HTTPS://DOI.ORG/10.25559/SITITO.022.202602.285-299
УДК 004.056.5
Оригинальная статья

# Прототип инструментальных средств для создания систем обмана на основе онтологического подхода

**А. С. Турашев*, В. А. Сухомлин**
ФГБОУ ВО «Московский государственный университет имени М. В. Ломоносова», г. Москва, Российская Федерация
Адрес: 119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1
* turashev.artem@mail.ru

**Аннотация**
Технологии киберобмана рассматриваются как перспективное направление активной киберзащиты, поскольку они позволяют выявлять нарушителя через управляемое взаимодействие с ложными объектами: приманочными узлами, сервисами, учетными записями и данными. Однако практическое применение таких технологий ограничивается трудоемкостью проектирования реалистичных ловушек, необходимостью поддерживать их согласованность с изменяющейся инфраструктурой. В статье предложен онтологический подход к созданию систем киберобмана и описан прототип инструментальных средств, обеспечивающий переход от формальной модели предметной области к автоматически создаваемой контейнерной обманной среде. Экземпляры модели параметризуются значениями, соответствующими эталонному поведению инфраструктуры. Показано, что онтологическая модель может использоваться как единый источник знаний для проектирования, конфигурирования, развертывания и последующей реконфигурации deception-сред.
Предложенный подход повышает интерпретируемость проектирования cyber deception-сред, снижает объем ручной настройки, создает основу для проверки правдоподобия ловушек и делает возможной адаптивную перестройку обманной инфраструктуры при изменении защищаемой системы.

**Ключевые слова:** онтология, киберобман, контейнеризация, эталонное поведение, автоматическое развертывание, активная киберзащита, правдоподобие ловушек, реконфигурация

**Конфликт интересов:** авторы заявляют об отсутствии конфликта интересов.

**Для цитирования:** Турашев А. С., Сухомлин В. А. Прототип инструментальных средств для создания систем обмана на основе онтологического подхода // Современные информационные технологии и ИТ-образование. 2026. Т. 22, № 2. С. 285-299. https://doi.org/10.25559/SITITO.022.202602.285-299

© Турашев А. С., Сухомлин В. А., 2026

Контент доступен под лицензией Creative Commons Attribution 4.0 License.
The content is available under Creative Commons Attribution 4.0 License.

Original article

# Prototype of Toolset for Creating Deception Systems Based on an Ontological Approach

**A. S. Turashev*, V. A. Sukhomlin**
Lomonosov Moscow State University, Moscow, Russian Federation
Address: 1 Leninskie gory, Moscow 119991, GSP-1, Russian Federation
* turashev.artem@mail.ru

## Abstract

Cyber deception technologies are considered a promising area of active cyber defense, as they enable intruder detection through controlled interaction with decoy objects: decoy nodes, services, accounts, and data. However, the practical application of such technologies is limited by the complexity of designing realistic honeypots and the need to maintain their consistency with a changing infrastructure. This article proposes an ontological approach to creating cyber deception systems and describes a prototype toolkit that enables the transition from a formal domain model to an automatically generated containerized deception environment. Model instances are parameterized by values corresponding to the reference behavior of the infrastructure. It is demonstrated that the ontological model can be used as a single source of knowledge for the design, configuration, deployment, and subsequent reconfiguration of deception environments. The proposed approach improves the interpretability of cyber deception environment design, reduces the amount of manual configuration, creates a basis for testing the plausibility of honeypots, and enables adaptive restructuring of the deception infrastructure as the protected system changes.

**Keywords:** cyber deception, honeypot, ontology, deception systems, containerization, baseline behaviour, automatic deployment, active cyber defence, decoy plausibility, reconfiguration

**Conflict of interests:** The authors declare no conflict of interest.

**For citation:** Turashev A.S., Sukhomlin V.A. Prototype of Toolset for Creating Deception Systems Based on an Ontological Approach. *Modern Information Technologies and IT-Education.* 2026;22(2):285-299. https://doi.org/10.25559/SITITO.022.202602.285-299

## Введение: актуальность проблемы и современные тенденции киберзащиты

Киберобман *(cyber deception)* относится к классу активных методов защиты, в которых защитник намеренно формирует для злоумышленника искаженную картину цифровой инфраструктуры и в дальнейшем блокирует нежелательные действия и анализирует уже произошедшие события. В такой картине присутствуют ложные узлы, сервисы, учетные записи, файлы, базы данных и иные элементы, с которыми легитимный пользователь обычно не должен взаимодействовать [2], [5-8].

В отличие от традиционных средств обнаружения вторжений, которые часто опираются на сигнатуры, правила корреляции или ретроспективный анализ журналов, в *deception*-подходе создается управляемая среда наблюдения, в которой действия нарушителя можно фиксировать. Если злоумышленник взаимодействует с ложным объектом, защитник получает не только тревожное событие, но и контекст: какой объект привлек внимание, какие команды или сетевые запросы были выполнены, какую цель мог преследовать нарушитель.

Актуальность технологии *cyber deception* усиливается изменением характера корпоративных инфраструктур. Современная сеть редко является статичным набором серверов и рабочих станций. В ней используются виртуальные машины, облачные сервисы, микросервисные приложения, программно-конфигурируемые сети и устройства интернета вещей. В результате реальная поверхность атаки становится более широкой. Если *deception*-среда не обновляется вместе с инфраструктурой, она начинает отличаться от реальных узлов по версиям сервисов, структуре каталогов, версиям логов. Все это снижает ценность ловушек и повышает риск распознавания ложных объектов.

Дополнительную сложность создает развитие методов обнаружения *honeypot*-систем. Исследования показывают, что злоумышленники могут выявлять ловушки по статическим баннерам, необычным ответам на некорректные пакеты, шаблонным сообщениям об ошибках, нетипичным таймингам, признакам виртуализации и повторно используемым конфигурациям [1]. Следовательно, эффективная система киберобмана должна быть не просто набором поднятых сервисов, а правдоподобной, семантически согласованной и регулярно актуализируемой моделью фрагмента инфраструктуры.

Проблема состоит в том, что проектирование обманной среды часто выполняется вручную. Специалист определяет набор приманок, сопоставляет их с реальными активами, выбирает сервисы, описывает сетевые параметры, готовит конфигурации и задает сценарии поведения. При изменении защищаемой инфраструктуры эти действия повторяются. В условиях большого числа сегментов, разнородных сервисов и ограниченного времени специалистов такой подход становится трудоемким и трудно воспроизводимым.

Онтологический подход позволяет рассматривать *deception*-систему как предметную область с явно заданными классами, отношениями, свойствами и экземплярами [9-11]. Инфраструктурный узел в такой модели описывается не набором разрозненных параметров, а как связанная система понятий: тип узла, операционная система, процессы, файловая активность, сетевое поведение, пользовательские действия, журналы событий, программные компоненты и логические признаки. Это делает модель понятной для эксперта, пригодной для автоматической обработки и расширяемой при появлении новых типов активов или источников телеметрии.

Центральная идея настоящей статьи состоит в применении онтологии не только как справочника терминов или средства последующего анализа, но и как операционного источника данных для создания обманной среды. Индивиды онтологии рассматриваются как прототипы будущих инфраструктурных узлов, скалярные свойства задают параметры поведения, а отношения между сущностями определяют структуру и ограничения создаваемой среды. Такой подход позволяет связать концептуальную модель, эталонное поведение и автоматическое развертывание в единый технологический контур.

## Обзор литературы

Разработка прототипа инструментальных средств для создания систем обмана опирается на исследования в трех ключевых направлениях: онтологическое моделирование, технологии *cyber deception* и методы повышения правдоподобия обманных сред. В данном разделе рассмотрены наиболее значимые источники, которые задают теоретическую и методическую основу предлагаемого подхода.

В контексте проблемы обнаружения ловушек особое значение имеет работа A. Chaudhry, C. Andersen, G. Choudhary, N. Dragoni “A Review of Honeypots: Fingerprinting Techniques, Detection, and Evasion Mechanisms” [1]. В ней рассматриваются методы распознавания *honeypot*-систем злоумышленниками. Авторы выделяют такие признаки обнаружения ловушек, как статические баннеры, ограниченная реализация протоколов, шаблонные сообщения об ошибках и особенности конфигурации. Для темы статьи этот источник особенно важен, поскольку он показывает: при создании систем обмана недостаточно автоматически развернуть ложный сервис, необходимо обеспечить его сетевую, поведенческую и системную правдоподобность.

