TRABAJO DE GRADO

Opción Seminario-Diplomado.

Mitigación de Riesgos de Seguridad y Protección de Propiedad Intelectual en la Externalización de Desarrollo de Software

Corporación Universitaria Remington.

Facultad de ingeniería de sistemas

Estudiante: Surelys Adriana Ynfante Valero

Tutor: Jorge Mauricio Sepúlveda Castaño

2025

DEDICATORIA

A mi familia, quienes con su amor incondicional, paciencia y apoyo constante, han sido mi motivación principal para alcanzar esta meta. Su confianza en mí fue el impulso necesario para superar cada desafío.

A mi propia perseverancia y disciplina, por permitirme culminar este proceso de formación, demostrando que el esfuerzo continuo y la búsqueda del conocimiento son las herramientas más poderosas para transformar mi realidad profesional.

AGRADECIMIENTOS

Quiero expresar mi gratitud a todas las personas e instituciones que aportaron a la realización de este trabajo:

A mi familia, por su comprensión durante las largas horas de estudio y dedicación que este proyecto requirió. Su respaldo emocional fue el cimiento sobre el cual construí este logro.

A la Corporación Universitaria Remington, por brindarme el espacio académico, los recursos y la formación integral que hoy me permite presentarme como un profesional competente en el área de ingeniería.

Finalmente, a mis profesores, colegas y compañeros de carrera, con quienes compartí debates, conocimientos y experiencias que enriquecieron mi visión profesional y personal durante esta etapa universitaria.

Tabla de Contenido

Tablas

RESUMEN

Este documento analiza los riesgos críticos de ciberseguridad inherentes al modelo de outsourcing tipo "Fábrica de Software". Si bien la externalización ofrece eficiencia operativa y acceso a talento especializado, introduce vectores de ataque específicos que difieren del BPO tradicional: la exfiltración de propiedad intelectual (IP) y la inclusión de vulnerabilidades en el código fuente entregado. Se propone un marco de gestión basado en la metodología DevSecOps, integrando controles automatizados (SAST, DAST, SCA) y cláusulas contractuales de "Quality Gates". El objetivo es transicionar de un modelo de confianza ciega a uno de verificación continua, mitigando la exposición a fugas de información y ataques a la cadena de suministro digital.

Palabras clave: fábrica de Software, Propiedad Intelectual, DevSecOps, Vulnerabilidades, OWASP, Zero Trust.

ABSTRACT

This document analyzes the critical cybersecurity risks inherent to the "Software Factory" (Project-based) outsourcing model. While outsourcing offers operational efficiency and access to specialized talent, it introduces specific attack vectors that differ from traditional BPO: intellectual property (IP) exfiltration and the inclusion of vulnerabilities in the delivered source code. A management framework based on the DevSecOps methodology is proposed, integrating automated controls (SAST, DAST, SCA) and "Quality Gates" contractual clauses. The objective is to transition from a blind trust model to one of continuous verification, mitigating exposure to information leaks and digital supply chain attacks.

Keywords: software Factory, Intellectual Property, DevSecOps, Vulnerabilities, OWASP, Zero Trust.

1. INTRODUCCIÓN Y CONTEXTO

La globalización y la transformación digital han impulsado el crecimiento sostenido del outsourcing, permitiendo a las organizaciones delegar funciones operativas para centrarse en su actividad o core principal. Sin embargo, la ciberseguridad en estos entornos se ha transformado en un desafío estratégico, especialmente cuando lo que se externaliza es la construcción de activos digitales.

A diferencia del outsourcing de procesos de negocio tradicionales, como contact centers o gestión documental, la externalización del desarrollo de software, también llamada Fábrica de Software, presenta una complejidad única, el producto entregable es el núcleo tecnológico del negocio. En este modelo, el cliente suele recibir el software final funcionando como una "Caja Negra", desconociendo a menudo las prácticas de seguridad o la ausencia de ellas empleadas durante su construcción.

