UDC 004.056.5

DEVELOPMENT OF A SYSTEM FOR MONITORING INFORMATION SECURITY OF SOCIAL NETWORKS

I.G. Tagangylyjov, K.S. Artyklyyev

Oguz han Engineering and Technology University of Turkmenistan, Ashgabat, Turkmenistan,
i.tagangylyjov@etut.edu.tm

Abstract: this paper addresses the design of a monitoring system intended to assess and maintain the information security of social network platforms and their users. The proposed system architecture combines automated content monitoring, account and bot-behavior analysis, malicious-link and phishing detection, and data-leakage tracking into a unified pipeline built on classical rule-based filters and machine-learning classifiers. The paper describes the functional modules of the system, the data flow between them, and the criteria used to prioritize alerts for human moderators. Limitations related to scalability, privacy constraints, and evasion by malicious actors are discussed, along with directions for further development.

Keywords: information security, social networks, security monitoring, bot detection, phishing detection, data leakage, machine learning.

I. INTRODUCTION

Social network platforms accumulate large volumes of personal, behavioral, and communication data, which makes them an attractive target for a wide range of threats: account compromise, phishing and malicious links distributed through posts and messages, automated bot networks used for disinformation or fraud, and unauthorized leakage of user data through third-party applications and application programming interfaces (APIs). Manual moderation cannot keep pace with the scale of user-generated content, which motivates the development of automated monitoring systems capable of continuously assessing the security state of a platform and flagging incidents for timely response [1-5].

The goal of this work is to describe the architecture and core functional components of a system for monitoring the information security of social networks, covering both platform-level threats (compromised accounts, bot activity) and content-level threats (malicious links, data leakage) within a single monitoring pipeline.

II. SYSTEM ARCHITECTURE

The proposed monitoring system consists of four principal modules operating over a shared event stream collected from platform activity logs, posts, messages, and account metadata: (1) a content monitoring module, (2) an account and behavior analysis module, (3) a link and phishing detection module, and (4) a data-leakage tracking module. Each module produces a risk score for the object under analysis (post, account, or outgoing data transfer), and an aggregation layer combines these scores using weighted rules to generate prioritized alerts for security analysts and moderators.

Incoming data is first normalized and passed through a rule-based pre-filter that removes low-risk, well-known benign patterns, reducing the load on the subsequent machine-learning components. This two-stage design allows the system to process high volumes of events while reserving computationally expensive analysis for content flagged as potentially suspicious.

III. CONTENT AND BEHAVIOR MONITORING

The content monitoring module applies natural-language processing techniques, including keyword and pattern matching combined with text classification models, to detect posts and messages containing harassment, hate speech, disinformation, or coordinated inauthentic messaging. Classification models are trained on labeled examples of policy-violating content and are periodically retrained to adapt to evolving language patterns used to evade detection.

The account and behavior analysis module evaluates account-level features such as posting frequency, temporal patterns, network of connections, and similarity to known bot signatures. Graph-based analysis of the social connection network helps identify clusters of accounts exhibiting coordinated behavior, which is characteristic of bot networks and inauthentic amplification campaigns, while anomaly-detection models flag individual accounts whose behavior deviates sharply from established user baselines, a pattern often associated with account takeover.

IV. LINK, PHISHING, AND DATA-LEAKAGE MONITORING

The link and phishing detection module inspects URLs shared within the platform against reputation databases, lexical features of the URL string, and, where feasible, sandboxed analysis of the linked page's

content and redirect chain, allowing detection of both known and newly registered phishing domains before they reach a significant number of users.
The data-leakage tracking module monitors data flows to third-party applications connected through the platform's API, comparing requested data scopes against the platform's data-sharing policy and flagging integrations that request excessive permissions or exhibit unusual data-export volume, which may indicate misuse of user data beyond its intended purpose.

V. ALERT AGGREGATION AND PRIORITIZATION

Risk scores produced by the four modules are combined by an aggregation layer that accounts for the severity, confidence, and potential reach of a detected issue – for example, a phishing link shared by a highly connected account is prioritized above the same link shared by an isolated one. Aggregated alerts are routed to human moderators through a triage queue, and confirmed incidents are fed back into the training data of the relevant module, enabling continuous improvement of detection accuracy over time.