В качестве базовой работы по *honeypot*-технологиям следует выделить книгу L. Spitzner “Honeypots: Tracking Hackers” [2]. В ней *honeypot* рассматривается как ложный ресурс, предназначенный для привлечения атакующего и наблюдения за его действиями. Основное значение этой работы для темы исследования состоит в формулировании базовой идеи киберобмана: ловушка должна быть достаточно правдоподобной, чтобы нарушитель воспринимал ее как реальный объект инфраструктуры. Это требование напрямую связано с необходимостью формального описания эталонного поведения и параметров ложных узлов.

С точки зрения автоматизированного развертывания ложных объектов важна работа N. Provos “A Virtual Honeypot Framework” [3], [25-29]. В ней рассматривается подход к созданию виртуальных *honeypot*-систем. Этот источник важен тем, что связывает идею *honeypot* с автоматизированным развертыванием виртуальных объектов. Для предлагаемого прототипа это особенно существенно, поскольку задача исследования состоит не только в описании *deception*-среды, но и в переходе от формальной модели к практически создаваемым инфраструктурным объектам.

Для понимания современных направлений развития *cyber deception* следует обратиться к работе A. Javadpour, F. Jafari, T.

Taleb, M. Shojafar, C. Benzaid "A Comprehensive Survey on Cyber Deception Techniques to Improve Honeypot Performance" [4]. В ней рассматриваются современные техники *cyber deception*, направленные на повышение эффективности *honeypot*-систем. Авторы анализируют типы ловушек, *deception*-стратегии и подходы к оценке эффективности. Для настоящего исследования данный источник важен тем, что показывает переход от отдельных *honeypot*-узлов к более сложным *deception*-средам, требующим автоматизации проектирования, параметризации и сопровождения.

Теоретическую основу онтологического подхода задает работа T. R. Gruber "A Translation Approach to Portable Ontology Specifications" [9]. В ней онтология рассматривается как формальная спецификация понятий предметной области. Для настоящего исследования эта работа важна тем, что задает базовое понимание онтологии как модели, пригодной для описания инфраструктурных объектов, их свойств и взаимосвязей. Это позволяет использовать онтологию не только как словарь терминов, но и как основу для автоматизированного проектирования *deception*-среды.

Практические принципы построения онтологий раскрываются в работе[1], в которой описывается процесс выделения классов, отношений, свойств и экземпляров. Данный источник важен для статьи, поскольку показывает, как перейти от неформального описания сетевой инфраструктуры к структурированной модели, на основе которой можно формировать профили ложных узлов, сервисов, процессов, файловой активности и журналов событий.

Применение онтологий непосредственно в задачах кибербезопасности рассматривается в работе D. Preuveneers, W. Joosen "An Ontology-Based Cybersecurity Framework for AI-Enabled Systems and Applications" [13-16]. Авторы показывают, как онтологический подход может применяться для формального описания угроз, контрмер и уязвимостей в системах кибербезопасности. Данный источник важен тем, что подтверждает применимость онтологий не только для классификации понятий, но и для поддержки анализа, логического вывода и автоматизации решений в киберзащите. Это методологически близко к использованию онтологии как ядра инструментальных средств для создания *cyber deception*-сред.

Общий вывод по существующим исследованиям состоит в том, что в литературе достаточно подробно рассмотрены отдельные аспекты: принципы *honeypot* и *cyber deception*, подходы к виртуализации ловушек, проблема их обнаружения злоумышленником и теория онтологического моделирования. Однако, вместе с тем сохраняется разрыв между формальным описанием *deception*-среды и ее автоматическим развертыванием. Большинство работ рассматривает либо концептуальные основы киберобмана, либо отдельные *honeypot*-решения, либо онтологии как средство анализа и классификации. Поэтому актуальной остается задача разработки прототипа инструментальных средств, в котором онтологическая модель используется как операционная основа для проектирования, параметризации и создания систем обмана.

## Теоретический анализ технологии cyber deception и проблемы правдоподобия ловушек

*Cyber Deception* – централизованная система, предназначенная для управления сетевыми ловушками. Системы DDP (*Distributed Deception Platform*, платформа для создания распределённой инфраструктуры ложных целей) обеспечивают развертывание сетевых ловушек в инфраструктуре, сбор и анализ данных, поступающих с этих ловушек, а также предоставление результатов пользователю системы [4], [6, 7].

Цель использования DDP – отвлечь внимание злоумышленника с реальной инфраструктуры на себя. Необходимо, чтобы ловушки являлись наиболее правдоподобными и имитировали реальные объекты, такие как базы данных, веб-серверы, сервисы удаленного управления. Таким образом, почти обязательным стало эмулирование работы таких протоколов и служб, как *HTTP/HTTPS, RDP, SSH, SMB, memcash, proxy, pptp, mqtt, mysql, mssql, mongo* и др. Это даёт возможность эффективно вписываться в инфраструктуру государственных, финансовых или промышленных организаций [2-4].

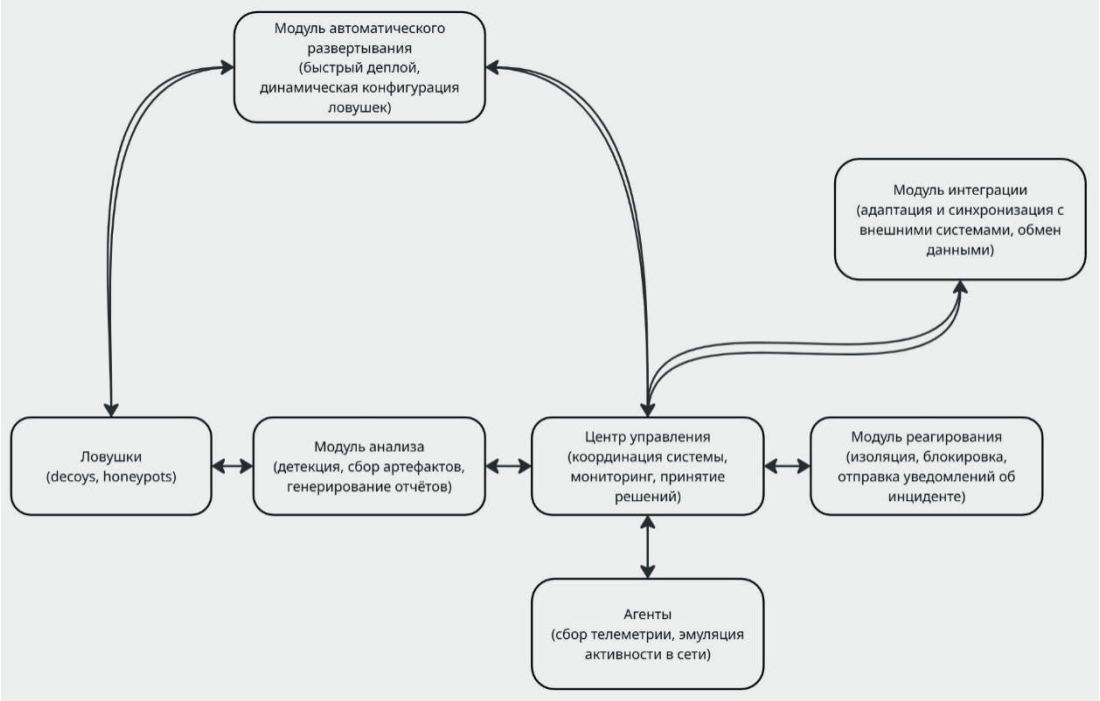
Основными преимуществами технологии *Cyber Deception* являются несколько её характерных особенностей:

- Во-первых, система позволяет выявлять атаки с использованием новейших или неизвестных инструментов, например, 0-day, то есть создаёт условия для проактивной защиты [5, 6], [8].
- Во-вторых, *Cyber Deception* – системы не задействованы в бизнес-процессах организации. Поскольку они не выполняют никаких полезных функций в обычной деятельности, любые попытки взаимодействия с ними считаются подозрительными. Это позволяет системе с достаточно высокой точностью обнаружить попытки вторжения или несанкционированного доступа, так как обращение к этим ложным объектам является явным признаком возможной угрозы [2], [5].