El presente trabajo tiene un alcance analítico y propositivo basado en un "caso tipo" empresarial. Para abordar esta problemática, se plantean los siguientes objetivos:

1. Analizar los vectores de riesgo específicos en la contratación de fábricas de software, diferenciándolos de los riesgos de infraestructura TI convencional.
2. Proponer un esquema de controles técnicos basado en DevSecOps que permita auditar la calidad del código entregado por terceros.
3. Recomendar lineamientos contractuales y herramientas tecnológicas que garanticen la protección de la propiedad intelectual y la integridad del software.

2. METODOLOGÍA

Para este desarrollo se empleó una metodología cualitativa de tipo analítica propositiva, estructurada en las siguientes fases:

- Revisión Dirigida de Literatura: se analizaron estándares internacionales de ciberseguridad, específicamente ISO/IEC 27001 y el NIST Cybersecurity Framework 2.0, así como guías técnicas de OWASP.

- Análisis de Casos: se estudiaron incidentes de seguridad en la cadena de suministro, tomando como referencia el caso SolarWinds documentado por FireEye, para extrapolar lecciones aprendidas hacia el entorno de desarrollo tercerizado.
- Diseño de Propuesta: se identificaron riesgos clave y se mapearon a controles técnicos y contractuales específicos, integrando conceptos de ingeniería de software segura para formular el modelo de mitigación.

3. CONTEXTO ORGANIZACIONAL Y MARCO CONCEPTUAL

3.1. Contexto del Problema

El análisis se enmarca en organizaciones financieras, de servicios o tecnológicas que delegan el desarrollo de aplicaciones "core" a terceros bajo modalidades outsourcing. En este escenario, la organización cliente transfiere lógica de negocio crítica a proveedores externos, quienes operan con sus propios estándares de seguridad, personal rotativo y dispositivos fuera del control directo del cliente. Esta desconexión crea una superficie de ataque extendida donde el proveedor se convierte en el eslabón más débil.

3.2. Marco Conceptual

Para comprender la propuesta, es necesario definir los conceptos base que articulan la solución:

- DevSecOps: metodología que integra la seguridad como responsabilidad compartida durante todo el ciclo de vida de TI, no solo al final.
- Cadena de Suministro Digital: red de recursos como código, librerías y herramientas necesarios para construir el software. Un ataque aquí implica comprometer los componentes antes de que lleguen al cliente.
- Zero Trust (Confianza Cero): modelo de seguridad que asume que ninguna entidad, usuario o dispositivo, incluso si está dentro del perímetro o es un proveedor contratado, es confiable por defecto.

- Quality Gates: puntos de control automatizados en el flujo de desarrollo que impiden el avance del código si no cumple con ciertos criterios de calidad y seguridad.

4. ANÁLISIS DE RIESGOS ESPECÍFICOS EN FACTORÍAS DE SOFTWARE

Mientras que las amenazas tradicionales en outsourcing incluyen phishing genérico o ransomware, en el desarrollo de software los riesgos más críticos son silenciosos y estructurales.

A continuación, se describen aterrizados a la realidad contractual:

4.1. Fuga de Propiedad Intelectual (IP)

El mayor activo en el desarrollo de software no es solo el dato, sino la lógica de negocio codificada.

- Reutilización no autorizada: existe el riesgo latente de que la fábrica de software reutilice módulos, algoritmos o arquitecturas propietarias (pagadas por el cliente) para acelerar proyectos de empresas competidoras.
- Exfiltración de Código Fuente: sin controles adecuados de prevención de pérdida de datos (DLP) en los endpoints del proveedor, un desarrollador externo puede clonar repositorios completos en nubes personales como GitHub público, Google Drive y otros, comprometiendo la exclusividad de la solución.