VI. LIMITATIONS AND CHALLENGES

Several challenges affect the practical deployment of such a system. First, scalability is a concern, since large platforms generate content at a rate that requires efficient, often distributed, real-time processing infrastructure. Second, privacy regulations constrain the extent to which user data and communications can be automatically analyzed, requiring careful design of data minimization and access-control measures within the monitoring pipeline itself. Third, malicious actors continuously adapt their tactics to evade detection, including the use of adversarially crafted content and slowly evolving bot behavior designed to mimic genuine users, which necessitates ongoing model retraining and periodic manual review of detection rules. Finally, an overly aggressive system risks a high rate of false positives, which can burden moderators and negatively affect legitimate users, making the calibration of alert thresholds an important design consideration.

VII. CONCLUSION

The described system combines content monitoring, behavioral analysis, phishing detection, and data-leakage tracking into a unified pipeline for maintaining the information security of social network platforms. A two-stage architecture, in which lightweight rule-based filtering precedes machine-learning-based analysis, allows the system to operate at scale while retaining the ability to detect complex, evolving threats. Future work should focus on improving robustness against adversarial evasion, refining alert prioritization to reduce moderator workload, and extending the architecture to comply with evolving data-protection requirements.

REFERENCES

[1] Fire, M., Goldschmidt, R., Elovici, Y. Online social networks: Threats and solutions // IEEE Communications Surveys & Tutorials. – 2014. – Vol. 16, No. 4. – P. 2019–2036.
[2] Cresci, S. A decade of social bot detection // Communications of the ACM. – 2020. – Vol. 63, No. 10. – P. 72–83.
[3] Varol, O., Ferrara, E., Davis, C., Menczer, F., Flammini, A. Online human-bot interactions: Detection, estimation, and characterization // Proceedings of the International AAAI Conference on Web and Social Media. – 2017.
[4] Aleroud, A., Zhou, L. Phishing environments, techniques, and countermeasures: A survey // Computers & Security. – 2017. – Vol. 68. – P. 160–196.
[5] Beigi, G., Liu, H. A survey on privacy in social media: Identification, mitigation, and applications // ACM Transactions on Data Science. – 2020. – Vol. 1, No. 1. – P. 1–38.

РАЗРАБОТКА СИСТЕМЫ МОНИТОРИНГА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ СОЦИАЛЬНЫХ СЕТЕЙ

Тагангылыджов И.Г., Артыклыев К.С.

Огузханский инженерно-технологический университет Туркменистана, Ашхабад, Туркменистан

Аннотация: в статье рассматривается разработка системы мониторинга, предназначенной для оценки и поддержания информационной безопасности платформ социальных сетей и их пользователей. Предложенная архитектура объединяет автоматизированный мониторинг контента, анализ поведения аккаунтов и ботов, обнаружение вредоносных ссылок и фишинга, а также отслеживание утечек данных в единый конвейер на основе классических правил и классификаторов машинного обучения. Описаны функциональные модули системы, потоки данных между ними и критерии приоритизации оповещений для модераторов. Рассмотрены ограничения, связанные с масштабируемостью, требованиями конфиденциальности и попытками обхода системы злоумышленниками, а также направления дальнейшего развития.

Ключевые слова: информационная безопасность, социальные сети, мониторинг безопасности, обнаружение ботов, обнаружение фишинга, утечка данных, машинное обучение.

УДК 004.056.52

АЛГОРИТМ ВНЕДРЕНИЯ ВОДЯНЫХ ЗНАКОВ В VHDL-ОПИСАНИЯ ЦИФРОВЫХ УСТРОЙСТВ, РЕАЛИЗУЕМЫХ НА FPGA

Лавренова А.С., Иванюк А.А.