Современная *deception*-платформа обычно состоит из нескольких логических компонентов. Первый компонент – набор ложных объектов, или ловушек: сетевые узлы, сервисы, учетные записи, файлы, базы данных, маркеры доступа и иные приманочные элементы, предназначенные для привлечения внимания нарушителя и фиксации его действий. Второй компонент – центр управления, который определяет общую конфигурацию *deception*-среды: какие ловушки должны быть развернуты, в каких сегментах инфраструктуры они размещаются, какие параметры им задаются и каким образом они должны обновляться. Третий компонент – агенты, размещаемые в отдельных сегментах инфраструктуры. Они обеспечивают локальное взаимодействие с развернутыми ловушками, контролируют их состояние, собирают первичную телеметрию и передают данные в центр управления. Четвертый компонент – средства сбора и анализа событий, которые обрабатывают обращения к приманкам, сетевые запросы, попытки аутентификации, действия атакующего и другие признаки взаимодействия с обманной средой. Пятый компонент – механизм реагирования, передающий значимые события в SOC, SIEM или

[1] Noy N. F., McGuinness D. L. Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory, Technical Report KSL-01-05, 2001. 25 p. [Электронный ресурс]. URL: https://protege.stanford.edu/publications/ontology_development/ontology101.pdf (дата обращения: 15.04.2026).

SOAR-систему и связывающий обнаруженное взаимодействие с процедурами расследования и реагирования на инциденты. Шестой компонент – модуль автоматического развертывания и обновления. Он отвечает за преобразование описания обманной среды в исполняемые инфраструктурные объекты, их запуск, пересоздание и актуализацию при изменении модели или параметров защищаемой инфраструктуры. Такая архитектура позволяет рассматривать *deception*-платформу не как набор отдельных honeypot-узлов, а как управляемую систему, в которой ловушки создаются, контролируются, анализируются и обновляются в едином технологическом контуре.




Р и с. 1. Диаграмма архитектуры Cyber Deception систем
F i g. 1. Architecture Diagram of Cyber Deception Systems

*Источник:* здесь и далее в статье все таблицы и рисунки составлены авторами.
*Source:* Hereinafter in this article all tables and figures were made by the authors.

Главный фактор ценности всей системы – неотличимость от реального актива в тех условиях, в которых действует нарушитель. Если атакующий распознает ловушку, он может прекратить взаимодействие, изменить тактику и обходить все сходные приманки в сети. Поэтому правдоподобие должно рассматриваться как некий заранее обдуманный концепт.

Например, ложный корпоративный веб-сервер должен иметь согласованный набор портов, сервисов, файлов, журналов, фоновой активности и сетевых задержек. Если он отвечает как веб-сервер, но не имеет характерных логов, не создает типичных служебных файлов, использует одинаковые сообщения об ошибках и демонстрирует слишком регулярное время отклика, опытный нарушитель может предположить, что объект является ловушкой. Аналогично рабочая станция должна отличаться от сервера: для нее важны пользовательские логины, следы работы приложений, изменения в пользовательских каталогах, обращения к внутренним ресурсам и периодическая активность в рабочее время.

Исследования методов распознавания *honeypot*-систем подчеркивают, что атакующие используют как активное, так и пассивное зондирование. При активном зондировании отправляются специально подготовленные запросы или некорректные пакеты, чтобы вызвать характерную реакцию системы. При пассивном зондировании анализируются уже наблюдаемые признаки: сертификаты, временные интервалы, последовательности ответов, параметры сетевого стека. В обоих случаях ловушка уязвима, если она реализована слишком упрощенно или не согласована с контекстом реальной инфраструктуры [1].

Отсюда следует важное инженерное требование: *deception*-среда должна проектироваться не как набор изолированных сервисов, а как модель фрагмента инфраструктуры. Такая модель должна отвечать на вопросы: какой тип узла имитируется, какие процессы на нем типичны, какие файлы появляются в результате работы, какие сетевые соединения ожидаемы, какие пользователи взаимодействуют с узлом, какие события пишутся в журналы и какие признаки могут раскрыть искусственность среды. Именно это делает онтологический подход естественным кандидатом для формализации *deception*-систем.

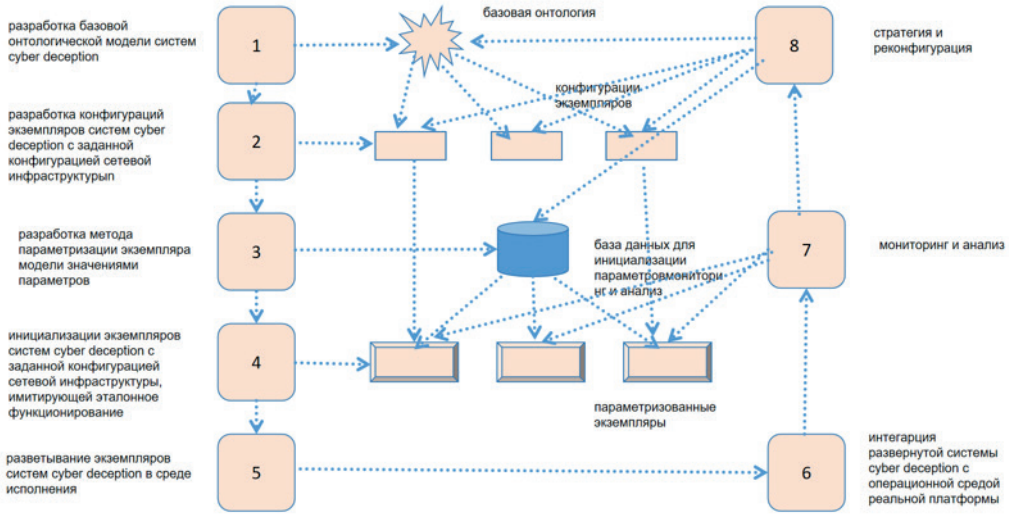
## Онтологический подход к представлению, построению и использованию систем cyber deception

В статье предложен новый подход построения и использования систем *cyber deception*, основанный на последовательном применении методов онтологического моделирования в среде промышленной онтологической платформы. Основными принципами этого подхода являются:

1. Использование в качестве базового формализма для систем *cyber deception* онтологического представления (модели) в среде промышленной онтологической платформы.
2. Построение реализаций систем *cyber deception* как экземпляров базового формализма для систем *cyber deception*.
3. Сопровождение процесса использования реализации системы *cyber deception* на основе ее онтологической модели.

На основе этих принципов разработана методология проектирования и использования обманных систем, включающая следующие виды деятельностей и иллюстрируемая на рис. 2:

- разработка базовой онтологической модели систем *cyber deception* в среде промышленной онтологической платформы, на основе которой в дальнейшем осуществляется построение экземпляров систем *cyber deception* с заданной конфигурацией сетевой инфраструктуры;
- определение на основе базовой онтологической модели конфигураций экземпляров систем *cyber deception* с заданной конфигурацией сетевой инфраструктуры;
- разработка метода параметризации экземпляра модели значениями параметров соответствующими эталонному поведению сетевой инфраструктуры и инициализации экземпляров систем *cyber deception* с заданной конфигурацией сетевой инфраструктуры, имитирующей эталонное функционирование;
- развертывание экземпляров систем *cyber deception* в среде исполнения;
- разработка модели жизненного цикла экземпляров систем *cyber deception* в среде онтологической платформы.




Р и с. 2. Методология проектирования и использования обманных систем на основе онтологического моделирования

F i g. 2. Methodology for Designing and Using Deception Systems Based on Ontological Modeling

## Построение онтологической модели: формирование классов, отношений и скалярных свойств эталонного поведения

Ранее было рассмотрено значение технологии *Cyber Deception* и показано, что её ключевая особенность заключается в построении синтетической модели эталонного поведения сетевой инфраструктуры. Для реализации этой задачи была разработана онтология, отражающая взаимосвязи между основными компонентами системы, их свойствами и динамическими характеристиками.