4.2. Inyección de Vulnerabilidades y Deuda Técnica

La falta de auditoría sobre el proceso de desarrollo externo deriva en la entrega de software funcionalmente correcto, pero inseguro estructuralmente.

- Vulnerabilidades OWASP en el entregable: por ejemplo, en un contrato para una aplicación bancaria, si el proveedor no válida las entradas, podría entregar código vulnerable a Inyección SQL, permitiendo a futuros atacantes manipular la base de datos del cliente.

- Secretos "Hardcodeados": una práctica negligente común donde los desarrolladores externos dejan credenciales (API Keys de AWS, contraseñas de bases de datos) escritas directamente en el código fuente para agilizar sus pruebas. Si este código se filtra, expone toda la infraestructura del cliente.
- Compromiso de la Cadena de Suministro: como se evidenció en el incidente de SolarWinds, un proveedor vulnerable puede ser un "Caballo de Troya". En el desarrollo, esto ocurre cuando la fábrica utiliza librerías de terceros (Open Source) infectadas o desactualizadas, introduciendo brechas de seguridad sin escribir una sola línea de código malicioso intencional.

5. ESTRATEGIA DE MITIGACIÓN: ENFOQUE DEVSECOPS

Para contrarrestar estos riesgos, se recomienda aplicar el principio de verificación continua, abandonando el modelo de confianza ciega. Esto se logra implementando un "pipeline" de DevSecOps que funciona como una auditoría en tiempo real.

5.1. Implementación de "Aduanas Digitales" (Quality Gates)

Proponemos establecer puntos de control obligatorios que funcionen como aduanas: el código no "entra" a la organización si no pasa la inspección.

- Gate de Commit (Aduana de Entrada): bloquea el guardado de código si herramientas como TruffleHog detectan credenciales expuestas.
- Gate de Build (Aduana de Construcción): detiene la compilación si el análisis estático, detecta vulnerabilidades de severidad Alta o Crítica según el estándar OWASP.
- Gate de Entrega (Aduana de Salida): rechaza el entregable final si el análisis de composición e identifica librerías de terceros con parches de seguridad pendientes.

6. HERRAMIENTAS TECNOLÓGICAS RECOMENDADAS

Más allá de herramientas perimetrales como Firewalls, el control de una fábrica de software requiere auditoría de código profunda.

Se recomienda el siguiente stack tecnológico:

Tabla 1. Herramientas recomendadas para el control de outsourcing de desarrollo.

Categoría	Herramienta Sugerida	Función Crítica en Outsourcing
SAST (Static Application Security Testing)	SonarQube / Checkmarx [13]	Analiza el código fuente ("Caja Blanca") para detectar errores de lógica y vulnerabilidades antes de compilar. Garantiza que el proveedor no entregue "código sucio".
SCA (Software Composition Analysis)	Snyk / OWASP Dependency Check [14]	Escanea las librerías de terceros utilizadas por la fábrica. Mitiga el riesgo de Cadena de Suministro detectando componentes obsoletos.
Secret Scanning	TruffleHog / GitGuardian [15]	Escanea el historial de cambios (commits) buscando contraseñas olvidadas. Previene "puertas traseras" accidentales.
DAST (Dynamic Application Security Testing)	OWASP ZAP / Burp Suite [16]	Ataca la aplicación en ejecución ("Caja Negra") simulando el comportamiento de un hacker externo para verificar la resistencia real del entregable.

IAM (Gestión de Identidad)	Azure AD / Okta	Asegura el Principio de Menor Privilegio, restringiendo el acceso de los desarrolladores externos solo a los recursos necesarios.

7. MARCO LEGAL Y CONTRACTUAL RECOMENDADO

Los Acuerdos de Nivel de Servicio (ANS) deben evolucionar desde la disponibilidad del servicio hacia la seguridad del entregable, alineándose con normativas como la Ley 1581 de 2012 en Colombia.

