ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

УДК 004.934.2

СЕМАНТИЧЕСКИЙ АНАЛИЗ ИНФОРМАЦИОННЫХ РИСКОВ И УГРОЗ НА ОСНОВЕ ОНТОЛОГИИ СТАНДАРТА ISO/IEC 27001

В. В. Гаршина, В. А. Степанцов, А. Ю. Данковцева

Воронежский государственный университет

Поступила в редакцию 30.08.2018 г.

Аннотация. В статье представлена реализация онтологического подхода к моделированию смысловых закономерностей для управления информационной безопасностью компании на основе стандарта ISO/IEC 27001. Предложена технологическая платформа разработки, базирующаяся на универсальных стандартах и использующая свободно-распространяемое ПО, на базе которой реализован прототип системы семантического анализа информационных рисков и угроз.
Ключевые слова: онтология, анализ информационных рисков и угроз, ISO/IEC 27001, SWRL-правила, SPARQL-запросы.
Annotation. The article presents the implementation of the ontological approach to the modeling of the semantic regularities for the management of the company's information security based on the ISO / IEC 27001 standard. A technology development platform based on universal standards and using free software. A prototype of an information semantic analysis system of risks threats has been made.
Keywords: ontology, information risk and threat analysis, ISO / IEC 27001, SWRL rules, SPARQL queries.

ВВЕДЕНИЕ

Разработка семантических систем аналитики в области информационной безопасности, несмотря на большой имеющийся теоретический задел, пока слабо представлена в реализованных информационных системах. В связи с этим, актуальным является формирование новых подходов к моделированию смысловых закономерностей для управления информационной безопасностью, выбор интеграционных технологических решений в качестве платформы разработки системы поддержки принятия решений для анализа информационных рисков и угроз.

Построение программных систем анализа информационной защищенности связано с проблемой представления, обработки и комплексирования разнородной информации (данных), которой присущи следующие особенности:

– поступает из разных источников (технические средства, экспертные мнения, результаты методик расчета, хранилища данных, результаты измерений и т. д.);

– обладает неполнотой, нечеткостью и неточностью, имеет разный уровень надежности;

– различные фрагменты вступают в противоречие друг с другом;

– изменяется во времени (может быть статической и динамической);

– носит как объективный (поступает из надежных источников, являются результатами измерений и оценок, обладающих свойствами точности, полноты, четкости (однозначной интерпретации)) так и субъективный (результаты экспертного оценивания, оценки квалифицированных специалистов, требуют

дополнительной обработки, согласований мнений и т. д.) характер.

Решение проблемы интеграции такой разнородной информации и построение системы логических выводов, обеспечивающих смысловой (семантический) анализ фиксируемых данных, процессов, событий, происходящих в информационной системе и влияющих на ее безопасность, возможно на основе онтологического подхода.

МАТЕРИАЛЫ И МЕТОДЫ ИССЛЕДОВАНИЯ

Для разработки системы, поддерживающей семантический анализ информационных рисков и угроз, была выбрана онтология в формате owl созданная в Линчёпингском университете Швеции [1, 2] и реализующая стандарт информационной безопасности ISO/IEC 27001. Данный стандарт используется в качестве модели для разработки, внедрения, функционирования, мониторинга, анализа, поддержки и улучшения системы менеджмента в области информационной безопасности и предоставляет возможность анализа и оценки рисков и уязвимостей для решения следующих задач:

1. Оценки ущерба для системы, который может быть нанесен в результате сбоя обеспечения безопасности, с учетом возможных последствий нарушения конфиденциальности, целостности или доступности.

2. Оценки реальной вероятности сбоя обеспечения безопасности с учетом угроз, уязвимостей и их последствий.

3. Оценки уровня рисков.