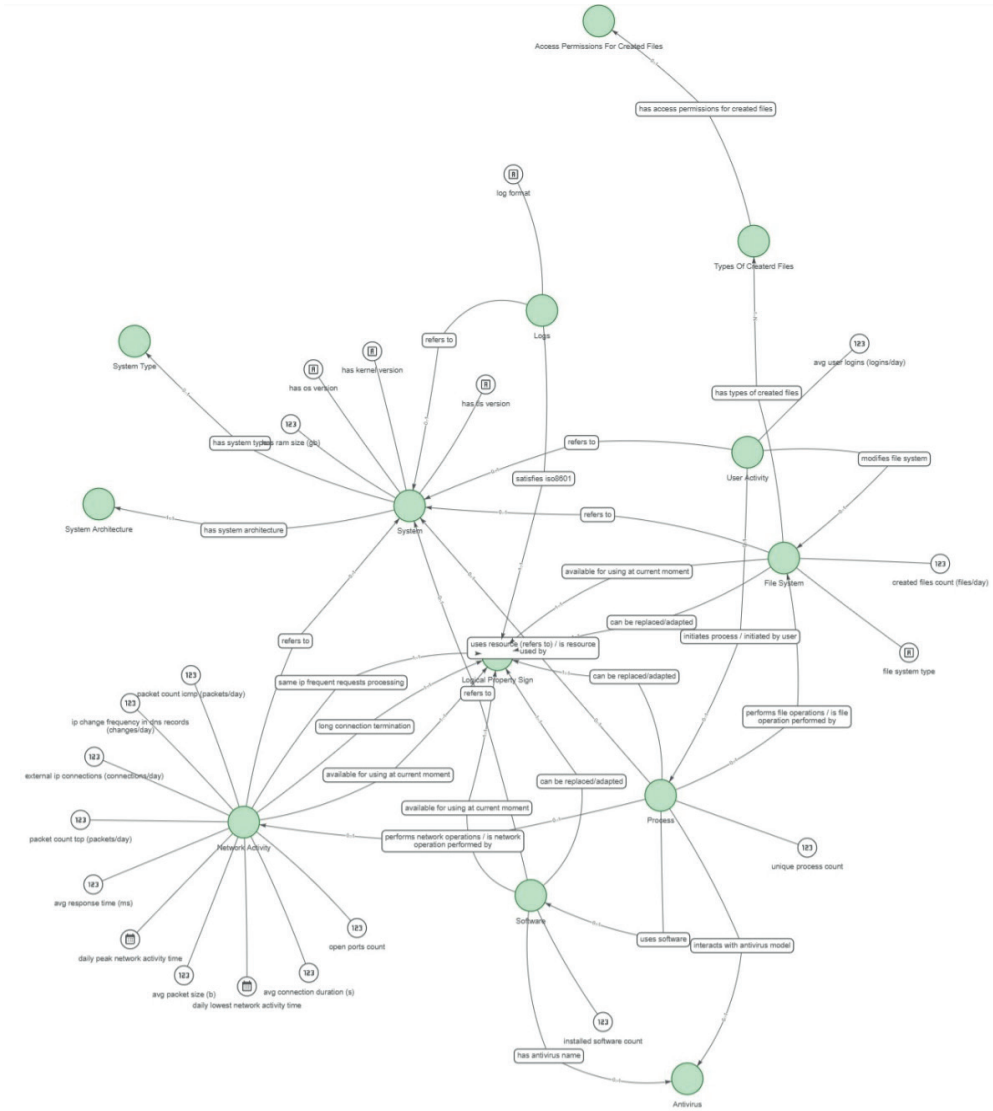
Онтологическая модель удобна тем, что позволяет зафиксировать все требования в явном виде (таб. 1).

В ней можно представить не только сам факт существования узла, но и его роль, связи с другими объектами, допустимые диапазоны параметров и ограничения на сочетание признаков. Благодаря этому ловушка проектируется как связанный объект, а не как набор случайно выбранных настроек. Если модель указывает, что объект относится к серверному классу, то для него ожидаются одни типы сетевой активности и журналов; если объект относится к пользовательскому классу, то набор признаков должен быть другим. Такая явная семантика снижает вероятность появления противоречивых профилей, которые легко распознаются опытным нарушителем.

Предлагаемая онтологическая модель строится вокруг центрального класса *System*, представляющего инфраструктурный узел. Узел характеризуется типом системы, архитектурой процессора, объемом оперативной памяти, версией опе-

Таблица 1. **Требования к правдоподобной cyber deception-среде**
Table 1. **Requirements for a Plausible Cyber Deception Environment**

| Группа требований | Содержание | Связь с онтологической моделью |
|---|---|---|
| Структурная согласованность | Тип узла, операционная система, файловая система, установленные сервисы и сетевые интерфейсы не должны противоречить друг другу | Классы System, File System, Software, Network Activity и отношения между ними |
| Поведенческая реалистичность | Активность процессов, пользователей, сетевых соединений и логов должна соответствовать выбранной роли узла | Скалярные свойства, профили эталонного поведения и сценарии имитации нагрузки |
| Автоматизируемость | Создание среды должно выполняться из единого машинно-читаемого описания без ручного переписывания множества файлов | Генератор инфраструктурных артефактов из онтологической модели |
| Поддерживаемость | Изменение инфраструктуры должно приводить к управляемой реконфигурации приманок | Механизм повторной генерации среды из обновленной модели |




Р и с. 3. Визуализация онтологии
F i g. 3. Ontology Visualization

рационной системы, версией ядра и рядом иных параметров, которые позволяют отличать сервер, рабочую станцию или встроенное устройство. С классом *System* связаны классы *Process, File System, Network Activity, User Activity, Software* и *Logs* (рис. 3).

Такое разбиение отражает наблюдаемую структуру реального узла и одновременно задает минимальный набор сущностей, необходимых для построения правдоподобной контейнерной имитации.

Класс *Process* описывает исполняемые процессы и их типы. Для *deception*-среды это важно потому, что список процессов и их активность часто используются атакующим как косвенный признак реальности системы. Класс *File System* задает тип файловой системы, количество создаваемых файлов, типы файлов и права доступа. Он нужен для формирования правдоподобных каталогов, служебных файлов, пользовательских следов и фиктивных документов. Класс *Network Activity* фиксирует параметры соединений, число пакетов разных типов, средний размер пакета, открытые порты, время отклика и частоту DNS-изменений. Эти параметры определяют сетевой профиль узла.

Класс *User Activity* описывает пользовательские логины, роли пользователей и типичные действия. Даже если *deception*-узел не предназначен для полноценной имитации рабочей станции, наличие разумных пользовательских следов повышает его правдоподобие. Класс *Software* отражает установленные программы, число приложений, наличие защитных средств и характеристики программной среды. Класс *Logs* задает формат, структуру и стандарт записи событий. Логи особенно важны для последующего анализа: они одновременно делают систему похожей на реальную и передают защитнику информацию о взаимодействии нарушителя с ловушкой.

Дополнительно в модель вводятся логические признаки, позволяющие задавать бинарные свойства типа «да/нет». Например, компонент может быть отмечен как заменяемый, сервис – как доступный для реконфигурации, а формат логов – как соответствующий заданному стандарту. Такие признаки удобны для автоматического выбора стратегии развертывания. Если определенный элемент допускает замену, система может подобрать другой вариант профиля при обновлении *deception*-среды. Если элемент не должен изменяться, он сохраняется как стабильная часть сценария.

Отношения модели разделяются на несколько групп. Структурные отношения фиксируют принадлежность и состав: узел имеет тип, программное обеспечение связано с процессами, файловая система принадлежит системе, журнал событий формируется определенными компонентами. Поведенческие отношения описывают динамику: процесс выполняет файловые операции, пользователь инициирует активность, программный компонент создает журнальную запись. Инфраструктурные отношения показывают влияние одной сущности на другую, например пользовательская активность модифицирует файловую систему, а сетевое взаимодействие порождает события мониторинга.

Важное отличие модели состоит в объединении семантических и количественных аспектов. Классы и отношения задают смысловую структуру, скалярные свойства задают измеримые параметры, а логические признаки позволяют выполнять проверки пригодности и согласованности. За счет этого онтология становится не только терминологическим описанием, но и источником данных для инструментальной системы. Она отвечает на три вопроса: какие объекты должны существовать, как они связаны и какие значения должны получить при создании конкретной обманной среды.

Такой подход особенно полезен для обеспечения интерпретируемости. Если в профиле узла появляется открытый порт, он связан не с произвольной строкой технической конфигурации, а с сущностью *Network Activity* и ролью узла. Если создается набор файлов, он связан с *File System* и пользовательскими сценариями. Если формируется журнал, он привязан к *Logs*, процессам и программным компонентам. Эксперт может проследить происхождение каждого параметра и понять, почему он присутствует в *deception*-среде.

## Сравнение онтологического подхода с существующими методами моделирования эталонного поведения

В практике информационной безопасности для задания эталона поведения используются две большие семьи методов.

Во-первых, статистические и эвристические подходы – пороговые правила, профили по распределениям, ядерная оценка плотности (KDE) и т. п. Их достоинства – простота внедрения и невысокие требования к данным. Ограничения – контекстная хрупкость порогов и ручная подстройка при изменениях инфраструктуры. Именно отсутствие богатого контекста (тип узла, роль процесса, политика логирования) часто приводит к ложным срабатываниям и трудностям интерпретации [20].

Во-вторых, модели машинного обучения и нейросети – *One-Class SVM, Isolation Forest*, автоэнкодеры и другие алгоритмы. Эти подходы усиливают чувствительность к сложным паттернам, в том числе нелинейным зависимостям и редким, тонким отклонениям. Однако цена за гибкость — повышенные требования к данным и MLOps-процедурам (сбор и разметка, контроль дрейфа, регулярное переобучение и валидация) [18, 19], [21-24].

В данной работе в качестве возможной альтернативы рассмотренным методам предложен онтологический подход (таб. 2). Ключевое отличие онтологии – интерпретируемость и управляемость. Отклонение объясняется как нарушение конкретного путевого ограничения или выход метрики за онтологически обусловленные диапазоны для данного узла [10-17]. У онтологии есть входной порог – требуется начальная формализация предметной области и поддержание актуальности модели по мере эволюции инфраструктуры[2] [11-12].

[2] Noy N. F., McGuinness D. L. Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory, Technical Report KSL-01-05, 2001. 25 p. [Электронный ресурс]. URL: https://protege.stanford.edu/publications/ontology_development/ontology101.pdf (дата обращения: 15.04.2026).

Таблица 2. **Сравнение онтологического подхода с существующими методами моделирования эталонного поведения**
Table 2. **Comparison of the Ontology-Based Approach with Existing Methods for Modeling Reference Behavior**

| Критерий | Онтологический подход | Статистика/эвристики | МО/нейросети |
|---|---|---|---|
| Представление эталона | Формальная карта: классы, отношения, скалярные свойства; проверяемые пути графа | Числовые профили и пороги на отдельных метриках; контекст задаётся вручную | Неявные представления в параметрах моделей; эталон – результат обучения |
| Интерпретируемость | Высокая: отклонение объясняется цепочкой «кто-что-где» и нарушением ограничений | Средняя: понятно «что вышло за порог», слабее – «почему» | Низкая-средняя: требуется *post-hoc* объяснение/*feature attribution* |
| Адаптация к изменениям | Естественная: меняются экземпляры/свойства – меняется эталон; возможность автоактуализации за счёт данных и IaC/контейнеров | Перенастройка порогов/правил; риск «ручной подгонки» | Переобучение/валидация, контроль дрейфа, поддержка датасетов |
| Требования к данным | Умеренные: структура + выборочные распределения | Низкие–умеренные: достаточно исторических рядов/экспертных правил | Высокие: нужны размеченные или прокси-датасеты, баланс классов |
| Реализм ловушек | Максимальный: профили берутся из эталона | Зависит от настройки параметров | Зависит от настройки параметров |
| Стоимость владения | Первичная формализация + поддержка онтологии; окупаемость за счёт снижения шума и ускорения расследований | Низкая на старте; растущие трудозатраты на ручную калибровку | Существенная: сбор/разметка данных, инфраструктура обучения, мониторинг дрейфа |

## Параметризация экземпляров эталонным поведением

После построения онтологической схемы необходимо перейти от абстрактных классов к экземплярам. Экземпляр *deception*-системы – это конкретизированный узел, обладающий набором свойств, связей и поведенческих характеристик. Экземпляры могут рассматриваться как базовые строительные блоки всей системы *cyber deception*. Формирование экземпляров начинается с выбора соответствующих индивидов онтологии, которые описывают типовые элементы инфраструктуры (например, *Server, Desktop, Embedded*). Каждый такой индивид интерпретируется как прототип будущего элемента обманной среды

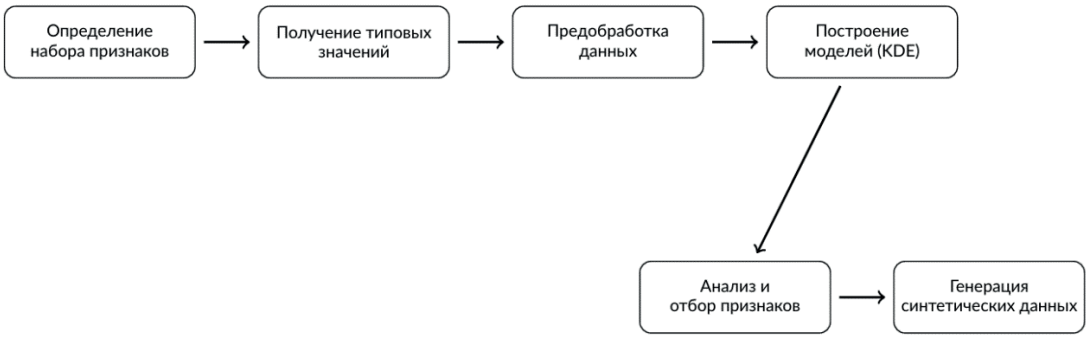
Предлагаемый метод параметризации включает несколько этапов. На первом этапе определяется набор признаков, существенных для выбранного типа узла. Для серверов это могут быть открытые сетевые порты, регулярные журнальные записи, фоновые процессы и устойчивые сетевые соединения. Для рабочих станций важнее пользовательские логины, прикладные процессы, файловые операции и обращения к внутренним ресурсам. Для встроенных устройств характерны ограниченные ресурсы, малое число сервисов, регулярная телеметрия и специфические протоколы взаимодействия.

На втором этапе формируется исходная совокупность данных. Она может включать открытые наборы сетевой телеметрии, результаты наблюдения тестовой инфраструктуры, экспертные ограничения и сведения о реальных конфигурациях организации.

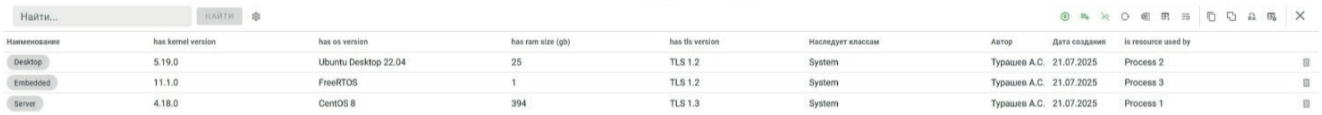
На третьем этапе выполняется генерация датасета синтетического эталонного поведения сетевой инфраструктуры. Для этого исходная совокупность данных, сформированная на предыдущем этапе, приводится к виду, пригодному для моделирования: устраняются выбросы, нормализуются числовые параметры, унифицируются форматы записей и проверяется согласованность данных с типом моделируемого узла. После предобработки для выбранных параметров строятся статистические модели, позволяющие сохранить характерные распределения нормальной активности: времени отклика, количества сетевых соединений, интенсивности пакетного обмена, частоты файловых операций, активности процессов и пользовательских действий. На основе этих моделей генерируются синтетические записи, которые образуют датасет эталонного поведения (рис. 4).

На последнем этапе значения сопоставляются со свойствами индивидов онтологии и формируют профиль узла. Важно, что параметризация выполняется не как заполнение таблицы, а как привязка значений к семантической структуре. Например, среднее время отклика относится к *Network Activity*, количество создаваемых файлов – к *File System*, число логинов – к *User Activity*, а формат записей – к *Logs*.