Cláusulas Esenciales Sugeridas:

1. Propiedad Intelectual Estricta: definición explícita de que todo código y script generado bajo el modelo "Work for hire" es propiedad exclusiva del cliente desde su creación.
2. Derecho a Auditoría de Código: facultad del cliente para auditar los repositorios del proveedor sin previo aviso.
3. Saneamiento de Vulnerabilidades: obligación contractual de entregar el software libre de vulnerabilidades críticas antes de cualquier pago final.
4. Destrucción Certificada: Obligación de eliminar todas las copias del código fuente y datos de prueba de los servidores del proveedor al finalizar el contrato.

8. CONCLUSIÓN

El análisis realizado permite concluir que la ciberseguridad en la externalización de desarrollo de software ya no puede abordarse únicamente desde la protección perimetral o la confianza contractual. La prevalencia de amenazas como la inyección de vulnerabilidades y el robo de propiedad intelectual exige un cambio de paradigma: pasar de un modelo de "Caja Negra", donde se confía ciegamente en el entregable, a un modelo de "Verificación Continua". La implementación de estrategias DevSecOps y controles automatizados (SAST/DAST) demuestra ser el mecanismo más efectivo para garantizar que la agilidad del outsourcing no comprometa la integridad de los activos digitales.

Desde la perspectiva de la práctica profesional de la ingeniería, este trabajo evidencia una evolución en el rol del ingeniero de sistemas y seguridad. El profesional actual no solo debe poseer habilidades técnicas para configurar herramientas de auditoría, sino también una visión estratégica para diseñar Acuerdos de Nivel de Servicio (ANS) que incluyan métricas de calidad de código. El ingeniero se convierte así en un arquitecto de confianza, actuando como el puente necesario entre los requisitos legales del negocio y la realidad técnica del software entregado por terceros.

Finalmente, esta investigación abre la puerta a nuevas líneas de trabajo futuro. Se sugiere profundizar en el estudio de la integración de Inteligencia Artificial para la detección predictiva de vulnerabilidades en código de terceros, así como validar este modelo propuesto mediante un caso de estudio experimental en una organización del sector financiero o salud. Estas futuras investigaciones permitirían medir cuantitativamente el retorno de inversión (ROI) de implementar "Aduanas Digitales", consolidando la seguridad no como un costo, sino como un habilitador indispensable para la competitividad en un mercado globalizado.

9. BIBLIOGRAFÍA

[1] International Organization for Standardization. (2022). ISO/IEC 27001: Information security, cybersecurity and privacy protection. https://www.iso.org/standard/27001

[2] National Institute of Standards and Technology. (2024). The NIST Cybersecurity Framework (CSF) 2.0. U.S. Department of Commerce. https://www.nist.gov/cyberframework

[3] OWASP Foundation. (2021). OWASP Top 10:2021 The Ten Most Critical Web Application Security Risks. https://owasp.org/www-project-top-ten/

[4] SonarSource. (s.f.). SonarQube: Code Quality and Code Security. Recuperado de https://www.sonarqube.org/

[5] Snyk. (s.f.). Snyk Open Source: Software Composition Analysis (SCA). Recuperado de https://snyk.io/product/open-source-security-management/

[6] Truffle Security. (s.f.). TruffleHog: Find leaked credentials. Recuperado de https://trufflesecurity.com/trufflehog

[7] GitGuardian. (s.f.). Automated Secrets Detection. Recuperado de https://www.gitguardian.com/

[8] OWASP Foundation. (s.f.). Zed Attack Proxy (ZAP). Recuperado de https://www.zaproxy.org/

[9] Congreso de la República de Colombia. (2012). Ley Estatutaria 1581 de 2012: Por la cual se dictan disposiciones generales para la protección de datos personales. https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=49981

[10] FireEye. (2020). Highly Evasive Attacker Leverages SolarWinds Supply Chain to Compromise Multiple Global Victims. Mandiant. https://cloud.google.com/blog/topics/threat-intelligence/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor