Entorno para Experimentación de Vulnerabilidades en la Enseñanza de Buenas Prácticas de Programación

Uri Yael y Benjamín Barán

Universidad Nacional de Asunción

Resumen Este trabajo propone un entorno sencillo y de fácil utilización para un instructor que permite a los alumnos experimentar con vulnerabilidades de seguridad en un contexto controlado, consolidando sus conocimientos teóricos de cursos anteriores y/o complementando lo aprendido en la carrera, con laboratorios prácticos que complementen el proceso de aprendizaje que muchas veces se limita a presentaciones teóricas por falta de recursos como tiempo o equipos. Se reportan resultados prometedores con alumnos de carreras universitarias en informática y electrónica.

Keywords: educación, seguridad, inyección de código, XSS, CSRF

1. Introducción

Los cursos de seguridad de la información deben cubrir una vasta cantidad de áreas y temas pero el tiempo disponible muchas veces impide ahondar suficientemente en todas las áreas, por lo que muchos autores intentan confeccionar guías generales que puedan ser aplicadas en forma transversal [17]. Es crucial seleccionar buenas técnicas pedagógicas para que los estudiantes puedan adquirir los conocimientos necesarios en un espacio de tiempo razonable [8,17].

1.1. Contexto y Motivación

Al momento de crear sistemas informáticos, la seguridad debe ser un objetivo primario desde el mismo inicio. En gran cantidad de casos, los sistemas que no tuvieron dicho objetivo ya en la etapa de diseño, deben luego pasar por reestructuración y diversos cambios, posiblemente con un importante costo en recursos (*e.g.* tiempo de desarrollo, atrasos en fechas límite, dinero) [14].

Los errores de código que pueden considerarse *bugs* clásicos, cuya detección puede ser automatizada y validada en sucesivas verificaciones, no son el único motivo por el que existen vulnerabilidades. Otras fuentes de vulnerabilidades –que pueden derivar en huecos de seguridad– son los errores conceptuales en protocolos e interacciones, susceptibilidad a la falta de autenticación, autorización o el chequeo de integridad.

En cierto sentido, las técnicas de desarrollo de software generalmente enseñadas en los cursos regulares son inadecuadas para crear programas seguros debido a que se centran en la correctitud. Una definición de correctitud es: "*cumplimiento de las especificaciones que determinan cómo los usuarios pueden interactuar con el sistema y cómo debe comportarse el sistema cuando es utilizado en forma correcta*". Mientras la correctitud se encarga de las funcionalidades del sistema, la seguridad se encarga de la falta de funcionalidades. Es decir, se intenta validar que sin importar las acciones llevadas a cabo, no se podrán efectuar ciertas acciones. Las comprobaciones automatizadas son métodos para validar la existencia de una funcionalidad, pero no existe un método que asegure la inexistencia de una funcionalidad.

En opinión de Matt Bishop y Jacob West [16], existen diversos mitos por los cuales muchas instituciones educativas no aumentan los esfuerzos por incorporar en la práctica, la enseñanza de codificación segura, entre ellos: (a) no hay espacio en la malla curricular. No se necesitan más cursos, sino revisar el contenido de los ya existentes; (b) si los alumnos aprenden a escribir programas seguros, cuando salgan al mercado laboral ¿aceptarán las compañías el aumento del costo y tiempo que toma crear un programa?, ¿los clientes pagarían precios más altos?, ¿realmente se requerirá a los alumnos que pongan en práctica lo que aprendieron?; y (c) la creencia de que ya lo estamos haciendo. Salvo ciertas excepciones, en los centros de formación, la evolución de los programas educativos y de las técnicas de enseñanza no es tan rápida como los avances en el campo de la seguridad de la información lo requiere.