Основные классы онтологии на основе стандарта ISO/IEC 27001: Asset – класс включает действия, необходимые для обеспечения

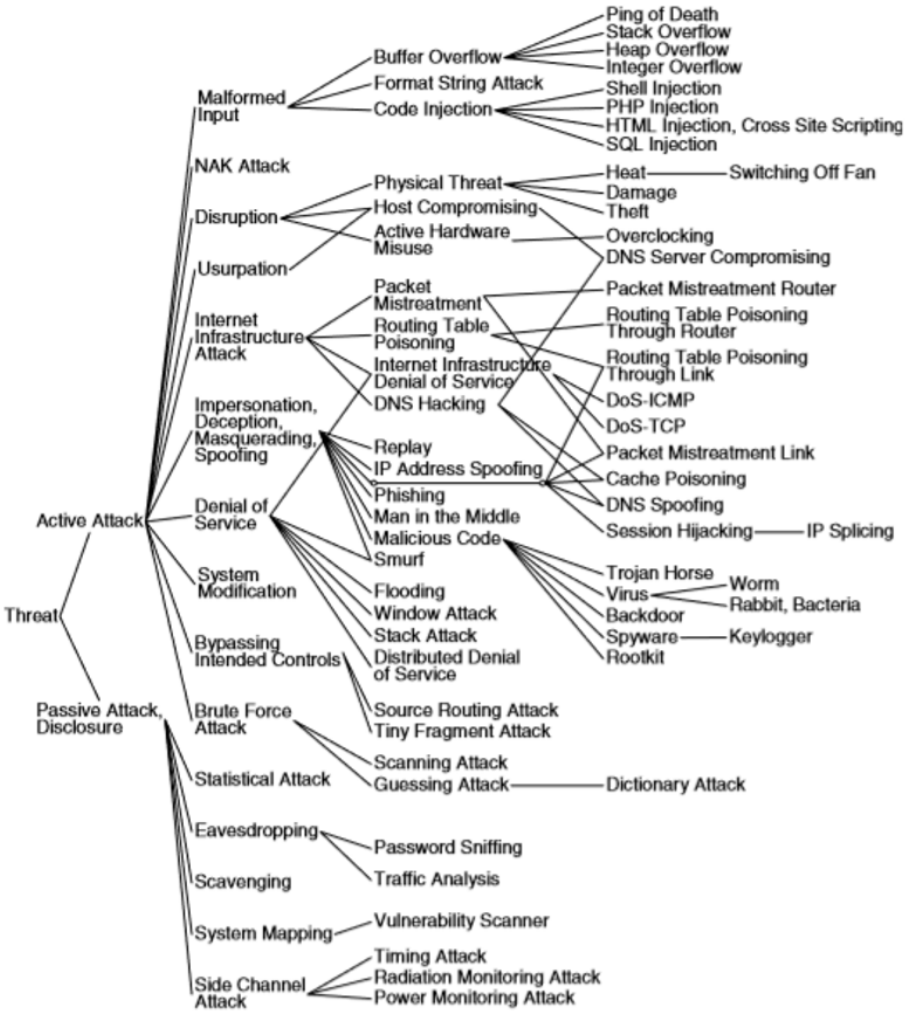

Рис 1. Используемая ветвь (Threat-угрозы) онтологии стандарта ISO/IEC 27001

безопасности; AssetLifeCyclePhase – класс, описывающий жизненный цикл; Countermeasure – класс включает в себя контрмеры, такие как антивирусные программы, пакетный фильтр проверки состояния, контрольная сумма, криптография и т. д.; DefenseStrategy – описывает стратегии защиты; Goal – цели анализа; Model – модели предпринимаемых действий для обеспечения безопасности; NaryRelation – отношения между классами; Product – анализируемые системы (базы данных, MacOS, Unix, Windows и другие); Threat – угрозы; Vulnerability – уязвимости.

В разрабатываемой системе была использована ветвь онтологии (рис. 1), предоставляющая возможность оценки 88 угроз, самыми распространёнными среди них можно выделить следующие: System Modification (изменения в системе, которые не были проконтролированы и могли привести к угрозам); DNS Spoofing (является формой взлома сети, в результате чего данные кэша доменных имен изменяются злоумышленником, с целью возврата ложного IP-адреса); IP Address Spoofing (вид атаки, в процессе которой используется чужой IP-адрес); Cache Poisoning (в результате данной атаки происходит повреждение целостности информации); Trojan Horse (вредоносные программы, которые выполняют несанкционированные пользователем действия, такие как удаление, блокировка, изменение данных); SQL Injection (является одним из самых доступных способов взлома сайтов путем внедрения SQL-кода злоумышленником); Stack Overflow (переполнение стека); DoS-TCP-атаки проводятся с целью довести вычислительную систему до отказа, позволяют вывести из строя практически любую систему, не оставляя при этом юридических улик; Spoofing – перенаправление трафика в компьютерных сетях под контролем злоумышленника, а также возможен запуск DoS атак ничего не подозревающими клиентами.

В данной онтологии понятия «угроза» и «атака» взаимозаменяемы, она ориентирована больше на конкретные атаки, чем на уровень угрозы. Верхний уровень представлен классификацией на пассивные и активные атаки. Пассивная атака – это атака, которая не изменяет систему, но нарушает её конфиденциальность. Распространенные примеры пассивной атаки – подслушивание, копирование данных. Типичные примеры активной угрозы – это несанкционированное изменение системы, атаки на отказ в обслуживании и многое другое. Концепция угрозы моделируется аксиомами, которые указывают на то, чему она может принести ущерб. В онтологии указано взаимодействие между различными угрозами и степень их влияния друг на друга.

РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЙ И ИХ ОБСУЖДЕНИЕ

Для разработки системы семантического анализа информационных рисков и угроз предложена технологическая платформа разработки, базирующаяся на универсальных стандартах и использующая свободно-распространяемое ПО: стандарты описания онтологий QWL, RDF; стандарты формирования запросов к онтологии SPARQL и формализации аксиом вывода SWRL; редактор и движок тестирования выводов на онтологии Protégé; язык Java в среде разработки IntelliJ IDEA; база данных PostgreSQL, взаимодействующая с основным модулем посредством стандарта jdbc.

Архитектура приложения представляет собой интеграцию нескольких компонент (рис. 2). Основной программный модуль, выполненный на языке Java, использующий программный интерфейс Apache Jena [4], для работы с семантическими онтологиями OWL API [5], позволяющем работать с хранилищами триплетов и машинами логического вывода. На вход поступают экспертные оценки и объективные результаты измерений, для работы с которыми используется база данных PostgreSQL, взаимодействующая с основным модулем посредством стандарта jdbc.

Заключение о безопасности анализируемой системы строится на основе SWRL-правил онтологии через фреймворк OWL API. Для получения необходимой информации о классах используются SPARQL-запросы, выполняемые через Apache Jena.

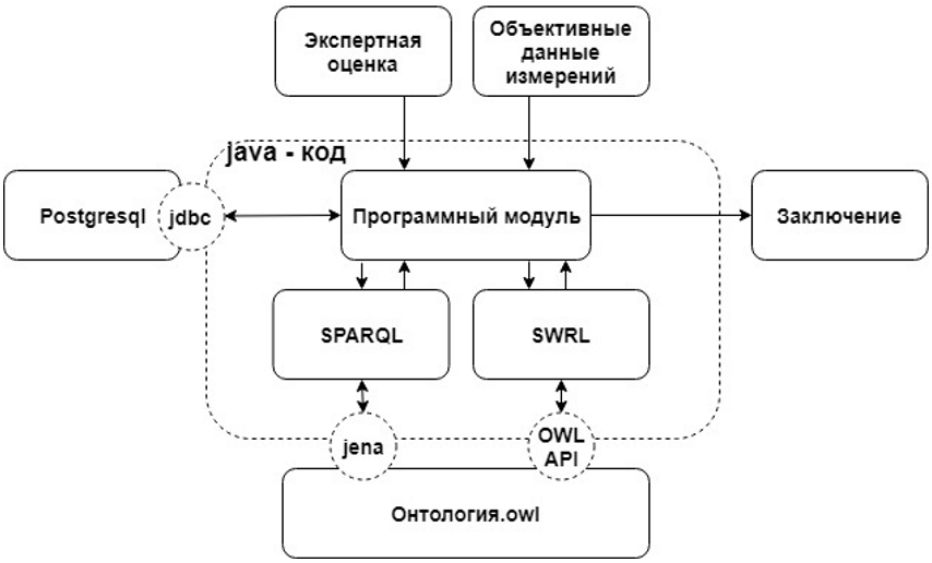

Рис 2. Архитектура программной системы комплексирования и анализа разнородной информации на основе онтологии для проведения семантического анализа информационной защищенности компании

```
String queryString =
                "prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>\n" +
                "select ?subclass where {\n" +
                "  ?subclass rdfs:subClassOf" +
                "<http://www.ida.liu.se/~iislab/projects/secont/Security.owl#Threat>" + "\n" +
                "}";
```

