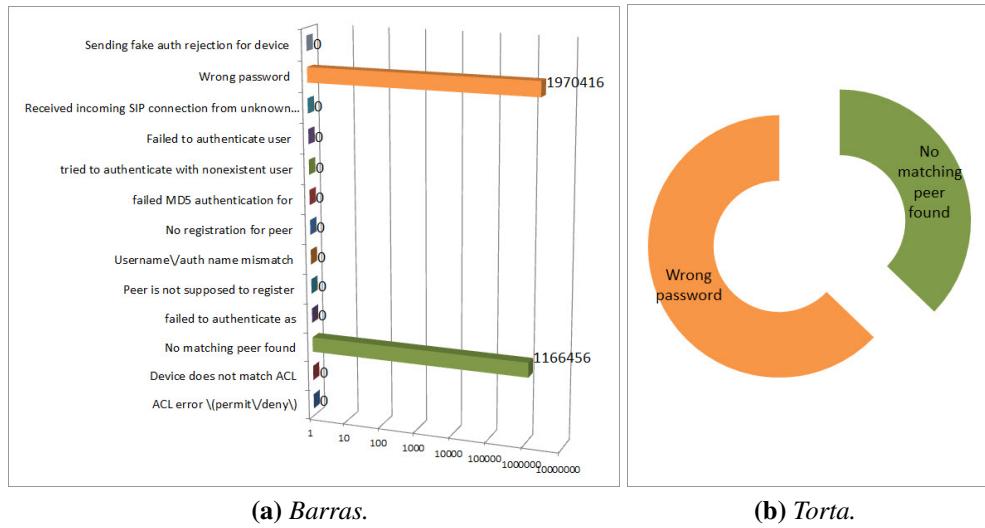


Figura 3.52: Cantidad de paquetes SIP por tipo ataque (Asterisk Metropolitan *registrar*).

Captura *brute-force attack "No matching peer found"* en el directorio /var/log/asterisk/messages.x:

```
[OK][Feb 22 19:01:06] [1;33mNOTICE [0m[1486]: [1;37mchan_sip.c
[0m:[1;37m21770[0m [1;37mhandle_request_register[0m: Registration from
'"1"<sip:1@186.137.221.179>' failed for '192.168.0.101' - No matching peer found
[OK][Feb 22 19:01:06] [1;33mNOTICE [0m[1486]: [1;37mchan_sip.c [0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"2"<sip:2@186.137.221.179>' failed for '192.168.0.101' - No matching peer found
[OK][Feb 22 19:01:06] [1;33mNOTICE [0m[1486]: [1;37mchan_sip.c [0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"3"<sip:3@186.137.221.179>' failed for '192.168.0.101' - No matching peer found
[OK][Feb 22 19:01:06] [1;33mNOTICE [0m[1486]: [1;37mchan_sip.c [0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"10"<sip:10@186.137.221.179>' failed for '192.168.0.101' - No matching peer found
[OK][Feb 22 19:01:06] [1;33mNOTICE [0m[1486]: [1;37mchan_sip.c [0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"11"<sip:11@186.137.221.179>' failed for '192.168.0.101' - No matching peer found
[OK][Feb 22 19:01:06] [1;33mNOTICE [0m[1486]: [1;37mchan_sip.c [0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"12"<sip:12@186.137.221.179>' failed for '192.168.0.101' - No matching peer found
[OK][Feb 22 19:01:06] [1;33mNOTICE [0m[1486]: [1;37mchan_sip.c [0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"13"<sip:13@186.137.221.179>' failed for '192.168.0.101' - No matching peer found
...
[OK][Feb 22 19:01:09] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"Administrator"<sip:Administrator@186.137.221.179>' failed for '192.168.0.101' - No matching peer found
[OK][Feb 22 19:01:09] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"administrator"<sip:administrator@186.137.221.179>' failed for '192.168.0.101' - No matching peer found
[OK][Feb 22 19:01:09] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"superuser"<sip:superuser@186.137.221.179>' failed for '192.168.0.101' - No matching peer found
[OK][Feb 22 19:01:09] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"security"<sip:security@186.137.221.179>' failed for '192.168.0.101' - No matching peer found
[OK][Feb 22 19:01:09] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from
```

```
'"sysadmin"<sip:sysadmin@186.137.221.179>' failed for '192.168.0.101' - No
matching peer found
[OK[Feb 22 19:01:09] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from
'"Manager"<sip:Manager@186.137.221.179>' failed for '192.168.0.101' - No
matching peer found
[OK[Feb 22 19:01:09] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from
'"manager"<sip:manager@186.137.221.179>' failed for '192.168.0.101' - No
matching peer found
...
[OK[Feb 22 19:01:15] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from
'"Agberto"<sip:Agberto@186.137.221.179>' failed for '192.168.0.101' - No
matching peer found
[OK[Feb 22 19:01:15] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from
'"Agilberto"<sip:Agilberto@186.137.221.179>' failed for '192.168.0.101' - No
matching peer found
[OK[Feb 22 19:01:15] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from
'"Albert"<sip:Albert@186.137.221.179>' failed for '192.168.0.101' - No matching
peer found
```

Contraseña incorrecta (*wrong password*): luego de poner en funcionamiento Asterisk, el siguiente paso es configurar los teléfonos en si mismos para comunicarlos con Asterisk. Por la manera en la que han sido configuradas las cuentas SIP en nuestro servidor de registro, este esperará que los teléfonos se registren a él. El proceso de registro es simplemente un mecanismo donde un agente de usuario comunica *Soy el teléfono X...aquí mi nombre de usuario* y password. *Y en caso de que reciba una llamada mi dirección IP es la siguiente.* Luego cuando Asterisk ha detectado que el password ingresado para un teléfono no coincide con el seteo secreto en la sección *sip.conf* acusara *Wrong password*.

Queda demostrado por los resultados, los cuales evidencian en base a la altísima cantidad de intentos de *password cracking* mediante *brute force* que *toll fraud* (fraude en tasas o tarifas telefónicas) evidentemente el mayor riesgo para los sistemas telefónicos en términos del potencial para costos que podrían llevar a la ruina a casi cualquier compañía. No es desconocido para los estafadores acumular miles de dólares en llamadas robadas en el curso de unos pocos días. *Toll fraud* no es algo nuevo, incluso ha existido antes de VoIP. Sin embargo, la naturaleza permisiva de VoIP significa que es más fácil para los fraudulentos tomar ventaja de sistemas no seguros. La mayoría de los proveedores de Internet no se harán responsables de estos costos, y de esta forma si su sistema ha sido comprometido usted podría tener que lidiar con una cuenta telefónica muy grande. A medida que los *carriers* van mejorando la forma de alertar a sus clientes de actividad sospechosa, esto no lo absuelve a uno de la responsabilidad que implica asegurar que el sistema del cliente está protegido contra una amenaza peligrosa y real. Dentro del sistema Asterisk, es de vital importancia que saber qué recursos en el sistema están expuestos al mundo exterior y garantizar que esos recursos están seguros.

La forma más común de fraude de tarifas telefónicas en estos días es alcanzada mediante ataques de fuerza bruta. En este escenario, los ladrones tendrán un script que va a contactar nuestro sistema e intentar registrarse como un usuario válido. Si son capaces de registrarse como un teléfono en su sistema, la inundación de llamadas va a comenzar, y usted, como dijimos anteriormente, se verá involucrado con la factura asociada a las mismas. Si el administrador usa números de extensiones simples, contraseñas fáciles de adivinar, y su entorno acepta registros externos a su *firewall*, es muy probable que eventualmente sea víctima de un fraude de cuenta telefónica.

Los *Brute-force attacks* pueden también causar problemas de performance en su sistema, ya que uno de estos scripts pueden inundar su *router* y PBX con números masivos de intentos de

registro. Se presentan de modo complementario algunas tácticas que han sido satisfactoriamente probadas en minimizar el riesgo de *toll fraud*:

1. No utilice contraseñas fáciles de adivinar (*easy-to-guess passwords*). Los *passwords* deberían ser de al menos ocho caracteres de largo y contener una combinación de dígitos, letras y caracteres. Ej.: “8aj03H” es una buena contraseña, mientras que “1234” is not.
2. No utilice números de extensiones para los dispositivos SIP a ser registrados en *sip.conf*, es decir, en lugar de [1000], use algo como una MAC address (por ejemplo: [0004f2123456] podría ser mucho más difícil de adivinar para un ataque de fuerza bruta).
3. Implemente un script de análisis como la conocida herramienta *fail2ban* para convertir su *firewall* interno con el fin de bloquear direcciones IP que están presentando comportamiento abusivo, como por ejemplo paquetes masivos de inundación.

[L. Madsen, 2011]

Captura *brute-force attack "Wrong password"* /var/log/asterisk/messages.x:

```
[OK][Feb 22 19:01:19] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"1010"
<sip:1010@186.137.221.179>' failed for '192.168.0.101' - Wrong password
[OK][Feb 22 19:01:19] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"1010"
<sip:1010@186.137.221.179>' failed for '192.168.0.101' - Wrong password
[OK][Feb 22 19:01:19] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"1010"
<sip:1010@186.137.221.179>' failed for '192.168.0.101' - Wrong password
[OK][Feb 22 19:01:19] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"1010"
<sip:1010@186.137.221.179>' failed for '192.168.0.101' - Wrong password
[OK][Feb 22 19:01:19] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"1010"
<sip:1010@186.137.221.179>' failed for '192.168.0.101' - Wrong password
...
[OK][Feb 23 13:14:47] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"1011"
<sip:1011@186.137.221.179>' failed for '192.168.0.101' - Wrong password
[Feb 23 13:14:47] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"1011"
<sip:1011@186.137.221.179>' failed for '192.168.0.101' - Wrong password
[Feb 23 13:14:47] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"1011"
<sip:1011@186.137.221.179>' failed for '192.168.0.101' - Wrong password
[Feb 23 13:14:47] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"1011"
<sip:1011@186.137.221.179>' failed for '192.168.0.101' - Wrong password
[Feb 23 13:14:47] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"1011"
<sip:1011@186.137.221.179>' failed for '192.168.0.101' - Wrong password
[Feb 23 13:14:47] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"1011"
<sip:1011@186.137.221.179>' failed for '192.168.0.101' - Wrong password
[Feb 23 13:14:47] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"1011"
<sip:1011@186.137.221.179>' failed for '192.168.0.101' - Wrong password
[Feb 23 13:14:47] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"1011"
<sip:1011@186.137.221.179>' failed for '192.168.0.101' - Wrong password
...
[Feb 23 13:16:45] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"1014"
<sip:1014@186.137.221.179>' failed for '192.168.0.101' - Wrong password
[Feb 23 13:16:45] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"1014"
<sip:1014@186.137.221.179>' failed for '192.168.0.101' - Wrong password
```

```

[Feb 23 13:16:45] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"1014"
<sip:1014@186.137.221.179>' failed for '192.168.0.101' - Wrong password
[Feb 23 13:16:45] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"1014"
<sip:1014@186.137.221.179>' failed for '192.168.0.101' - Wrong password
[Feb 23 13:16:45] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"1014"
<sip:1014@186.137.221.179>' failed for '192.168.0.101' - Wrong password
[Feb 23 13:16:45] [1;33mNOTICE[0m[1486]: [1;37mchan_sip.c[0m:[1;37m21770[0m
[1;37mhandle_request_register[0m: Registration from '"1014"
<sip:1014@186.137.221.179>' failed for '192.168.0.101' - Wrong password

```

Se adjuntan tablas y gráficos relacionadas en Anexo C para mayor información.

3.6.8. Estadística tráfico VoIP SIP

Las Tablas 3.10 y 3.11 y Figuras 3.54, 3.55, 3.56 y 3.57 del *testbed* Metropolitan, como también las Tablas 3.12 y 3.13 y Figuras 3.58, 3.59, 3.60 y 3.61 del *testbed* UBP, presentan una comparativa entre el tráfico generado por un UA o un posible atacante con origen externo a nuestro dominio, es decir desde Internet, al cual llamaremos tráfico público/externo con IP de origen pública. Contrastado con el tráfico restante, el cuál será el originado dentro de nuestra red LAN, es decir tráfico privado/interno, los paquetes tendrán IP origen asociadas al *registrar*, Artemisa o UA del entorno privado en cuestión. Ya sea Metropolitan *testbed* o UBP *testbed*.

De la misma manera, se trato al tráfico con destino externo a nuestro entorno (IP destino pública/externa) para reflejarlo en las tablas y gráficos mencionados en el párrafo anterior. Éste, claramente en respuesta a las solicitudes desde Internet descriptas anteriormente. Y por otro lado las IP de destino privadas/internas de paquetes encaminados dentro de nuestra LAN por todas las entidades SIP. Luego de este estudio y considerando la arquitectura de los *testbeds* implementados, podemos concluir que prevalece ampliamente el tráfico privado (IP origen/destino internas). Ésto se debe al comportamiento inherente de un entorno VoIP SIP, donde existe un intercambio constante entre el *registrar* y los UA, más aún en las implementaciones con Asterisk (Metropolitan *testbed*) donde todo el tráfico de señalización SIP debe transitar a través de este. Habrá que recordar que a modo de *keepalive* los UAs se vuelven a registrar cada cierto *timeout* (tiempo muerto). Cabe aclarar, que al capturar con una herramienta ejecutada en ciertos dispositivos del dominio, más específicamente en el SIP *proxy/registrar* estamos captando todo el tráfico de señalización y encaminamiento SIP de todos los componentes del dominio. Se puede señalar que para ambos *testbed* se aprecia el tráfico analizado según IP de origen y como refleja la Figura 3.54 tenemos un porcentaje considerable, del 28,21 % de tráfico público/externo para Metropolitan *testbed*, y como se observa en la Figura 3.58 un 10,56 % para UBP *testbed*, lo cual explica el origen de los ataques de enumeración y de *password crack* sufridos por Asterisk.

De manera complementaria se clasificó el tráfico con IP de origen o IP destino pública con su correspondencia por zona geográfica, más precisamente a que país está asociado ese rango de IP pública. Esto fue consultado a través del servicio web de geolocalización IP *iplocation*¹⁷. Y también se tuvo de referencia la distribución de direcciones IP supervisada por Autoridad de Asignación de Números de Internet (IANA), y la manera en que las direcciones IP de los países son emitidas por sus agencias gobierno. Ciertamente, la actual emisión de direcciones IP es manejada por diferentes agencias de RIRs en diferentes partes del mundo, como se muestra en la Figura 3.53: [Hundley, 2009]

¹⁷<https://www.iplocation.net/>

Some of the IP address allocations managed by RIR (Regional Internet Registry)

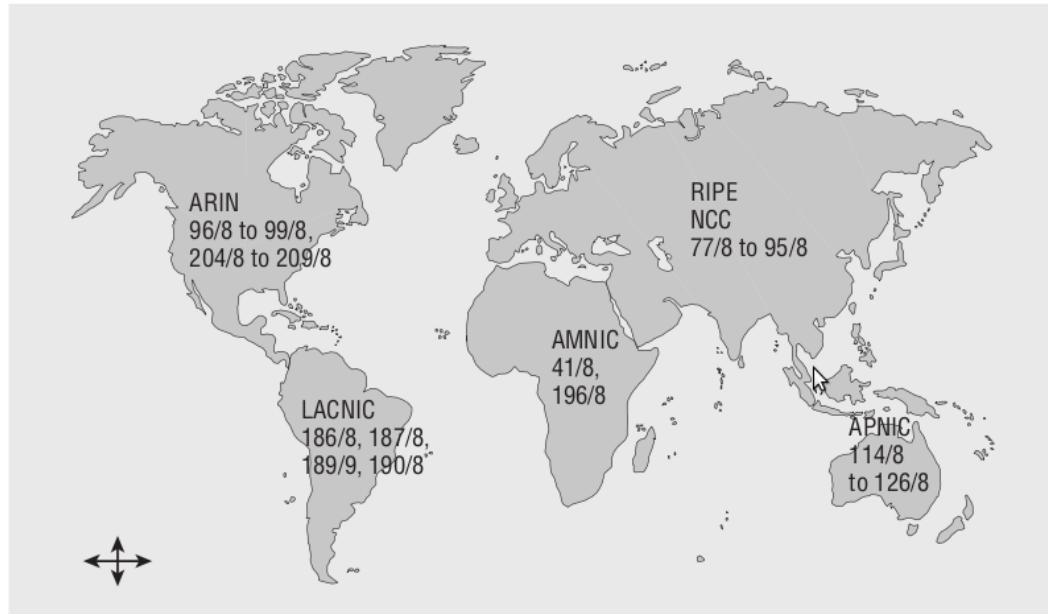


Figura 3.53: Los RIR alocan direcciones IP públicas. [Hundley, 2009]

Tal puede verse en los gráficos de las Figuras 3.55, 3.57, 3.59 y 3.61, que integra todo el tráfico con origen o destino a direcciones IP públicas podemos destacar que Argentina y Alemania fueron los países de mayor protagonismo. Clarificamos que todo el tráfico recibido de una nacionalidad diferente a la Argentina puede considerarse sospechoso. A su vez será menester puntualizar que si bien es 100 % objetivo el origen del tráfico, éste podría estar siendo encaminado a través de *proxies* intermedios. Al utilizar esta técnica de encaminamiento es posible enmascarar el origen real del tráfico. En tercer lugar está Estados Unidos, habiendo generado un tráfico también considerable. Y por último Corea, Irlanda y Polonia con un porcentaje mucho menor.

Tabla 3.10: Estadísticas por dirección IP de origen (Metropolitan testbed).

| IP Statistics/Source and Dest IP Addresses - Metropolitan_TestBed | | | | | |
|---|-----------------|-----------|---------|------------------|------------------|
| Source IP Addresses | Count (packets) | Rate (ms) | Percent | Burst rate (p/s) | private/public |
| TOTAL | 4154958 | 0,0002 | 100% | 2,12 | Private & Public |
| 192.168.0.101 | 1034040 | 0 | 24,89% | 0,96 | Private |
| 192.168.0.106 | 864007 | 0 | 20,79% | 0,4 | Private |
| 217.9.36.145 | 780718 | 0 | 18,79% | 1,88 | Public / DEU |
| 192.168.0.104 | 674785 | 0 | 16,24% | 0,32 | Private |
| 192.168.0.105 | 207149 | 0 | 4,99% | 0,04 | Private |
| 192.168.0.103 | 201873 | 0 | 4,86% | 0,23 | Private |
| 174.35.23.242 | 98446 | 0 | 2,37% | 2 | Public / USA |
| 66.114.54.12 | 90456 | 0 | 2,18% | 2,12 | Public / USA |
| 174.35.35.18 | 88596 | 0 | 2,13% | 1,02 | Public / USA |
| 174.35.35.30 | 43196 | 0 | 1,04% | 2,02 | Public / USA |
| 174.35.35.38 | 32722 | 0 | 0,79% | 2,04 | Public / USA |
| 66.114.54.35 | 22020 | 0 | 0,53% | 2,1 | Public / USA |
| 208.80.154.225 | 12250 | 0 | 0,29% | 2,08 | Public / USA |
| 217.9.54.27 | 2940 | 0 | 0,07% | 0,84 | Public / DEU |
| 174.35.35.23 | 1004 | 0 | 0,02% | 1,52 | Public / USA |
| 192.168.0.1 | 756 | 0 | 0,02% | 1,26 | Private |

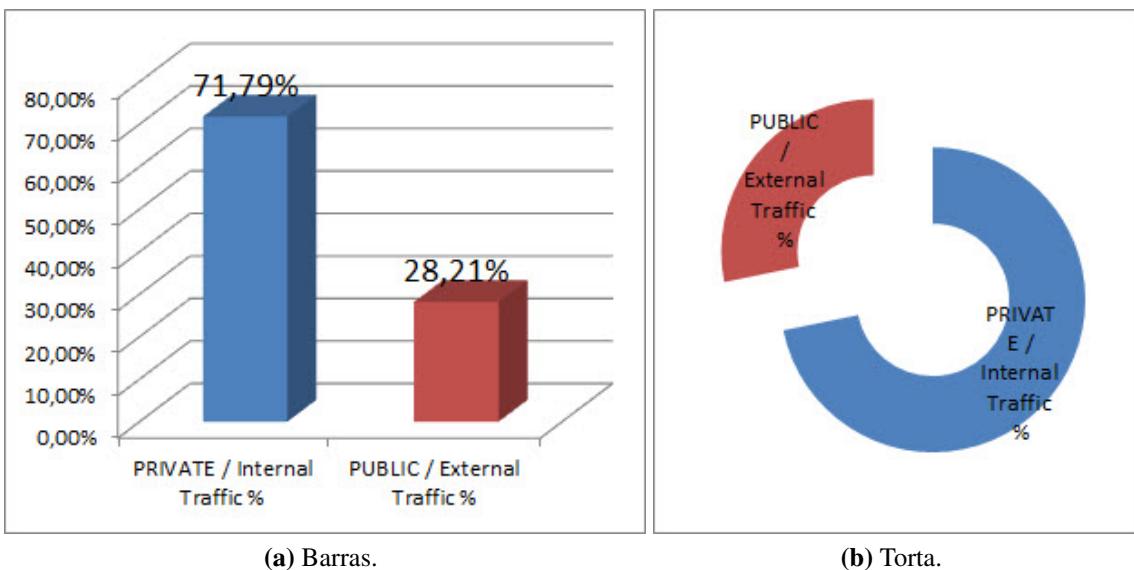


Figura 3.54: Estadísticas por dirección IP de origen pública/privada (Metropolitan *testbed*).

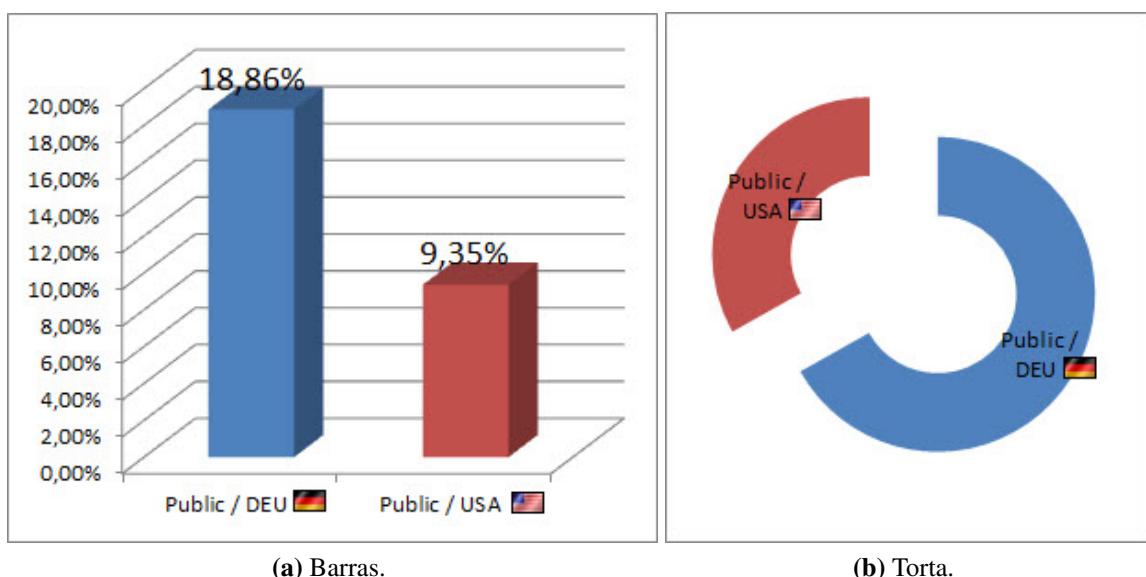


Figura 3.55: Estadísticas por dirección IP de origen por país (Metropolitan *testbed*).

Tabla 3.11: Estadísticas por dirección IP de destino (Metropolitan *testbed*).

| IP Statistics/Source and Dest IP Addresses - Metropolitan_TestBed | | | | | |
|---|-----------------|-----------|---------|------------------|--|
| Destination IP Addresses | Count (packets) | Rate (ms) | Percent | Burst rate (p/s) | private/public |
| TOTAL | 4154958 | 0,0002 | 100% | 2,12 | Private & Public |
| 192.168.0.104 | 1382893 | 0,0001 | 33,28% | 0,79 | Private |
| 192.168.0.105 | 1317329 | 0,0001 | 31,70% | 2,12 | Private |
| 192.168.0.106 | 729073 | 0 | 17,55% | 0,32 | Private |
| 192.168.0.103 | 260077 | 0 | 6,26% | 0,96 | Private |
| 192.168.0.101 | 255938 | 0 | 6,16% | 0,23 | Private |
| 217.9.36.145 | 197417 | 0 | 4,75% | 0,02 | Public / DEU  |
| 103.6.174.75 | 3080 | 0 | 0,07% | 0,02 | Public / KOR  |
| 186.137.221.179 | 2511 | 0 | 0,06% | 0,36 | Public / ARG  |
| 23.2.17.170 | 2365 | 0 | 0,06% | 0,01 | Public / USA  |
| 31.13.85.16 | 572 | 0 | 0,01% | 0,01 | Public / IRL  |
| 174.35.23.242 | 528 | 0 | 0,01% | 0,02 | Public / USA  |
| 66.114.54.12 | 450 | 0 | 0,01% | 0,01 | Public / USA  |
| 174.35.35.18 | 439 | 0 | 0,01% | 0,01 | Public / USA  |
| 174.35.35.30 | 263 | 0 | 0,01% | 0,01 | Public / USA  |
| 200.89.148.74 | 220 | 0 | 0,01% | 0,01 | Public / ARG  |
| 200.89.148.104 | 209 | 0 | 0,01% | 0,01 | Public / ARG  |
| 207.171.163.226 | 187 | 0 | 0,00% | 0,01 | Public / USA  |
| 174.35.35.38 | 175 | 0 | 0,00% | 0,01 | Public / USA  |
| 209.170.75.235 | 165 | 0 | 0,00% | 0,01 | Public / SWE  |
| 72.21.91.9 | 143 | 0 | 0,00% | 0,01 | Public / USA  |
| 66.114.54.35 | 110 | 0 | 0,00% | 0,01 | Public / USA  |
| 200.89.148.98 | 99 | 0 | 0,00% | 0,01 | Public / ARG  |
| 184.51.200.26 | 99 | 0 | 0,00% | 0,01 | Public / ARG  |
| 173.194.42.4 | 88 | 0 | 0,00% | 0,01 | Public / USA  |
| 23.2.17.138 | 66 | 0 | 0,00% | 0,01 | Public / USA  |
| 23.12.167.41 | 66 | 0 | 0,00% | 0,01 | Public / USA  |
| 208.80.154.225 | 66 | 0 | 0,00% | 0,01 | Public / USA  |
| 64.94.107.48 | 55 | 0 | 0,00% | 0,01 | Public / USA  |
| 200.89.148.97 | 44 | 0 | 0,00% | 0,01 | Public / ARG  |
| 200.89.148.90 | 33 | 0 | 0,00% | 0,01 | Public / ARG  |
| 173.252.73.52 | 33 | 0 | 0,00% | 0,01 | Public / USA  |
| 216.205.25.33 | 22 | 0 | 0,00% | 0,01 | Public / USA  |
| 200.89.148.89 | 22 | 0 | 0,00% | 0,01 | Public / ARG  |
| 200.89.148.105 | 22 | 0 | 0,00% | 0,01 | Public / ARG  |
| 74.125.137.121 | 11 | 0 | 0,00% | 0,01 | Public / USA  |
| 72.21.194.168 | 11 | 0 | 0,00% | 0,01 | Public / USA  |
| 64.94.107.44 | 11 | 0 | 0,00% | 0,01 | Public / USA  |
| 64.12.106.9 | 11 | 0 | 0,00% | 0,01 | Public / USA  |
| 200.42.33.115 | 11 | 0 | 0,00% | 0,01 | Public / ARG  |
| 176.32.100.69 | 11 | 0 | 0,00% | 0,01 | Public / USA  |
| 174.35.35.23 | 11 | 0 | 0,00% | 0,01 | Public / USA  |
| 118.214.160.225 | 11 | 0 | 0,00% | 0,01 | Public / SGP  |
| 118.214.160.217 | 11 | 0 | 0,00% | 0,01 | Public / SGP  |

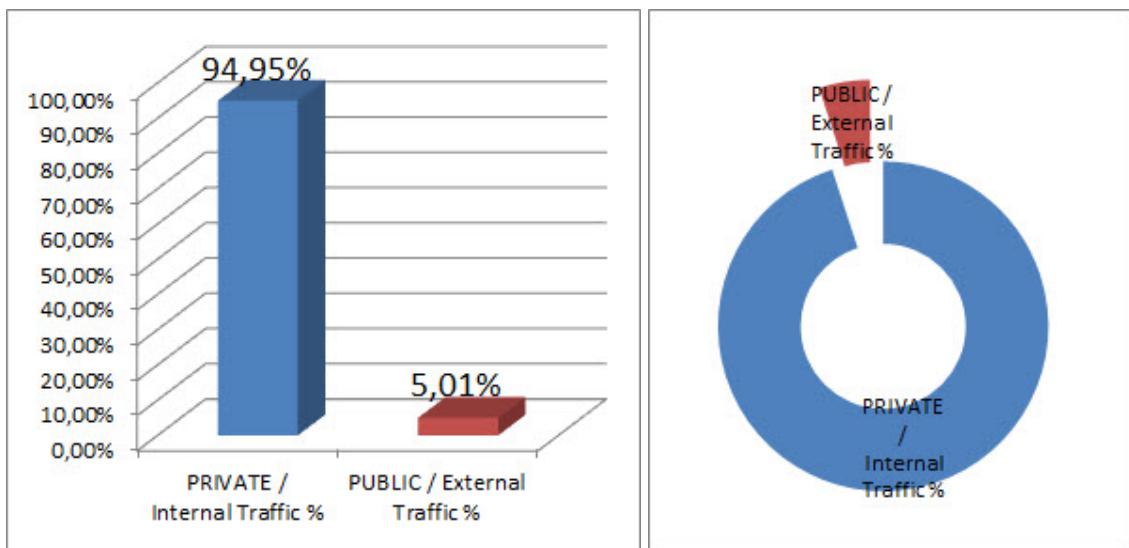


Figura 3.56: Estadísticas por dirección IP de destino pública/privada (Metropolitan testbed).

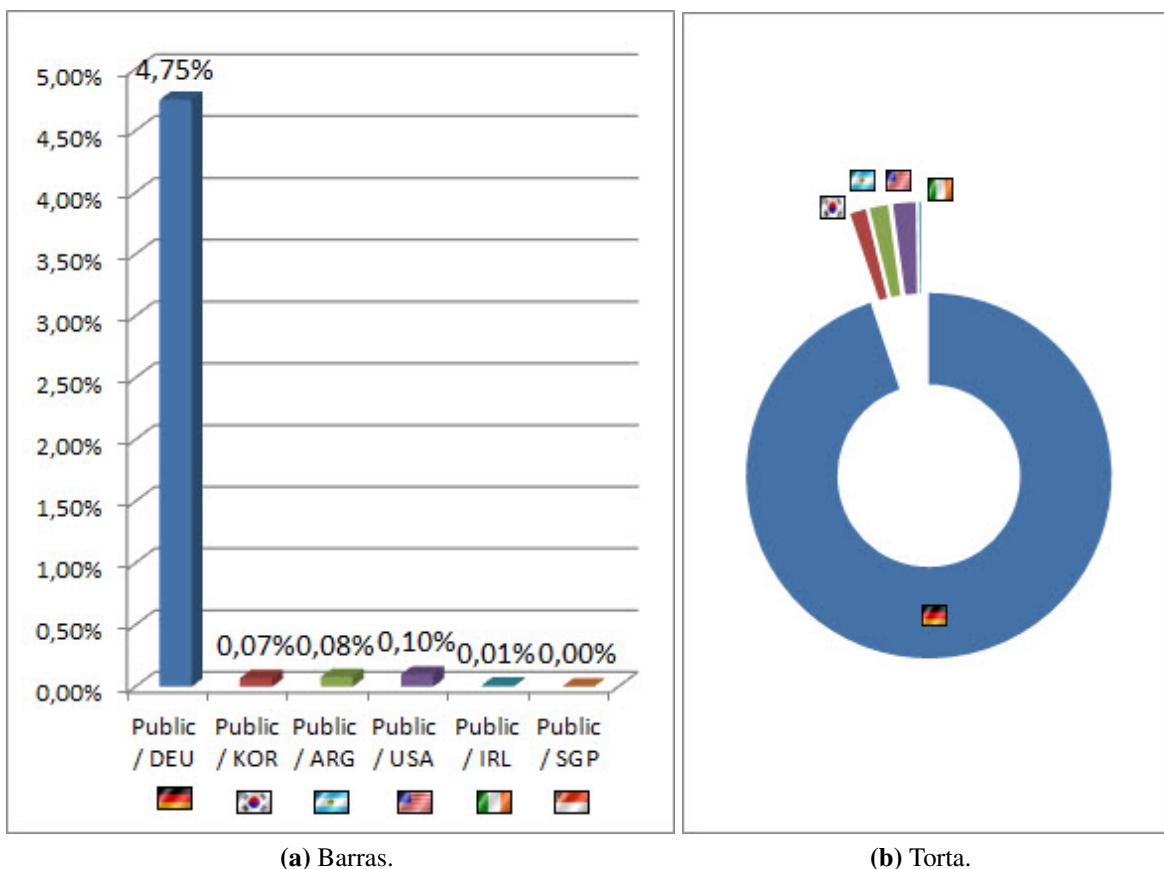


Figura 3.57: Estadísticas por dirección IP de destino por país (Metropolitan testbed).

Tabla 3.12: Estadísticas por dirección IP de origen (UBP *testbed*).

| IP Statistics/Source and Dest IP Addresses - UBP_TestBed | | | | | |
|--|-----------------|-----------|---------|------------------|--|
| Source IP Addresses | Count (packets) | Rate (ms) | Percent | Burst rate (p/s) | private/public |
| TOTAL | 3920963 | 0,0002 | 100% | 10,35 | Private & Public |
| 10.10.0.240 | 2191723 | 0,0001 | 55,90% | 5,39 | Private |
| 10.10.0.250 | 1125352 | 0,0001 | 28,70% | 0,62 | Private |
| 10.10.0.230 | 80815 | 0 | 2,06% | 0,02 | Private |
| 217.9.36.145 | 77669 | 0 | 1,98% | 0,04 | Public / DEU  |
| 186.109.7.117 | 68948 | 0 | 1,76% | 3,9 | Public / ARG  |
| 201.252.59.95 | 61149 | 0 | 1,56% | 4,97 | Public / ARG  |
| 10.10.0.254 | 56804 | 0 | 1,45% | 0,44 | Private |
| 10.10.0.30 | 44184 | 0 | 1,13% | 0,3 | Private |
| 186.108.147.99 | 25196 | 0 | 0,64% | 4,56 | Public / ARG  |
| 186.109.2.218 | 23765 | 0 | 0,61% | 4,13 | Public / ARG  |
| 186.108.144.154 | 22641 | 0 | 0,58% | 3,47 | Public / ARG  |
| 186.153.255.252 | 20278 | 0 | 0,52% | 4,17 | Public / ARG  |
| 181.95.65.153 | 16140 | 0 | 0,41% | 4,54 | Public / ARG  |
| 201.252.49.31 | 10600 | 0 | 0,27% | 3,92 | Public / ARG  |
| 186.108.147.78 | 9970 | 0 | 0,25% | 3,6 | Public / ARG  |
| 201.252.23.85 | 8752 | 0 | 0,22% | 2,73 | Public / ARG  |
| 181.95.71.10 | 8279 | 0 | 0,21% | 3,77 | Public / ARG  |
| 186.109.1.172 | 6281 | 0 | 0,16% | 4,51 | Public / ARG  |
| 181.95.70.43 | 5848 | 0 | 0,15% | 3,19 | Public / ARG  |
| 186.153.246.22 | 5440 | 0 | 0,14% | 2,78 | Public / ARG  |
| 200.45.97.75 | 5391 | 0 | 0,14% | 3,95 | Public / ARG  |
| 181.95.66.104 | 5267 | 0 | 0,13% | 3,49 | Public / ARG  |
| 10.10.0.50 | 5042 | 0 | 0,13% | 0,02 | Private |
| 186.153.252.167 | 4125 | 0 | 0,11% | 2,31 | Public / ARG  |
| 186.109.5.165 | 4085 | 0 | 0,10% | 4,08 | Public / ARG  |
| 186.109.0.222 | 4082 | 0 | 0,10% | 1,94 | Public / ARG  |
| 201.252.49.242 | 4040 | 0 | 0,10% | 3,88 | Public / ARG  |
| 181.95.70.246 | 3440 | 0 | 0,09% | 3,27 | Public / ARG  |
| 186.108.145.167 | 3348 | 0 | 0,09% | 4,96 | Public / ARG  |
| 190.136.104.149 | 2671 | 0 | 0,07% | 2,78 | Public / ARG  |
| 10.10.0.40 | 2027 | 0 | 0,05% | 0,02 | Private |
| 186.153.236.142 | 1987 | 0 | 0,05% | 2,31 | Public / ARG  |
| 186.153.249.14 | 1621 | 0 | 0,04% | 2,4 | Public / ARG  |
| 186.153.253.58 | 1082 | 0 | 0,03% | 2,49 | Public / ARG  |
| 186.153.250.47 | 837 | 0 | 0,02% | 2,05 | Public / ARG  |
| 190.136.105.76 | 755 | 0 | 0,02% | 1,67 | Public / ARG  |
| 186.153.232.13 | 384 | 0 | 0,01% | 0,33 | Public / ARG  |
| 190.226.27.93 | 181 | 0 | 0,00% | 0,14 | Public / ARG  |
| 201.252.51.65 | 168 | 0 | 0,00% | 0,14 | Public / ARG  |
| 186.153.237.34 | 126 | 0 | 0,00% | 0,14 | Public / ARG  |
| 190.226.25.187 | 84 | 0 | 0,00% | 0,14 | Public / ARG  |
| 217.9.54.27 | 82 | 0 | 0,00% | 0,02 | Public / DEU  |
| 181.164.196.114 | 48 | 0 | 0,00% | 0,06 | Public / ARG  |
| 190.30.11.231 | 42 | 0 | 0,00% | 0,14 | Public / ARG  |
| 181.95.66.90 | 42 | 0 | 0,00% | 0,14 | Public / ARG  |
| 10.10.0.3 | 34 | 0 | 0,00% | 0,06 | Private |
| 201.253.206.207 | 28 | 0 | 0,00% | 0,14 | Public / ARG  |
| 181.95.73.190 | 28 | 0 | 0,00% | 0,14 | Public / ARG  |
| 181.95.69.8 | 28 | 0 | 0,00% | 0,14 | Public / ARG  |
| 181.164.215.193 | 15 | 0 | 0,00% | 0,01 | Public / ARG  |
| 201.252.45.134 | 14 | 0 | 0,00% | 0,14 | Public / ARG  |
| 10.10.0.6 | 10 | 0 | 0,00% | 0,04 | Private |
| 10.10.0.2 | 10 | 0 | 0,00% | 0,02 | Private |
| 10.10.0.9 | 5 | 0 | 0,00% | 0,04 | Private |

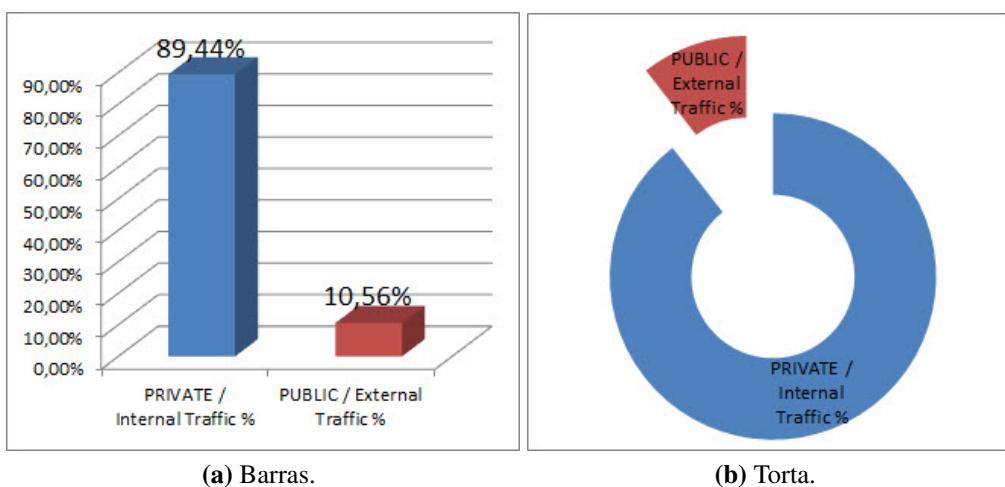


Figura 3.58: Estadísticas por dirección IP de origen pública/privada (UBP testbed).

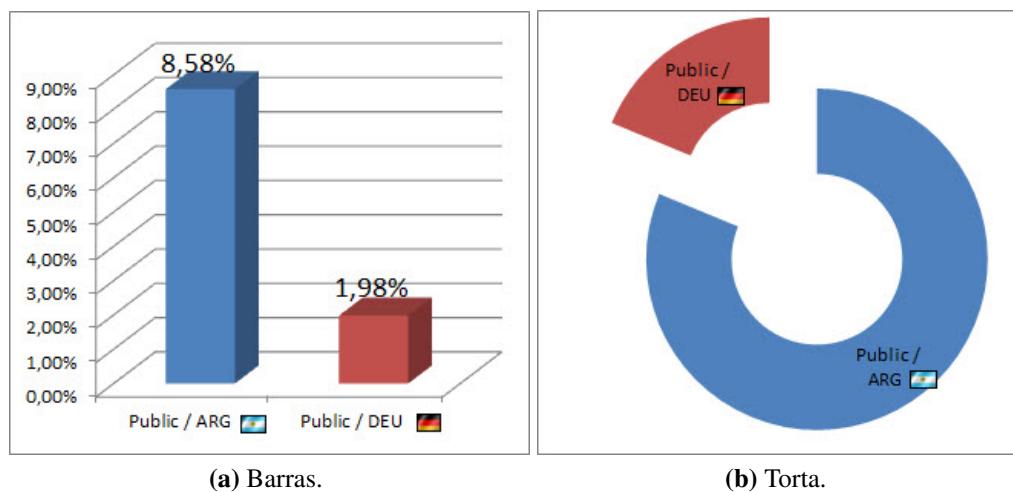


Figura 3.59: Estadísticas por dirección IP de origen por país (UBP testbed).

Tabla 3.13: Estadísticas por dirección IP de destino (UBP *testbed*).

| IP Statistics/Source and Dest IP Addresses - UBP_TestBed | | | | | |
|--|-----------------|-----------|---------|------------------|------------------|
| Destination IP Addresses | Count (packets) | Rate (ms) | Percent | Burst rate (p/s) | private/public |
| TOTAL | 3920963 | 0,0002 | 100% | 10,35 | Private & Public |
| 10.10.0.250 | 1743163 | 0,0001 | 44,46% | 0,81 | Private |
| 10.10.0.240 | 1570674 | 0,0001 | 40,06% | 5,03 | Private |
| 217.9.36.145 | 80815 | 0 | 2,06% | 0,02 | Public / DEU |
| 10.10.0.230 | 77751 | 0 | 1,98% | 0,04 | Private |
| 186.109.7.117 | 68972 | 0 | 1,76% | 3,91 | Public / ARG |
| 201.252.59.95 | 61389 | 0 | 1,57% | 4,91 | Public / ARG |
| 10.10.0.254 | 59504 | 0 | 1,52% | 1,33 | Private |
| 10.10.0.30 | 43201 | 0 | 1,10% | 0,2 | Private |
| 186.108.147.99 | 25305 | 0 | 0,65% | 4,55 | Public / ARG |
| 186.109.2.218 | 23801 | 0 | 0,61% | 4,12 | Public / ARG |
| 186.108.144.154 | 22693 | 0 | 0,58% | 3,47 | Public / ARG |
| 186.153.255.252 | 20386 | 0 | 0,52% | 4,18 | Public / ARG |
| 181.95.65.153 | 16252 | 0 | 0,41% | 4,53 | Public / ARG |
| 201.252.49.31 | 10630 | 0 | 0,27% | 3,93 | Public / ARG |
| 186.108.147.78 | 9985 | 0 | 0,25% | 3,6 | Public / ARG |
| 201.252.23.85 | 8750 | 0 | 0,22% | 2,72 | Public / ARG |
| 181.95.71.10 | 8338 | 0 | 0,21% | 3,77 | Public / ARG |
| 186.109.1.172 | 6296 | 0 | 0,16% | 4,53 | Public / ARG |
| 181.95.70.43 | 5907 | 0 | 0,15% | 3,1 | Public / ARG |
| 186.153.246.22 | 5459 | 0 | 0,14% | 2,77 | Public / ARG |
| 200.45.97.75 | 5375 | 0 | 0,14% | 3,78 | Public / ARG |
| 181.95.66.104 | 5301 | 0 | 0,14% | 3,5 | Public / ARG |
| 10.10.0.50 | 4997 | 0 | 0,13% | 0,04 | Private |
| 186.153.252.167 | 4131 | 0 | 0,11% | 2,31 | Public / ARG |
| 186.109.5.165 | 4091 | 0 | 0,10% | 4,11 | Public / ARG |
| 186.109.0.222 | 4087 | 0 | 0,10% | 1,96 | Public / ARG |
| 201.252.49.242 | 4036 | 0 | 0,10% | 3,88 | Public / ARG |
| 181.95.70.246 | 3460 | 0 | 0,09% | 3,34 | Public / ARG |
| 186.108.145.167 | 3349 | 0 | 0,09% | 4,96 | Public / ARG |
| 190.136.104.149 | 2689 | 0 | 0,07% | 2,8 | Public / ARG |
| 186.153.236.142 | 1988 | 0 | 0,05% | 2,32 | Public / ARG |
| 10.10.0.40 | 1921 | 0 | 0,05% | 0,03 | Private |
| 186.153.249.14 | 1627 | 0 | 0,04% | 2,4 | Public / ARG |
| 186.153.253.58 | 1091 | 0 | 0,03% | 2,5 | Public / ARG |
| 186.153.250.47 | 845 | 0 | 0,02% | 2,14 | Public / ARG |
| 190.136.105.76 | 755 | 0 | 0,02% | 1,65 | Public / ARG |
| 186.153.232.13 | 492 | 0 | 0,01% | 0,33 | Public / ARG |
| 181.164.196.114 | 251 | 0 | 0,01% | 0,04 | Public / ARG |
| 190.226.27.93 | 180 | 0 | 0,00% | 0,14 | Public / ARG |
| 201.252.51.65 | 168 | 0 | 0,00% | 0,14 | Public / ARG |
| 186.153.237.34 | 126 | 0 | 0,00% | 0,14 | Public / ARG |
| 10.0.0.240 | 108 | 0 | 0,00% | 0,03 | Private |
| 82.98.86.174 | 90 | 0 | 0,00% | 0,09 | Public / DEU |
| 190.226.25.187 | 84 | 0 | 0,00% | 0,14 | Public / ARG |
| 1.1.1.1 | 62 | 0 | 0,00% | 0,03 | Private |
| 190.30.11.231 | 42 | 0 | 0,00% | 0,14 | Public / ARG |
| 181.95.66.90 | 42 | 0 | 0,00% | 0,14 | Public / ARG |
| 201.253.206.207 | 28 | 0 | 0,00% | 0,14 | Public / ARG |
| 181.95.73.190 | 28 | 0 | 0,00% | 0,14 | Public / ARG |
| 181.95.69.8 | 28 | 0 | 0,00% | 0,14 | Public / ARG |
| 10.0.0.65 | 27 | 0 | 0,00% | 0,03 | Private |
| 10.0.0.182 | 27 | 0 | 0,00% | 0,03 | Private |
| 181.164.215.193 | 26 | 0 | 0,00% | 0,02 | Public / ARG |
| 10.10.0.3 | 26 | 0 | 0,00% | 0,04 | Private |
| 181.9.123.152 | 23 | 0 | 0,00% | 0,02 | Public / ARG |
| 192.0.43.10 | 18 | 0 | 0,00% | 0,01 | Private |
| 181.6.191.136 | 15 | 0 | 0,00% | 0,02 | Public / ARG |
| 201.252.45.134 | 14 | 0 | 0,00% | 0,14 | Public / ARG |
| 10.10.0.6 | 10 | 0 | 0,00% | 0,04 | Private |
| 181.95.66.94 | 9 | 0 | 0,00% | 0,01 | Public / ARG |
| 20.20.0.50 | 8 | 0 | 0,00% | 0,02 | Private |
| 181.92.171.68 | 8 | 0 | 0,00% | 0,01 | Public / ARG |
| 10.10.0.2 | 6 | 0 | 0,00% | 0,02 | Private |
| 181.9.114.163 | 2 | 0 | 0,00% | 0,01 | Public / ARG |
| 10.10.0.9 | 1 | 0 | 0,00% | 0,01 | Private |

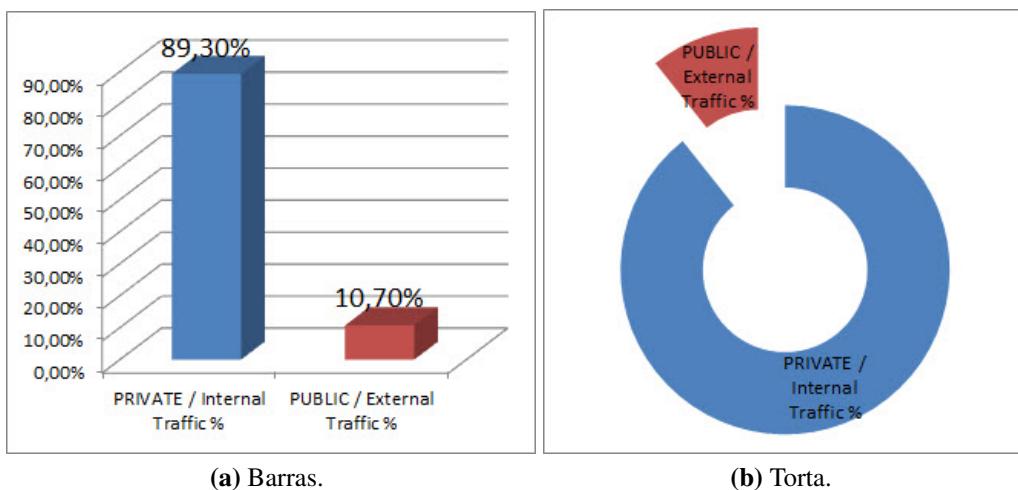


Figura 3.60: Estadísticas por dirección IP de destino pública/privada (UBP testbed).

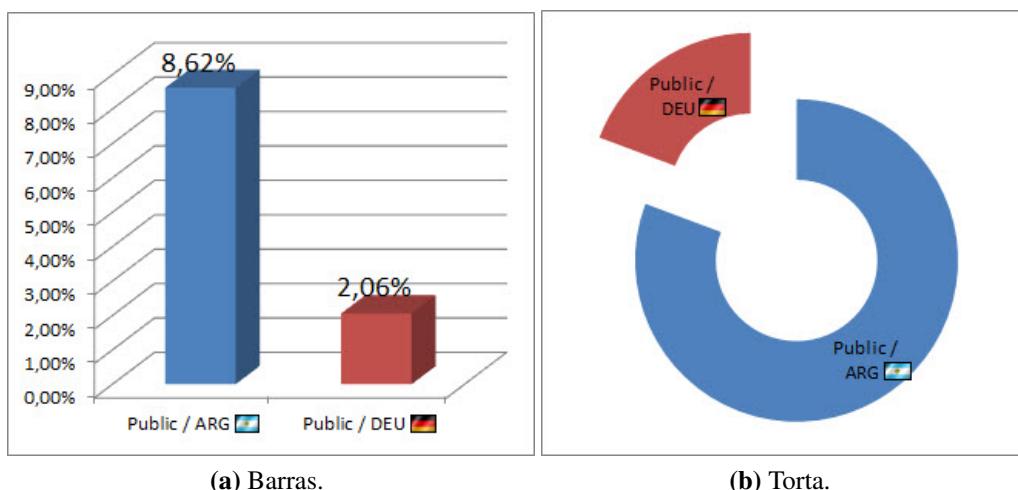


Figura 3.61: Estadísticas por dirección IP de destino por país (UBP testbed).

¹⁸Geolocalización de las direcciones IP a través de <http://www.iplocation.net/>

3.6.9. Resumen de resultados

Sobre el estudio descriptivo estadístico de las características de Artemisa *honeypot* en entornos reales en mesa de prueba:

- Los registros de ataques colectados muestran la existencia de ataques reales a los que se enfrentan a diario los servidores del entorno VoIP. Se hace evidente la necesidad de contar con herramientas que provean facilidades para llevar adelante el análisis de los *logs* de los servidores buscando una detección temprana de ataques.
- Al exponer al *honeypot* a Internet y recibir un sin numero de intentos de comunicaciones SIP se descubrió predominancia de mensajes *500 Server Internal Error* en el *environment* de prueba. Sabemos que estos mensajes de respuesta están íntimamente relacionados a que el servidor, en nuestro caso Artemisa, se enfrentó con una condición inesperada que impidió al mismo completar la solicitud SIP. Se notó que esto está asociado a que Artemisa solo tiene la capacidad de responder a ciertos SIP *messages* considerando que el mismo actua como un UA y no tiene la capacidad de un *proxy SIP* o *registrar*. A pesar de esto llegamos a la conclusión de que como herramienta de estudio sobre ataques en la red pública, Artemisa, o mejor aún, una segunda instancia de Artemisa debe configurarse idealmente para que entregue como su *fingerprint*, el de un *proxy SIP* por ejemplo: Asterisk PBX 1.6.2.9-2+squeeze5, e instalarse a la escucha sobre el puerto estándar SIP (5060) de una IP pública para disuadir al atacante.
- A través de las capturas se pudo corroborar que los tipos de mensajes SIP que predominan en los *testbeds* son *401 unauthorized* por error en la autenticación contra Artemisa, asociados a intentos de registro fallidos. En su gran mayoría no esperados, por lo que podemos decir que fueron generados por un atacante.
- Se puede inferir directamente del análisis estadístico realizado en las dos camas de prueba que existe un muy elevado número de REGISTERS asociados a intentos de intrusión.
- En cuanto a la cantidad de llamadas VoIP SIP total (*registrars*) se validó un número importantísimo de llamadas rechazadas, lo que se ve directamente reflejado por las estadísticas del método REGISTER, como así también de los mensajes SIP *401 unauthorized* y *404 not found* presentes en gran número e involucrados en este proceso. Los cuales resultan de una gran cantidad de registros fallidos. Por su número exagerado y contrastando con los *logs* de Asterisk es que se acreditaron como intentos de registro mal intencionados.
- Cabe destacar que para ambos *testbed* se aprecia que el tráfico analizado según IP de origen, tenemos un porcentaje considerable, del 28,21 % de tafico público/externo, lo cual refleja los ataques de enumeración y de *password crack* sufridos por Asterisk.
- Integrando todo el tráfico con origen o destino a direcciones IP públicas se destaca que Argentina y Alemania fueron los países de mayor protagonismo. Todo el tráfico recibido de una nacionalidad diferente a la Argentina, puede considerarse sospechoso. A su vez será menester aclarar que si bien es 100 % objetivo el origen del tráfico, este podría estar siendo encaminado a través de *proxies* intermedios. Al utilizar esta técnica de enrutamiento es posible enmascarar el origen real del tráfico. En tercer lugar Estados Unidos habiendo generado un tráfico también considerable. Y por último Corea, Irlanda y Polonia con un porcentaje mucho menor.

- La efectividad que tuvo la plataforma en los entornos propuestos para detectar los ataques cubiertos por la misma. Principalmente, los ataques de inundación en un número mayor a 180.000 durante el período de exposición de seis meses. Sin descontar los 102 ataques de *ringing* y *scanning attemps*. Luego podríamos decir que los entornos no recibieron ataques de SPIT.
- Quedó en evidencia que si se expone al sistema Asterisk a la Internet pública, una de las cosas que casi con seguridad veremos es un *scan* para descubrir cuentas válidas. Esto quedó reflejado a través de nuestro caso de estudio, la instancia de Asterisk expuesta reportó en sus *logs* números muy significativos de eventos: *No matching peer found* y *Wrong password*. Este tipo de escaneos comienza chequeando varios nombres de usuarios comunes, y a continuación de estos se comienza a escanear por cuentas numeradas. Esto muestra que el atacante supone como patrón común dentro de los usuarios, usar el número de extensión de la PBX como nombre para la cuenta SIP.
- En base a la altísima cantidad de intentos de *password cracking* mediante *brute force* que *toll fraud* (fraude en tasas o tarifas telefónicas) es evidentemente el mayor riesgo para los sistemas telefónicos en términos del potencial para costos que podrían llevar a la ruina a casi cualquier compañía. No es desconocido para los estafadores acumular miles de dólares en llamadas robadas en el curso de unos pocos días. *Toll fraud* no es algo nuevo, incluso ha existido antes de VoIP. Sin embargo, la naturaleza permisiva de VoIP significa que es más fácil para los fraudulentos tomar ventaja de sistemas no seguros.
- Tomando como referencia todas las conclusiones antes presentadas sobre el estudio descriptivo estadístico de Artemisa, se hace notorio que permitiría proteger de forma más efectiva el dominio de comunicaciones SIP si se incrementa el alcance del *honeypot* para simular el rol de un servidor de registro SIP, es decir, agregar algunos métodos para que tenga funciones básicas de un *registrar* y así lograr desviar todos los ataques a los que la actual versión de Artemisa no se ve expuesto por su rol de *back-end user-agent*.

3.7. Desarrollos Funcionales sobre Artemisa

3.7.1. XML-RPC API

Como el lector habrá apreciado, en primera instancia se efectuó la implementación de Artemisa en dos *testbeds*. Por un lado en el laboratorio de redes de la UBP, y paralelamente, en otro punto de la ciudad de Córdoba. Se apuntó a lograr mejoras funcionales y concluir en configuraciones óptimas. Se persiguió desarrollar nuevas funcionalidades, y sus consecuentes ajustes al código como se verá en esta sección. Fue de carácter primordial mantener esta tendencia a lo largo del proyecto.

Se ha desarrollado una API implementando el módulo *SimpleXMLRPCServer* de la librería estándar de Python. Permitiendo de esta manera crear un servidor XML-RPC. El cual utiliza en su código fuente el protocolo de Internet, como también el módulo de sistema *socket* embebido en la mayoría de los OS.

El módulo *SimpleXMLRPCServer* provee un *framework* básico para servidores XML-RPC escritos en Python.

Antes de detallar los parámetros que espera Artemisa en su interfaz RPC resulta de interés presentar de manera resumida al protocolo XML-RPC.

El mismo es la especificación y un set de implementaciones que permite a software corriendo en diferentes OS y entornos, realizar una llamada de procedimiento (*call procedure*) a través de Internet.

Su procedimiento de llamada remoto (*RPC - Remote Procedure Call*) usa HTTP como protocolo de transporte y Lenguaje de Marcas Extensibles (XML) como lenguaje de codificación. XML-RPC esta diseñado para ser lo más simple posible, y a su vez permitir transmitir, procesar y devolver estructuras complejas.

Puede ser utilizado con lenguajes de programación como Perl, Java, C, C++, PHP, entre otros, en nuestro caso Python. Sus implementaciones están disponibles para Unix, Windows y Macintosh. [Kidd E., 2012]

3.7.1.1. Clase XML-RPC Server

```
class SimpleXMLRPCServer.SimpleXMLRPCServer  
(addr[, requestHandler[, logRequests[, allow_none[, encoding[,  
bind_and_activate]]]])
```

Esta clase permitirá crear una nueva instancia de servidor. También provee métodos para el registro de funciones que podrán ser llamadas por el protocolo XML-RPC. El parámetro *requestHandler* (solicitud de manejo) debería ser un factor para las instancias de *SimpleXMLServer*; resulta por defecto ser *SimpleXMLRPCRequestHandler*. Los parámetros dirección y *requestHandler* son pasados al *SocketServer.TCPServer*.

3.7.1.2. Clase XML-RPC Server para manejo de solicitudes

```
class SimpleXMLRPCServer.SimpleXMLRPCRequestHandler
```

Crea una nueva instancia de manejo de solicitudes. Este *request handler* soporta solicitudes de publicación *POST request* y modifica el *logging* de manera que el parámetro *logRequests* para el constructor de la clase *SimpleXMLRPCServer* sea honorado.
[Python Software Foundation, 2014].

Se presenta al pie de este párrafo el código desarrollado para este trabajo final de carrera. A fines prácticos y de simplificar su exposición se obvian algunas líneas de código relacionadas a importar la clase, librerías, asignación de variables, comienzo, terminación del XML-RPC server. Como así también obtención y seteo de algunas variables relacionadas al mismo. El código completo puede ser descargado desde el repositorio de Artemisa en SourceForge¹⁹.

3.7.1.3. Código método XmlServer()

```
def XmlServer(self):
    """
        Creation of the XML Server
    """
    global xml_serv
    xml_serv = SimpleXMLRPCServer((self.Local_IP,
int(self.Local_xml_port)),
requestHandler=RequestHandler)
    ### Function to allow Clients (XML-RPC connections) to access XML-RPC
service methos - API Interface
    xml_serv.register_introspection_functions()

    print '' # necessary to correct the console for a better output
visualization
    #logger.info('XML-RPC service running...')

    xml_serv.register_function(self.ModifyExt,'modify_extension')

    def RestartArtemisa():
        f= open('/dev/stdin','w')
        f.write('restart\n')
        #self.ArtemisaRestart()
    xml_serv.register_function(RestartArtemisa,'restart')

    ##### Run the XML-RPC_service main loop
    xml_serv.serve_forever()
```

3.7.1.4. Código método ModifyExt()

```
def ModifyExt(self,mod,ext,user,passwd):
    """
        Modify extensions from file extensions.conf
    """

    config = ConfigParser.ConfigParser()
    config.read(EXTENSIONS_FILE_PATH)

    config1 = ConfigParser.ConfigParser()
    config1.read(SERVERS_FILE_PATH)

    ext_aux = []

    # se usara la instancia configServer de la clase ConfigParser para
    # escribir en server.conf
    #configServer = ConfigParser.ConfigParser()
    #configServer.read(EXTENSIONS_FILE_PATH)

    if mod == 'add':
        if config.has_section(ext):
            logger.info ('Extension '+ext+' already exists in
extensions.conf')
            return 'Extension '+ext+' already exists'

    elif len(config.sections()) <= 7:

        config.add_section(ext)
        config.set(ext,'username','"' +user+'"')
        config.set(ext,'password',passwd)
```

¹⁹<https://sourceforge.net/p/artemisa/code/HEAD/tree/trunk/1.1/>

```

        with open(EXTENSIONS_FILE_PATH,'wb') as configfile:
            config.write(configfile)

        archi=open(EXTENSIONS_FILE_PATH,'r')
        lineS=archi.readlines()
        archi.close()

        archi=open(EXTENSIONS_FILE_PATH,'w')
        archi.write("# Artemisa - Extensions configuration file...")
        archi.writelines(lineS)
        archi.close()

        logger.info ('Extension '+ext+' added in extensions.conf')

        if config1.has_option('myproxy','exten'):
            b = False
            ext_aux = config1.get('myproxy','exten').split(',')
            for i in ext_aux:
                if ext == i:
                    logger.info ('Extension '+ext+' already exists in
servers.conf')
                    return 'Extension '+ext+' already exists.'

            aux_str = '%s' % ','.join(map(str, ext_aux))
            config1.set('myproxy','exten',str(aux_str)+','+ext)
            with open(SERVERS_FILE_PATH,'wb') as configfile:
                config1.write(configfile)

            archi=open(SERVERS_FILE_PATH,'r')
            lineS=archi.readlines()
            archi.close()

            archi=open(SERVERS_FILE_PATH,'w')
            archi.write("# Artemisa - Servers configuration
file...")
            archi.writelines(lineS)
            archi.close()

            logger.info ('Extension '+ext+' added in servers.conf')
            return 'Extension '+ext+' added'

        else:
            logger.info ('Max number of extensions registered. Run another
artemisa instance to register more extensions.')
            return 'Max number of extensions registered. Run another
artemisa instance to register more extensions.'
        elif mod == 'delete':
            if config.has_section(ext):
                config.remove_section(ext)
                with open(EXTENSIONS_FILE_PATH,'wb') as configfile:
                    config.write(configfile)

                archi=open(EXTENSIONS_FILE_PATH,'r')
                lineS=archi.readlines()
                archi.close()

                archi=open(EXTENSIONS_FILE_PATH,'w')
                archi.write("# Artemisa - Extensions configuration file...")
                archi.writelines(lineS)
                archi.close()

                logger.info ('Extension '+ext+' deleted in extensions.conf')

            else:
                logger.info ('Extension '+ext+' does not exists in
extensions.conf')

                if config1.has_option('myproxy','exten'):
                    b = False
                    ext_aux = config1.get('myproxy','exten').split(',')

```

```

#len_list_aux = len(ext_aux)
j=0
while j<len(ext_aux):
    if ext == ext_aux[j]:
        del(ext_aux[j])
        #len_list_aux = len(ext_aux)
        b = True
    else:
        j=j+1

aux_str = '%s' % ','.join(map(str, ext_aux))
config1.set('myproxy','exten',aux_str)
with open(SERVERS_FILE_PATH,'wb') as configfile:
    config1.write(configfile)

if not b :

    archi=open(SERVERS_FILE_PATH,'r')
    lineS=archi.readlines()
    archi.close()

    archi=open(SERVERS_FILE_PATH,'w')
    archi.write("# Artemisa - Extensions configuration
file")
    archi.writelines(lines)
    archi.close()

    logger.info ('Extension '+ext+' does not exists in
servers.conf')
    return 'Extension '+ext+' does not exists'

    archi=open(SERVERS_FILE_PATH,'r')
    lineS=archi.readlines()
    archi.close()

    archi=open(SERVERS_FILE_PATH,'w')
    archi.write("# Artemisa - Servers configuration file...")
    archi.writelines(lineS)
    archi.close()

    logger.info ('Extension '+ext+' deleted in servers.conf')
    return 'extension '+ext+' deleted'

else:
    logger.info ('Extension '+ext+' does not exists.')

else:
    logger.info ('wrong request please try with "add" or "delete"')
    return 'wrong request please try with "add" or "delete"'

del config

```

3.7.1.5. modify_extension() para agregar o quitar extensiones

```

XML-RPC_Object_instance.modify_extension('add','extension','username',
'password')

XML-RPC_Object_instance.modify_extension('delete','extension','username',
'password')

```

Se propone de modo genérico el nombre “XML-RPC_Object_instance” como nombre que referencia a una instancia de un constructor XML-RPC de cualquier lenguaje de programación como se detalló con anterioridad. Luego *modify_extension()* se corresponde con el método declarado en *core.py* dentro del programa Artemisa. Como se puede apreciar este método (o función) espera cuatro parámetros del tipo *string*. Que están íntimamente relacionados al registro de un nuevo usuario en las configuraciones de Asterisk. Recordar el formato del URI SIP *sip:user@registerIP:port*, o de forma más explícita la salida de pantalla de la consola de

Artemisa Extension "1006"sip:user@registrarIP:5060 para entender el porqué de los parámetros esperados por la API que se detallan debajo.

1. **add || delete**: es un parámetro estático y mandatorio que indicará el agregado o borrado y posterior registro de una nueva extensión en nuestro *honeypot*.
2. **extension**: será la extensión a registrar en la central.
3. **username**: usuario para autenticación en el *SIP-registrar*.
4. **password**: contraseña asociada a la autenticación del usuario en el *SIP-registrar*.

3.7.1.6. restart() para reiniciar el proceso de Artemisa

```
XML-RPC_Object_instance.restart()
```

restart() se corresponde con el método declarado en *core.py* dentro del programa Artemisa el cual nos permitirá reiniciar el proceso de Artemisa. De esta manera los cambios realizados con *XML-RPC_Object_instance.modify_extension()* tomarán efecto inmediato registrando o borrando las nuevas extensiones virtuales a nuestro señuelo.

3.7.2. XML-RPC ejemplo de implementación

A continuación se expondrá mediante un ejemplo la nueva funcionalidad programada en Artemisa. Este caso de estudio permitirá ver más claramente el alcance de esta nueva *feature* (característica).

Consideremos el siguiente escenario presentado en la Figura 3.62:

Como fue desarrollado en la sección 3.4 "Honeypots VoIP específicos y Artemisa *honeypot*", nuestra instancia de Artemisa habrá registrado las extensiones virtuales según los valores de sus archivos de configuración en los correspondientes *SIP-registrar/proxy servers*.

A partir de la nueva versión desarrollada, será posible para nuestro administrador realizar modificaciones vía la línea de comandos *CLI* de nuestro software. Como así también, permitir modificaciones de manera remota mediante un cliente XML-RPC adaptado a sus necesidades.

Considerando la capacidad de añadir y eliminar nuevas extensiones SIP vía *API server*. Se plantea un escenario donde el administrador de sistemas integre los demás componentes de su arquitectura de VoIP, para incrementar los niveles de seguridad del entorno de voz.

En este caso mediante el uso de un aplicativo conformado por ciertos Python scripts se integra y automatiza el registro de nuevas extensiones virtuales en tiempo real, que resultasen sospechosas en nuestra central VoIP (Asterisk).

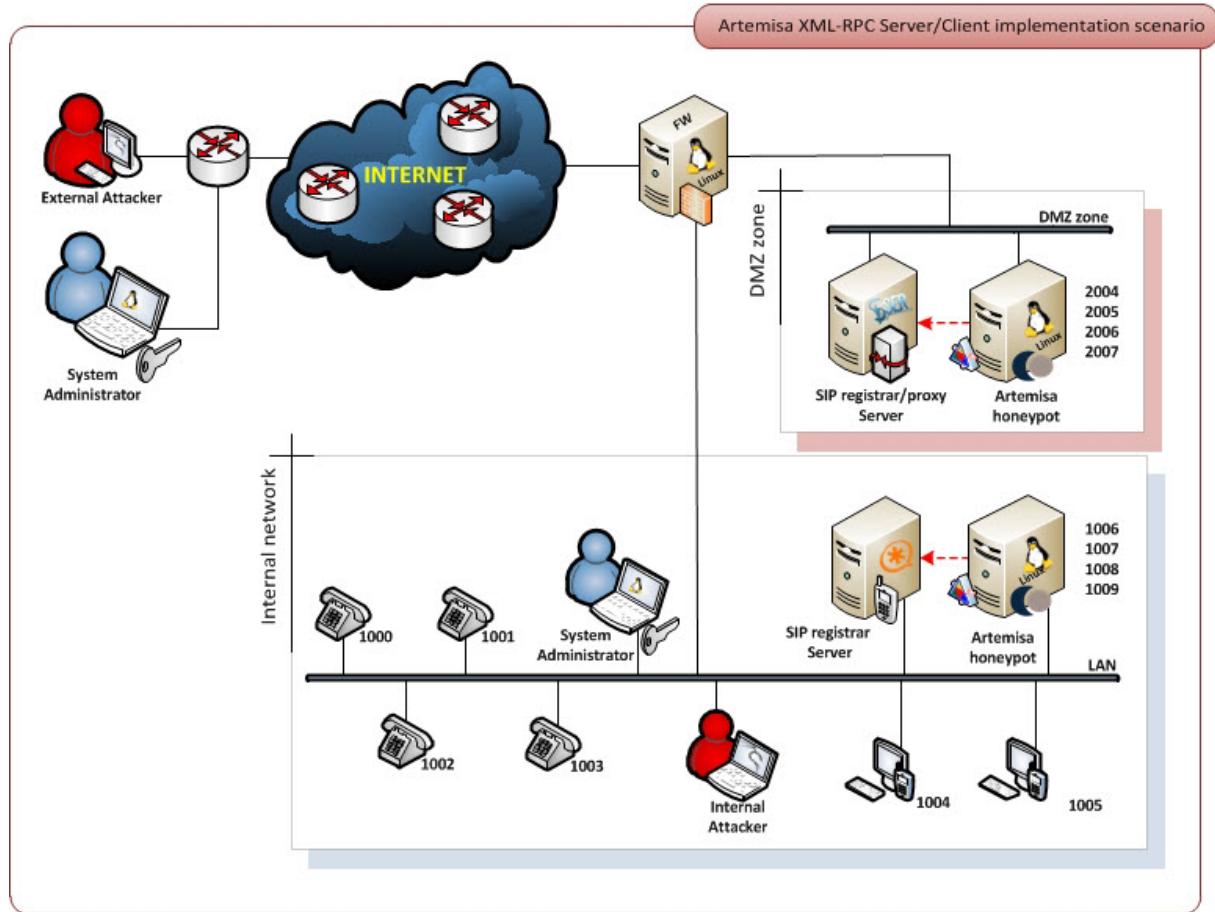


Figura 3.62: Diagrama topología XML-RPC cliente/servidor implementación.

Ejecución de Artemisa según el escenario planteado

```
Artemisa vrepvernumber Copyright (C) 2009-2013 Mohamed Nassar, Rodrigo do Carmo,
Pablo Masri, Mauro Villarroel and Exequiel Barirrero
```

```
This program comes with ABSOLUTELY NO WARRANTY; for details type 'show
warranty'.
This is free software, and you are welcome to redistribute it under certain
conditions; type 'show license' for details.
```

```
Type 'help' for help.
```

```
19:42:46.528 os_core_unix.c !pjlib 2.0.1 for POSIX initialized
19:42:46.528 sip_endpoint.c .Creating endpoint instance...
19:42:46.528     pjlib .select() I/O Queue created (0x9788760)
19:42:46.528 sip_endpoint.c .Module "mod-msg-print" registered
19:42:46.528 sip_transport. .Transport manager created.
19:42:46.528 pjsua_core.c .PJSUA state changed: NULL --> CREATED
SIP User-Agent listening on: 10.10.0.250:5061
XML-RPC service listening on: 10.10.0.250:9000
Behaviour mode: active
Starting extensions registration process...
2014-03-05 19:42:46,624 Extension 1006 registration sent. Status: 100 (In
Progress)
2014-03-05 19:42:46,634 Extension 1007 registration sent. Status: 100 (In
Progress)
2014-03-05 19:42:46,638 Extension 1008 registration sent. Status: 100 (In
Progress)
2014-03-05 19:42:46,646 Extension 1009 registration sent. Status: 100 (In
Progress)
2014-03-05 19:42:46,704 Extension "1006" <sip:1006@10.10.0.245:5060> registered,
status=200 (OK)
```

```

2014-03-05 19:42:46,706 Extension "1007" <sip:1007@10.10.0.245:5060> registered,
status=200 (OK)
2014-03-05 19:42:46,713 Extension "1008" <sip:1008@10.10.0.245:5060> registered,
status=200 (OK)
2014-03-05 19:42:46,716 Extension "1009" <sip:1009@10.10.0.245:5060> registered,
status=200 (OK)
delivery>

```

Mediante el diagrama de la Figura 3.63, se resume un posible caso de aplicación. Se observan en escena dos atacantes, interno y externo. Y por otro lado el administrador de red o sistemas quien tiene la capacidad de ejecutar comandos desde la CLI de Artemisa de manera local o mediante una conexión remota al propio *server*, vía SSH por ejemplo. O sacar provecho de la nueva funcionalidad pudiendo ejecutar modificaciones a las configuraciones de Artemisa antes mencionadas de forma remota con un XML-RPC *client*. Cabe aclarar que, idealmente nuestro XML-RPC *client* será ejecutado de forma dinámica y persistente en el mismo *SIP proxy/registrar* (Asterisk o SER), ya se demostrará el por qué en el desarrollo que continúa en esta sección.

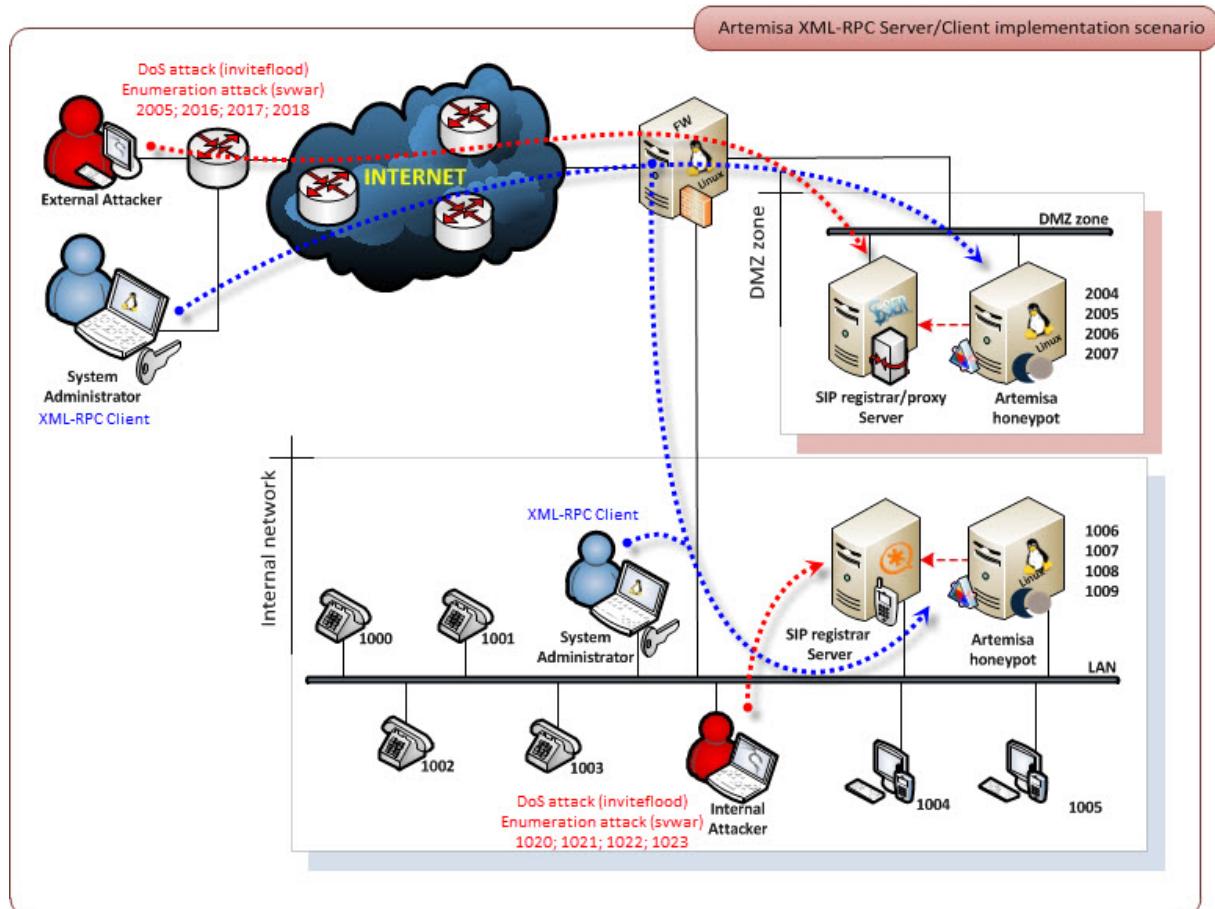


Figura 3.63: Diagrama topología XML-RPC cliente/servidor *enumeration/DoS*.

Para nuestro caso de estudio los usuarios maliciosos detrás de un ataque de enumeración de extensiones y/o de un ataque de denegación de servicio. Cabe aclarar que los mismos son complementarios, debido que en base a los resultados de la enumeración se podrá generar un ataque de DoS más efectivo.

La enumeración de extensiones puede favorecer al atacante permitiéndole descubrir extensiones válidas en un sistema VoIP que más tarde podrá guiar a un (*brute force attack*) sobre las

cuentas SIP. El método de enumeración de extensiones trabaja mediante procesamiento de los errores devueltos a los (*SIP request methods*) como REGISTER, OPTIONS e INVITE.

Mientras que, un ataque de DoS en servicios VoIP puede lograr su cometido causando un daño intencional a la red y a la disponibilidad de sistemas VoIP. Este ataque puede ocurrir en dos niveles, ataques DoS estándar y ataques DoS específicos para VoIP. Generalmente enviaríamos altísimas cargas de tráfico para inundar la red y consumir todos sus recursos o un protocolo específico para saturarlo con un gran número de peticiones.

Se tomarán como referencia la herramientas Inviteflood & SVWAR incluidas en *Linux Backtrack R2 distribution*.

- SVWAR: es una herramienta pertenecientes a la *suite* SIPVicious que permite enumerar extensiones mediante la utilización de un rango de extensiones o usando un archivo de diccionario. Este aplicativo soporta los tres métodos de enumeración posibles como fueron detallados previamente, el método por defecto para la enumeración es REGISTER.

Sintaxis de utilización básica:

```
root@bt:/pentest/voip/sipvicious# ./svwar.py
Usage: svwar.py [options] target
examples:
svwar.py -e100-999 10.0.0.1
svwar.py -d dictionary.txt 10.0.0.2
```

- Inviteflood: esta en una herramienta que puede ser utilizada para inundar un objetivo con peticiones SIP INVITE. También se implementa para alcanzar SIP *gateways/proxies* y teléfonos SIP.

Inviteflood cli

```
root@bt:/pentest/voip/inviteflood# ./inviteflood
inviteflood - Version 2.0
                June 09, 2006
Usage:
Mandatory -
    interface (e.g. eth0)
    target user (e.g. "" or john.doe or 5000 or "1+210-555-1212")
    target domain (e.g. enterprise.com or an IPv4 address)
    IPv4 addr of flood target (ddd.ddd.ddd.ddd)
    flood stage (i.e. number of packets)
Optional -
    -a flood tool "From:" alias (e.g. jane.doe)
    -i IPv4 source IP address [default is IP address of interface]
    -S srcPort (0 - 65535) [default is well-known discard port 9]
    -D destPort (0 - 65535) [default is well-known SIP port 5060]
    -l lineString line used by SNOM [default is blank]
    -s sleep time btwn INVITE msgs (usec)
    -h help - print this usage
    -v verbose output mode
```

Sintaxis de utilización básica:

```
./inviteflood eth0 target_extension target_domain target_ip number_of_packets
```

Recordando los escenarios presentado en la sección 3.6.1 "Implementación *testbed UBP*". Se adjunta a continuación los output relacionados de la consola de Backtrack para Inviteflood y SVWAR para ser consideradas como ejemplo. Cabe aclarar que se obtendrían salidas en pantalla

similares para el caso de estudio de esta unidad.

Atacante interno - LAN SVWAR:

```
root@bt:/pentest/voip/sipvicious# ./svwar.py -d
/root/Desktop/svwar_wordlist/extensions_list.dic ubpsipserver.myvnc.com -m
INVITE -p 5065
WARNING:TakeASip:extension '2520696194' probably exists but the response is
unexpected
WARNING:TakeASip:extension '1892677372' probably exists but the response is
unexpected
| Extension | Authentication |
-----
| 1008      | reqauth      |
| 1009      | reqauth      |
| 1006      | reqauth      |
| 1007      | reqauth      |
| 1004      | reqauth      |
| 1005      | reqauth      |
| 1002      | reqauth      |
| 1003      | reqauth      |
| 1000      | reqauth      |
| 1001      | reqauth      |
| 1892677372 | weird        |
| 2520696194 | weird        |
```

Atacante externo - WAN SVWAR:

```
root@bt:/pentest/voip/sipvicious# ./svwar.py -d
/root/Desktop/svwar_wordlist/extensions_list.dic ubpsipserver.myvnc.com -m
INVITE -p 5065
WARNING:TakeASip:extension '486246286' probably exists but the response is%\begin{center}
%\\textbf{\large\color[rgb]{1,0,0}{An\'alisis formal de attack responses -
%PENDIENTE}}}}
%end{center}
unexpected
WARNING:TakeASip:extension '3742607132' probably exists but the response is
unexpected
| Extension | Authentication |
-----
| 486246286 | weird        |
| 2008      | reqauth      |
| 2009      | reqauth      |
| 2006      | reqauth      |
| 2007      | reqauth      |
| 2004      | reqauth      |
| 2005      | reqauth      |
| 2002      | reqauth      |
| 2003      | reqauth      |
| 2000      | reqauth      |
| 2001      | reqauth      |
| 3742607132 | weird        |
```

Atacante interno - LAN Inviteflood:

```
root@bt:/pentest/voip/inviteflood# ./inviteflood eth0 1000
ubpsipserver.myvnc.com 20.20.0.240 5 -D 5065

inviteflood - Version 2.0
                June 09, 2006

source IPv4 addr:port    = 20.20.0.7:9
dest   IPv4 addr:port    = 20.20.0.240:5065
targeted UA                 = 1000@ubpsipserver.myvnc.com

Flooding destination with 5 packets
sent: 5s
```

Atacante externo - WAN Inviteflood:

```
root@bt:/pentest/voip/inviteflood# ./inviteflood eth0 1000
ubpsipserver.myvnc.com 186.109.2.218 5 -D 5065

inviteflood - Version 2.0
June 09, 2006

source IPv4 addr:port    = 10.0.0.86:9
dest   IPv4 addr:port    = 186.109.2.218:5065
targeted UA                = 1000@ubpsipserver.myvnc.com

Flooding destination with 5 packets
sent: 5
```

[Shai R., 2011]

Estas herramientas maliciosas alcanzarán de forma directa a nuestro SIP *registrar* sin ser advertidas por Artemisa en caso de que las extensiones no fueran previamente cargadas en los archivos de configuración. Es aquí cuando mediante un XML-RPC *client* que facilite la integración SIP *registrar/proxy - honeypot* se podrá aumentar la eficacia de este sistema de detección.

Debajo se muestra un sencillo XML-RPC Python *client* que fue desarrollado para demostrar lo expuesto en el párrafo inmediatamente anterior. Este sencillo programa al recibir como input un archivo de texto, en este caso el log perteneciente a el *registrar SIP Asterisk 1.6*. Exactamente /var/log/asterisk/messages.x realizará una búsqueda en el mismo, de extensiones que resultasen sospechosas. Para posteriormente registrarlas en Artemisa en caso de un futuro ataque. Lo que resultaría en neutralizar un próximo DoS y/o *enumeration attack* con similares características. Por ejemplo donde el atacante haga uso del mismo diccionario de extensiones para realizar una enumeración, un intento de registro o un DoS.

El XML-RPC *client* está compuesto por dos scripts en Python: ast_log_pars.py permite obtener a partir de un archivo de log de Asterisk las extensiones que han sido alcanzadas por el usuario malintencionado.

ast_log_pars.py

```
class AsteriskLogParser():

    def ExtLogParser(self):

        fileLog = open('./asterisk_logs/messages_svwar_WAN.log','r')

        ext=[]
        ext_return=[]

        line=fileLog.readline()
        while line!='':

            if line.find('Extension')!=-1:
                index=line.find('Extension')
                ext.append(line[index+11:index+15])

            line=fileLog.readline()

        # Clearing repeated entries in returned extension list
        for i in ext:
            if i not in ext_return:
                ext_return.append(i)

        return ext_return
```

Conjuntamente, artemisa_XML-RPC_client.py el cual por un lado instancia a la clase donde se codifico el *parser* de texto para obtener las extensiones involucradas. Y por el otro, en base a estas extensiones las pasará como parámetros a nuestro método principal para agregar o quitar números registrados:

En el XML-RPC *client*

```
XML-RPC\__Object\__in.modify\__extension('add','extension','username',
'password')

XML-RPC\__Object\__in.modify\__extension('delete','extension','username',
'password')

XML-RPC_Object_in.restart()
```

Finalmente llegará al XML-RPC *server*

```
ModifyExt(self,mod,ext,user,passwd)

RestartArtemisa()
```

Recordar la sintaxis de la XML-RPC API presentada en la sección 3.7.1.5

artemisa_XML-RPC_client.py

```
from ast_log_pars import AsteriskLogParser
import xmlrpclib

asteriskExt = AsteriskLogParser()
extension = asteriskExt.ExtLogParser()
print 'List of Extension to be registered based on suspisous registar logs:'
print extension

s = xmlrpclib.ServerProxy('http://10.10.0.250:9000')

for i in extension:
    print 'Trying to modify the extension '+i
    print s.modify_extension('add',i,i,i)
    #print s.modify_extension('delete','phone_7','phone_7','1007')
    #print s.modify_extension('add','phone_8','phone_8','1008')

s.restart()

# Print list of available methods.
# Hope you feel confident to add new methods to XML-RPC service.
print s.system.listMethods()
```

Será menester a la hora de realizar un Cliente XML-RPC, tomar en consideración todas las variables y validaciones necesarias por parte del administrador de sistemas para evitar cualquier posible conflicto o inconsistencia en cuanto al registro y borrado de usuarios. Como así también cualquier tipo de configuración que sea modificada en Artemisa que pueda impactar en el entorno de voz sobre IP. Por lo que, el ejemplo planteado puede ser entendido como un sencillo código cliente a modo de referencia, que para ser utilizado en la práctica debería ser estrictamente revisado y modificado según corresponda. En otras palabras, el administrador deberá ajustar su scripts considerando solo ataques y extensiones sospechosas evitando secuestrar una sesión SIP de un usuario válido.

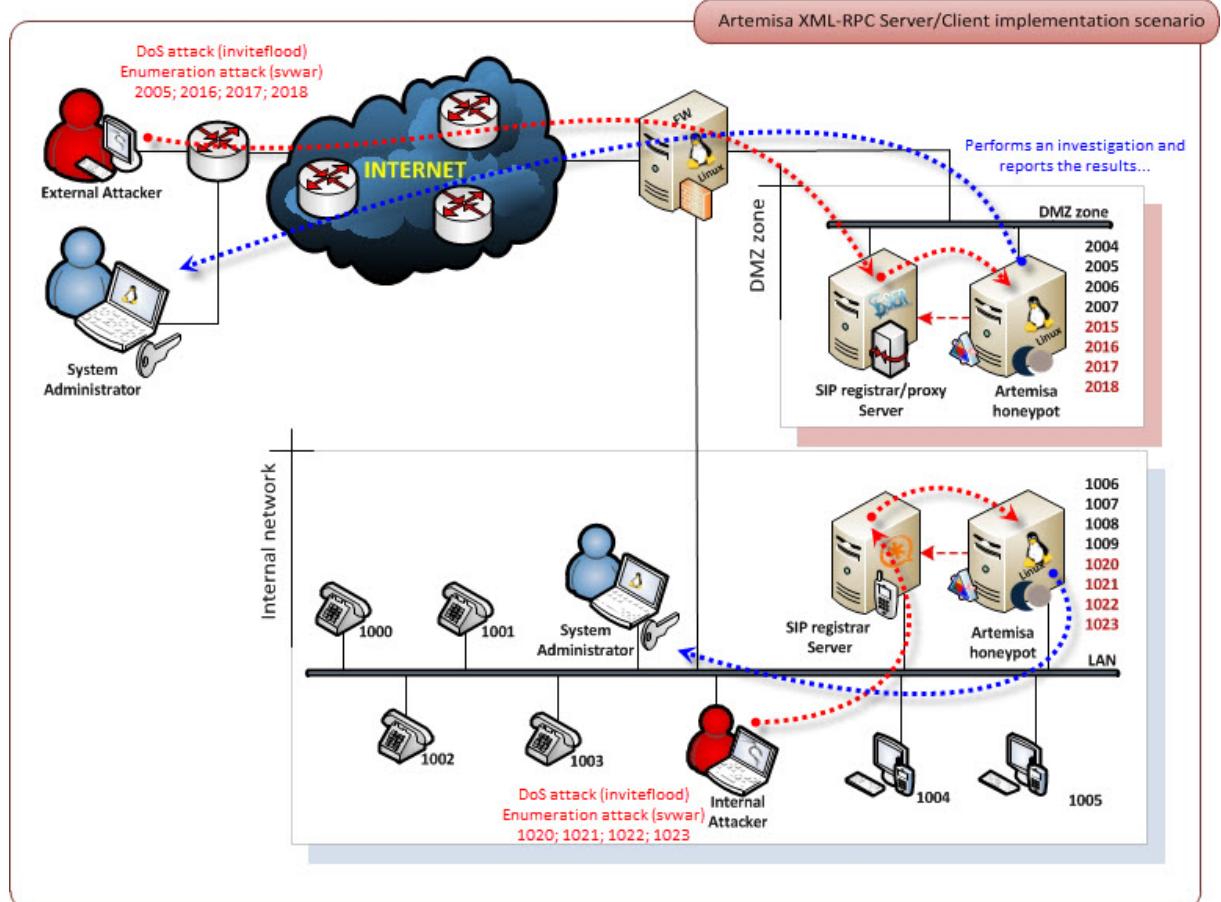


Figura 3.64: Diagrama topología XML-RPC cliente/servidor luego de su ejecución.

Como se aprecia en la Figura 3.64, se acotará nuestro ejemplo al resultante de un ataque LAN/WAN con SVWAR/Inviteflood a las extensiones WAN 2015, 2016, 2017 y 2018. Y LAN 1020, 1021, 1022, 1023 y 1024. Se realizará el desarrollo en detalle solo de este último, es decir en un entorno de área local.

Luego de recibir una enumeración con mensajes SIP INVITE mediante SVWAR contra nuestro SIP *registrar* Asterisk 1.6 sus *logs* acusarán el evento en /var/log/asterisk-/messages.x

Fragmento de /var/log/asterisk/messages.x

```
[Feb 13 17:09:31] VERBOSE[9054] pbx.c:      -- Executing [1020@default:1]
Verbose("SIP/ubpsipserver.myvnc.com-00000017", "1,Extension 1020") in new stack
[Feb 13 17:09:31] VERBOSE[9054] app_verbose.c: Extension 1020
[Feb 13 17:09:31] VERBOSE[9054] pbx.c:      -- Executing [1020@default:2]
Playback("SIP/ubpsipserver.myvnc.com-00000017", "queue-thankyou") in new stack
[Feb 13 17:09:31] VERBOSE[8906] netsock.c:    == Using SIP RTP CoS mark 5
[Feb 13 17:09:31] VERBOSE[9055] pbx.c:      -- Executing [1021@default:1]
Verbose("SIP/ubpsipserver.myvnc.com-00000018", "1,Extension 1021") in new stack
[Feb 13 17:09:31] VERBOSE[9055] app_verbose.c: Extension 1021
[Feb 13 17:09:31] VERBOSE[9055] pbx.c:      -- Executing [1021@default:2]
Playback("SIP/ubpsipserver.myvnc.com-00000018", "queue-thankyou") in new stack
[Feb 13 17:09:31] VERBOSE[8906] netsock.c:    == Using SIP RTP CoS mark 5
[Feb 13 17:09:31] VERBOSE[9056] pbx.c:      -- Executing [1022@default:1]
Wait("SIP/ubpsipserver.myvnc.com-00000019", "1") in new stack
[Feb 13 17:09:31] VERBOSE[8906] netsock.c:    == Using SIP RTP CoS mark 5
[Feb 13 17:09:31] VERBOSE[9057] pbx.c:      -- Executing [1023@default:1]
Verbose("SIP/ubpsipserver.myvnc.com-0000001a", "1,Extension 1023") in new stack
[Feb 13 17:09:31] VERBOSE[9057] app_verbose.c: Extension 1023
[Feb 13 17:09:31] VERBOSE[9057] pbx.c:      -- Executing [1023@default:2]
```

```

Playback("SIP/ubpsipserver.myvnc.com-0000001a", "queue-thankyou") in new stack
[Feb 13 17:09:31] VERBOSE[8906] netsock.c: == Using SIP RTP CoS mark 5
[Feb 13 17:09:31] VERBOSE[9057] pbx.c: -- Executing [1024@default:1]
Verbose("SIP/ubpsipserver.myvnc.com-0000001a", "1,Extension 1024") in new stack
[Feb 13 17:09:31] VERBOSE[9057] app_verbose.c: Extension 1024
[Feb 13 17:09:31] VERBOSE[9057] pbx.c: -- Executing [1024@default:2]
Playback("SIP/ubpsipserver.myvnc.com-0000001a", "queue-thankyou") in new stack

```

Frente a esto podría ejecutarse nuestro sencillo aplicativo XML mediante un *cron* bajo cualquier plataforma Linux de manera de obtener a partir de los *logs* de nuestro SIP registrar las posibles extensiones utilizadas por el atacante como objetivo. Se podría leer en tiempo real cualquier tipo de *logs* para poder realizar un ajuste más eficiente de los *users* SIP logueados a la central. Desde el punto de vista del administrador de sistemas del entorno, podrían proponerse diferentes alternativas en base a las características de este habiendo infinidad de posibilidades para ser mejorado.

Se expone la salida en pantalla de ejecución de nuestro Cliente XML-RPC. Quien luego de identificar las extensiones comprometidas intentará registrarlas siempre y cuando no existan previamente. Y no se haya superado la capacidad de ocho usuarios virtuales que soporta Artemisa actualmente.

Output XML-RPC client

```

List of Extension to be registered based on suspisous registar logs:
['1006', '1007', '1008', '1009', '1015','1020', '1021', '1022', '1023', '1024']
Trying to modify the extension 1006
Extension 1006 already exists
Trying to modify the extension 1007
Extension 1007 already exists
Trying to modify the extension 1008
Extension 1008 already exists
Trying to modify the extension 1009
Extension 1009 already exists
Trying to modify the extension 1020
Extension 1020 added
Trying to modify the extension 1021
Extension 1021 added
Trying to modify the extension 1022
Extension 1022 added
Trying to modify the extension 1023
Extension 1023 added
Trying to modify the extension 1024
Max number of extensions registered. Run another artemisa instance to register
more extensions.

['modify_extension', 'restart', 'system.listMethods', 'system.methodHelp',
'system.methodSignature']

```

Desde el punto de vista del servidor XML-RPC ejecutado con el *honeypot* se aprecia claramente el mismo comportamiento:

Output de Artemisa CLI a una XML-RPC client request

```

2014-03-12 21:07:55,470 Extension 1006 already exists in extensions.conf
10.10.0.250 -- [12/Mar/2014 21:07:55] "POST /RPC2 HTTP/1.0" 200 -
2014-03-12 21:07:55,473 Extension 1007 already exists in extensions.conf
10.10.0.250 -- [12/Mar/2014 21:07:55] "POST /RPC2 HTTP/1.0" 200 -
2014-03-12 21:07:55,470 Extension 1008 already exists in extensions.conf
10.10.0.250 -- [12/Mar/2014 21:07:55] "POST /RPC2 HTTP/1.0" 200 -
2014-03-12 21:07:55,473 Extension 1009 already exists in extensions.conf
10.10.0.250 -- [12/Mar/2014 21:07:55] "POST /RPC2 HTTP/1.0" 200 -
2014-03-12 21:07:55,478 Extension 1020 added in extensions.conf
2014-03-12 21:07:55,479 Extension 1020 added in servers.conf
10.10.0.250 -- [12/Mar/2014 21:07:55] "POST /RPC2 HTTP/1.0" 200 -

```

```

2014-03-12 21:07:55,484 Extension 1021 added in extensions.conf
2014-03-12 21:07:55,484 Extension 1021 added in servers.conf
10.10.0.250 -- [12/Mar/2014 21:07:55] "POST /RPC2 HTTP/1.0" 200 -
2014-03-12 21:07:55,488 Extension 1022 added in extensions.conf
2014-03-12 21:07:55,489 Extension 1022 added in servers.conf
10.10.0.250 -- [12/Mar/2014 21:07:55] "POST /RPC2 HTTP/1.0" 200 -
2014-03-12 21:07:55,494 Extension 1023 added in extensions.conf
2014-03-12 21:07:55,494 Extension 1023 added in servers.conf
10.10.0.250 -- [12/Mar/2014 21:07:55] "POST /RPC2 HTTP/1.0" 200 -
2014-03-12 21:07:55,501 Max number of extensions registered. Run another
artemisa instance to register more extensions.

```

**Resultando finalmente en una configuración de Artemisa en base sus archivos
artemisa_1.1.8/conf/extension.conf y ../../conf/servers.conf**

artemisa_1.1.8/conf/extension.conf

```

# Artemisa - Extensions configuration file
#
# Be careful when modifying this file!

# Here you are able to set up the extensions that shall be used by Artemisa in
# the registration process. In order to use them, they must be defined in the
# servers.conf file.
#
# The sections name hereunder, such as 3000 in section [3000], refers to a SIP
# extension and it must be unique in this file, as well as correctly configured in
# the registrar server.

[1006]
username = "1006"
password = 1006

[1007]
username = "1007"
password = 1007

[1008]
username = "1008"
password = 1008

[1009]
username = "1009"
password = 1009

[1020]
username = "1020"
password = 1020

[1021]
username = "1021"
password = 1021

[1022]
username = "1022"
password = 1022

[1023]
username = "1023"
password = 1023

```

artemisa_1.1.8/conf/servers.conf

```

# Artemisa - Servers configuration file
#
# Be careful when modifying this file!

```

```

# Here you are able to set the registrar servers configuration that Artemisa
shall use to register itself.
#
# registrar_time=
# Is the time in minutes between automatic registrations. This is performed in
order to avoid
# being disconnected from the server because of a lack of activity.
#
# nat_keepalive_interval=
# When dealing with NAT proxies, you can set a value in seconds which indicates
the time interval between keep alive messages. If zero is written, then the NAT
keep alive messages shall not be sent.
#
# exten=
# In this field you should set the extensions to be used. They must be declared
in extensions.conf.

[myproxy]
nat_keepalive_interval = 30
exten = 1006,1007,1008,1009,1020,1021,1022,1023
registrar_ip = 10.10.0.245
registrar_time = 15
registrar_port = 5060

```

Ejecución de Artemisa luego de correr el cliente XML-RPC donde se puede ver el registro de las nuevas extensiones sospechosas.

Artemisa cli output

```
Artemisa vrepvernumber Copyright (C) 2009-2013 Mohamed Nassar, Rodrigo do Carmo,
Pablo Masri, Mauro Villarroel and Exequiel Barirero
```

```
This program comes with ABSOLUTELY NO WARRANTY; for details type 'show
warranty'.
This is free software, and you are welcome to redistribute it under certain
conditions; type 'show license' for details.
```

Type 'help' for help.

```

19:42:46.528 os_core_unix.c !pjlib 2.0.1 for POSIX initialized
19:42:46.528 sip_endpoint.c .Creating endpoint instance...
19:42:46.528          pjlib .select() I/O Queue created (0x9788760)
19:42:46.528 sip_endpoint.c .Module "mod-msg-print" registered
19:42:46.528 sip_transport. .Transport manager created.
19:42:46.528 pjsua_core.c .PJSUA state changed: NULL --> CREATED
SIP User-Agent listening on: 10.10.0.250:5061
XML-RPC service listening on: 10.10.0.250:9000
Behaviour mode: active
Starting extensions registration process...
2014-03-05 19:42:46,624 Extension 1006 registration sent. Status: 100 (In
Progress)
2014-03-05 19:42:46,634 Extension 1007 registration sent. Status: 100 (In
Progress)
2014-03-05 19:42:46,638 Extension 1008 registration sent. Status: 100 (In
Progress)
2014-03-05 19:42:46,646 Extension 1009 registration sent. Status: 100 (In
Progress)
2014-03-05 19:42:46,624 Extension 1020 registration sent. Status: 100 (In
Progress)
2014-03-05 19:42:46,634 Extension 1021 registration sent. Status: 100 (In
Progress)
2014-03-05 19:42:46,638 Extension 1022 registration sent. Status: 100 (In
Progress)
2014-03-05 19:42:46,638 Extension 1023 registration sent. Status: 100 (In
Progress)
2014-03-05 19:42:46,704 Extension "1006" <sip:1006@10.10.0.245:5060> registered,
status=200 (OK)
2014-03-05 19:42:46,706 Extension "1007" <sip:1007@10.10.0.245:5060> registered,
status=200 (OK)
2014-03-05 19:42:46,713 Extension "1008" <sip:1008@10.10.0.245:5060> registered,
```

```

status=200 (OK)
2014-03-05 19:42:46,716 Extension "1009" <sip:1009@10.10.0.245:5060> registered,
status=200 (OK)
2014-03-05 19:42:46,704 Extension "1020" <sip:1006@10.10.0.245:5060> registered,
status=200 (OK)
2014-03-05 19:42:46,706 Extension "1021" <sip:1007@10.10.0.245:5060> registered,
status=200 (OK)
2014-03-05 19:42:46,713 Extension "1022" <sip:1008@10.10.0.245:5060> registered,
status=200 (OK)
2014-03-05 19:42:46,716 Extension "1023" <sip:1009@10.10.0.245:5060> registered,
status=200 (OK)
delivery>

```

Luego de analizar el escenario exhibido quedan plasmadas todas las posibilidades y ventajas que brinda una interfaz programable al *honeypot* Artemisa. Una interfaz de programación representa la capacidad de comunicación entre componentes o entidades de software, en este caso, todos los que integran la arquitectura de VoIP-SIP (*SIP phone, proxy, registrar*, Ramales Privados de Comunicación IP (IPBX), Protocolo Ligero/Simplificado de Acceso a Directorios (LDAP), FW,SIP-PSTN GW, entre otros). Resultando en el incremento del nivel de seguridad del entorno de voz sobre IP mediante la integración de las entidades antedichas. Como se estudió mediante un monitoreo preventivo, se toman acciones anticipadas y se disparan en tiempo real.

Para la cama de prueba se definió el empleo de *inviteflood*, mientras la herramienta se mantenga inundando el *gateway SIP* va a prevenir a los usuarios cursar llamadas. Por ejemplo, podría inundarse el *proxy SIP* con una extensión inexistente, de esta manera de mantenerlo ocupado al ir generando mensajes de respuesta *404 not found*. Este ataque se vería anegado al integrar nuestro *registrar* para acusar todo tipo de eventos no deseados o sospechosos.

Ejemplo de *strings* no esperados en /var/log/asterisk/messages.x

```

NOTICE.* .*: Registration from '.*' failed for '<HOST>:.*' - Wrong password
        NOTICE.* .*: Registration from '.*' failed for '<HOST>:.*' - No
matching peer found
        NOTICE.* .*: Registration from '.*' failed for '<HOST>:.*' - No
matching peer found
        NOTICE.* .*: Registration from '.*' failed for '<HOST>:.*' -
Username/auth name mismatch
        NOTICE.* .*: Registration from '.*' failed for '<HOST>:.*' - Device
does not match ACL
        NOTICE.* .*: Registration from '.*' failed for '<HOST>:.*' - Peer is
not supposed to register
        NOTICE.* .*: Registration from '.*' failed for '<HOST>:.*' - ACL
error (permit/deny)
        NOTICE.* .*: Registration from '.*' failed for '<HOST>:.*' - Device
does not match ACL
        NOTICE.* .*: Registration from '\".*\\".*' failed for '<HOST>:.*' -
No matching peer found
        NOTICE.* .*: Registration from '\\".*\\\".*' failed for '<HOST>:.*' -
Wrong password
        NOTICE.* <HOST> failed to authenticate as '.*$'
        NOTICE.* .*: No registration for peer '.*' \ (from <HOST>\)
        NOTICE.* .*: Host <HOST> failed MD5 authentication for '.*' (.*)
        NOTICE.* .*: Failed to authenticate user .*@<HOST>.*
        NOTICE.* .*: <HOST> failed to authenticate as '.*'
        NOTICE.* .*: <HOST> tried to authenticate with nonexistent user
'..'
        VERBOSE.*SIP/<HOST>--.*Received incoming SIP connection from unknown
peer

```

De esta forma, los administradores de sistema y programadores podrán utilizar las funcionalidades de la API o incluso mejorar la misma para elevar las defensas de la infraestructura SIP de una organización cuanto crean necesario.