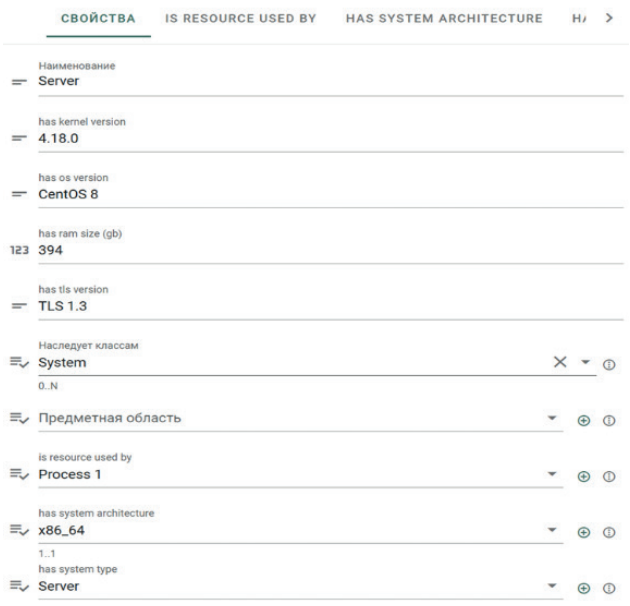
Практическая ценность параметризации состоит в том, что ловушка получает жизненные признаки. Пассивный узел с одним открытым портом и пустыми журналами быстро выглядит искусственным. Узел с согласованной фоновой активностью, реалистичными файлами, умеренными сетевыми задержками и логами, соответствующими выбранной роли, воспринимается более естественно (рис. 5, рис. 6).




Р и с. 4. Методика моделирования синтетического эталонного поведения сетевой инфраструктуры
F i g. 4. Methodology for Modeling Synthetic Reference Behavior of a Network Infrastructure




Р и с. 5. Объекты внутри класса System: Server, Desktop и Embedded
F i g. 5. Objects within the System Class: Server, Desktop, and Embedded




Р и с. 6. Пример экземпляра онтологии – Server
F i g. 6. Example of an Ontology Instance: Server

## Интеграция развернутой системы cyber deception с операционной средой реальной платформы. Этапы мониторинга, анализа и реконфигурации

После развёртывания экземпляров систем необходимо обеспечить их интеграцию с операционной средой реальной платформы. Данный этап направлен на включение обманной инфраструктуры в существующую информационную систему таким образом, чтобы она воспринималась как её естественная часть. Интеграция предполагает подключение развернутых элементов к сетевой инфраструктуре, настройку взаимодействия с существующими сервисами и обеспечение доступности обманных ресурсов для потенциальных атакующих. При этом важно сохранить баланс между доступностью и изоляцией: с одной стороны, обманные объекты должны быть видимыми и привлекательными для злоумышленника, с другой – их взаимодействие с реальной системой должно быть контролируемым и безопасным. Кроме того, интеграция включает взаимодействие с системами мониторинга и управления безопасностью. Это позволяет передавать информацию о событиях, происходящих в обманной среде, в централизованные системы анализа, обеспечивая единое представление о состоянии безопасности инфраструктуры.

После интеграции системы *cyber deception* начинается этап её эксплуатации, ключевым элементом которого является мониторинг и анализ происходящих событий. Основной целью данного этапа является выявление и интерпретация взаимодействий с обманными объектами, которые с высокой вероят-

ностью свидетельствуют о наличии вредоносной активности. Мониторинг включает сбор информации о всех действиях, направленных на обманную инфраструктуру: попытках подключения, обращениях к сервисам, выполнении операций и других видах активности. Поскольку легитимные пользователи, как правило, не взаимодействуют с обманными объектами, любые такие события рассматриваются как потенциально подозрительные и подлежат дальнейшему анализу. Анализ полученных данных позволяет выявить поведенческие паттерны злоумышленников, определить используемые ими методы и инструменты, а также установить последовательность действий в рамках атаки. Это даёт возможность не только обнаружить инцидент на ранней стадии, но и получить дополнительную информацию для совершенствования механизмов защиты.

Заключительным этапом жизненного цикла системы *cyber deception* является формирование стратегии её развития и последующая реконфигурация. Данный этап направлен на адаптацию обманной среды к изменяющимся условиям, включая эволюцию инфраструктуры и появление новых типов угроз. Стратегия развития основывается на результатах мониторинга и анализа, а также на изменениях в реальной инфраструктуре организации. На основе полученных данных принимаются решения о необходимости изменения структуры обманной среды, добавления новых экземпляров систем, модификации существующих параметров или удаления устаревших элементов. Реконфигурация системы осуществляется путём обновления онтологической модели и повторного выполнения этапов параметризации и развёртывания. Такой подход обеспечивает гибкость и управляемость системы, позволяя быстро адаптироваться к новым условиям.

## Прототип инструментальных средств и его экспериментальная оценка

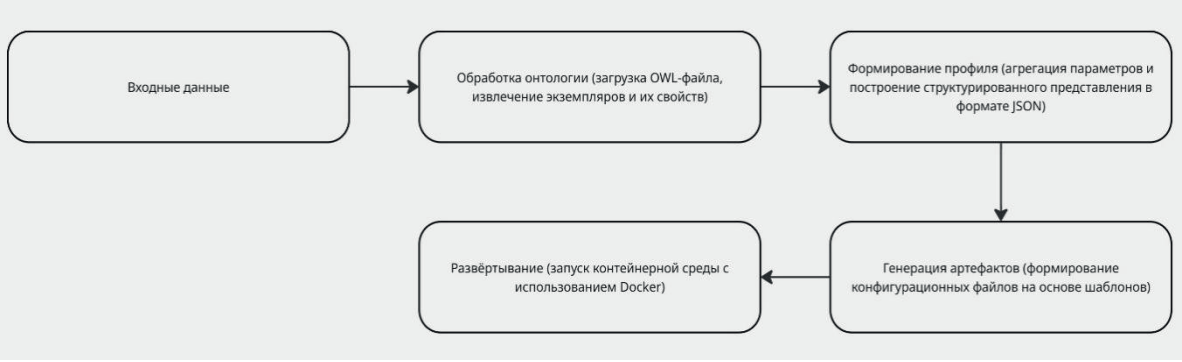
Прототип инструментальных средств предназначен для того, чтобы сделать процесс создания *deception*-среды управляемым и воспроизводимым. В традиционном подходе специалисту приходится вручную связывать между собой описание инфраструктуры, параметры ложных узлов, шаблоны конфигураций. При изменении инфраструктуры такой набор трудно обновлять. Предлагаемый прототип можно рассматривать как связующее звено между семантическим описанием и средой исполнения. На вход он получает онтологическое описание, в котором заданы классы инфраструктурных сущностей, их экземпляры, свойства и отношения. На выходе формируется набор артефактов, достаточных для создания контейнерной обманной среды.

Разработка прототипа инструментальных средств автоматизации создания обманных сред на основе онтологической модели требует формализации набора функциональных и архитектурных требований, обеспечивающих возможность преобразования семантического описания инфраструктуры в исполняемую контейнерную среду. К основным поддерживаемым операциям относятся: загрузка и парсинг онтологии, выбор целевого индивида, представляющего тип инфраструктурного узла (например, *Server, Desktop* или *Embedded*), извлечение значений его свойств и связей, формирование внутреннего профиля системы, генерация конфигурационных файлов, а также автоматический запуск контейнерной среды. Дополнительно система должна поддерживать повторную генерацию и развёртывание среды при изменении исходного онтологического описания.

Важным требованием является поддержка различных типов генерируемых артефактов, соответствующих современной практике развёртывания инфраструктуры. В частности, прототип должен автоматически формировать файлы описания контейнерной среды *(docker-compose)*, файлы сборки образов *(Dockerfile)*, сценарии инициализации контейнеров *(entrypoint)*, сценарии имитации поведения *(workload)*, а также профильные JSON-файлы, содержащие параметры системы, извлечённые из онтологии. Такой набор артефактов обеспечивает не только запуск инфраструктуры, но и моделирование реалистичного поведения её компонентов.

Неотъемлемой частью требований является интеграция с контейнерной инфраструктурой. Прототип должен быть совместим с технологией *Docker*, обеспечивать автоматическую сборку контейнерных образов, запуск контейнеров в изолированной среде и поддержку управления жизненным циклом контейнеров. Данный подход позволяет поддерживать воспроизводимость создаваемых обманных сред.

С учётом сформулированных требований была разработана архитектура прототипа (рис. 7), основанная на модульном принципе и обеспечивающая последовательное преобразование онтологического описания в исполняемую инфраструктуру.




Р и с. 7. Архитектура прототипа инструментальных средств

F i g. 7. Prototype Toolset Architecture

## Проверка корректности работы прототипа и критерии экспериментальной оценки

Экспериментальная оценка прототипа направлена прежде всего на проверку корректности преобразования онтологического описания в инфраструктурные объекты. В рамках такой оценки важно не просто показать, что контейнеры могут быть запущены, а подтвердить, что они создаются на основе значений параметров экземпляров заданной модели. Если значения параметров развернутой среды совпадают со значениями параметров экземпляра онтологического описания, можно говорить о правильности базового механизма генерации (таб. 3).