Рис. 3 Пример SPARQL-запроса

Механизмы логического вывода позволяют вычислять значения логических выражений, проверять правильность модели, автоматически помещать в онтологию новую информацию в соответствии с правилами, позволяют оперировать именами классов, свойств и сущностей, и «задавать модели вопросы», абстрагируя пользователя от подробностей внутреннего строения модели.

Стандарт SPARQL описывает синтаксис запросов к онтологическим моделям (является аналогом языка SQL). Применяется преимущественно для выборки необходимых данных с помощью SELECT-запросов с возможной их фильтрацией и сортировкой. SPARQL поддерживает вопросы, требующие однозначных ответов да или нет, сортировку, фильтрацию, сопоставление строк.

На рис. 3 представлен запрос, используемый в программном коде разработанного приложения, который позволяет получить подклассы, в данном случае, для класса Threat. Он необходим для заполнения базы данных угрозами безопасности информационной системы, получаемыми из онтологии, которые разработаны экспертами в данной области, с целью дальнейшей работы с выбранными критериями.

Для задания собственных правил логических выводов используется стандарт SWRL. Каждое правило состоит из двух частей – условия и вывода, который формируется, если условие выполнено. И условие, и вывод могут состоять из нескольких атомов – элементарных логических выражений. Каждый атом представляет собой предикат – утверждение о каких-либо объектах онтологии. Правила SWRL позволяют создавать гибкие условия для получения новых знаний. На основе таких правил строится вывод о защищенности системы.

Изначально выбранная онтология не содержала правил, поэтому они были разработаны дополнительно. Для того чтобы применять правила, необходимо было выполнить заполнение онтологии данными, на основе которых будут приниматься решения. Для каждого анализируемого класса было созда-