3.7.3. Actualización de *fingerprints*

Adición de *fingerprints* - SMAP: se definirá el concepto de *fingerprint* (identificador de tipo de agente) en base a Nmap (Escaner de seguridad). Nmap (Network Mapper) es una utilidad gratuita y de código abierto para descubrimiento de red y auditorías de seguridad informática. Muchos administradores de sistemas y redes también lo encuentran útil para obtener inventarios de red, administrar tareas de actualización de servicios de manera programada, y monitorear disponibilidad de servicios o monitorear dispositivos de red. Nmap utiliza paquetes IP en bruto de una manera novedosa para determinar cuales *hosts* están disponibles en la red, qué servicios (nombre y versión del aplicativo) estos *hosts* están ofreciendo, qué OS y versión de OS éstos están ejecutando, qué tipo de corta fuegos/filtros de paquetes están en uso, y decenas de otras características. Fue diseñado para rápidamente poder realizar escaneos a grandes redes de datos, a su vez funciona muy bien contra un terminal individual. Nmap corre en todos los OS de computadoras, paquetes binarios oficiales están disponibles para Linux, Windows, y MAC OS X. Una de las características más conocidas de Nmap es la detección de OS remota utilizando *fingerprinting* (identificación de tipo de agente). Nmap envía una serie de paquetes TCP y UDP al *host* remoto y examina prácticamente cada bit en las respuestas a estos. Luego de realizar decenas de pruebas tales como muestreo TCP, Número de Secuencia Inicial (ISN)²⁰, soporte y ordenamiento de opciones TCP, muestreo de ID IP y la verificación de tamaño de ventana inicial. Nmap compara los resultados contra la base de datos nmap-os-db de más de 2600 *fingerprints* de OS conocidos y presenta en pantalla los detalles del OS que coincide con la búsqueda comparativa. Cada *fingerprint* incluye una descripción de forma textual libre del OS en cuestión, y una clasificación que provee el nombre del proveedor o compañía (ej.: Sun), nombre OS (ej.: Solaris), generación o versión del OS (ej.: 10), y tipo de dispositivo (de propósito general, *router*, *switch*, consola de juegos, etc). La mayor parte de los *fingerprints* también poseen una representación de Plataforma Común de Enumeración (CPE), tal como cpe:/o:linux:linux_kernel:2.6. [Lyon G., 2014]

Y complementariamente, recordando la descripción de SMAP presentada en la sección 3.5.3.1 caso 1, herramienta SMAP el lector ya habrá ganado el entendimiento del concepto de *fingerprint*.

Como se analizó previamente Artemisa *honeypot* dispone de un archivo donde se encuentran los *fingerprints* utilizados para identificar las herramientas de ataques contra nuestro entorno de voz sobre IP, más precisamente en el directorio

/home/delivery/Desktop/artemisa_1.1.8/fingerprint. A continuación se presentará el agregado de un nuevo *fingerprint* para reconocer la herramienta SMAP con sus aparejadas capturas. La misma se ejecutó en un entorno de prueba simple, con la extensión virtual "1000" registrada en un SIP registrar Asterisk 1.6. El ataque de prueba se ejecuto con *Backtrack Linux 5 R2*.

Sintaxis SMAP 0.6.0

```
root@bt:/pentest/voip/smap# ./smap
smap 0.6.0 <hs@123.org> http://www.wormulon.net/
usage: smap [ Options ] <ip | ip:port | ip/mask | host>
```

²⁰ISN (Initial Sequence Number): intenta determinar el patrón de generación de secuencia de los números de secuencia iniciales de TCP (también conocido como muestreo TCP ISN), los números de identificación IP (también conocidos como muestreo IPID), y los números de *time stamp* (estampa de tiempo). La prueba es realizada mediante el envío de seis paquetes TCP con la bandera SYN habilitada para un puerto TCP abierto. [EC-Council, 2010]

```

-h: this help
-d: increase debugging
-o: enable fingerprinting
-O: enable more verbose fingerprinting
-l: fingerprint learning mode
-t: TCP transport
-u: UDP transport (default)
-p0: Treat all hosts as online - skip host discovery
-p <port>: destination port
-r <rate>: messages per second rate limit
-D <domain>: SIP domain to use without leading sip:
-w <timeout>: timeout in msec

```

Escaneo con SMAP contra Artemisa

```

root@bt:/pentest/voip/smap# ./smap -o 10.10.0.250 -p 5062
smap 0.6.0 <hs@123.org> http://www.wormulon.net/
sendto: Network is unreachable
NOTICE: STUN based IP discovery failed, falling back to ioctl()
NOTICE: Could not obtain local port 5060. Scanning may be unreliable!
10.10.0.250: ICMP reachable, SIP enabled
    Guess: Cisco 79x0 series Phone (version 7.5)
    User-Agent: Twinkle/1.4.2
1 host scanned, 1 ICMP reachable, 1 SIP enabled (100.0%)

```

Correlación final Artemisa sin *fingerprint* SMAP

```

***** Correlation *****
Artemisa concludes that the arrived message is likely to be:

* A scanning attempt.

Executing bash ./scripts/on_scanning.sh 10.10.0.60 12345 10.10.0.60 12345 smap
...
* The message belongs to a ringing attack.

This report has been saved on file ./results/2014-05-10_1.txt
NOTICE This report has been saved on file ./results/2014-05-10_1.html
E-mail notification is disabled.
Call from <sip:smap@localhost> is DISCONNCTD, last code = 406 (Not Acceptable)
Error while closing the conferences in method on_state().
Call from <sip:smap@localhost> is DISCONNCTD, last code = 406 (Not Acceptable)
Error while closing the conferences in method on_state().

```

Correlación final Artemisa con *fingerprint* SMAP

```

***** Correlation *****
Artemisa concludes that the arrived message is likely to be:

* The attack was created employing the tool SMAP.
* A scanning attempt.

Executing bash ./scripts/on_scanning.sh 10.10.0.60 12345 10.10.0.60 12345 smap
SMAP ...
* The message belongs to a ringing attack.

This report has been saved on file ./results/2014-05-10_7.txt
NOTICE This report has been saved on file ./results/2014-05-10_7.html
E-mail notification is disabled.
Call from <sip:smap@localhost> is DISCONNCTD, last code = 406 (Not Acceptable)
Error while closing the conferences in method on_state().
Call from <sip:smap@localhost> is DISCONNCTD, last code = 406 (Not Acceptable)
Error while closing the conferences in method on_state().

```

Se podrá notar que el *honeypot* concluye que el dominio VoIP recibió un ataque de escaneo utilizando SMAP: “*The attack was created employing the tool SMAP*”.

3.7.4. Resumen de resultados

Sobre los desarrollos Funcionales sobre Artemisa:

- La interfaz XML-RPC añadida al código fuente agiliza y facilita la configuración, y permite la integración del *honeypot* con otros agentes de la red. Por su parte, a partir de la misma podrían fácilmente agregarse otros módulos para impactar cambios en todas las configuraciones de Artemisa y no solo el agregado dinámico de extensiones a proteger.
- A través del ejemplo de implementación XML-RPC quedó demostrada la efectividad de Artemisa con la nueva API en un escenario probable de la vida real, es decir, contra un atacante externo que hace uso de herramientas maliciosas desde la WAN. El *honeypot* puede responder de forma dinámica y en tiempo real protegiendo las extensiones que están siendo atacadas en ese instante de tiempo.

3.8. Conclusiones

Sobre la evaluación de respuestas empíricas a partir de su implementación y experimentación:

- Este *honeypot* se hace fuerte en entornos de red privada, acusando en tiempo real y con registros detallados la presencia de un atacante interno.
- Una configuración en el SIP *registrar* del entorno que encamine hacia el *honeypot* todo intento de comunicación contra un agente SIP que no pertenezca al dominio, aumenta el grado de detección de atacantes, tanto internos como externos.
- Considerar en el caso de un atacante interno, que disponer de dos instancias de Artemisa, una con *fingerprint* de un *softphone* y otra de *registrar* será muy ventajoso. Ya que de esta manera podremos detectar a través de nuestros *honeypots* tanto las IP de origen y de destino de los ataques dentro de la LAN.

Sobre el estudio descriptivo estadístico de las características de Artemisa *honeypot* en entornos reales en mesa de prueba:

- Al exponer al *honeypot* a Internet se descubrió predominancia de mensajes *500 Server Internal Error* asociado a que Artemisa solo tiene la capacidad de responder a ciertos SIP *messages* considerando que el mismo actúa como un UA y no tiene la capacidad de un *proxy SIP* o *registrar*. A pesar de esto como herramienta de estudio sobre ataques en la red pública, Artemisa, o mejor aún, una segunda instancia de Artemisa debe configurarse idealmente para que entregue como su *fingerprint*, el de un *proxy SIP* para disuadir al atacante.
- En cuanto a la cantidad de llamadas VoIP SIP total (*registrars*) se validó un número importantísimo de llamadas rechazadas, lo que se ve directamente reflejado por las estadísticas del método REGISTER, como así también de los mensajes SIP *401 unauthorized* y *404 not found* presentes en gran número e involucrados en este proceso. Los cuales resultan de una gran cantidad de registros fallidos. Por su número exagerado y contrastando con los *logs* de Asterisk es que se acreditaron como intentos de registro mal intencionados.
- Integrando todo el tráfico con origen o destino a direcciones IP públicas se destaca que Argentina y Alemania fueron los países de mayor protagonismo. Todo el tráfico recibido de una nacionalidad diferente a la Argentina, puede considerarse sospechoso. En tercer lugar Estados Unidos habiendo generado un tráfico también considerable. Y por último Corea, Irlanda y Polonia con un porcentaje mucho menor.
- La efectividad que tuvo la plataforma en los entornos propuestos para detectar los ataques cubiertos por la misma. Principalmente, los ataques de inundación en un número mayor a 180.000 durante el período de exposición de seis meses. Sin descontar los 102 ataques de *ringing* y *scanning attempts*. Luego podríamos decir que los entornos no recibieron ataques de SPIT.
- Tomando como referencia todas las conclusiones antes presentadas sobre el estudio descriptivo estadístico de Artemisa, se hace notorio que permitiría proteger de forma más efectiva el dominio de comunicaciones SIP si se incrementa el alcance del *honeypot* para simular el rol de un servidor de registro SIP, es decir, agregar algunos métodos para que

tenga funciones básicas de un *registrar* y así lograr desviar todos los ataques a los que la actual versión de Artemisa no se ve expuesto por su rol de *back-end user-agent*.

Sobre los desarrollos Funcionales sobre Artemisa:

- La interfaz XML-RPC añadida al código fuente agiliza y facilita la configuración, y permite la integración del *honeypot* con otros agentes de la red. Por su parte, a partir de la misma podrían fácilmente agregarse otros módulos para impactar cambios en todas las configuraciones de Artemisa y no solo el agregado dinámico de extensiones a proteger.
- A través de la implementación XML-RPC el *honeypot* puede responder de forma dinámica y en tiempo real protegiendo las extensiones que están siendo atacadas en ese instante de tiempo.

Sobre el proyecto en general:

Este proyecto presenta una variante de prácticas de laboratorio para alumnos avanzados de ingeniería en materias como seguridad de la información, protocolos y gestión de red. Por su parte, permite la integración de estas materias con otras relacionadas con redes de computadoras y sistemas operativos. Se comenta lo anterior, ya que siguiendo las configuraciones que fueron presentadas y haciendo uso de los servidores, teléfonos SIP y dispositivos de red instalados para el mismo, podrán implementarse entornos de comunicaciones seguras de voz sobre IP. Esto posibilitaría a los alumnos estudiar el comportamiento de dichos entornos y protocolos, como su respuesta a una infinidad de modificaciones y pruebas. Si bien aquí se apuntó a la implementación y desarrollos sobre un señuelo para VoIP *Open Source*, utilizando las bases de este trabajo podría instalarse y poner a prueba cualquier dispositivo o *soft VoIP* basado principalmente en tecnologías SIP. Los entornos virtuales pueden crearse en muy poco tiempo y sin costo, para un gran número de servicios de red como *web servers*, servidores DNS, proxy, NAT, *web mail*, Protocolo de Transferencia de Archivos (FTP), NFS, monitoreo (Nagios, Cacti, OpenNMS), Red Privada Virtual (VPN) aportando una gran flexibilidad al proceso de aprendizaje dentro del ámbito teórico práctico del futuro ingeniero. La UBP ya cuenta con todo lo necesario, por lo que es posible aplicar el proyecto en materias como las antes mencionadas para la puesta en marcha y desarrollo de aplicativos, entornos en mesa de prueba y configuraciones para prácticas de laboratorio. Incluso en el Anexo F se proponen algunos *To-Do* (ha realizar) que podrían considerarse para un futuro TFC de la carrera de Ing. en Telecomunicaciones. La combinación con varias materias ofrece mayor amplitud y un sin numero de posibilidades para la realización de nuevos casos de estudio. Todo esto contribuye con experiencia muy valiosa al perfil profesional del egresado.

Bibliografía

- Acunetix© (2015). XSS – the Underestimated Exploit. What is XSS (Cross Site Scripting)? <https://www.acunetix.com/websitedevelopment/xss/>. [07-11-2015].
- Antonopoulos A. M. (2011). Honeypots for hacker detection. <http://www.networkworld.com/article/2213251/network-security/honeypots-for-hacker-detection.html>. [06-03-2012].
- Camarillo G. (2002). *SIP Demystified*. Ed. McGraw-Hill, USA.
- Collier M. D. (2006). Enterprise Telecom Security Threats. <http://www.securelogix.com/telecom-security.html>. [06-02-2013].
- Computer Security Institute (2010/2011). 15th annual csi computer crime and security survey. <http://www.GoCSI.com>. [06-02-2013].
- De Laet G., S. G. (2005). *Cisco Network Security Fundamentals*. Cisco, USA.
- Digium© (2015). Get Started. What is Asterisk? <http://www.asterisk.org/get-started>. [05-03-2015].
- Do Carmo R., Festor O., N. M. (2011a). Artemisa: an Open-source Honeypot Back-end to Support Security in Voip Domains. *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on. Volumen Anual*, pages 361–368.
- Do Carmo R., Masri P., N. M. (2011b). “Artemisa v1.0 documentation.”. <http://artemisa.sourceforge.net/documentation/Introduction.html>. [03-02-2012].
- EC-Council (2010). *Ethical Hacking and Countermeasures: Attack Phases*. Ed Cengage Learning, USA.
- Espinoza M. P., Montalván H., P. R. (2009). “Honeynet como Apoyo a la Investigación de Seguridad de Redes en entornos Universitarios”. <http://www.utpl.edu.ec/honeynet/wp-content/uploads/2009/09/proyecto-honeynet-utpl.pdf>. [03-01-2012].
- Ewald T., Nassar M., N. S. S. R. (2007). Holistic Voip intrusion detection and prevention system. *Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications. Volumen IPTComm '07*, pages 1–9.
- Greene T. (2011). Botnets, cloud computing power may be fueling attacks against VoIP. <http://www.networkworld.com/article/2200806/voip/botnets--cloud-computing-power-may-be-fueling-attacks-against-voip.html>. [17-04-2012].

Hundley, K. (2009). *Alcatel-Lucent Scalable IP Networks Self-Study Guide: Preparing for the Network Routing Specialist I (NRS1) Certification Exam (4A0-100)*. Wiley Publishing Inc., Canada.

Internet World Stats (2015). Internet Usage Statistics for the Americas. <http://www.internetworldstats.com/stats.htm>. [18-12-2015].

Johnston A., Donovan S., S. R. (2003). Sip Basic Call Flow Examples - Best Current Practice. <https://tools.ietf.org/html/rfc3665>. [RFC: 3665] [05-01-2012].

Kidd E. (2012). XML-RPC HOWTO. <http://www.tldp.org/HOWTO/XML-RPC-HOWTO>. [17-04-2014].

L. Madsen, J. Van Megelen, R. B. (2011). Asterisk™: The Definitive Guide. http://www.asteriskdocs.org/en/3rd_Edition/asterisk-book-html-chunk/index.html. [06-01-2014].

Lyon G. (2014). Nmap Security Scanner. <https://nmap.org>. [14-12-2012].

Messmer E. (2011). Massive DDos attacks a growing threat to VoIP services. <http://www.networkworld.com/article/2181743/voip/massive-ddos-attacks-a-growing-threat-to-voip-services.html>. [16-12-2012].

Microsoft® (2015). SQL Injection. <https://technet.microsoft.com/en-us/library/ms161953%28v=sql.105%29.aspx?f=255&MSPPError=-2147217396>. [06-01-2015].

Nassar M. (2009). *VoIP Networks Monitoring and Instrusion Detection*. Tesis de doctorado, Universidad Henri Poincaré, Francia.

Neira Ayuso P., W. H. (2014). The netfilter.org iptables project. What is iptables? <http://www.netfilter.org/projects/iptables/>. [05-04-2014].

Niemi, A. (2004). Session Initiation Protocol (sip) Extension for Event State Publication. <https://tools.ietf.org/rfc/rfc3903.txt>. [RFC: 3903] [09-01-2012].

Pachghare, V. K. (2008). *Cryptography and Information Security*. PHI Learning Pvt. Ltd., India.

Python Software Foundation (2014). Python 2.7 - The Python Standard Library documentation. <http://docs.python.org/2/library/>. [10-04-2015].

Rapid7© (2015). Metasploit documentation. What is Metasploit? <https://help.rapid7.com/metasploit/index.html>. [28-05-2015].

Reingold E. (2009). The Turing Test: Alan Turing and the Imitation Game. <http://psych.utoronto.ca/users/reingold/courses/ai/turing.html>. [10-04-2012].

Riehle D. (2000). Framework Design: A Role Modeling Approach. <http://dirkriehle.com/computer-science/research/dissertation/diss-a4.pdf>. [09-01-2014].

- Rosenberg J., Schulzrinne H., C. G. J. A. (2002). Sip: Session initiation protocol. <https://tools.ietf.org/html/rfc3261#section-21.5.1>. [RFC: 3261] [07-01-2012].
- Shai R. (2011). Pentesting VOIP. <http://www.backtrack-linux.org/wiki/index.php/Pentesting\VOIP>. [07-06-2013].
- Symantec TM® TechNet (2011). Vulnerability Trends. Zero-Day Vulnerabilities. http://securityresponse.symantec.com/threatreport/topic.jsp?id=vulnerability_trends&aid=zero_day_vulnerabilities. [15-06-2012].
- Trapp Flores J. W. (2009). *ICIHONEY: Diseño e Implementación de una Red Trampa en el Instituto de Informática de la Universidad Austral de Chile*. Tesis, Escuela de Ingeniería Civil en Informática, Universidad Austral de Chile Instituto de Informática, Chile.
- Wei Chen (2006). Sip programming for the Java developer. <http://www.javaworld.com/article/2071781/java-web-development/sip-programming-for-the-java-developer.html>. [05-01-2012].
- Wheeler D. (2004). Secure Programming for Linux and Unix howto. <http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/index.html>. [05-02-2012].

Siglas y Acrónimos

AGI Interfaz de Puerta de Enlace de Asterisk

ANI Identificación de Número Automática

AoR Registros de Direcciones

API Interfaz de Programación de Aplicaciones

ARP Protocolo de Resolución de Direcciones

B2BUA Agente de Usuario Espalda a Espalda

AstDb Base Datos de Asterisk

CAS Señalización Asociada a Circuito

CDN Entrega de Numero Llamante

CDR Registros de Detalle de Llamadas

CLI Interfaz de linea de comandos

CPE Equipamiento de Permisos de Usuario

DHCP Protocolo de Configuración de Clientes Dinámico

DMZ Zona Desmilitarizada

DoS Denegación de Servicio

DDoS Denegación de Servicio Distribuida

DNS Sistema de Nombre de Dominio

FBI Oficina Federal de Investigación

FTP Protocolo de Transferencia de Archivos

FW Corta Fuego

GNU GNU's Not Unix!

GW Puerta de Enlace

HTTP Protocolo de Transferencia de Hipertexto

IANA Autoridad de Asignación de Números de Internet

- ICMP** Protocolo de Mensajes de Control de Internet
- IP** Protocolo de Internet
- IPBX** Ramales Privados de Comutación IP
- ISN** Número de Secuencia Inicial
- ISP** Proveedor de Servicios de Internet
- IT** Tecnología de la Información
- LAN** Red de Área Local
- LDAP** Protocolo Ligero/Simplificado de Acceso a Directorios
- MG** Puerta de Enlace de Medios
- MGC** Controlador de Puerta de Enlace de Medios
- MITM** Hombre en el Medio
- NAT** Traductor de Direcciones de Red
- NFS** Sistema de Archivos de Red
- NAPTR** Autoridad Puntero de Nombramiento
- OS** Sistema Operativo
- PBX** Ramales Privados de Comutación
- PSTN** Red de Telefonía Pública Comutada
- QoS** Calidad de Servicio
- RIR** Registros de Internet Regionales
- RTP** Protocolo de Transporte en Tiempo Real
- SDP** Protocolo de Descripción de Sesión
- SER** SIP Express Router
- SIP** Protocolo de Inicio de Sesión
- SPIT** Spam sobre Telefonía IP
- STDOUT** Salida Estándar
- STDERR** Salida de Error Estándar
- TFC** Trabajo Final de Carrera
- TFTP** Protocolo de Transferencia de Archivos Trivial
- TRIP** Telefonía ruteada sobre IP

TGREP Protocolo de Registro de Puerta de Enlace de Telefonía

TCP Protocolo de Control de Transporte

UA Agente de Usuario

UAC Agente de Usuario Cliente

UAS Servidor de Agente de Usuario

UBP Universidad Blas Pascal

UDP Protocolo de Datagrama de Usuario

URI Identificadores de Recursos Uniformes

VM Máquina Virtual

VoIP Voz sobre IP

VPN Red Privada Virtual

WAN Red de Area Extensa

WLAN Redes de Acceso Local Inalámbricas

XML Lenguaje de Marcas Extensibles

XML-RPC Llamada de procedimiento remoto de Lenguaje de Marcas Extensibles

Anexo A

Carta de participación en foros y comunidades de seguridad informática

La misma fue presentada en diferentes sitios entre estos:

<http://www.comptia.org>
<http://www.dragonjar.org/>
<http://www.linkedin.com>
<http://www.hack3r.com/>
<http://www.hackthissite.org/>
<http://www.hackxcrack.es/>
<http://krebsonsecurity.com/2012/05/service-automates-boobytrapping-of-hacked-sites/>
<http://www.security-forums.com>
<http://www.segu-info.com.ar/>
<http://www.soldierx.com/>
<http://www.textfiles.com/hacking/>
<http://www.wilderssecurity.com/>

Otros sitios:

http://www.livingInternet.com/i/ia_hackers_sites.htm
<http://www.protectivehacks.com/hacking-sites.html>
<http://www.eh1infotech.com/hackingtopsites/>

La misma se presenta debajo:

Dear VoIP enthusiasts and Community members,

We're writing in order to invite you all to participate in a research Open Source project, we've been working in since the beginning of this year. This project is sponsored by the Science and Technology Government Department & Blas Pascal University, both organizations from Córdoba province, Argentina.

The link to the project: <http://artemisa.sourceforge.net/>. Our work is related to an Open Source honeypot, named Artemisa, for VoIP networks deploying the SIP protocol. We'll really appreciate the participation of anyone that has interest to play the attacker's role, if possible concentrating in the VoIP service we are exposing. So as to let us capture relevant information related with real attacks. As a result, we'll be able to do an analysis of the efficiency of the platform. Furthermore, a statistical analysis, of all the received attacks, will be performed.

Target SIP extensions:

- 1) *sip:ubp1@iptel.org or sip:229049@iptel.org (Public Free ext)*
- 2) *sip:ubp2@iptel.org or sip:229056@iptel.org (Public Free ext)*
- 3) *sip:2xxx@ubpsipserver.myvnc.com (SER)*
- 4) *sip:metropolitan1@iptel.org or sip:229061@iptel.org (Public Free ext)*
- 5) *sip:metropolitan2@iptel.org or sip:229063@iptel.org (Public Free ext)*
- 6) *sip:1xxx@asteriskserver.myvnc.com (Asterisk)*
- 7) *sip:2xxx@asteriskserver.myvnc.com:5061 (SER)*
- 8) *sip:1xxx@asterisksipserver.myvnc.com (Asterisk)*

We briefly give you a description of the honeypot:

Artemisa is a VoIP/SIP-specific honeypot software designed to connect to a VoIP enterprise

domain as a user-agent back-end in order to detect malicious activity at an early stage. It registers multiple SIP accounts, which do not represent real human subscribers, at one or more VoIP service providers, and wait for incoming attacks. Besides, Artemisa can play a role in the real-time adjustment of the security policies of the enterprise domain where it is deployed (e.g. setting rules in a firewall to ban IPs or in the VoIP PBX to ban caller-IDs).

Thanks for your time, hope you'll actively participate (attack!) the SIP extension presented above. To contact us: artemisa.notifications@gmail.com

Best regards, Exequiel Barrirero / Mauro Villarroel.-

Anexo B

SIP - RFC 3665 [Johnston A., 2003]

Successful Session Establishment



In this scenario, Alice completes a call to Bob directly.

Message Details

F1 INVITE Alice -> Bob

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76s1
To: Bob <sip:bob@biloxi.example.com>
```

```
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 151
```

```
v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F2 180 Ringing Bob -> Alice

```
SIP/2.0 180 Ringing
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76s1
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com;transport=tcp>
Content-Length: 0
```

F3 200 OK Bob -> Alice

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76s1
```

```

To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 147

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000

```

F4 ACK Alice -> Bob

```

ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bd5
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76s1
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 ACK
Content-Length: 0

/* RTP streams are established between Alice and Bob */

/* Bob Hangs Up with Alice. Note that the CSeq is NOT 2, since
Alice and Bob maintain their own independent CSeq counts.
(The INVITE was request 1 generated by Alice, and the BYE is
request 1 generated by Bob) */

```

F5 BYE Bob -> Alice

```

BYE sip:alice@client.atlanta.example.com SIP/2.0
Via: SIP/2.0/TCP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
Max-Forwards: 70
From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76s1
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 BYE
Content-Length: 0

```

F6 200 OK Alice -> Bob

```

SIP/2.0 200 OK
Via: SIP/2.0/TCP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
;received=192.0.2.201
From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76s1
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 BYE
Content-Length: 0

```

[Johnston A., 2003]

Anexo C

Anexo unidad estudio descriptivo estadístico en entornos reales en mesa de prueba (testbeds)

Tablas unidad estudio descriptivo estadístico en entornos reales en mesa de prueba (testbeds)

Tabla 3.14: Cantidad de paquetes SIP total por tipo de mensaje SIP (Artemisa).

| Mensajes SIP Artemisa | Cantidad de Paquetes SIP Metro_Centos_Artemisa | Cantidad de Paquetes SIP Metro_Debian_Artemisa | Cantidad de Paquetes SIP Metro_Ubuntu_Artemisa | Cantidad de Paquetes SIP UBP_Centos_Artemisa | Cantidad de Paquetes SIP UBP_Debian_Artemisa | Cantidad de Paquetes Self-Hosted SIP TOTAL | Cantidad de Paquetes Hosted-Services SIP TOTAL | Cantidad de Paquetes SIP TOTAL |
|--------------------------------|--|--|--|--|--|--|--|--------------------------------|
| SIP 180 Ringing | 1 | 3 | 0 | 0 | 9 | 12 | 1 | 13 |
| SIP 100 Trying | 1 | 3 | 0 | 0 | 9 | 12 | 1 | 13 |
| SIP 199 Informational - Others | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SIP 200 OK | 240320 | 437072 | 455 | 26790 | 496159 | 933686 | 267110 | 1200796 |
| SIP 401 Unauthorized | 45833 | 570951 | 455 | 13506 | 494483 | 1065889 | 59339 | 1125228 |
| SIP 404 Not Found | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SIP 405 Method Not Allowed | 0 | 0 | 0 | 0 | 6 | 6 | 0 | 6 |
| SIP 408 Request Timeout | 28 | 2 | 0 | 1 | 1 | 3 | 29 | 32 |
| SIP 486 Busy Here | 0 | 0 | 0 | 0 | 53 | 53 | 0 | 53 |
| SIP 487 Request Terminated | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SIP 499 Client Error - Others | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SIP 483 Too Many Hops | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SIP 489 Bad Event | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| SIP 488 Not Acceptable Here | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SIP 482 Loop Detected | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SIP 513 Message Too Large | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SIP 500 Server Internal Error | 2349 | 143714 | 0 | 428 | 0 | 143714 | 2777 | 146491 |
| SIP 501 Not Implemented | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SIP 603 Decline | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Tabla 3.15: Cantidad de paquetes SIP total por tipo de método SIP (Artemisa).

| Métodos SIP Artemisa | Cantidad de Paquetes SIP | Cantidad de Paquetes Self-Hosted SIP TOTAL | Cantidad de Paquetes Hosted-Services SIP TOTAL | Cantidad de Paquetes SIP TOTAL |
|----------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--|--|--------------------------------|
| BYE | 2 | 16 | 0 | 0 | 64 | 80 | 2 | 82 |
| NEWMETHOD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUBLISH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| INVITE | 2 | 6 | 0 | 0 | 63 | 69 | 2 | 71 |
| OPTIONS | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PING | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ACK | 2 | 20 | 0 | 0 | 114 | 134 | 2 | 136 |
| CANCEL | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PRACK | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| REGISTER (x100) | 2071,3 | 4435,38 | 9,1 | 808,15 | 3701,05 | 8145,53 | 2879,45 | 11024,98 |
| SUBSCRIBE | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |

Tabla 3.16: Cantidad de llamadas VoIP total (Artemisa).

| Llamadas VoIP SIP Artemisa | Cantidad de Llamadas Self-Hosted | Cantidad de Llamadas Hosted-Services | Cantidad de Llamadas TOTAL |
|----------------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------------------|--------------------------------------|----------------------------|
| Completadas | 1 | 3 | 0 | 0 | 64 | 67 | 1 | 68 |
| Rechazadas | 0 | 0 | 0 | 0 | 53 | 53 | 0 | 53 |

Tabla 3.17: Cantidad de paquetes SIP total por tipo de mensaje SIP (*registrars*).

| Llamadas VoIP SIP Artemisa | Cantidad de Llamadas Self-Hosted | Cantidad de Llamadas Hosted-Services | Cantidad de Llamadas TOTAL |
|----------------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------------------|--------------------------------------|----------------------------|
| Completadas | 1 | 3 | 0 | 0 | 64 | 67 | 1 | 68 |
| Rechazadas | 0 | 0 | 0 | 0 | 53 | 53 | 0 | 53 |

Tabla 3.18: Cantidad de paquetes SIP total por tipo de método SIP (*registrars*).

| Mensajes SIP Registrars | Cantidad de Paquetes SIP Metro_Debian_Asterisk-1.6 | Cantidad de Paquetes SIP Metro_Debian_SER | Cantidad de Paquetes SIP UBP_Debian_SER | Cantidad de Paquetes SIP TOTAL |
|---|--|---|---|--------------------------------|
| SIP 180 Ringing | 0 | 4 | 290 | 294 |
| SIP 100 Trying | 12 | 4 | 13488 | 13504 |
| SIP 199 Informational - Others | 0 | 0 | 6 | 6 |
| SIP 200 OK | 196 | 94599 | 409217 | 504012 |
| SIP 401 Unauthorized | 106794 | 141807 | 410967 | 659568 |
| SIP 404 Not Found | 13177 | 0 | 450 | 13627 |
| SIP 405 Method Not Allowed | 0 | 0 | 4 | 4 |
| SIP 408 Call/Transaction Does Not Exist | 0 | 1 | 21 | 22 |
| SIP 486 Busy Here | 0 | 0 | 600 | 600 |
| SIP 487 Request Terminated | 0 | 0 | 34 | 34 |
| SIP 499 Client Error - Others | 0 | 0 | 1603 | 1603 |
| SIP 483 Too Many Hops | 0 | 0 | 186153 | 186153 |
| SIP 489 Bad Event | 0 | 0 | 1930 | 1930 |
| SIP 488 Not Acceptable Here | 0 | 0 | 25 | 25 |
| SIP 482 Loop Detected | 0 | 0 | 4 | 4 |
| SIP 513 Message Too Large | 0 | 0 | 51909 | 51909 |
| SIP 500 Server Internal Error | 0 | 48763 | 86779 | 135542 |
| SIP 501 Not Implemented | 0 | 0 | 705 | 705 |
| SIP 603 Decline | 0 | 0 | 3 | 3 |

Tabla 3.19: Cantidad de llamadas VoIP SIP total (*registrars*).

| Métodos SIP Registrars | Cantidad de Paquetes SIP TOTAL |
|------------------------|--------------------------|--------------------------|--------------------------|--------------------------------|
| BYE | 0 | 13 | 581 | 594 |
| NEWMETHOD | 0 | 0 | 11 | 11 |
| PUBLISH | 0 | 0 | 1075 | 1075 |
| INVITE | 22 | 1119 | 7792 | 8933 |
| OPTIONS | 116 | 0 | 7087 | 7203 |
| PING | 0 | 0 | 10 | 10 |
| ACK | 13314 | 10 | 16 | 13340 |
| CANCEL | 0 | 0 | 38 | 38 |
| PRACK | 0 | 0 | 16 | 16 |
| REGISTER (x10) | 24377,5 | 28518,6 | 119726,2 | 172622,3 |
| SUBSCRIBE | 0 | 0 | 2261 | 2261 |

Tabla 3.20: Cantidad de paquetes SIP total por tipo de ataque.

| Llamadas VoIP SIP Registrars | Cantidad de Llamadas | Cantidad de Llamadas | Cantidad de Llamadas | Cantidad de Llamadas TOTAL |
|------------------------------|----------------------|----------------------|----------------------|----------------------------|
| Call Setup | 1 | 21 | 0 | 22 |
| Rechazadas | 10 | 0 | 9984 | 9994 |
| Completadas | 0 | 13 | 581 | 594 |

Tabla 3.21: Cantidad de paquetes SIP por tipo ataque (Asterisk Metropolitan).

| Tipo de ataque Asterisk | Cantidad de Paquetes SIP TOTAL |
|--|-----------------------------------|
| ACL error \(permit\ deny\) | 0 |
| Device does not match ACL | 0 |
| No matching peer found | 1166456 |
| failed to authenticate as | 0 |
| Peer is not supposed to register | 0 |
| Username\auth name mismatch | 0 |
| No registration for peer | 0 |
| failed MD5 authentication for | 0 |
| tried to authenticate with nonexistent user | 0 |
| Failed to authenticate user | 0 |
| Received incoming SIP connection from unknown peer | 0 |
| Wrong password | 1970416 |
| Sending fake auth rejection for device | 0 |

Tablas y gráficos Wireshark unidad estudio descriptivo estadístico en entornos reales en mesa de prueba (*testbeds*)

Considerar que solo se adjuntan Tablas y Figuras con diagramas topológicos para la implementación de una de las camas de prueba a modo de referencia. En este caso del UBP testbed, tanto del *registrar* (SER) y de la instancia de Artemisa protegiendo el mismo. Puede generizarse el análisis mediante Wireshark para entender de que manera se generaron las tablas y gráficos de la unidad de análisis estadístico.

Tabla 3.22: Wireshark SIP Statistics with filter (UBP SER Registrar).

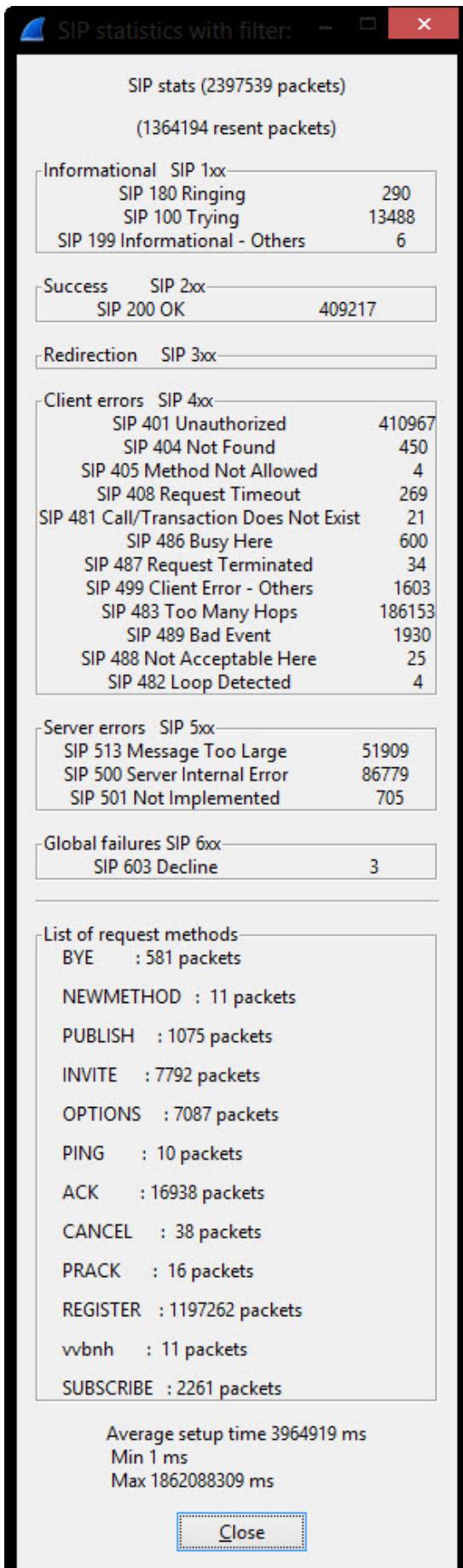


Tabla 3.23: Wireshark VoIP calls (UBP SER *registrar*).

| Detected 1920 VoIP Calls. Selected 1 Call. | | | | | | | | |
|--|--------------|-----------------|---------------------------|---------------------|----------|---------|----------|----------|
| Start Time | Stop Time | Initial Speaker | From | To | Protocol | Packets | State | Comments |
| 8847,256236 | 8847,256391 | 10.10.0.254 | sip:smap@localhost:1234 | sip:smap@localhost | SIP | 2 | REJECTED | |
| 8885,997387 | 8887,350193 | 10.10.0.254 | <sip:smap@localhost:12345 | <sip:smap@localhost | SIP | 49 | REJECTED | |
| 8888,036022 | 8889,313604 | 10.10.0.254 | sip:smap@localhost:1234 | sip:smap@localhost | SIP | 49 | REJECTED | |
| 12747,689541 | 12747,785146 | 10.10.0.254 | <sip:smap@localhost:12345 | <sip:smap@localhost | SIP | 3 | REJECTED | |
| 12749,681899 | 12749,758773 | 10.10.0.254 | sip:smap@localhost:1234 | sip:smap@localhost | SIP | 3 | REJECTED | |
| 14855,576614 | 14855,576787 | 181.164.215.193 | <sip:smap@localhost:12345 | <sip:smap@localhost | SIP | 2 | REJECTED | |

Total: Calls: 1920 Start packets: 0 Completed calls: 581 Rejected calls: 9984

Prepare Filter Flow Player Select All Close

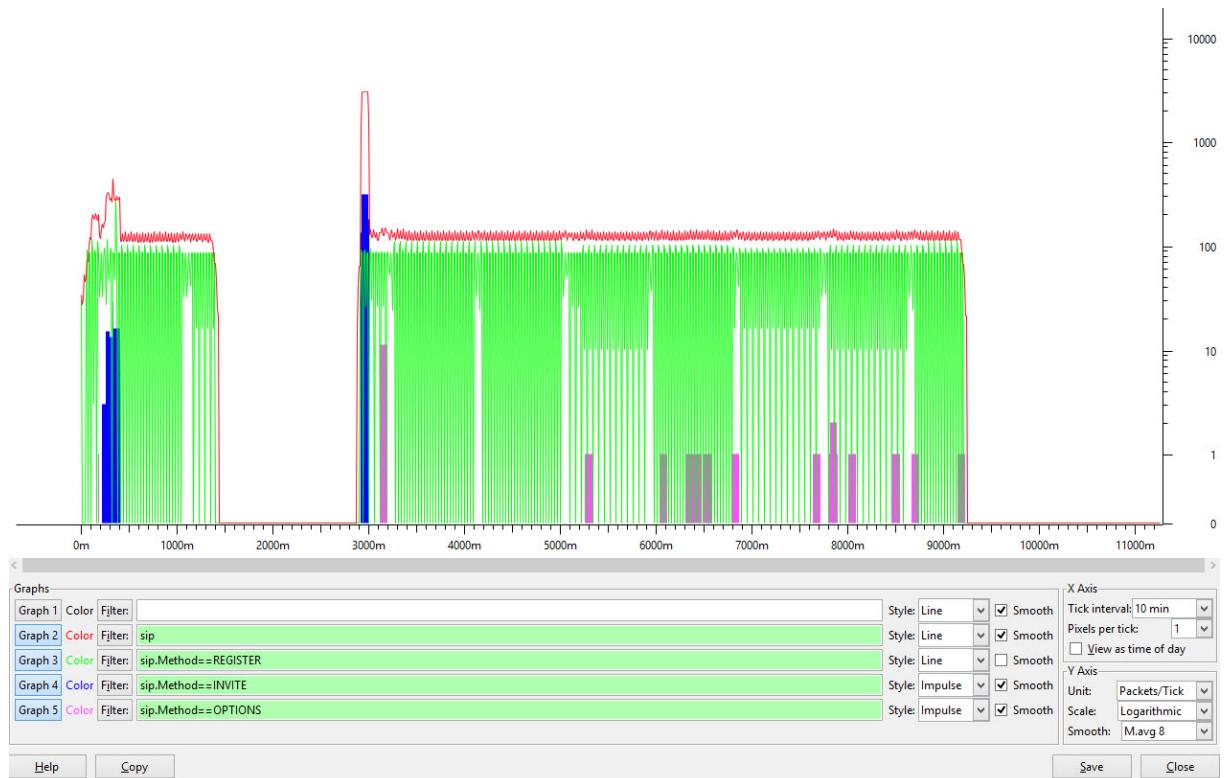


Figura 3.65: Wireshark I/O Graph Statistics w/ SIP filters (UBP SER *registrar*).

Tabla 3.24: Wireshark SIP Statistics with filter (UBP SER registered Artemisa).

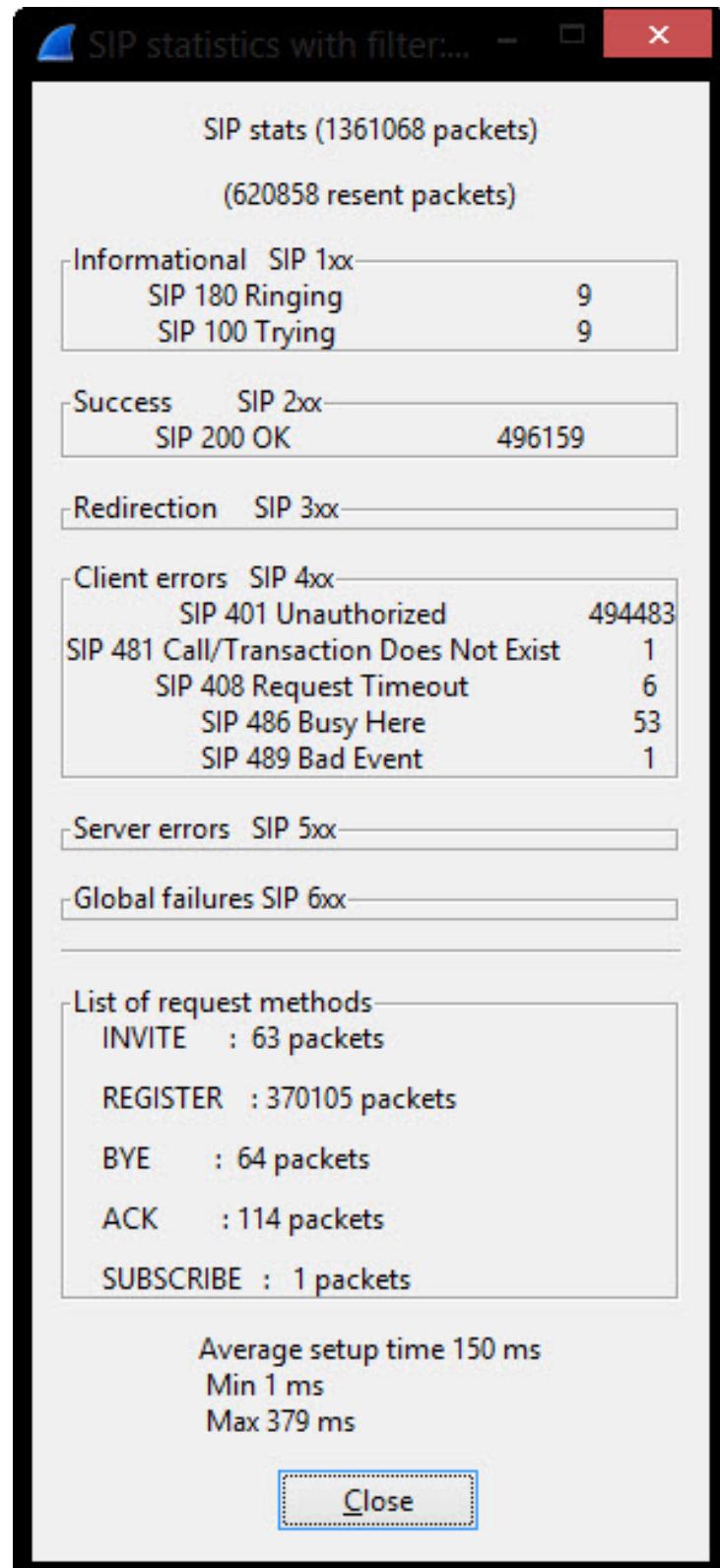


Tabla 3.25: Wireshark VoIP Calls (UBP SER registered Artemisa).

(Untitled) – VoIP Calls

Detected 62 VoIP Calls. Selected 0 Calls.

| Start Time | Stop Time | Initial Speaker | From | To | Protocol | Packets | State | Comments |
|---------------|---------------|-----------------|--|------------------------------|----------|---------|-----------|----------|
| 13712,932725 | 13745,111526 | 10.10.0.240 | "ubp1" <sip:artemisa@iptel.org | <sip:2000@ubpsipserver.n SIP | | 16 | COMPLETED | |
| 14320,735971 | 14352,962571 | 10.10.0.240 | "artemisa@iptel.org" <sip:artemisa@iptel.org | <sip:2000@ubpsipserver.n SIP | | 16 | COMPLETED | |
| 173517,279505 | 173578,558384 | 10.10.0.240 | <sip:10.0.0.182:9 | 2000 <sip:2000@ubpsipsei SIP | | 25 | COMPLETED | |
| 173517,310899 | 173578,558556 | 10.10.0.240 | <sip:10.0.0.182:9 | 2000 <sip:2000@ubpsipsei SIP | | 25 | COMPLETED | |
| 173517,316933 | 173578,558714 | 10.10.0.240 | <sip:10.0.0.182:9 | 2000 <sip:2000@ubpsipsei SIP | | 25 | COMPLETED | |
| 173517,322485 | 173517,366823 | 10.10.0.240 | <sip:10.0.0.182:9 | 2000 <sip:2000@ubpsipsei SIP | | 4 | REJECTED | |
| 173517,329452 | 173517,368444 | 10.10.0.240 | <sip:10.0.0.182:9 | 2000 <sip:2000@ubpsipsei SIP | | 4 | REJECTED | |
| 173517,336358 | 173517,370072 | 10.10.0.240 | <sip:10.0.0.182:9 | 2000 <sip:2000@ubpsipsei SIP | | 4 | REJECTED | |
| 173517,342738 | 173517,371869 | 10.10.0.240 | <sip:10.0.0.182:9 | 2000 <sip:2000@ubpsipsei SIP | | 4 | REJECTED | |

Total: Calls: 62 Start packets: 0 Completed calls: 64 Rejected calls: 53

Prepare Filter Flow Player Select All Close

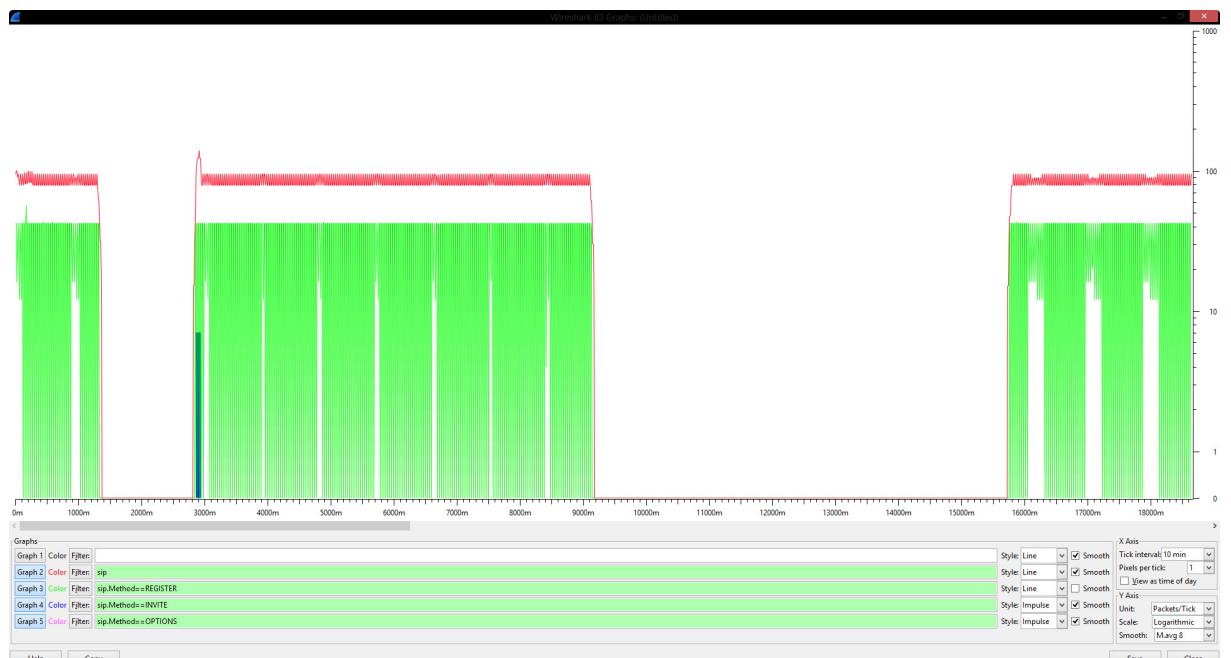


Figura 3.66: Wireshark I/O Graph Statistics w/ SIP filters (UBP SER registered Artemisa).

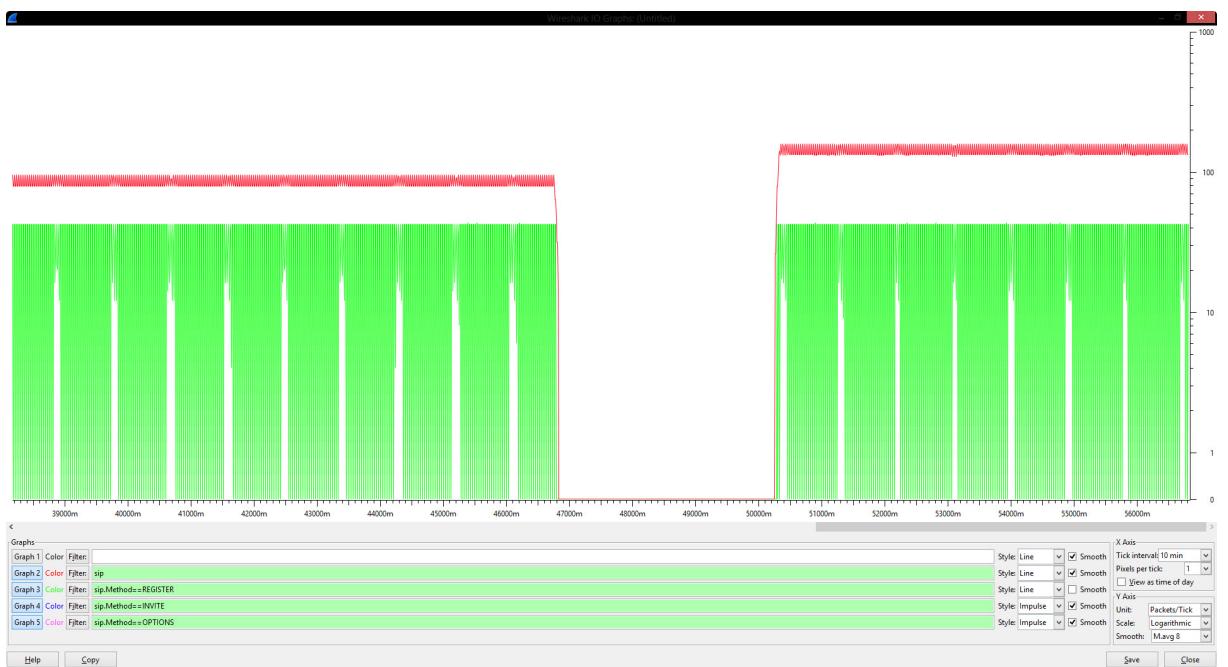


Figura 3.67: Wireshark I/O Graph Statistics w/ SIP filters (UBP SER registered Artemisa).

Anexo D

Configuraciones

En el Anexo subsiguiente se presentan las configuraciones de los componentes fundamentales implementados en los entornos de prueba en cuestión. Las cuales son equivalentes, esto se debe a que los servicios fueron implementados en Máquina Virtual (VM) como se detalla a continuación:

Firewall Iptables

root@firewall# iptables-save

```
# Generated by iptables-save v1.4.8 on Mon Jul 30 17:25:24 2012
*nat
:PREROUTING ACCEPT [979:290313]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [1035:61633]
-A POSTROUTING -o eth0 -j MASQUERADE
COMMIT
# Completed on Mon Jul 30 17:25:24 2012
# Generated by iptables-save v1.4.8 on Mon Jul 30 17:25:24 2012
*mangle
:PREROUTING ACCEPT [5728:3386520]
:INPUT ACCEPT [4774:3104304]
:FORWARD ACCEPT [30:1620]
:OUTPUT ACCEPT [5258:529761]
:POSTROUTING ACCEPT [4872:502687]
COMMIT
# Completed on Mon Jul 30 17:25:24 2012
# Generated by iptables-save v1.4.8 on Mon Jul 30 17:25:24 2012
*filter
:INPUT DROP [24:7872]
:FORWARD DROP [30:1620]
:OUTPUT DROP [0:0]
:INBOUND - [0:0]
:LOG_FILTER - [0:0]
:LSI - [0:0]
:LSO - [0:0]
:OUTBOUND - [0:0]
-A INPUT -s 10.10.0.1/32 -p tcp -m tcp ! --tcp-flags FIN,SYN,RST,ACK SYN -j
ACCEPT
-A INPUT -s 10.10.0.1/32 -p udp -j ACCEPT
-A INPUT -s 10.0.0.1/32 -p tcp -m tcp ! --tcp-flags FIN,SYN,RST,ACK SYN -j
ACCEPT
-A INPUT -s 10.0.0.1/32 -p udp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp -m limit --limit 10/sec -j ACCEPT
-A INPUT -d 255.255.255.255/32 -i eth0 -j DROP
-A INPUT -d 10.10.0.255/32 -j DROP
-A INPUT -s 224.0.0.0/8 -j DROP
-A INPUT -d 224.0.0.0/8 -j DROP
-A INPUT -s 255.255.255.255/32 -j DROP
-A INPUT -d 0.0.0.0/32 -j DROP
-A INPUT -m state --state INVALID -j DROP
-A INPUT -f -m limit --limit 10/min -j LSI
-A INPUT -i eth0 -j INBOUND
-A INPUT -d 10.10.0.254/32 -i eth0 -j INBOUND
-A INPUT -d 10.10.0.255/32 -i eth0 -j INBOUND
-A INPUT -d 20.20.0.254/32 -i eth1 -j INBOUND
-A INPUT -d 10.10.0.254/32 -i eth1 -j INBOUND
-A INPUT -d 20.20.0.255/32 -i eth1 -j INBOUND
-A INPUT -j LOG_FILTER
-A INPUT -j LOG --log-prefix "Unknown Input" --log-level 6
-A FORWARD -p icmp -m limit --limit 10/sec -j ACCEPT
-A FORWARD -p tcp -m tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu
-A FORWARD -i eth0 -j OUTBOUND
-A FORWARD -i eth1 -j OUTBOUND
-A FORWARD -d 10.10.0.0/24 -p tcp -m state --state RELATED,ESTABLISHED -j
ACCEPT
-A FORWARD -d 10.10.0.0/24 -p udp -m state --state RELATED,ESTABLISHED -j
ACCEPT
```

```

-A FORWARD -d 20.20.0.0/24 -p tcp -m state --state RELATED,ESTABLISHED -j
ACCEPT
-A FORWARD -d 20.20.0.0/24 -p udp -m state --state RELATED,ESTABLISHED -j
ACCEPT
-A FORWARD -j LOG_FILTER
-A FORWARD -j LOG --log-prefix "Unknown Forward" --log-level 6
-A OUTPUT -s 10.10.0.254/32 -d 10.10.0.1/32 -p tcp -m tcp --dport 53 -j ACCEPT
-A OUTPUT -s 10.10.0.254/32 -d 10.10.0.1/32 -p udp -m udp --dport 53 -j ACCEPT
-A OUTPUT -s 10.10.0.254/32 -d 10.0.0.1/32 -p tcp -m tcp --dport 53 -j ACCEPT
-A OUTPUT -s 10.10.0.254/32 -d 10.0.0.1/32 -p udp -m udp --dport 53 -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -s 224.0.0.0/8 -j DROP
-A OUTPUT -d 224.0.0.0/8 -j DROP
-A OUTPUT -s 255.255.255.255/32 -j DROP
-A OUTPUT -d 0.0.0.0/32 -j DROP
-A OUTPUT -m state --state INVALID -j DROP
-A OUTPUT -o eth0 -j OUTBOUND
-A OUTPUT -o eth1 -j OUTBOUND
-A OUTPUT -j LOG_FILTER
-A OUTPUT -j LOG --log-prefix "Unknown Output" --log-level 6
-A INBOUND -p tcp -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INBOUND -p udp -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INBOUND -s 10.10.0.0/24 -j ACCEPT
-A INBOUND -s 20.20.0.0/24 -j ACCEPT
-A INBOUND -p tcp -m tcp --dport 22 -j ACCEPT
-A INBOUND -p udp -m udp --dport 22 -j ACCEPT
-A INBOUND -p tcp -m tcp --dport 5060 -j ACCEPT
-A INBOUND -p udp -m udp --dport 5060 -j ACCEPT
-A INBOUND -p tcp -m tcp --dport 10001:20000 -j ACCEPT
-A INBOUND -p udp -m udp --dport 10001:20000 -j ACCEPT
#-A INBOUND -p tcp -m tcp --dport 80 -j ACCEPT
#-A INBOUND -p udp -m udp --dport 80 -j ACCEPT
-A INBOUND -j LSI
-A LSI -j LOG_FILTER
-A LSI -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK SYN -m limit --limit 1/sec -j
LOG
--log-prefix "Inbound" --log-level 6
-A LSI -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK SYN -j DROP
-A LSI -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK RST -m limit --limit 1/sec -j
LOG
--log-prefix "Inbound" --log-level 6
-A LSI -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK RST -j DROP
-A LSI -p icmp -m icmp --icmp-type 8 -m limit --limit 1/sec -j LOG --log-prefix
"Inbound" --log-level 6
-A LSI -p icmp -m icmp --icmp-type 8 -j DROP
-A LSI -m limit --limit 5/sec -j LOG --log-prefix "Inbound" --log-level 6
-A LSI -j DROP
-A LSO -j LOG_FILTER
-A LSO -m limit --limit 5/sec -j LOG --log-prefix "Outbound" --log-level 6
-A LSO -j REJECT --reject-with icmp-port-unreachable
-A OUTBOUND -p icmp -j ACCEPT
-A OUTBOUND -p tcp -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTBOUND -p udp -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTBOUND -j ACCEPT
COMMIT
# Completed on Mon Jul 30 17:25:24 2012

```

root@firewall# iptables -L

| Chain | INPUT (policy DROP) | target | prot | opt | origin | destination | |
|-------|---------------------|---------|------|-----|-----------|-----------------|-------------------|
| | | ACCEPT | tcp | -- | 10.10.0.1 | anywhere | tcp |
| | | ACCEPT | udp | -- | 10.10.0.1 | anywhere | |
| | | ACCEPT | tcp | -- | 10.0.0.1 | anywhere | tcp |
| | | ACCEPT | udp | -- | 10.0.0.1 | anywhere | |
| | | ACCEPT | all | -- | anywhere | anywhere | |
| | | ACCEPT | icmp | -- | anywhere | anywhere | limit: avg 10/sec |
| | | burst 5 | | | | | |
| | | DROP | all | -- | anywhere | 255.255.255.255 | |
| | | DROP | all | -- | anywhere | 10.10.0.255 | |

```

DROP      all  --  base-address.mcast.net/8 anywhere
DROP      all  --  anywhere          base-address.mcast.net/8
DROP      all  --  255.255.255.255 anywhere
DROP      all  --  anywhere          default
DROP      all  --  anywhere          anywhere        state INVALID
LSI       all  -f  anywhere          anywhere        limit: avg 10/min
burst 5
INBOUND   all  --  anywhere          anywhere
INBOUND   all  --  anywhere          10.10.0.254
INBOUND   all  --  anywhere          10.10.0.255
INBOUND   all  --  anywhere          20.20.0.254
INBOUND   all  --  anywhere          10.10.0.254
INBOUND   all  --  anywhere          20.20.0.255
LOG_FILTER all  --  anywhere          anywhere
LOG       all  --  anywhere          anywhere        LOG level info
prefix
'Unknown Input'

Chain FORWARD (policy DROP)
target    prot opt source          destination
ACCEPT   icmp -- anywhere        anywhere        limit: avg 10/sec
burst 5
TCPMSS   tcp  -- anywhere        anywhere
flags:SYN,RST/SYN
TCPMSS clamp to PMTU
OUTBOUND all  -- anywhere        anywhere
OUTBOUND all  -- anywhere        anywhere
ACCEPT   tcp  -- anywhere        10.10.0.0/24     state
RELATED,ESTABLISHED
ACCEPT   udp  -- anywhere        10.10.0.0/24     state
RELATED,ESTABLISHED
ACCEPT   tcp  -- anywhere        20.20.0.0/24     state
RELATED,ESTABLISHED
ACCEPT   udp  -- anywhere        20.20.0.0/24     state
RELATED,ESTABLISHED
LOG_FILTER all  -- anywhere        anywhere
LOG      all  -- anywhere        anywhere        LOG level info
prefix
'Unknown Forward'

Chain OUTPUT (policy DROP)
target    prot opt source          destination
ACCEPT   tcp  -- 10.10.0.254      10.10.0.1      tcp dpt:domain
ACCEPT   udp  -- 10.10.0.254      10.10.0.1      udp dpt:domain
ACCEPT   tcp  -- 10.10.0.254      10.0.0.1       tcp dpt:domain
ACCEPT   udp  -- 10.10.0.254      10.0.0.1       udp dpt:domain
ACCEPT   all  -- anywhere        anywhere
DROP     all  -- base-address.mcast.net/8 anywhere
DROP     all  -- anywhere        base-address.mcast.net/8
DROP     all  -- 255.255.255.255 anywhere
DROP     all  -- anywhere        default
DROP     all  -- anywhere        anywhere        state INVALID
OUTBOUND all  -- anywhere        anywhere
OUTBOUND all  -- anywhere        anywhere
LOG_FILTER all  -- anywhere        anywhere
LOG      all  -- anywhere        anywhere        LOG level info
prefix
'Unknown Output'

Chain INBOUND (6 references)
target    prot opt source          destination
ACCEPT   tcp  -- anywhere        anywhere        state
RELATED,ESTABLISHED
ACCEPT   udp  -- anywhere        anywhere
RELATED,ESTABLISHED
ACCEPT   all  -- 10.10.0.0/24    anywhere
ACCEPT   all  -- 20.20.0.0/24    anywhere
ACCEPT   tcp  -- anywhere        anywhere
ACCEPT   udp  -- anywhere        anywhere
ACCEPT   tcp  -- anywhere        anywhere
ACCEPT   udp  -- anywhere        anywhere
ACCEPT   tcp  -- anywhere        anywhere
ACCEPT   udp  -- anywhere        anywhere

```

```

LSI      all  --  anywhere          anywhere
Chain LOG_FILTER (5 references)
target    prot opt source          destination
Chain LSI (2 references)
target    prot opt source          destination
LOG_FILTER all  --  anywhere      anywhere
LOG      tcp  --  anywhere        anywhere      tcp
flags:FIN,SYN,RST,ACK/SYN limit: avg 1/sec burst 5 LOG level info prefix
'Inbound
'
DROP      tcp  --  anywhere        anywhere      tcp
flags:FIN,SYN,RST,ACK/SYN
LOG      tcp  --  anywhere        anywhere      tcp
flags:FIN,SYN,RST,ACK/RST limit: avg 1/sec burst 5 LOG level info prefix
'Inbound
'
DROP      tcp  --  anywhere        anywhere      tcp
flags:FIN,SYN,RST,ACK/RST
LOG      icmp --  anywhere       anywhere      icmp echo-request
limit: avg 1/sec burst 5 LOG level info prefix 'Inbound '
DROP      icmp --  anywhere       anywhere      icmp echo-request
LOG      all  --  anywhere        anywhere      limit: avg 5/sec
burst 5 LOG level info prefix 'Inbound '
DROP      all  --  anywhere        anywhere

Chain LSO (0 references)
target    prot opt source          destination
LOG_FILTER all  --  anywhere      anywhere
LOG      all  --  anywhere        anywhere      limit: avg 5/sec
burst 5 LOG level info prefix 'Outbound '
REJECT    all  --  anywhere       anywhere      reject-with
icmp-port-unreachable

Chain OUTBOUND (4 references)
target    prot opt source          destination
ACCEPT    icmp --  anywhere       anywhere
ACCEPT    tcp  --  anywhere       anywhere      state
RELATED,ESTABLISHED
ACCEPT    udp  --  anywhere       anywhere      state
RELATED,ESTABLISHED
ACCEPT    all  --  anywhere       anywhere

```

Sip Express Router (SER)

ser.cfg

```
#  
# $Id: ser.cfg,v 1.25.2.1 2005/02/18 14:30:44 andrei Exp $  
#  
# simple quick-start config script  
#  
# ----- global configuration parameters -----  
  
debug=3          # debug level (cmd line: -ddddddddd)  
fork=yes  
log_stderror=no # (cmd line: -E)  
  
listen=10.10.0.240  
listen=127.0.0.1  
  
# hostname matching an alias will satisfy the condition uri==myself".  
alias=ubpsipserver.myvnc.com  
alias=10.10.0.240  
  
# dns - Uses dns to check if it is necessary to add a "received=" field  
# to a via. Default is no.  
# rev_dns - Same as dns but use reverse DNS.  
  
dns=no  
rev_dns=no  
  
port=5060  
children=4  
  
# check_via - Turn on or off Via host checking when forwarding replies.  
# Default is no. arcane. looks for discrepancy between name and  
# ip address when forwarding replies.  
  
check_via=yes  
  
# syn_branch - Shall the server use stateful synonym branches? It is  
# faster but not reboot-safe. Default is yes.  
  
syn_branch=yes  
  
# memlog - Debugging level for final memory statistics report. Default  
# is L_DBG -- memory statistics are dumped only if debug is set high.  
  
memlog=3  
  
# sip_warning - Should replies include extensive warnings? By default  
# yes, it is good for trouble-shooting.  
  
sip_warning=yes  
  
# fifo - FIFO special file pathname  
  
fifo="/tmp/ser_fifo"  
  
# server_signature - Should locally-generated messages include server's  
# signature? By default yes, it is good for trouble-shooting.  
  
server_signature=yes  
  
# reply_to_via - A hint to reply modules whether they should send reply  
# to IP advertised in Via. Turned off by default, which means that  
# replies are sent to IP address from which requests came.  
  
reply_to_via=no  
  
# ----- module loading -----  
  
# Uncomment this if you want to use SQL database
```

```

loadmodule "/usr/local/lib/ser/modules/mysql.so"

loadmodule "/usr/local/lib/ser/modules/sl.so"
loadmodule "/usr/local/lib/ser/modules/tm.so"
loadmodule "/usr/local/lib/ser/modules/rr.so"
loadmodule "/usr/local/lib/ser/modules/maxfwd.so"
loadmodule "/usr/local/lib/ser/modules/usrloc.so"
loadmodule "/usr/local/lib/ser/modules/registrar.so"
loadmodule "/usr/local/lib/ser/modules/textops.so"

# Uncomment this if you want digest authentication
# mysql.so must be loaded !
loadmodule "/usr/local/lib/ser/modules/auth.so"
loadmodule "/usr/local/lib/ser/modules/auth_db.so"

# ----- setting module-specific parameters -----

# -- usrloc params --
#modparam("usrloc", "db_mode", 0)

# Uncomment this if you want to use SQL database
# for persistent storage and comment the previous line
modparam("usrloc", "db_mode", 2)

# -- auth params --
# Uncomment if you are using auth module
#
modparam("auth_db", "calculate_ha1", yes)
#
# If you set "calculate_ha1" parameter to yes (which true in this
# config),
# uncomment also the following parameter
#
modparam("auth_db", "password_column", "password")

# -- rr params --
# add value to ;lr param to make some broken UAs happy
modparam("rr", "enable_full_lr", 1)

# database location
modparam("usrloc", "db_url", "mysql://ser:heslo@localhost/ser")

# ----- auth parameters

# database location
modparam("auth_db", "db_url", "mysql://ser:heslo@localhost/ser")

# ----- request routing logic -----

# main routing logic

route{

# initial sanity checks -- messages with
# max_forwards==0, or excessively long requests
if (!mf_process_maxfwd_header("10")) {
sl_send_reply("483", "Too Many Hops");
break;
};
if (msg:len >= 2048 ) {
sl_send_reply("513", "Message too big");
break;
};

# we record-route all messages -- to make sure that
# subsequent messages will go through our proxy; that's
# particularly good if upstream and downstream entities
# use different transport protocol
if (!method=="REGISTER") record_route();

# subsequent messages withing a dialog should take the

```

```

# path determined by record-routing
if (loose_route()) {
# mark routing logic in request
append_hf("P-hint: rr-enforced\r\n");
route(1);
break;
};

if (!uri==myself) {
# mark routing logic in request
append_hf("P-hint: outbound\r\n");
route(1);
break;
};

# if the request is for other domain use UsrLoc
# (in case, it does not work, use the following command
# with proper names and addresses in it)
if (uri==myself) {

if (method=="REGISTER") {

# Uncomment this if you want to use digest authentication
if (!www_authorize("localhost", "subscriber")) {
www_challenge("localhost", "0");
break;
};

save("location");
break;
};

lookup("aliases");
if (!uri==myself) {
append_hf("P-hint: outbound alias\r\n");
route(1);
break;
};

# native SIP destinations are handled using our USRLOC
DB
if (!lookup("location")) {
sl_send_reply("404", "Not Found");
break;
};
append_hf("P-hint: usrloc applied\r\n");
route(1);
}

route[1]
{
# send it out now; use stateful forwarding as it works reliably
# even for UDP2TCP
if (!t_relay()) {
sl_reply_error();
};
}

```

Asterisk PBX

extension.conf

```
[globals]
[general]
autofallthrough=yes
clearglobalvars=no

[default]

exten => s,1,Verbose(1,Unrouted call handler)
exten => s,n,Answer()
exten => s,n,Wait(1)
exten => s,n,Playback(pbx-invalidpark)
include => internal
exten => s,n,WaitExten(5)
exten => s,n,Hangup()

exten => i,1,Verbose(1,Unrouted call handler)
exten => i,n,Answer()
exten => i,n,Playback(pbx-invalidpark)
exten => i,n,WaitExten()
exten => i,n,Hangup()

[internal]

exten => 1000,1,Verbose(3,Extension 1001)
exten => 1000,n,Playback(queue-thankyou)
exten => 1000,n,Dial(SIP/phone_0,20,Tt)
exten => 1000,n,Voicemail(1001)
exten => 1000,n,Hangup()

exten => 1001,1,Verbose(1,Extension 1001)
exten => 1001,n,Playback(queue-thankyou)
exten => 1001,n,Dial(SIP/phone_1,20,Tt)
exten => 1001,n,Voicemail(1001)
exten => 1001,n,Hangup()

exten => 1002,1,Verbose(3,Extension 1002)
exten => 1002,n,Playback(queue-thankyou)
exten => 1002,n,Dial(SIP/phone_2,20,Tt)
exten => 1002,n,Voicemail(1002)
exten => 1002,n,Hangup()

exten => 1003,1,Verbose(3,Extension 1003)
exten => 1003,n,Wait(1)
exten => 1003,n,Playback(queue-thankyou)
exten => 1003,n,Dial(SIP/phone_3,20,Tt)
exten => 1003,n,Voicemail(1003)
exten => 1003,n,Hangup()

...
exten => 1015,1,Verbose(1,Extension 1015)
exten => 1015,n,Playback(queue-thankyou)
exten => 1015,n,Dial(SIP/phone_15,20,Tt)
exten => 1015,n,Voicemail(1015)
exten => 1015,n,Hangup()

exten => echo,1,Playback(demo-echotest)
exten => echo,n,Echo
exten => echo,n,Playback(demo-echodone)
exten => echo,n,Hangup

exten => *555,1,VoicemailMain
exten => _X.,1,Goto(default,s,1)
```

rtp.conf

```
;  
; RTP Configuration  
;  
[general]  
;  
; RTP start and RTP end configure start and end addresses  
;  
; Defaults are rtpstart=5000 and rt pend=31000  
;  
rtpstart=10001  
rt pend=20000  
;  
; Whether to enable or disable UDP checksums on RTP traffic  
; RTP  
; rtpchecksums=no  
;  
; The amount of time a DTMF digit with no 'end' marker should be  
; allowed to continue (in 'samples', 1/8000 of a second)  
;  
; dtmftimeout=3000  
; rtcpinterval = 5000 ; Milliseconds between rtcp reports  
;(min 500, max 60000, default 5000)  
;  
; Enable strict RTP protection. This will drop RTP packets that  
; do not come from the source of the RTP stream. This option is  
; disabled by default.  
  
; strictrtp=yes
```

sip.conf

```
[general]  
  
context=default  
  
allowoverlap=no  
  
bindport=5060  
  
bindaddr=10.10.0.254  
  
externip=asteriskserver.myvnc.com  
;externip=asterissipkserver.myvnc.com  
;externip=UBPsipserver.myvnc.com  
  
localnet=192.168.18.0/255.255.255.0  
;localnet=10.10.0.0/255.255.255.0  
;localnet=172.16.0.0/255.255.255.0  
  
srvlookup=yes ; Activates the DNS SRV Server  
  
[phone_0]  
type=friend  
secret=1000  
username=phone_0  
callerid=phone_0 <1000>  
qualify=no  
nat=yes  
context=phones  
host=dynamic  
careinvite=no  
context=internal  
mailbox=1000@default  
  
[phone_1]  
type=friend  
secret=1001  
username=phone_1; suele ser el mismo que el titulo de la seccion  
callerid=phone_1 <1001>  
qualify=no
```

```

nat=yes
context=phones
host=dynamic
canreinvite=no
context=internal
mailbox=1001@default

[phone_2]
type=friend
secret=1002
username=phone_2
callerid=phone_2 <1002>
qualify=no
nat=yes
context=phones
host=dynamic
careinvite=no
context=internal
mailbox=1002@default

[phone_3]
type=friend
secret=1003
username=phone_3
callerid=phone_3 <1003>
qualify=no
nat=yes
context=phones
host=dynamic
careinvite=no
context=internal
mailbox=1003@default

...
[phone_15]
type=friend
secret=1015
username=phone_15
callerid=phone_15 <1015>
qualify=no
nat=yes
context=phones
host=dynamic
careinvite=no
context=internal
mailbox=1015@default

```

voicemail.conf

```

[general]
attach=yes
format=wav
maxmessage=180
minmessage=3
saycid=yes
maxlogins=3
cidinternalcontexts=house_local,house_toll,house_admin
[default]
1000 => 1000,Phone_0,artemisa.notifications@gmail.com
1001 => 1001,Phone_1,artemisa.notifications@gmail.com
1002 => 1002,Phone_2,artemisa.notifications@gmail.com
1003 => 1003,Phone_3,artemisa.notifications@gmail.com

```

```

1004 => 1004,Phone_4,artemisa.notifications@gmail.com
1005 => 1005,Phone_5,artemisa.notifications@gmail.com
1006 => 1006,Phone_6,artemisa.notifications@gmail.com
1007 => 1007,Phone_7,artemisa.notifications@gmail.com
1008 => 1008,Phone_8,artemisa.notifications@gmail.com
1009 => 1009,Phone_9,artemisa.notifications@gmail.com
1010 => 1010,Phone_10,artemisa.notifications@gmail.com
1011 => 1011,Phone_11,artemisa.notifications@gmail.com
1012 => 1012,Phone_12,artemisa.notifications@gmail.com
1013 => 1013,Phone_13,artemisa.notifications@gmail.com
1014 => 1014,Phone_14,artemisa.notifications@gmail.com
1015 => 1015,Phone_15,artemisa.notifications@gmail.com

```

Asterisk CLI

```

debiannv*CLI> sip show settings

Global Settings:
-----
    UDP SIP Port:          5060
    UDP Bindaddress:        10.10.0.245
    TCP SIP Port:          Disabled
    TLS SIP Port:          Disabled
    Videosupport:          No
    Textsupport:            No
    Ignore SDP sess. ver.: No
    AutoCreate Peer:        No
    Match Auth Username:   No
    Allow unknown access:  Yes
    Allow subscriptions:   Yes
    Allow overlap dialing: No
    Allow promsic. redir:  No
    Enable call counters:  No
    SIP domain support:   No
    Realm. auth:           No
    Our auth realm:        asterisk
    Call to non-local dom.: Yes
    URI user is phone no: No
    Always auth rejects: No
    Direct RTP setup:     No
    User Agent:            Asterisk PBX 1.6.2.9-2+squeeze5
    SDP Session Name:      Asterisk PBX 1.6.2.9-2+squeeze5
    SDP Owner Name:        root
    Reg. context:          (not set)
    Regexten on Qualify:  No
    Caller ID:             asterisk
    From: Domain:
    Record SIP history:   Off
    Call Events:           Off
    Auth. Failure Events: Off
    T.38 support:          No
    T.38 EC mode:          Unknown
    T.38 MaxDtgrm:         -1
    SIP realtime:          Disabled
    Qualify Freq :         60000 ms

Network QoS Settings:
-----
    IP ToS SIP:            CS0
    IP ToS RTP audio:      CS0
    IP ToS RTP video:      CS0
    IP ToS RTP text:       CS0
    802.1p CoS SIP:        4
    802.1p CoS RTP audio:  5
    802.1p CoS RTP video:  6
    802.1p CoS RTP text:   5
    Jitterbuffer enabled:  No
    Jitterbuffer forced:   No
    Jitterbuffer max size: -1
    Jitterbuffer resync:   -1
    Jitterbuffer impl:     Jitterbuffer
    Jitterbuffer log:      No

```

```

Network Settings:
-----
SIP address remapping: Disabled, externip is 0.0.0.0
Externhost: <none>
Externip: 0.0.0.0:5060
Externrefresh: 10
Internal IP: 10.10.0.245:5060
Localnet: 10.10.0.0/255.255.255.0
STUN server: 0.0.0.0:0

Global Signalling Settings:
-----
Codecs: 0x8000e (gsm|ulaw|alaw|h263)
Codec Order: none
Relax DTMF: No
RFC2833 Compensation: No
Compact SIP headers: No
RTP Keepalive: 0 (Disabled)
RTP Timeout: 0 (Disabled)
RTP Hold Timeout: 0 (Disabled)
MWI NOTIFY mime type: application/simple-message-summary
DNS SRV lookup: Yes
Pedantic SIP support: No
Reg. min duration: 60 secs
Reg. max duration: 3600 secs
Reg. default duration: 120 secs
Outbound reg. timeout: 20 secs
Outbound reg. attempts: 0
Notify ringing state: Yes
Include CID: No
Notify hold state: No
SIP Transfer mode: open
Max Call Bitrate: 384 kbps
Auto-Framing: No
Outb. proxy: <not set>
Session Timers: Accept
Session Refresher: uas
Session Expires: 1800 secs
Session Min-SE: 90 secs
Timer T1: 500
Timer T1 minimum: 100
Timer B: 32000
No premature media: Yes

Default Settings:
-----
Allowed transports: UDP
Outbound transport: UDP
Context: default
Nat: Always
DTMF: rfc2833
Qualify: 0
Use ClientCode: No
Progress inband: Never
Language:
MOH Interpret: default
MOH Suggest:
Voice Mail Extension: asterisk

```

Artemisa honeypot

Librerías y Dependencias

Se pueden encontrar las librerías mandatorias en la documentación oficial del aplicativo artemisa [Do Carmo R., 2011b]. Debajo se detallan todas las utilizadas en la distribución de Linux Debian aplicadas para el TFC. Recordar que estas son fundamentales para poder ejecutar Artemisa con todas sus funciones. Como por ejemplo para que detecte la *sound card* en Artemisa y puede reproducir audio al responder una *call*.