1.2. Aspectos Educativos y Pedagógicos

Las personas aprenden a través de distintos procesos. La estrategia de aprendizaje o de enseñanza debe ser seleccionada teniendo en cuenta tanto los contenidos como el público destinatario. En [4] Bishop y Orvis presentan un enfoque para realzar buenas prácticas e implementar un estilo de programación adecuado a lo largo de una malla curricular. Según Bishop y Orvis, la técnica es también utilizada por escuelas de abogados para enseñar escritura legal. El concepto de buenas prácticas debe ser entendido como un conjunto de sugerencias relacionadas a cómo realizar procesos (*e.g.* ciclo de vida del *software*), normas de cómo escribir código fuente, entre otros [1,2,3].

Los autores del presente trabajo, en clases de la cátedra Redes de Computadoras dictadas en la Universidad Católica *"Nuestra Señora de la Asunción"*, tuvieron la oportunidad de introducir conceptos generales de seguridad y hacer énfasis en aquellos que se relacionan a programación y protocolos. En diversas ocasiones, y aunque existen excepciones, la mayoría de los alumnos captan las ideas a partir de ejemplos concretos pero tienen dificultades al momento de abstraerse y comprender el caso general. Esto se puede deber a la poca o nula cantidad de experiencias prácticas relacionadas a seguridad de la información que se desarrollan en la carrera. Los alumnos encuestados dan tres motivos principales por los que creen que no es mayor el número de experiencias similares en su carrera: falta de tiempo tanto en clase como de los profesores, alta dificultad

de actualización y mantenimiento para los recursos con los que se cuenta, y la complejidad de crear y ejecutar las experiencias.

La inclusión de experiencias prácticas en el aula podría ayudar a los participantes a fijar los conceptos teóricos. Este trabajo está motivado por el hecho de que muchos instructores y profesores se dedican a la enseñanza como una actividad laboral relevante intelectualmente, pero no económicamente. Es decir, el tiempo dedicado a la preparación de los cursos en los que participan en general es reducido y la adaptación, configuración y realización de experiencias prácticas excede al tiempo disponible; todo esto sin considerar otros aspectos como los requerimientos de equipos y configuración.

1.3. Objetivos del Trabajo

El presente trabajo tiene como objetivo brindar una herramienta práctica que por su simplicidad de instalación, mantenimiento y gestión así como por los escasos recursos que requiere, podría disminuir notoriamente el costo en tiempo y en materiales, al realizar actividades prácticas de seguridad en clase.

Conocidos los beneficios reportados al realizar en clase experiencias como laboratorios prácticos, y revisando las dificultades encontradas por los instructores en las distintas técnicas de enseñanza [4,10,14,15,16,17], la herramienta aquí presentada se diferencia de otras en la simplicidad de preparación y administración del entorno, ya que según los datos recabados, la mayoría está de acuerdo en que estas experiencias son beneficiosas para los estudiantes, pero que los instructores no cuentan con recursos suficientes (*e.g.* tiempo, componentes) para llevarlas a cabo.

En lo que sigue del documento, la Sección 2 presenta el entorno de experimentación con sus características y escenarios, finalizando con la Sección 3 donde se presentan las conclusiones y posibles trabajos futuros.

2. Entorno de Experimentación

El lenguaje de programación utilizado en la construcción de la herramienta es PHP debido a su uso extendido y popularizado. Otros lenguajes que pudieron haber sido seleccionados son Ruby o Python pero los autores priorizaron el hecho de que PHP es anterior a dichos lenguajes por lo que es más probable que los instructores, profesores o guías estén familiarizados con él. Como motor de base de datos fue elegido PostgreSQL y con poco esfuerzo podrían realizarse las adaptaciones necesarias para utilizar MySQL, Oracle u otro motor, dada la simplicidad de las funciones y demás objetos de base de datos.

En la Figura 1 pueden observarse las tablas y relaciones que conforman el Diagrama de Entidad Relación. La base de datos tiene dos funciones principales: almacenar en forma de bitácora las acciones relevantes realizadas por los participantes, y almacenar los datos utilizados en los distintos escenarios. Las tablas `base_personas` y `base_usuarios` almacenan los datos originales que son insertados en `datos_personas` y `datos_usuarios` por cada nuevo participante

que se registra en la plataforma de forma que los participantes tengan cada uno un ambiente aislado. Los datos base son también utilizados cuando se acciona un mecanismo de restauración para un participante dado. La tabla `cupones` es utilizada en el octavo escenario, y las demás tablas mantienen información sobre los participantes y las acciones que realizan.

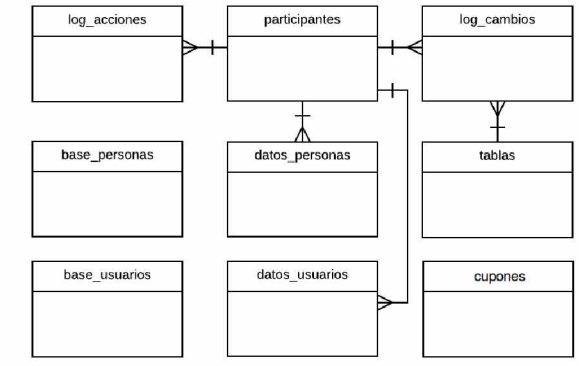

Figura 1. Diagrama entidad relación simplificado.

El entorno desarrollado se conforma principalmente de escenarios que son de nivel básico, y no son suficientemente complejos como para satisfacer las necesidades de una materia que se especialice en seguridad pero permite a los participantes llevar a la práctica las nociones mínimas y adentrarse en el área de seguridad. Entre las listas de vulnerabilidades más críticas y relevantes se destacan la de OWASP (*Open Web Application Security Project*) [11], y la de CWE (*Common Weakness Enumeration*) y SANS (*SysAdmin Audit, Networking and Security Institute*) [9]. Cada escenario plantea un desafío distinto basado en las vulnerabilidades abarcadas por las listas antes mencionadas de manera a abarcar situaciones actuales y de alto impacto. En particular, el quinto escenario plantea una situación real no cubierta por dichas listas pero sí por algunas guías [6,12].

En el diseño del entorno se tuvo en cuenta que el instructor puede agregar nuevos escenarios o modificar a los escenarios existentes. Para ello se necesitan algunos conocimientos de programación. Los escenarios podrán ser modificados, ampliados o extendidos en la medida que las listas de OWASP y CWE/SANS sean actualizadas. Una forma alternativa de utilizar la herramienta propuesta es que los participantes puedan crear o modificar escenarios y que los mismos sean probados por sus pares.

2.1. Escenarios

A continuación se describen los ocho escenarios incorporados actualmente, cuya relación con vulnerabilidades frecuentes se resumen en el Cuadro 1.

Cuadro 1. Resumen de relación entre escenarios y vulnerabilidades

Escenario	CWE/SANS	OWASP
1	CWE–89	A1, A2
2	CWE–79, CWE–134	A3
3	CWE–79, CWE–352, CWE–601	A8, A10
4	CWE–79, CWE–862, CWE–829	A6, A7
5		
6	CWE–89, CWE–250	A1, A3, A5
7	CWE–79, CWE–311	A2, A3
8		A5, A9

Escenario 1. Demuestra una de las consecuencias de un error típico de programación relacionado al limpiado de datos. La vulnerabilidad generada por el error de programación permite que se inyecte código SQL (*Structured Query Language*) y ocupa el primer lugar en ambas listas: CWE/SANS con 93,8 puntos bajo el identificador CWE–89, y OWASP con identificador A1. El escenario también se relaciona con el segundo puesto de la lista OWASP sobre déficit en autenticación y manejo de sesiones. Específicamente, el escenario se centra en explotar la vulnerabilidad en el método de autenticación. El objetivo planteado es ingresar sin conocer ni adivinar usuario y contraseña, como puede verse en la Figura 2. Se espera que el participante, aunque requiera de una pequeña investigación en Internet (e.g. `http://www.redeszone.net/seguridad-informatica/inyeccion-sql-manual-basico` o `http://www.securityartwork.es/2013/11/21/evasion-de-autenticacion-con-inyeccion-sql`), realice una inyección de código que cambie la sentencia lógica evaluada por el motor de base de datos en la aceptación del usuario permitiendo así su acceso al sistema.