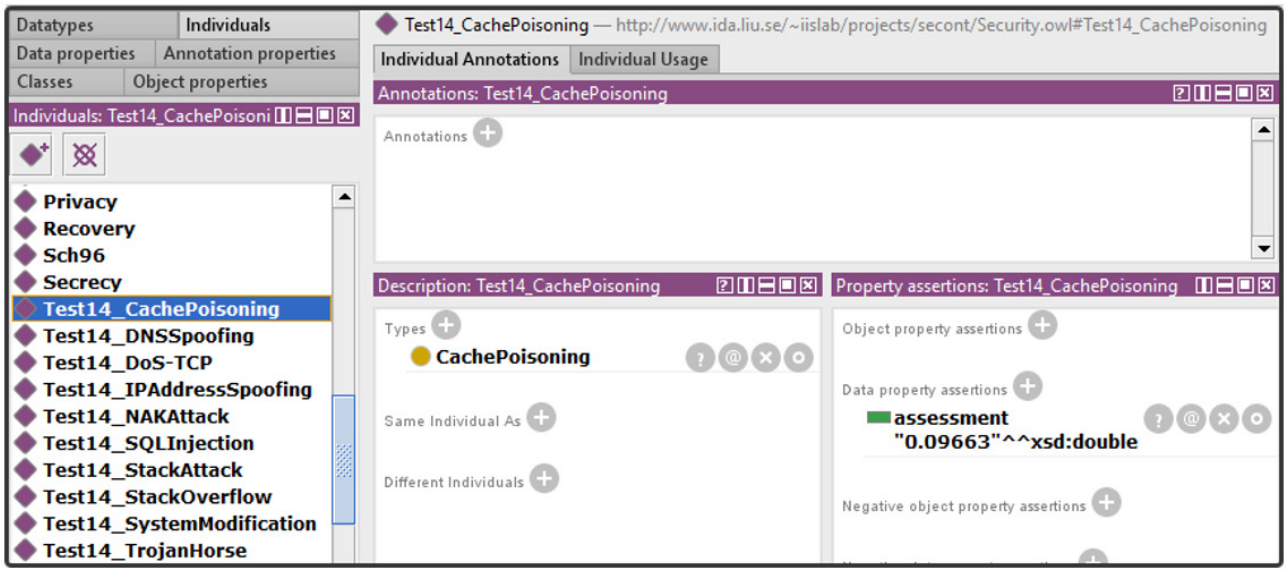

Рис. 4. Данные онтологии

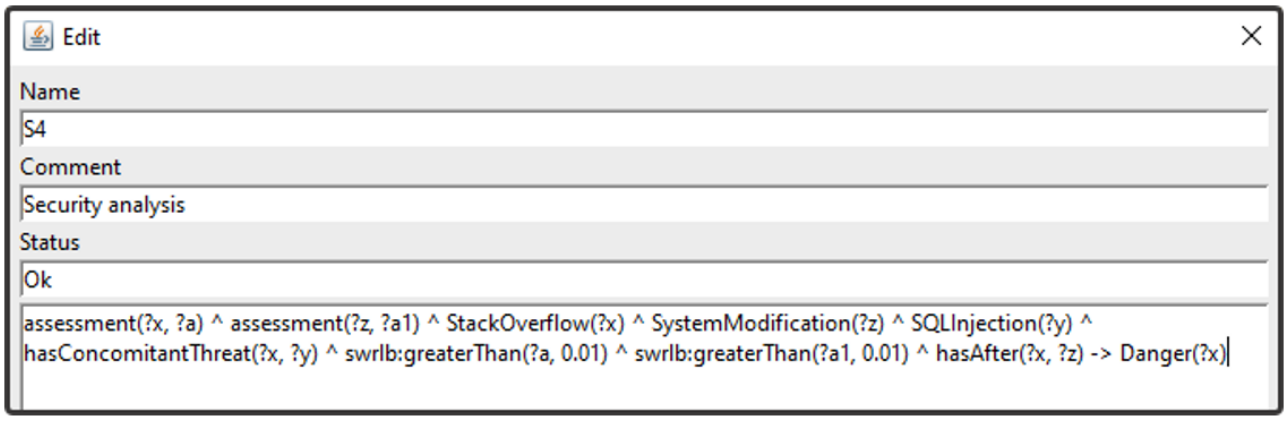

Рис 5. Пример SWRL-правила

но собственное индивидуальное значение, соответствующее названию класса, и присвоено значение свойства assessment (оценка), которые получено по результатам опроса и согласования мнений экспертов в ходе тестирования.

Например, индивидуальному значению класса CachePoisoning (отправление кэша), присвоено значение оценки равное 0.09663. Данные онтологии представлены на рис. 4.

В ходе создания программного приложения разработан ряд SWRL-правил, которые прошли тестирование на полученной онтологии. Пример на рис. 5.

Правило можно интерпретировать как: *влияние_фактора(?фактор_x,?порог_влияния_a)∧влияние_фактора(?фактор_z,?порог_влияния_a1) ∧ переполнение_стека(?фактор_x) ∧ произошедшие_изменения_в_системе (?фактор_z) ∧SQLиньекция(?y) ∧сопутствующие_факторы(?x,?y) ∧ функция_проверки_превышения_фактором_порога(?порог_влияния_a,0.01) ∧ функция_проверки_превышения_фактором_порога(?порог_влияния_a1,0.01) ∧ факторы_последовательны(?фактор_x,?фактор_z) →угроза (?фактор_x)*