```
delivery@debianDUH:~$ dpkg -l | egrep -i  
'alsa|dnsutils|gcc|jwhois|make|nmap|pjlib|pjstun|python|ssl|tcpdump|twinkle'|  
grep  
-v grep  
ii  alsa-base                      1.0.23+dfsg-2          ALSA  
  driver configuration files  
ii  alsa-oss                        1.0.17-4            ALSA  
  wrapper for OSS applications  
ii  alsa-utils                      1.0.23-3  
  Utilities  
    for configuring and using ALSA  
ii  dnsutils                         1:9.7.3.dfsg-1~squeeze5   Clients  
  provided with BIND  
ii  gcc                             4:4.4.5-1           The GNU  
C  
  compiler  
ii  gcc-4.3                          4.3.5-4            The GNU  
C  
  compiler  
ii  gcc-4.3-base                     4.3.5-4            The GNU  
Compiler Collection (base package)  
ii  gcc-4.4                          4.4.5-8            The GNU  
C  
  compiler  
ii  gcc-4.4-base                     4.4.5-8            The GNU  
Compiler Collection (base package)  
ii  gstreamer0.10-alsa                0.10.30-1          GStreamer  
  plugin for ALSA  
ii  jwhois                           4.0-1              Improved  
  Caching Whois client  
ii  libasound2                       1.0.23-2.1          shared  
  library for ALSA applications  
ii  libcurl3                          7.21.0-2.1+squeeze2  
  Multi-protocol file transfer library (OpenSSL)  
ii  libflac8                           1.2.1-2+b1          Free  
  Lossless Audio Codec - runtime C library  
ii  libgcc1                           1:4.4.5-8           GCC  
  support library  
ii  libgomp1                          4.4.5-8            GCC  
OpenMP  
  (GOMP) support library  
ii  libpython2.6                      2.6.6-8+b1          Shared  
  Python runtime library (version 2.6)  
ii  libsdll1.2debian-alsa             1.2.14-6.1          Simple  
  DirectMedia Layer (with X11 and ALSA options)  
ii  libssl-dev                         0.9.8o-4squeeze13  SSL  
  development libraries, header files and documentation  
ii  libssl0.9.8                        0.9.8o-4squeeze13  SSL  
shared  
  libraries  
ii  libwvpack1                        4.60.1-1           an audio  
  codec (lossy and lossless) - library  
ii  linux-sound-base                  1.0.23+dfsg-2          base  
  package for ALSA and OSS sound systems  
ii  make                            3.81-8             An  
utility  
for Directing compilation.  
ii  nmap                            5.00-3              The
```

```

Network Mapper
ii  openssl                      0.9.8o-4squeeze13          Secure
Socket Layer (SSL) binary and related cryptographic tools
ii  python                       2.6.6-3+squeeze7
interactive high-level object-oriented language (default version)
ii  python-all                   2.6.6-3+squeeze7
depending on all supported Python runtime versions
ii  python-all-dev               2.6.6-3+squeeze7
depending on all supported Python development packages
ii  python-apt                  0.7.100.1+squeeze1
interface to libapt-pkg
ii  python-apt-common            0.7.100.1+squeeze1
interface to libapt-pkg (locales)
ii  python-cairo                1.8.8-1+b1
bindings for the Cairo vector graphics library
ii  python-central              0.6.16+nmul
and build utility for Python packages
ii  python-dbus                 0.83.1-1
interprocess messaging system (Python interface)
ii  python-dev                  2.6.6-3+squeeze7
files and a static library for Python (default)
ii  python-gconf                2.28.1-1
bindings for the GConf configuration database system
ii  python-glade2               2.17.0-4
bindings: Glade support
ii  python-gmenu                2.30.3-1
implementation of the freedesktop menu specification for GNOME
ii  python-gnome2               2.28.1-1
bindings for the GNOME desktop environment
ii  python-gobject              2.21.4+is.2.21.3-1
bindings for the GObject library
ii  python-gtk2                 2.17.0-4
bindings for the GTK+ widget set
ii  python-gtksourceview2        2.10.1-1
bindings for the GtkSourceView widget
ii  python-libxml2              2.7.8.dfsg-2+squeeze4
bindings for the GNOME XML library
ii  python-minimal              2.6.6-3+squeeze7
subset of the Python language (default version)
ii  python-notify               0.1.1-2+b2
bindings for libnotify
ii  python-numpy                1:1.4.1-5
Numerical
Python adds a fast array facility to the Python language
ii  python-pyorbit              2.24.0-6
language binding for the ORBit2 CORBA implementation
ii  python-reportbug             4.12.6
modules for interacting with bug tracking systems
ii  python-support              1.0.10
automated
rebuilding support for Python modules
ii  python-vte                  1:0.24.3-3
bindings for the VTE widget set
ii  python2.5                   2.5.5-11
interactive high-level object-oriented language (version 2.5)
ii  python2.5-dev               2.5.5-11
files and a static library for Python (v2.5)
ii  python2.5-minimal           2.5.5-11
minimal
subset of the Python language (version 2.5)
ii  python2.6                   2.6.6-8+b1
interactive high-level object-oriented language (version 2.6)
ii  python2.6-dev               2.6.6-8+b1
files and a static library for Python (v2.6)
ii  python2.6-minimal           2.6.6-8+b1
minimal
subset of the Python language (version 2.6)
ii  ssl-cert                    1.0.28
simple
debconf wrapper for OpenSSL
ii  tcpdump                     4.1.1-1
powerful
tool for network monitoring and data acquisition

```

actions.conf

```
# Artemisa - Configuration file for the actions of the scripts
#
# Be careful when modifying this file!

# In this configuration file you can define the parameters used when
# calling the user-configurable
# scripts such as "on_flood.sh".
# Use a blank space to separate the parameters.

# List of available parameters to insert in each section:
#
# $From_Extension$ 
# $From_IP$ 
# $From_Port$ 
# $From_Transport$ 
# $Contact_IP$ 
# $Contact_Port$ 
# $Contact_Transport$ 
# $Connection_IP$ 
# $Owner_IP$ 
# $Via_IP$ This is the first Via detected
# $Via_Port$ 
# $Tool_name$ 
#
# You can put your own parameters also. For example:
# on_flood=$From_IP$ DROP
# This will make that Artemisa run the command:
# bash ./scripts/on_flood.sh xxx.xxx.xxx.xxx DROP
# where xxx.xxx.xxx.xxx is the replaced variable $From_IP$

# [actions]
# on_flood contains the parameters to be sent when calling the file
on_flood.sh
# on_spit contains the parameters to be sent when calling the file
on_spit.sh
# on_scanning contains the parameters to be sent when calling the file
on_scanning.sh

[actions]
on_flood=$Contact_IP$ $Contact_Port$ $From_Extension$ $Tool_name$
on_spit=$From_IP$ $From_Port$ $From_Extension$ $Owner_IP$ $Tool_name$
on_scanning=$Contact_IP$ $Contact_Port$ $Via_IP$ $Via_Port$ 
$From_Extension$ $Tool_name$
```

artemisa.conf

```
# Artemisa - Main configuration file
#
# Be careful when modifying this file!

# [environment]
#
# behaviour_mode=
# There are three available values for behaviour_mode: active, passive
and aggressive.
#     active: messages are analyzed.
#     passive: messages are not analyzed.
#     aggressive: messages are analyzed and Artemisa tries to
counter-attack the caller.
#
# These modes are defined in file behaviour.conf.
# The call is answered in all cases and the messages are also logged.
#
# max_calls=
# Number of calls that Artemisa can handle at the same time. Use this
number to prevent flooding.
#
# playfile=
```

```

# Name of the WAV file that will be played after sending an 200 OK to
the caller. The file
# is searched into the ./audiofiles directory. The format must be
uncompressed 16 bit PCM
# or compressed G.711 A-law/U-law.

[environment]
local_ip=10.10.0.250
local_port=5060
sip_domain=10.10.0.240
user_agent=Twinkle/1.4.2
playfile=/sample.wav

behaviour_mode=active

max_calls=3
#fingerprint_mode=passive           NOT YET IMPLEMENTED

# [sound]
# In this section some sound parameters can be configured. Please, do
not simply modify it unless
# you have troubles with the sound recording process.
#
# enabled=
# If it contains 'false' the media shall not be recorded.
#
# device=
# Index of the sound device to be used. DO NOT simply modify this line
unless you are sure of what you are changing. In order to get a list of
the available sound devices run Artemisa with command-line parameter
'-g'.

[sound]
enabled=true
device=0
rate=44100

# [email]
# In this section you can configure the e-mail account that Artemisa
shall use to report detections.

[email]
# If enabled is set to 'false', no e-mail shall be sent.
enabled=true
smtp_server_ip=smtp.gmail.com
smtp_server_port=587
smtp_server_use tsl_ssl=true
smtp_server_username=artemisa.notifications@gmail.com
smtp_server_password=*****
from_mail=artemisa.notifications@gmail.com

# Here you can add more than one recipient address.
# Example: recipients_mail=john@admin.yourcompany.com,
mike@admin.yourcompany.com
# The e-mails shall be delivered as BCC to each one.

recipients_mail=artemisa.notifications@gmail.com
to_header="Administrator"
subject="Artemisa's report of activity 2xxx@ubpsipserver.myvnc.com
[2000-2007]"

```

behaviour.conf

```

# Artemisa - Program behaviour modes configuration file
#
# Be careful when modifying this file!

# In this file you can define each behaviour mode.

```

```

# List of available commands to insert in each section:
#
# NOT YET IMPLEMENTED!

[passive]
send_180
send_200

[active]
send_180
send_200

[aggressive]

```

extensions.conf

```

# Artemisa - Extensions configuration file
#
# Be careful when modifying this file!

# Here you are able to set up the extensions that shall be used by
Artemisa in the registration process. In order to use them, they must be
defined in the servers.conf file.
#
# The sections name hereunder, such as 3000 in section [3000], refers to
a SIP extension and it must be unique in this file, as well as correctly
configured in the registrar \emph{server}.

[1000]
username="1000"
password=1000

[1001]
username="1001"
password=1001

[1002]
username="1002"
password=1002

[1003]
username="1003"
password=1003

...
[1007]
username="1007"
password=1007

```

servers.conf

```

# Artemisa - Servers configuration file
#
# Be careful when modifying this file!

# Here you are able to set the registrar servers configuration that
Artemisa shall use to register itself.
#
# registrar_time=
# Is the time in minutes between automatic registrations. This is
performed in order to avoid
# being disconnected from the \emph{server} because of a lack of activity.
#
# nat_keepalive_interal=
# When dealing with NAT proxies, you can set a value in seconds which

```

```
indicates the time interval between keep alive messages. If zero is
written, then the NAT keep alive messages shall not be sent.
#
# exten=
# In this field you should set the extensions to be used. They must be
declared in extensions.conf.

[myproxy]
registrar_ip=10.10.0.240
registrar_port=5060
registrar_time=15
nat_keepalive_interval=30
exten=1000,1001,1002,1003,1004,1005,1006,1007
```

Anexo E

Informe generado por Artemisa honeypot para el ataque con SMAP a su dirección DoS

```
2012-07-11 18:02:37,496 Audio device not found. Calls will not be recorded.
2012-07-11 18:02:37,498 Extension phone_0 registration sent. Status: 100 (In Progress)
2012-07-11 18:02:37,499 Extension phone_1 registration sent. Status: 100 (In Progress)
2012-07-11 18:02:37,500 Extension phone_7 registration sent. Status: 100 (In Progress)
2012-07-11 18:02:37,502 Extension phone_8 registration sent. Status: 100 (In Progress)
2012-07-11 18:02:37,503 Extension phone_9 registration sent. Status: 100 (In Progress)
2012-07-11 18:02:37,604 Extension "phone_0" <sip:iphone_0@10.10.0.240:5060> registered, status=200 (OK)
2012-07-11 18:02:37,605 Extension "phone_1" <sip:iphone_1@10.10.0.240:5060> registered, status=200 (OK)
2012-07-11 18:02:37,656 Extension "phone_7" <sip:iphone_7@10.10.0.240:5060> registered, status=200 (OK)
2012-07-11 18:02:37,657 Extension "phone_8" <sip:iphone_8@10.10.0.240:5060> registered, status=200 (OK)
2012-07-11 18:02:37,658 Extension "phone_9" <sip:iphone_9@10.10.0.240:5060> registered, status=200 (OK)
2012-07-11 18:02:43,186 OPTIONS message detected in extension smap from 201.252.57.89:12345
2012-07-11 18:02:44,157 OPTIONS message detected in extension from 201.252.57.89:12345
2012-07-11 18:02:47,172 INVITE message detected.
2012-07-11 18:02:47,172 Waiting for SIP messages (5)...
2012-07-11 18:02:47,174 Incoming call from <sip:smap@localhost>
2012-07-11 18:02:47,175 Call from <sip:smap@localhost> is EARLY, last code = 180 (Ringing)
2012-07-11 18:02:47,176 Call from <sip:smap@localhost> is CONNECTING, last code = 200 (OK)
2012-07-11 18:02:48,174 Waiting for SIP messages (4)...
2012-07-11 18:02:49,172 INVITE message detected.
2012-07-11 18:02:49,172 Waiting for SIP messages (5)...
2012-07-11 18:02:49,174 Incoming call from <sip:smap@localhost>
2012-07-11 18:02:49,175 Call from <sip:smap@localhost> is EARLY, last code = 180 (Ringing)
2012-07-11 18:02:49,175 Waiting for SIP messages (3)...
2012-07-11 18:02:49,176 Call from <sip:smap@localhost> is CONNECTING, last code = 200 (OK)
2012-07-11 18:02:50,173 Waiting for SIP messages (4)...
2012-07-11 18:02:50,176 Waiting for SIP messages (2)...
2012-07-11 18:02:51,175 Waiting for SIP messages (3)...
2012-07-11 18:02:51,178 Waiting for SIP messages (1)...
2012-07-11 18:02:52,176 Waiting for SIP messages (2)...
2012-07-11 18:02:52,179
2012-07-11 18:02:52,179 ***** Information about the call *****
2012-07-11 18:02:52,179
2012-07-11 18:02:52,179 From: smap in localhost:12345/udp
2012-07-11 18:02:52,179 To: smap in localhost
2012-07-11 18:02:52,179 Contact: smap in 201.252.57.89:12345/udp
2012-07-11 18:02:52,180 Connection:
2012-07-11 18:02:52,180 Owner:
2012-07-11 18:02:52,180 Via 0: 201.252.57.89:12345/udp
2012-07-11 18:02:52,180 User-Agent: smap 0.6.0
2012-07-11 18:02:52,180
2012-07-11 18:02:52,180 ***** Classification *****
2012-07-11 18:02:52,180
2012-07-11 18:02:52,180 + Checking fingerprint...
2012-07-11 18:02:52,181 |
2012-07-11 18:02:52,181 | User-Agent: smap 0.6.0
2012-07-11 18:02:52,193 |
2012-07-11 18:02:52,193 | No fingerprint found.
2012-07-11 18:02:52,193
2012-07-11 18:02:52,193 + Checking DNS...
2012-07-11 18:02:52,193 |
2012-07-11 18:02:52,193 | + Checking localhost...
2012-07-11 18:02:52,194 |
2012-07-11 18:02:52,235 | | IP resolved: 127.0.0.1
2012-07-11 18:02:52,235 | | WHOIS data found.
2012-07-11 18:02:52,235 |
2012-07-11 18:02:52,235 | | Category: Interactive attack
2012-07-11 18:02:52,235 |
2012-07-11 18:02:52,236 | + Checking 201.252.57.89...
2012-07-11 18:02:52,236 |
2012-07-11 18:02:52,236 | | This is already an IP address. Nothing done.
2012-07-11 18:02:52,236 |
2012-07-11 18:02:52,236 | + Checking ...
2012-07-11 18:02:52,236 |
2012-07-11 18:02:52,236 | | IP cannot be resolved.
2012-07-11 18:02:52,236 |
2012-07-11 18:02:52,237 | | Category: Spoofed message
2012-07-11 18:02:52,237
2012-07-11 18:02:52,237 + Checking if SIP port is opened...
2012-07-11 18:02:52,237 |
2012-07-11 18:02:52,237 | + Checking 201.252.57.89:12345/udp...
2012-07-11 18:02:52,237 |
2012-07-11 18:02:52,528 | | Port state: closed
2012-07-11 18:02:52,528 |
2012-07-11 18:02:52,528 | | Category: Spoofed message
2012-07-11 18:02:52,528
2012-07-11 18:02:52,528 + Checking if media port is opened...
2012-07-11 18:02:52,528 |
2012-07-11 18:02:52,528 | No RTP info delivered.
2012-07-11 18:02:52,529 |
2012-07-11 18:02:52,529 | Category: Spoofed message
2012-07-11 18:02:52,529
2012-07-11 18:02:52,529 + Checking request URI...
2012-07-11 18:02:52,529 |
2012-07-11 18:02:52,529 | Extension in field To: smap
2012-07-11 18:02:52,529 |
2012-07-11 18:02:52,529 | Request addressed to the honeypot? No
2012-07-11 18:02:52,530
2012-07-11 18:02:52,530 + Checking if proxy in Via...
2012-07-11 18:02:52,530 |
2012-07-11 18:02:52,530 | + Checking 201.252.57.89:12345/udp...
2012-07-11 18:02:52,530 |
2012-07-11 18:02:52,703 | | Result: There is no SIP proxy
2012-07-11 18:02:52,704 |
2012-07-11 18:02:52,704 | | Category: Dial plan fault
2012-07-11 18:02:52,704
2012-07-11 18:02:52,704 + Checking for ACK...
```

```

2012-07-11 18:02:52,704 |
2012-07-11 18:02:52,704 | ACK received: No
2012-07-11 18:02:52,704 |
2012-07-11 18:02:52,704 | Category: Scanning
2012-07-11 18:02:52,705
2012-07-11 18:02:52,705 + Checking for received media...
2012-07-11 18:02:52,705 |
2012-07-11 18:02:52,705 | Media received: No
2012-07-11 18:02:52,705 |
2012-07-11 18:02:52,705 | Category: Ringing
2012-07-11 18:02:52,705
2012-07-11 18:02:52,705 + The message is classified as:
2012-07-11 18:02:52,705 | Interactive attack
2012-07-11 18:02:52,706 | Spoofed message
2012-07-11 18:02:52,706 | Dial plan fault
2012-07-11 18:02:52,706 | Scanning
2012-07-11 18:02:52,706 | Ringing
2012-07-11 18:02:52,706
2012-07-11 18:02:52,706 ***** Correlation *****
2012-07-11 18:02:52,706
2012-07-11 18:02:52,706 Artemisa concludes that the arrived message is likely to be:
2012-07-11 18:02:52,707
2012-07-11 18:02:52,708 * A scanning attempt.
2012-07-11 18:02:52,708
2012-07-11 18:02:52,708 Executing bash ./scripts/on_scanning.sh 201.252.57.89 12345 201.252.57.89 12345 smap ...
2012-07-11 18:02:52,710 * The message belongs to a ringing attack.
2012-07-11 18:02:52,711
2012-07-11 18:02:52,711 This report has been saved on file ./results/2012-07-11_5.txt
2012-07-11 18:02:52,711 NOTICE This report has been saved on file ./results/2012-07-11_5.html
2012-07-11 18:02:52,711 Sending this report by e-mail...
2012-07-11 18:02:53,177 Waiting for SIP messages (1)...
2012-07-11 18:02:54,179
2012-07-11 18:02:54,179 ***** Information about the call *****
2012-07-11 18:02:54,179
2012-07-11 18:02:54,179 From: smap in localhost:12345/udp
2012-07-11 18:02:54,179 To: smap in localhost
2012-07-11 18:02:54,179 Contact: smap in 201.252.57.89:12345/udp
2012-07-11 18:02:54,179 Connection:
2012-07-11 18:02:54,179 Owner:
2012-07-11 18:02:54,180 Via 0: 201.252.57.89:12345/udp
2012-07-11 18:02:54,180 User-Agent: smap 0.6.0
2012-07-11 18:02:54,180
2012-07-11 18:02:54,180 ***** Classification *****
2012-07-11 18:02:54,180
2012-07-11 18:02:54,180 + Checking fingerprint...
2012-07-11 18:02:54,180 |
2012-07-11 18:02:54,180 | User-Agent: smap 0.6.0
2012-07-11 18:02:54,194
2012-07-11 18:02:54,194 | No fingerprint found.
2012-07-11 18:02:54,194
2012-07-11 18:02:54,194 + Checking DNS...
2012-07-11 18:02:54,194 |
2012-07-11 18:02:54,194 | + Checking localhost...
2012-07-11 18:02:54,194 |
2012-07-11 18:02:54,226 | | IP resolved: 127.0.0.1
2012-07-11 18:02:54,226 | | WHOIS data found.
2012-07-11 18:02:54,226 | |
2012-07-11 18:02:54,226 | | Category: Interactive attack
2012-07-11 18:02:54,226 |
2012-07-11 18:02:54,227 | + Checking 201.252.57.89...
2012-07-11 18:02:54,227 | |
2012-07-11 18:02:54,227 | | This is already an IP address. Nothing done.
2012-07-11 18:02:54,227 | |
2012-07-11 18:02:54,227 | + Checking ...
2012-07-11 18:02:54,227 | |
2012-07-11 18:02:54,227 | | IP cannot be resolved.
2012-07-11 18:02:54,227 | |
2012-07-11 18:02:54,228 | | Category: Spoofed message
2012-07-11 18:02:54,228
2012-07-11 18:02:54,228 + Checking if SIP port is opened...
2012-07-11 18:02:54,228 |
2012-07-11 18:02:54,228 | + Checking 201.252.57.89:12345/udp...
2012-07-11 18:02:54,228 | |
2012-07-11 18:02:54,401 | | Port state: closed
2012-07-11 18:02:54,401 | |
2012-07-11 18:02:54,402 | | Category: Spoofed message
2012-07-11 18:02:54,402
2012-07-11 18:02:54,402 + Checking if media port is opened...
2012-07-11 18:02:54,402 |
2012-07-11 18:02:54,402 | No RTP info delivered.
2012-07-11 18:02:54,402 |
2012-07-11 18:02:54,402 | Category: Spoofed message
2012-07-11 18:02:54,402
2012-07-11 18:02:54,402 + Checking request URI...
2012-07-11 18:02:54,403 |
2012-07-11 18:02:54,403 | Extension in field To: smap
2012-07-11 18:02:54,403 |
2012-07-11 18:02:54,403 | Request addressed to the honeypot? No
2012-07-11 18:02:54,403
2012-07-11 18:02:54,403 + Checking if proxy in Via...
2012-07-11 18:02:54,403 |
2012-07-11 18:02:54,403 | + Checking 201.252.57.89:12345/udp...
2012-07-11 18:02:54,404 | |
2012-07-11 18:02:54,576 | | Result: There is no SIP proxy
2012-07-11 18:02:54,576 | |
2012-07-11 18:02:54,576 | | Category: Dial plan fault
2012-07-11 18:02:54,576
2012-07-11 18:02:54,577 + Checking for ACK...
2012-07-11 18:02:54,577 |

```

```

2012-07-11 18:02:54,577 | ACK received: No
2012-07-11 18:02:54,577 |
2012-07-11 18:02:54,577 | Category: Scanning
2012-07-11 18:02:54,577
2012-07-11 18:02:54,577 + Checking for received media...
2012-07-11 18:02:54,577 |
2012-07-11 18:02:54,577 | Media received: No
2012-07-11 18:02:54,578 |
2012-07-11 18:02:54,578 | Category: Ringing
2012-07-11 18:02:54,578
2012-07-11 18:02:54,578 + The message is classified as:
2012-07-11 18:02:54,578 | Interactive attack
2012-07-11 18:02:54,578 | Spoofed message
2012-07-11 18:02:54,578 | Dial plan fault
2012-07-11 18:02:54,578 | Scanning
2012-07-11 18:02:54,579 | Ringing
2012-07-11 18:02:54,579
2012-07-11 18:02:54,579 **** Correlation ****
2012-07-11 18:02:54,579
2012-07-11 18:02:54,579 Artemisa concludes that the arrived message is likely to be:
2012-07-11 18:02:54,579
2012-07-11 18:02:54,579 * A scanning attempt.
2012-07-11 18:02:54,579
2012-07-11 18:02:54,579 Executing bash ./scripts/on_scanning.sh 201.252.57.89 12345 201.252.57.89 12345 smap ...
2012-07-11 18:02:54,582 * The message belongs to a ringing attack.
2012-07-11 18:02:54,582
2012-07-11 18:02:54,582 This report has been saved on file ./results/2012-07-11_6.txt
2012-07-11 18:02:54,583 NOTICE This report has been saved on file ./results/2012-07-11_6.html
2012-07-11 18:02:54,583 Sending this report by e-mail...
2012-07-11 18:03:19,255 Call from <sip:smap@localhost> is DISCONNECTD, last code = 406 (Not Acceptable)
2012-07-11 18:03:19,255 Error while closing the conferences in method on_state().
2012-07-11 18:03:21,251 Call from <sip:smap@localhost> is DISCONNECTD, last code = 406 (Not Acceptable)
2012-07-11 18:03:21,251 Error while closing the conferences in method on_state().
2012-07-11 18:03:56,598 Extension "phone_0" <sip:phone_0@10.10.0.240:5060> unregistered, status=200 (OK)
2012-07-11 18:03:56,615 Extension "phone_1" <sip:phone_1@10.10.0.240:5060> unregistered, status=200 (OK)
2012-07-11 18:03:56,632 Extension "phone_7" <sip:phone_7@10.10.0.240:5060> unregistered, status=200 (OK)
2012-07-11 18:03:56,649 Extension "phone_8" <sip:phone_8@10.10.0.240:5060> unregistered, status=200 (OK)
2012-07-11 18:03:56,666 Extension "phone_9" <sip:phone_9@10.10.0.240:5060> unregistered, status=200 (OK)
Artemisa v1.0.91 Copyright (C) 2009-2011 Mohamed Nassar, Rodrigo do Carmo, and Pablo Masri

```

This program comes with ABSOLUTELY NO WARRANTY; for details type 'show warranty'.
This is free software, and you are welcome to redistribute it under certain
conditions; type 'show license' for details.

Type 'help' for help.

```

18:02:37.481 os_core_unix.c pjlib 1.14.0 for POSIX initialized
18:02:37.481 sip_endpoint.c Creating endpoint instance...
18:02:37.481 pjlib select() I/O Queue created (0x8858140)
18:02:37.481 sip_endpoint.c Module "mod-msg-print" registered
18:02:37.481 sip_transport. Transport manager created.
SIP User-Agent listening on: 10.10.0.250:5060
Behaviour mode: active
Starting extensions registration process...
root>

```

Anexo F

To-Do list:

Implementación corporativa (entorno en producción)

A fines prácticos se propone su implementación para la continuación de una 2da etapa del proyecto.

Realizar prueba exhaustiva con herramientas de *penetration testing* VoIP

Realizar pruebas con todos los aplicativos presentes en la lista VoIP Security Tool List (<http://www.voipsa.org/Resources/tools.php>) de modo de por una parte ampliar más el diccionario de *fingerprints* de Artemisa. Como también, mesurar cuantas de estas herramientas son detectadas por nuestro *honeypot*.

Más herramientas de seguridad VoIP:

<http://securitytools.wikidot.com/voip>

<http://www.0daysecurity.com/penetration-testing/VoIP-security.html>

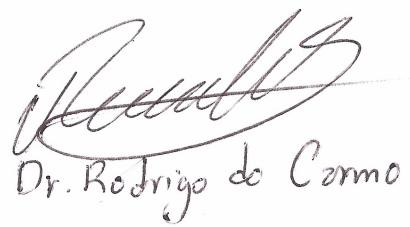
Modos activos y agresivo de Artemisa

Considerar leer campo *user-agent* dentro del mensaje SIP (puede verse en el output de la consola de Artemisa - realizar parse - o guardarlo en variable utilizando pjsua como lo hace Artemisa actualmente y agregar al archivo *fingerprint*).

Interoperabilidad de Artemisa con VoIP honey

Lograr la interoperabilidad de Artemisa con VoIP honey²¹ pudiendo Artemisa formar parte (si el usuario así lo desea) del conjunto de herramientas VoIP honey. Todo ello a nivel de plugins que permitan configurar Artemisa como un honeypot externo aunque integrado en la solución VoIP honey.

²¹<http://voiphoney.sourceforge.net/> Por su parte, sera de gran utilidad considerar los comentarios realizados en la publicación <http://bytecoders.net/content/honeypot-voip-artemisa.html>



A handwritten signature in black ink, appearing to read "Reinaldo".

Dr. Rodrigo do Carmo