Escenario 2. La inyección de código se encuentra en cuarto puesto del CWE/SANS con identificador CWE–79, cuya raíz es la inapropiada neutralización de datos de entrada durante la generación del contenido. La vulnerabilidad denominada CWE–134 trata acerca de la falta de control sobre los formatos de cadena. Según OWASP, el tercer riesgo más crítico es el de XSS (*Cross-Site Scripting*) causado por la falta de limpiado de datos ingresados por el usuario y el correspondiente escape de ciertos caracteres como los símbolos mayor y menor. El participante debe lograr inyectar código JavaScrip para que se despliegue una alerta a los usuarios que carguen la página. Cada persona listada posee un botón que permite ingresar a la edición de los campos correspondientes a sus datos (ver Figura 3). El ejercicio plantea sólo que se despliegue un cartel emergente pero el

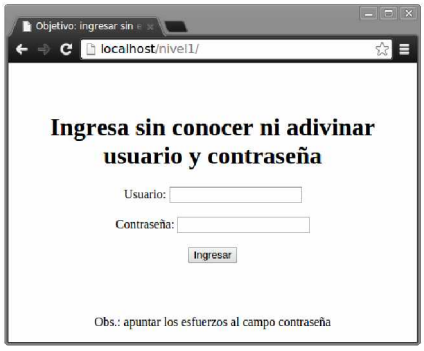

Figura 2. Pantalla del escenario 1.

ataque tiene gran potencial ya que podría descargar datos al cliente y acceder a información existente en el DOM (*Document Object Model*), entre otros métodos que forman parte de los ataques. En concreto, el participante realiza inyección de código a través de uno de los campos y verifica si tuvo o no éxito reingresando al formulario.

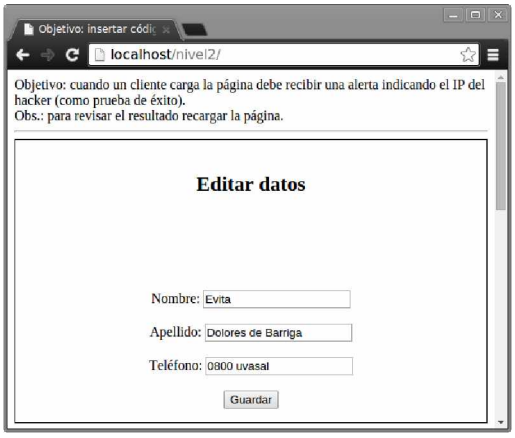

Figura 3. Pantalla del escenario 2.

Escenario 3. Es similar al anterior y provee la misma interfaz al participante, pero difiere en el objetivo ya que el código JavaScript inyectado debe redirigir al usuario en forma automática a otro sitio completamente diferente. La peligrosidad de este ataque aumenta si el sitio vulnerable utiliza marcos (*frames*) ya que en tal caso el usuario podría no notar el hecho de que parte del sitio que observa fue reemplazado por contenido malicioso que podría imitar al original. Este escenario está relacionado al error con identificador CWE–79 que trata sobre la

inapropiada neutralización de datos de entrada, con CWE–352 que se encuentra en el puesto once y trata sobre CSRF (*Cross-Site Request Forgery*), y con CWE–601 que se relaciona con las redirecciones a sitios no confiables. Revisando la lista de OWASP, la vulnerabilidad se relaciona al octavo puesto sobre CSRF y al décimo puesto sobre no validación de redirecciones y reenvíos. Debe destacarse que actualmente algunos entornos como Django y algunos navegadores web implementan cabeceras que controlan el origen del contenido y no permiten que un marco *iframe* sea cargado con contenido de un dominio distinto a los autorizados o al del sitio principal. Esta vez, el participante realiza inyección de código JavaScript y verifica si tuvo o no éxito cargando nuevamente la página que lo debería redirigir al sitio externo inyectado por el atacante.

Escenario 4. Se centra en el método utilizado por un formulario al enviar los valores y en la mala práctica de utilizar campos ocultos para definir el comportamiento de una funcionalidad. El escenario está relacionado con el error CWE–79 ya referido anteriormente, y con los errores CWE–862 (autorización faltante) y CWE–829 (inclusión de funcionalidades desde ámbitos de control no confiables). En lo referente al listado de OWASP, el escenario se relaciona al sexto puesto (exposición de información sensible) y al séptimo puesto (falta de control de nivel de acceso). Como objetivo del escenario se solicita al participante que obtenga la contraseña de un usuario aprovechando una vulnerabilidad de diseño. Este escenario demuestra la peligrosidad de que el usuario tenga acceso a modificar el funcionamiento del sistema por falta de validación, exposición de código y en general, por mal diseño. En este caso, el participante debe observar el código HTML (*HyperText Markup Language*) y cambiarlo de forma que en vez de obtener el correo electrónico recupere la contraseña. Se hace notar aquí la función del instructor que podría necesitar dar pistas a los participantes sobre los posibles campos que podrá tener la tabla en cuestión.

Escenario 5. Enfoca un error humano que puede dar lugar a diversas vulnerabilidades. El objetivo es demostrar la gravedad del error en forma general. Para ello, se muestra al participante un texto que habla sobre la importancia de tener copias de respaldo antes de realizar modificaciones importantes a ciertos archivos. El error humano surge debido a que algunas personas dejan esas copias de respaldo en ubicaciones alcanzables por un usuario, y luego se olvidan de borrar los archivos temporales. Para realizar una copia de respaldo rápida, una forma muy utilizada es copiar el archivo y agregar un sufijo que indique que el mismo es una copia vieja o de respaldo (*e.g.* utilizando extensiones como .old, .bkp). Un servidor web, a menos que sea configurado en forma distinta, descarga al cliente los archivos cuyas extensiones no sean bloqueadas o indiquen que el archivo debe ser interpretado (*e.g.* .php, .aspx). Otra forma de exposición de código se da cuando el programador deja comentarios que incluyen datos sensibles como contraseñas, nombres de usuario u otra información privada. El error de procedimiento no está incluido en las listas referidas debido a que estas listas se centran en las aplicaciones y no en la configuración de servidores, pero figura en guías como [6,12]. En resumen, el participante podría modificar la dirección

de un recurso como `conexion.php` intentando acceder a una copia de respaldo, por ejemplo `conexion.php.bkp` o `conexion.old`.

Escenario 6. Se enfoca en el alcance que podría tener la inyección de código SQL solicitando al participante que borre de la base de datos todos los datos de una tabla. El escenario está asociado en forma directa a la vulnerabilidad de mayor riesgo según CWE/SANS: CWE–89 (relacionada a la falta de neutralización de código SQL), y se ve potenciada por CWE–250 (relacionada a la ejecución con permisos innecesarios). También se relaciona al primer puesto de la lista OWASP debido a la posibilidad de inyectar código SQL que resulte en el borrado de datos, al tercer puesto debido a la falta de limpieza de datos de entrada que posibilita un ataque de tipo XSS y al quinto puesto relacionado a una mala configuración de seguridad ya que la operación de borrado no se ve obstruida por permisos. Básicamente, el participante debe aprovechar un campo del formulario para inyectar código SQL con instrucción de borrado de datos, por ejemplo, utilizando la sentencia `delete` y verificando el éxito de su ataque al no encontrar datos en la lista de usuarios existentes que muestra la página.