Таблица 3. **Степень соответствия между концептуальным описанием инфраструктуры, заданным в онтологии, и её фактической реализацией в контейнерной среде**
Table 3. **Degree of Correspondence Between the Conceptual Description of the Infrastructure Specified in the Ontology and Its Actual Implementation in a Containerized Environment**

| Название параметра инфраструктуры | Значения параметра в онтологии | Значение параметра в контейнерной среде Docker | Команды вывода значений для контейнерной среды Docker | Результаты проверки |
|---|---|---|---|---|
| os version | Server: CentOS 8<br>Desktop: Ubuntu Desktop 22.04<br>Embeded: FreeRTOS | Server: CentOS 8<br>Desktop: Ubuntu Desktop 22.04<br>Embeded: FreeRTOS | docker exec <container> sh -c «grep -o '\»has os version\»: *\»[^\»]*\»' /opt/ontology/profile.json \| sed 's/.*: *\»//; s/\»$//'» | Совпадение |
| kernel version | Server: 4.18.0<br>Desktop: 5.19.0<br>Embeded: 11.1.0 | Server: 4.18.0<br>Desktop: 5.19.0<br>Embeded: 11.1.0 | docker exec <container> sh -c «grep -o '\»has kernel version\»: *\»[^\»]*\»' /opt/ontology/profile.json \| sed 's/.*: *\»//; s/\»$//'» | Совпадение |
| tls version | Server: 1.3<br>Desktop: 1.2<br>Embeded: 1.2 | Server: 1.3<br>Desktop: 1.2<br>Embeded: 1.2 | docker exec <container> sh -c «grep -o '\»has tls version\»: *\»[^\»]*\»' /opt/ontology/profile.json \| sed 's/.*: *\»//; s/\»$//'» | Совпадение |
| ram size (gb) | Server: 394<br>Desktop: 25<br>Embeded: 1 | Server: 394<br>Desktop: 25<br>Embeded: 1 | docker exec <container> sh -c «grep -o '\»has ram size (gb)\»: *\»[^\»]*\»' /opt/ontology/profile.json \| sed 's/.*: *\»//; s/\»$//'» | Совпадение |
| system architecture | Server: x86_64<br>Desktop: x86_64<br>Embeded: ARM64 | Server: x86_64<br>Desktop: x86_64<br>Embeded: ARM64 | docker exec <container> sh -c «sed -n '/has system architecture/,+4p' /opt/ontology/profile.json \| grep '\»label\»' \| head -n 1 \| sed 's/.*: *\»//; s/\»,*$//'» | Совпадение |

Проведённый анализ показывает, что разработанный прототип корректно преобразует элементы онтологической модели в параметры контейнерной инфраструктуры. Обеспечивается прямая связь между концептуальным описанием инфраструктуры и её фактической реализацией в контейнерной среде. Это подтверждает возможность использования онтологической модели в качестве источника конфигурационных данных и позволяет наглядно проследить соответствие между онтологическими конструкциями и объектами контейнерной инфраструктуры.

Оценка степени автоматизации разработанного прототипа может быть выполнена по нескольким измеряемым критериям, отражающим удобство проектирования и развёртывания контейнерной среды.

Первым критерием является количество ручных действий, необходимых для получения рабочей инфраструктуры. При использовании традиционного подхода разработчику необходимо вручную создавать и настраивать несколько типов конфигурационных файлов: писать *Dockerfile* для каждого контейнера, формировать файлы *docker-compose*, настраивать переменные окружения, сетевые параметры, сервисы внутри контейнеров, а также выполнять их отладку и проверку работоспособности. Такой процесс требует значительного количества ручных операций и повышает вероятность ошибок конфигурации. В разработанном прототипе данный процесс существенно упрощается: достаточно запустить один скрипт, который автоматически извлекает параметры из онтологической модели, генерирует необходимые конфигурационные файлы и подготавливает инфраструктуру к запуску.

Вторым критерием является время до готовности среды *(time-to-deploy)*. Этот показатель определяется как промежуток времени от запуска скрипта до момента, когда все контейнеры находятся в поднятом состоянии. В традиционном подходе время развёртывания может существенно увеличиваться из-за необходимости ручного редактирования конфигураций, проверки корректности *Dockerfile* и устранения ошибок сборки. В разработанном прототипе большая часть этих действий автоматизирована, поскольку конфигурационные файлы фор-

мируются программно на основе онтологической модели, что сокращает общее время подготовки среды.

Третьим критерием является автоматическая генерация поведения инфраструктурных узлов. В прототипе предусмотрены специальные сценарии *workload*, которые запускаются автоматически внутри контейнеров после их инициализации. Эти сценарии имитируют типичную активность узлов инфраструктуры, например сетевые обращения, файловые операции или обмен сообщениями. Благодаря этому контейнеры не только запускаются, но и начинают воспроизводить реалистичное поведение без дополнительной ручной настройки. Таким образом, разработанный прототип обеспечивает не только автоматическое развёртывание инфраструктуры, но и автоматическое моделирование её активности.

Далее представлено сравнительное сопоставление предложенного метода с существующими подходами к развёртыванию и моделированию инфраструктурных сред (таб. 4).

Таблица 4. **Сравнение традиционных подходов и предложенного метода**
Table 4. **Comparison of Traditional Approaches and the Proposed Method**

| **Критерий** | **Ручное конфигурирование инфраструктуры** | **Статические Docker-шаблоны** | **IaC без онтологии (Terraform / Ansible)** | **Honeypot-фреймворки** | **Эмуляция инфраструктуры (VM, GNS3, EVE-NG)** | **Предложенный метод** |
|---|---|---|---|---|---|---|
| Источник конфигурации | Разрозненные конфигурационные файлы | Готовые шаблоны docker-compose | Конфигурация как код | Настройки фреймворка | Ручная настройка виртуальных машин | Онтологическая модель |
| Формальная модель инфраструктуры | Отсутствует | Отсутствует | Отсутствует | Отсутствует | Отсутствует | Присутствует (онтология) |
| Автоматизация развертывания | Низкая | Частичная | Высокая | Высокая | Низкая | Высокая |
| Автогенерация конфигураций | Нет | Нет | Частично | Нет | Нет | Да |
| Связь между концепцией и реализацией | Отсутствует | Косвенная | Косвенная | Отсутствует | Отсутствует | Прямая (OWL → конфигурация среды) |
| Изменение требований | Ручная правка всех файлов | Ручная корректировка параметров | Изменение IaC-скриптов | Ограниченная настройка | Ручная реконфигурация | Изменение онтологии |
| Масштабируемость | Низкая | Средняя | Высокая | Средняя | Ограниченная | Высокая |
| Использование контейнеризации | Да | Да | Да | Частично | Нет | Да |
| Ресурсоемкость | Средняя | Средняя | Средняя | Низкая | Высокая | Низкая |
| Скорость развертывания среды | Низкая | Средняя | Высокая | Высокая | Низкая | Высокая |
| Поддержка имитации поведения | Нет | Нет | Нет | Частично | Частично | Да (workload-сценарии) |

## Заключение

В статье рассмотрен подход к созданию систем киберобмана на основе онтологического моделирования. Показано, что онтологическая модель может выступать единым источником знаний для описания инфраструктурных узлов, их свойств, связей и поведенческих характеристик. В рамках такого подхода элементы обманной среды представляются не как разрозненные конфигурационные параметры, а как взаимосвязанные сущности: системы, процессы, файловые структуры, сетевые активности, пользовательские действия, программные компоненты и журналы событий. Это повышает интерпретируемость проектирования обманной системы.

Разработанный на основе онтологического подхода прототип инструментальных средств для создания систем обмана продемонстрировал высокую степень автоматизации развертывания и возможность генерации реалистичного поведения инфраструктурных узлов. Дальнейшее развитие работы может быть связано с расширением онтологической модели, добавлением новых типов инфраструктурных узлов, более глубокой интеграцией с SIEM/SOAR-системами, расширением модуля анализа событий при взаимодействии с ловушками и разработкой механизмов адаптивного изменения *deception*-среды на основе результатов мониторинга.