Белорусский государственный университет информатики и радиоэлектроники, Минск, Республика Беларусь, anastasialavrenova566@gmail.com

Аннотация: в работе представлен подход к автоматизированному исследованию устойчивости аппаратных водяных знаков, внедряемых в параметры INIT LUT-блоков FPGA, к синтезу и оптимизации в САПР [1-2]. Ключевым результатом исследования является разработка программного средства, позволяющего автоматизировать процесс защиты интеллектуальной собственности, а также детально изучать механизмы сохранения скрытой информации на этапах синтеза, размещения и трассировки. Для повышения криптостойкости водяного знака предложен метод псевдослучайного распределения битов по свободным ячейкам памяти конфигурации LUT-блоков с использованием LFSR. Эксперименты выявили уязвимость водяного знака к оптимизации САПР и показали, что для его сохранения необходимо применять специальные приёмы обеспечения устойчивости.

Ключевые слова: аппаратные водяные знаки, VHDL, FPGA, САПР, LUT-блок, LFSR.

I. ВВЕДЕНИЕ

Разработка цифровых устройств на языке VHDL с последующей реализацией в программируемых логических интегральных схемах (ПЛИС) требует значительных интеллектуальных затрат. Традиционные правовые механизмы не всегда предотвращают несанкционированное копирование, обратное проектирование и модификацию проектных описаний [3]. В связи с этим актуальной задачей становится создание эффективных методов защиты HDL-описаний, среди которых особое место занимают аппаратные водяные знаки, которые позволяют скрытно внедрить идентификационную информацию в структуру устройства.

Основной проблемой внедрения водяных знаков через модификацию INIT-параметров LUT-блоков является их уязвимость к процессу синтеза в системах автоматизированного проектирования (САПР). Синтезатор оптимизирует схему, удаляя избыточные логические связи, что может привести к стиранию скрытой идентификационной информации. В рамках исследования проведена проверка того, как оптимизации САПР влияют на сохранность водяных знаков. В качестве инструмента решения задачи разработано программное средство, обеспечивающий автоматизированный анализ VHDL-кода, внедрение водяных знаков и их защиту от оптимизации САПР.

II. ОБОСНОВАНИЕ ПРЕДЛАГАЕМОГО СПОСОБА ВНЕДРЕНИЯ ВОДЯНЫХ ЗНАКОВ

Базовым элементом комбинационной логики в архитектурах FPGA являются LUT-блоки. LUT-блок представляет собой конфигурируемое статическое оперативное запоминающее устройство (СОЗУ), где логическая функция определяется таблицей истинности [4], задаваемой параметром инициализации INIT. Для LUT-блока с n-входами размер параметра INIT составляет 2^n бит.

Водяной знак (watermark, далее – WM) – это скрытая идентификационная информация, внедряемая в цифровой объект для подтверждения прав интеллектуальной собственности. В предлагаемом подходе в качестве контейнера для водяного знака используются недостижимые (свободные) адреса таблицы истинности LUT-блока, не влияющие на реализуемую логическую функцию. Анализ VHDL-проектов показал, что среднее количество LUT2-блоков относительно всех LUT-блоков составляет 24,2%, LUT3 – 20,2%, LUT4 – 24,2%, LUT5 – 16,1%, LUT6 – 15,3%. Таким образом среднее количество свободного места в проектах составило 2640 бит, что показывает возможность внедрения водяного знака соответствующей длины без использования дополнительных аппаратных ресурсов.

При реализации логической функции с числом входов меньшим, чем присутствует в выбранном LUT-блоке часть адресного пространства оказывается функционально неиспользуемой. Количество недостижимых (свободных) адресов, которые могут быть использованы для скрытого хранения данных без нарушения работоспособности схемы, вычисляется по формуле:

$$C_{wm} = 2^n - 2^m, \quad (1)$$

где C_{wm} – ёмкость контейнера водяного знака в LUT-блоке, m - количество входов которые требуются для реализации целевой логической функции.

На рисунке 1 показан базовый принцип использования свободных адресов таблицы истинности LUT4-блока для скрытого хранения данных без изменения функционального поведения схемы.

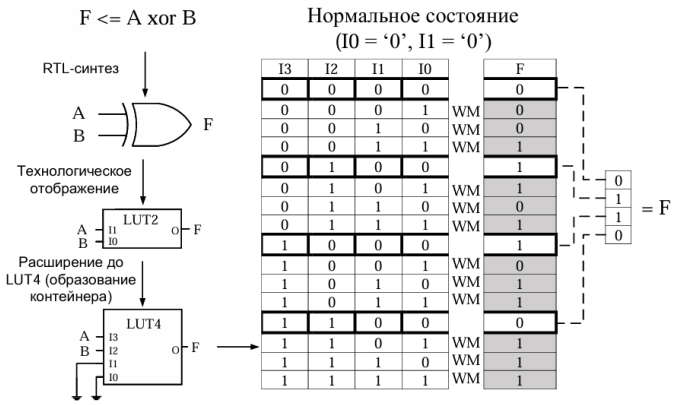


Рисунок 1. Принцип использования свободных ячеек LUT4-блока для внедрения водяного знака

Водяной знак может быть любой размерности, не превышающей максимальную ёмкость контейнера. Простое линейное расположение битов по возрастанию адресов уязвимо к анализу, так как делает закономерности внедрения предсказуемыми для злоумышленника. Для повышения криптостойкости метода внедрения и устойчивости к атакам на обнаружение знака используется механизм псевдослучайной перестановки индексов свободных ячеек на основе линейного регистра сдвига с обратной связью (LFSR) [5]. LFSR, инициализированный секретным ключом и произвольный примитивный неприводимый полином обратной связи, генерирует последовательность псевдослучайных чисел, используемую для построения перестановки. Алгоритм внедрения водяного знака с использованием LFSR проиллюстрировано на рисунке 2.



Рисунок 2. Алгоритм внедрения водяного знака с использованием LFSR

Алгоритм внедрения включает анализ VHDL-кода, определение свободных ячеек LUT-блока и генерацию псевдослучайной перестановки индексов на основе LFSR с заданным ключом и полиномом. Для определения множества $\boldsymbol{A}$ достижимых адресов ячеек памяти параметра INIT используется следующая формула:

$$\boldsymbol{A} = (j \mid \boldsymbol{OC}(j) = 1), j \in \{0, 1, \dots, 2^n - 1\}, |\boldsymbol{A}| = 2^{HW(\boldsymbol{OP})}, \quad (2)$$

где n – число входов LUT-блока; $\boldsymbol{OP} = (p^{n-1}, \dots, p^1, p^0)$ – вектор занятости входов LUT-блока; $p^i \in \{0,1\}$, причём $p^i = 1$ означает, что вход i используется, а $p^i = 0$ – что вход зафиксирован константой; $HW(\boldsymbol{OP})$ – вес Хэмминга вектора $\boldsymbol{OP}$, равный числу используемых входов; j – номер ячейки памяти конфигурации LUT-блока (параметр INIT); $\boldsymbol{OC}$ – вектор занятости памяти конфигурации LUT-блока; $\boldsymbol{OC}(j) \in \{0,1\}$ – признак занятости ячейки, где $\boldsymbol{OC}(j) = 1$ соответствует достижимому адресу, а $\boldsymbol{OC}(j) = 0$ – свободному адресу. В результате множество $\boldsymbol{A}$ содержит номера всех достижимых ячеек INIT, количество которых равно $2^{HW(\boldsymbol{OP})}$.

Адреса, не вошедшие в этот вектор, образуют множество свободных ячеек, которое и используется в качестве контейнера для водяного знака.

III. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ И АНАЛИЗ РЕЗУЛЬТАТОВ ИССЛЕДОВАНИЯ

Эксперименты проводились на тестовом проекте – 8-входовом мультиплексоре, реализованном на трех блоках LUT6 в среде Xilinx Vivado 2021.1. Третий блок LUT имеет три входа, фиксированные в логическом нуле, что предоставляет 56 свободных ячеек для внедрения знака. В качестве тестового водяного знака использовалась 16-битная последовательность 0001001000110100 (hex: 0x1234). Первый эксперимент показал, что при внедрении знака без защитных атрибутов синтезатор Vivado выполняет агрессивную оптимизацию: блок LUT6 с фиксированными входами сокращается до LUT3, а

недостижимые адреса удаляются. В результате водяной знак теряется полностью. Для сохранения знака исследовались три метода маскирования, исключающих оптимизацию LUT-блока:

1. Атрибут DONT_TOUCH: запрещает синтезатору структурные преобразования блока.

2. Ложные сигналы: замена констант '0' на выражения вида $d(i)\ xor\ d(i)$ с атрибутом DONT_TOUCH, что маскирует факт фиксированных входов.

3. Генератор константных значений (CVG – Constant Value Generator [2]): введение последовательностной логики для усложнения структуры схемы.

Структурные схемы синтезированных устройств для указанных конфигураций приведены на рисунках 3 и 4. Для наглядности на схемах скрыты буферы ввода-вывода (IBUF, OBUF), автоматически вставляемые САПР, логическая структура и параметры LUT-блоков отображены без изменений.

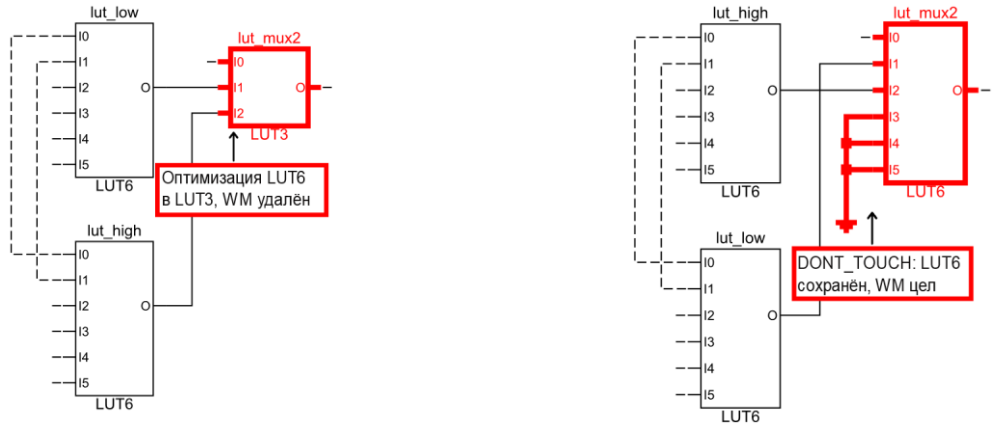


Рисунок 3. Схема после синтеза без метода защиты и с атрибутом DONT_TOUCH

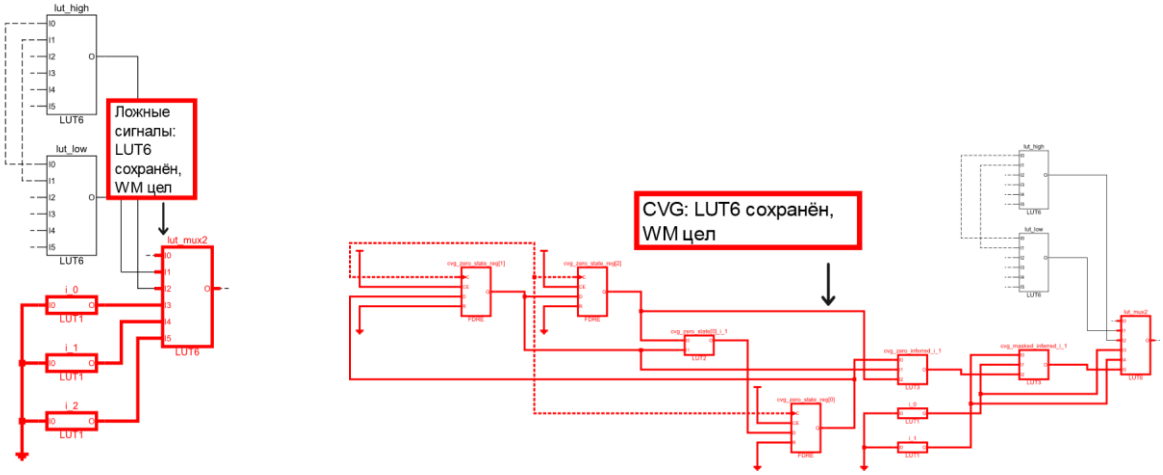


Рисунок 4. Схема после синтеза с использованием ложных сигналов и CVG

Сравнительный анализ влияния различных методов защиты на аппаратные ресурсы приведен в таблице 1.

Таблица 1. Сравнительный анализ ресурсоемкости и энергопотребления

Вариант реализации	Кол-во LUT-блоков	Типы LUT-блоков	Триггеры	Полная мощность (Вт)
Без защиты	3	2 × LUT6, 1 × LUT3	0	1,657
DONT_TOUCH	3	3 × LUT6	0	1,657
Ложные сигналы	6	3 × LUT6, 3 × LUT1	0	1,654
CVG	8	3 × LUT6, 2 × LUT3, LUT2, 2 × LUT1	3	1,711

Результаты показывают, что:

1. Метод DONT_TOUCH не увеличивает ресурсы, но структура схемы остается прозрачной для анализа, что снижает степень скрытности знака.

2. Метод с ложными сигналами удваивает число LUT-блоков (добавляются LUT1 для генерации нулей), но эффективно маскирует константы, усложняя обратное проектирование.

3. Использование CVG переводит часть схемы в синхронный режим, увеличивая потребление энергии до 1,711 Вт из-за переключательной активности триггеров, однако обеспечивает наивысшую сложность анализа структуры для злоумышленника.

Во всех защищенных конфигурациях функциональная эквивалентность сохранялась, а максимальная рабочая частота не снижалась критически. Экспериментально подтверждено, что простой метод внедрения уязвим к оптимизациям САПР, тогда как предложенные механизмы маскирования обеспечивают сохранность знака с приемлемыми накладными расходами.

IV. ЗАКЛЮЧЕНИЕ

В ходе исследования установлено, что внедрение водяных знаков в VHDL-описания через модификацию INIT-параметров LUT-блоков является уязвимым к оптимизациям современных САПР без применения дополнительных защитных мер. Простое внедрение приводит к удалению скрытой информации на этапе синтеза. Разработанный программный модуль позволил автоматизировать процесс оценки устойчивости знака и доказать, что эффективным решением является использование механизмов маскирования. Результаты работы могут быть использованы для защиты HDL-проектов при разработке сложных цифровых устройств.

БЛАГОДАРНОСТЬ

Авторы выражают искреннюю благодарность резиденту ПВТ компании «Инженерный Центр Ядро», которая является одним из центров разработки YADRO, за предоставленное оборудование для проведения экспериментов в рамках работы совместной учебной лаборатории с Белорусским государственным университетом информатики и радиоэлектроники.

ЛИТЕРАТУРА

[1] Проектирование встраиваемых цифровых устройств и систем / А. А. Иванюк. Минск, 2012.

[2] Обзор методов реализации аппаратных водяных знаков в цифровых устройствах программируемой логики / В. В. Сергейчик, А. А Иванюк. Минск, 2015.

[3] Majzoobi, M. Introduction to hardware security and trust / M. Majzoobi, F. Koushanfar, M. Potkonjak. N.Y.: Springer, 2011.

[4] Метод внедрения цифровых водяных знаков в аппаратные контейнеры с LUT-ориентированной архитектурой / К. Защелкин, Е. Иванова // Информатика и моделирование, 2013.

[5] Digital Watermarking and Steganography / I. Cox [et al.] – Burlington: Elsevier, 2008.

ALGORITHM FOR EMBEDDING WATERMARKS IN VHDL DESCRIPTIONS OF DIGITAL DEVICES IMPLEMENTED ON FPGA

A.S. Lavrenova, A.A. Ivaniuk

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus,
anastasialavrenova566@gmail.com

Abstract: this paper presents an approach to the automated study of the robustness of hardware watermarks embedded in the parameters INIT of FPGA LUT-blocks against synthesis and optimization in CAD [1-2]. The key result of the research is the development of specialized tools that automate the process of intellectual property protection and also enable a detailed study of the mechanisms for preserving hidden information during synthesis, placement, and routing. To increase the cryptographic strength of the watermark, a method of pseudorandom distribution of bits among free cells of LUT-blocks using an LFSR is proposed. Experiments revealed the vulnerability of the watermark to CAD optimization and showed that special techniques for ensuring robustness must be applied to preserve it.

Keywords: hardware watermarks, VHDL, FPGA, CAD, LUT-block, LFSR.