Escenario 7. Tiene como meta el robo de información de las *cookies* a través de inyección de código JavaScrip. El objetivo del escenario es demostrar cómo un atacante podría robar y alterar la información que contienen dichos archivos. Este ataque podría causar graves daños como alteración de información o permitir que el atacante gane acceso a una aplicación a través de la cuenta de la víctima aún sin conocer nombre de usuario ni contraseña. Este ataque posee características de XSS encontrándose en el cuarto puesto de la lista de CWE/SANS bajo el identificador CWE–79 y en el tercer puesto de la lista OWASP, y al segundo puesto de ésta última relacionado al déficit en autenticación y manejo de sesiones. El esfuerzo del atacante para modificar los datos es menor si las *cookies* son almacenadas en texto plano en vez de encriptado, considerando que el octavo puesto de la lista CWE/SANS lo ocupa la vulnerabilidad CWE–311 relacionada a la falta de encriptación de datos sensibles. El participante debe inyectar código JavaScript en un formulario para acceder a la *cookie* ya creada por la herramienta donde se almacena la información de sesión.

Escenario 8. El octavo escenario se centra en un área del diseño de protocolos seguros. Demuestra en forma práctica lo que podría ocurrir cuando se requiere que un dato sea aleatorio, pero debido a la implementación se asignan variables con una aleatoriedad pobre, razón por la cual no es aconsejable que personas no expertas utilicen generadores pseudo-aleatorios de diseño propio [5,13]. A modo de ejemplo, HTTP es un protocolo sin estado, por lo que en cada transacción el navegador envía al servidor el valor de una variable de sesión para recuperar la información. Si el atacante adivina dicho valor puede ganar acceso a la sesión sin necesidad de interceptar paquetes que intercambian entre cliente y servidor, ni de realizar un ataque directo a la víctima. En el octavo escenario se presenta una pantalla desde donde solicitar cupones para analizar la aleatoriedad de los códigos. El objetivo es que el participante deduzca la lógica detrás de la generación de códigos y encuentre el cupón por valor de un millón de Dólares. En ocasiones se hace uso de componentes de terceros con vulnerabilidades conocidas o que

requieren cierta configuración para funcionar en forma segura. Estas dos últimas posibilidades ocupan el quinto y el noveno puesto en la lista de OWASP bajo los títulos de "Mala configuración de seguridad" y "Utilización de componentes con vulnerabilidades conocidas", respectivamente. En concreto, el participante puede observar la relación entre cupón y premio para luego intentar deducir el código del cupón especificado en el ejercicio, para lo cual hace uso de una entrada del formulario donde puede verificar el premio que corresponde a cada código que el participante desea consultar. El participante puede probar tantos códigos de cupones como desee, pero claramente le será muy difícil lograr el objetivo por búsqueda exhaustiva.

3. Conclusiones

Los cursos de seguridad de la información deben cubrir una vasta cantidad de áreas y temas; el tiempo disponible impide ahondar suficientemente en todas las áreas, por lo que muchos autores intentan confeccionar guías generales que puedan ser aplicadas en forma transversal, y que sean mayormente válidas para cada caso [17]. El tiempo y los recursos son los principales motivos por los que muchas instituciones educativas no aumentan la cantidad de prácticas, pero existen formas de incluirlas a lo largo de la malla curricular [4,16]. Es crucial seleccionar buenas técnicas pedagógicas para que los estudiantes puedan adquirir los conocimientos necesarios en un espacio de tiempo razonable. Así, este trabajo está motivado por el hecho de que, en general, el tiempo del que disponen muchos instructores y profesores para preparar los cursos es menor al que requiere la adaptación, configuración y realización de experiencias prácticas. El presente trabajo tiene como objetivo brindar una herramienta práctica que por su simplicidad de instalación, mantenimiento y gestión así como por los escasos recursos que requiere, podría disminuir notoriamente el costo en tiempo y en materiales de realizar actividades prácticas en clase. Dichas actividades reportan valiosos beneficios sorteando dificultades encontradas por los instructores en las distintas técnicas de enseñanza [4,10,14,15,16,17].

La herramienta presentada se diferencia de otras en la simplicidad de preparación y administración del entorno, el cual se conforma principalmente de escenarios y de componentes destinados a la administración y recuperación de datos. Los escenarios permiten a los participantes llevar a la práctica nociones básicas y adentrarse en el área de seguridad. Cada escenario plantea un desafío distinto basado en las vulnerabilidades frecuentes mencionadas en las listas de OWASP y CWE/SANS de manera a abarcar situaciones actuales y de alto impacto, aunque uno de los escenarios propuestos plantea también situación real no cubierta por dichas listas, pero sí por algunas guías [6,12].

En el diseño del entorno se consideró que el instructor puede agregar nuevos escenarios o modificar los existentes. Los escenarios deberán ser actualizados, ampliados o extendidos en la medida que las listas de OWASP y CWE/SANS sean actualizadas. Una forma alternativa de utilizar la herramienta propuesta es que el objetivo de los participantes sea crear o modificar escenarios y que los mis-

mos sean probados por sus pares. Los autores –en su calidad de profesores de la cátedra Redes de Computadoras– realizaron experiencias que resultaron ser muy alentadoras tanto en las carreras de Ingeniería Electrónica como de Ingeniería Informática de la Universidad Católica *Nuestra Señora de la Asunción*.

Como trabajos futuros los autores siguen trabajando en: (a) la inclusión de más escenarios aumentando la dificultad o abarcando vulnerabilidades tidavía no incluidas en los existentes, por ejemplo, la carga de archivos al servidor; (b) adaptar la herramienta para simplificar la creación de escenarios destinados a ataques de denegación de servicio; y (c) agregar un componente que permita al participante, una vez que descubra la contraseña, acceder a una consola donde pueda ejecutar código SQL en la base de datos, entre otras alternativas.

Referencias

1. Abran, A., Bourque, P.: SWEBOK: Guide to the software engineering Body of Knowledge. IEEE Computer Society. 2014
2. Apple Inc: Secure Coding Guide. 2014
3. Belk, Coles, Goldschmidt, Howard, Randolph, Saario, Sondhi, Tarandach, Vähä–Sipilä, Yonchev: Fundamental Practices for Secure Software Development. 2011
4. Bishop, M., Orvis, B. J.: A clinic to teach good programming practices, Proceedings of the 10th Colloquium for Information Systems Security Education. 168–1174 (2006)
5. Ferguson, N., Schneier, B., Kohno, T.: Cryptography Engineering: Design Principles and Practical Applications. ISBN 9780470474242. Wiley Publishing. 2010
6. Heckathorn, M.: Network Monitoring for Web-Based Threats. 2011
7. Kim, P.: The Hacker Playbook: Practical Guide to Penetration Testing. ISBN 9781494932633. Createspace Independent Pub. 2014
8. Kölling, M.: The problem of teaching object-oriented programming, Part 1: Languages. SIGS Publications. Journal of Object-oriented programming. Vol. 11, Num. 8, 8–15 (1999)
9. Martin, B., Brown, M., Paller, A., Kirby, D., Christey, S.: 2011 CWE/SANS top 25 most dangerous software errors. Common Weakness Enumeration. 2011
10. NESG – Network Engineering & Security Group: NETA: A NETwork Attacks Framework. Architecture and Usage. University of Granada. 2013
11. OWASP Foundation: OWASP Top 10. `https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project`. 2013
12. OWASP Foundation: OWASP Testing Guide v4. `https://www.owasp.org/index.php/OWASP_Testing_Project`. 2014
13. Paar, C., Pelzl, J.: Understanding cryptography : a textbook for students and practitioners. ISBN 978-3-642-04101-3. 2010
14. Schindler, F.: Coping with Security in Programming. Acta Polytechnica Hungarica. Vol. 3, Num. 2, 65–72 (2006)
15. Talib, M. A., Khelifi, A., Jololian, L.: Secure Software Engineering: A New Teaching Perspective Based on the SWEBOK. Interdisciplinary Journal of Information, Knowledge, and Management. Vol. 5, 83–99 (2010)
16. West, J., Bishop, M.: Security Education for the new Generation, Proc. RSA Conference. 2014
17. Yurcik, W., Doss, D.: Different approaches in the teaching of information systems security, Proceedings of the Information Systems Education Conference. 2001