## References

[1] Chaudhry A., Andersen C., Choudhary G., Dragoni N. A Review of Honeypots: Fingerprinting Techniques, Detection, and Evasion Mechanisms. *Future Internet.* 2026;18(4):190. https://doi.org/10.3390/fi18040190
[2] Spitzner L. Honeypots: Tracking Hackers. Boston: Addison-Wesley Professional; 2003. 480 p.
[3] Provos N. A Virtual Honeypot Framework. In: Proceedings of the 13th USENIX Security Symposium. San Diego, CA: USENIX Association; 2004. p. 1-14. Available at: https://www.usenix.org/conference/13th-usenix-security-symposium/virtual-honeypot-framework (accessed 15.04.2026).
[4] Javadpour A., Ja'fari F., Taleb T., Shojafar M., Benzaid C. A Comprehensive Survey on Cyber Deception Techniques to Improve Honeypot Performance. *Computers & Security*. 2024;140:103792. https://doi.org/10.1016/j.cose.2024.103792
[5] Almeshekah M.H., Spafford E.H. Planning and Integrating Deception into Computer Security Defenses. In: Proceedings of the 2014 New Security Paradigms Workshop. New York: ACM; 2014. p. 127-138. https://doi.org/10.1145/2683467.2683482
[6] Jajodia S., Subrahmanian V.S., Swarup V., Wang C. Cyber Deception: Building the Scientific Foundation. Cham: Springer; 2016. 312 p. https://doi.org/10.1007/978-3-319-32699-3
[7] Rowe N.C., Rrushi J. Introduction to Cyberdeception. Cham: Springer, 2016. 334 p. https://doi.org/10.1007/978-3-319-41187-3
[8] Pawlick J., Colbert E., Zhu Q. A Game-Theoretic Taxonomy and Survey of Defensive Deception for Cybersecurity and Privacy. *ACM Computing Surveys*. 2019;52(4):82. https://doi.org/10.1145/3337772
[9] Gruber T.R. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition.* 1993;5(2):199-220. https://doi.org/10.1006/knac.1993.1008
[10] Guarino N. Formal Ontology and Information Systems. In: Proceedings of the First International Conference on Formal Ontology in Information Systems (FOIS 1998). Amsterdam: IOS Press; 1998. p. 3-15. Available at: https://www.loa.istc.cnr.it/old/Papers/FOIS98.pdf (accessed 15.04.2026).
[11] Guarino N., Welty C. Evaluating Ontological Decisions with OntoClean. *Communications of the ACM*. 2002;45(2):61-65. https://doi.org/10.1145/503124.503150
[12] Musen M.A. The Protégé Project: A Look Back and a Look Forward. *AI Matters*. 2015;1(4):4-12. https://doi.org/10.1145/2757001.2757003
[13] Obrst L., Chase P., Markeloff R. Developing an Ontology of the Cyber Security Domain. *CEUR Workshop Proceedings.* 2013;966:49-56. Available at: https://ceur-ws.org/Vol-966/STIDS2012_T06_ObrstEtAl_CyberOntology.pdf (accessed 15.04.2026).
[14] Iannacone M., Bohn S., Nakamura G., Gerth J., Huffer K., Bridges R., Ferragut E., Goodall J. Developing an Ontology for Cyber Security Knowledge Graphs. In: Proceedings of the 10th Annual Cyber and Information Security Research Conference (CISR 2015). New York: ACM; 2015. Article number: 12. https://doi.org/10.1145/2746266.2746278
[15] Ben-Asher N., Oltramari A., Erbacher R.F., Gonzalez C. Ontology-Based Adaptive Systems of Cyber Defense. *CEUR Workshop Proceedings*. 2015;1523:34-41. Available at: https://ceur-ws.org/Vol-1523/STIDS_2015_T05_BenAsher_etal.pdf (accessed 15.04.2026).
[16] Preuveneers D., Joosen W. An Ontology-Based Cybersecurity Framework for AI-Enabled Systems and Applications. *Future Internet*. 2024;16(3):69. https://doi.org/10.3390/fi16030069
[17] Hoseini S., Theissen-Lipp J., Quix C. A survey on semantic data management as intersection of ontology-based data access, semantic modeling and data lakes. *Journal of Web Semantics*. 2024;81:100819. https://doi.org/10.1016/j.websem.2024.100819
[18] Patki N., Wedge R., Veeramachaneni K. The Synthetic Data Vault. In: 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA). Montreal, QC, Canada: IEEE Press; 2016. p. 399-410. https://doi.org/10.1109/DSAA.2016.49
[19] Xu L., Skoularidou M., Cuesta-Infante A., Veeramachaneni K. Modeling Tabular Data Using Conditional GAN. In: Advances in Neural Information Processing Systems 32 (NeurIPS 2019). Curran Associates, Inc.; 2019. p. 7333-7343.
[20] Terrell G.R., Scott D.W. Variable Kernel Density Estimation. *The Annals of Statistics*. 1992;20(3):1236-1265. https://doi.org/10.1214/aos/1176348768
[21] Zoghi Z., Serpen G. UNSW-NB15 computer security dataset: Analysis through visualization. *Security and Privacy.* 2024;7(1):e331. https://doi.org/10.1002/spy2.33
[22] Putra M.A.R., Hostiadi D.P., Ahmad T. Botnet Dataset with Simultaneous Attack Activity. *Data in Brief*. 2022;45:108628. https://doi.org/10.1016/j.dib.2022.108628
[23] Koroniotis N., Moustafa N., Sitnikova E., Turnbull B. Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Datase. *Future Generation Computer Systems*. 2019;100:779-796. https://doi.org/10.1016/j.future.2019.05.041
[24] Moustafa N., Slay J. UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems. In: Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS). Piscataway: IEEE Press; 2015. p. 1-6. https://doi.org/10.1109/MilCIS.2015.7348942
[25] Morris K. Infrastructure as Code: Dynamic Systems for the Cloud Age. 2nd ed. Sebastopol: O'Reilly Media; 2021. 430 p.
[26] Felter W., Ferreira A., Rajamony R., Rubio J. An Updated Performance Comparison of Virtual Machines and Linux Containers. In: Proceedings of the 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). Philadelphia, PA, USA: IEEE Press; 2015. p. 171-172. https://doi.org/10.1109/ISPASS.2015.7095802

[27] Bernstein D. Containers and Cloud: From LXC to Docker to Kubernetes. *IEEE Cloud Computing*. 2014;1(3):81-84. https://doi.org/10.1109/MCC.2014.51
[28] Pahl C. Containerization and the PaaS Cloud. *IEEE Cloud Computing*. 2015;2(3):24-31. https://doi.org/10.1109/MCC.2015.51
[29] Merkel D. Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux Journal.* 2014;2014(239):2.

*Поступила 15.04.2026; одобрена после рецензирования 14.05.2026; принята к публикации 24.06.2026.*
*Submitted 15.04.2026; approved after reviewing 14.05.2026; accepted for publication 24.06.2026.*

Об авторах:

**Турашев Артём Сергеевич,** магистрант кафедры информационной безопасности факультета вычислительной математики и кибернетики, ФГБОУ ВО «Московский государственный университет имени М. В. Ломоносова» (119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1), **ORCID: https://orcid.org/0000-0001-8391-4948,** turashev.artem@mail.ru

**Сухомлин Владимир Александрович**, заведующий лабораторией открытых информационных технологий кафедры информационной безопасности факультета вычислительной математики и кибернетики, ФГБОУ ВО «Московский государственный университет имени М. В. Ломоносова» (119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1), доктор технических наук, профессор, **ORCID: https://orcid.org/0000-0001-9468-7138**, sukhomlin@mail.ru

*Все авторы прочитали и одобрили окончательный вариант рукописи.*

About the authors:

**Artem S. Turashev**, Master Degree Student of the Department of Information Security, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1 Leninskie gory, Moscow 119991, GSP-1, Russian Federation), **ORCID: https://orcid.org/0000-0001-8391-4948,** turashev.artem@mail.ru

**Vladimir A. Sukhomlin,** Head of the Open Information Technologies Lab, Department of Information Security, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1 Leninskie gory, Moscow 119991, GSP-1, Russian Federation), Dr. Sci. (Tech.), Professor, **ORCID: https://orcid.org/0000-0001-9468-7138**, sukhomlin@mail.ru

*All authors have read and approved the final manuscript.*