В соответствие с ним проверяется оценка assessment подаваемого на вход правила фактора x, где a – значение оценки, определенное в ходе работы программы по результатам опроса экспертов. swrlb:greaterThan(?a, 0.01) – встроенная функция проверки того, превышает ли значение a определенный порог 0.01. hasConcomitantThreat(?x, ?y) – проверка являются ли факторы x и y сопутствующими, что несет более высокую опасность. hasAfter(?x, ?z) – проверка на последовательность появления факторов. В случае удовлетворения всем этим условиям, делается вывод Danger(?x) является ли введенный фактор опасным для

```
Danger(Test14_StackOverflow)
StackOverflow(Test14_StackOverflow)
hasAfter(Test14_StackOverflow, Test14_SystemModification)
hasConcomitantThreat(Test14_StackOverflow, Test14_SQLInjection)
assessment(Test14_StackOverflow, 0.09459)
SystemModification(Test14_SystemModification)
assessment(Test14_SystemModification, 0.12101)
Danger(?x) ← assessment(?x, ?a) ∧ assessment(?z, ?a1) ∧ StackOverflow(?x)
∧ SystemModification(?z) ∧ SQLInjection(?y) ∧ hasConcomitantThreat(?x, ?y) ∧
http://www.w3.org/2003/11/swrlb#greaterThan((?a) , (0.01)) ∧ http://www.w3.
org/2003/11/swrlb#greaterThan((?a1) , (0.01)) ∧ hasAfter(?x, ?z)
SQLInjection(Test14_SQLInjection)
```

Рис. 6. Пример вывода на основе SWRL-правила

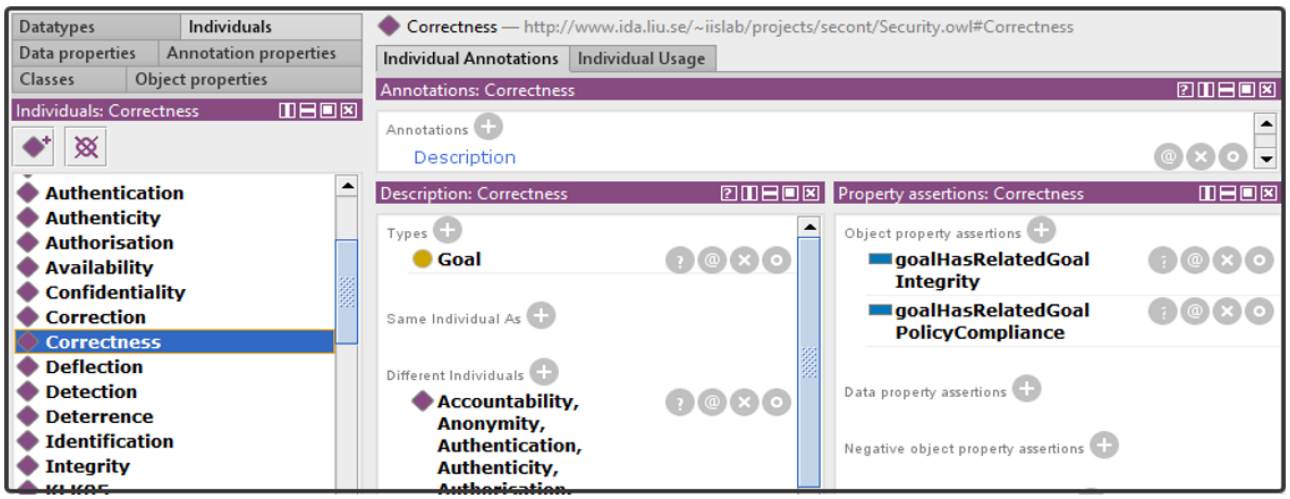

Рис. 7. Пример индивидуальной сущности цели онтологии

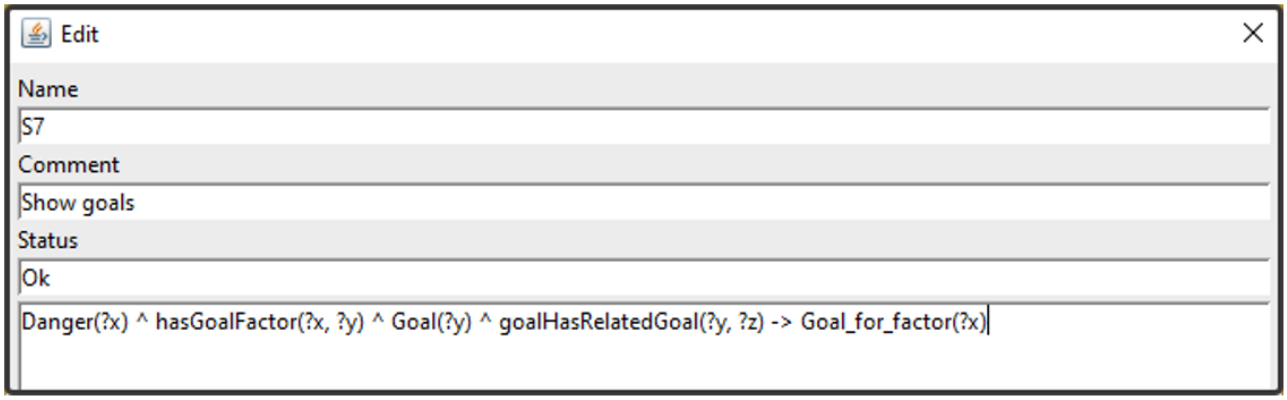

Рис. 8. Пример SWRL-правила, на основе другого SWRL-правила

системы. Преимуществом данного подхода является то, что строится дерево вывода, в соответствие с которым было принято решение. Это даёт возможность получить не количественную оценку защищенности, которая не несет подробной информации, а увидеть условия, влияющие на принятое решение (рис. 6).

В выбранной онтологии, основанной на стандарте ISO/IEC 27001, предусмотрены следующие цели: Authentication Goals (цели аутентификации данных); Confidentiality Goals (целостности); Integrity Goals (доступности).

Онтология уже изначально содержала набор индивидуальных сущностей, для которых определены связанные цели. Например, на рис. 6 для сущности Correctness (Корректность) видим, что она принадлежит классу Goal (целей) и содержит связанные сущности: goalHasRelatedGoal Integrity – имеет связанную сущность Доступность; goalHasRelatedGoal PolicyCompliance – имеет связанную сущность Соблюдение политики.

На основании данных отношений принимаются решения в правилах SWRL. Пре-

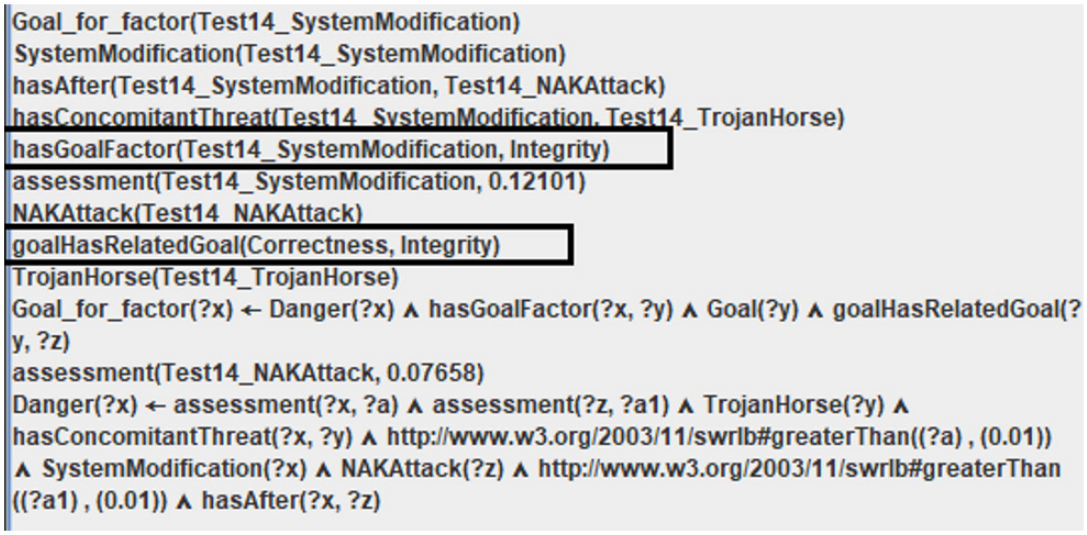

Рис. 9. Пример вывода на основе SWRL-правила

имуществом является то, что возможно построение правила на основе других правил. Рассмотрим пример (рис. 8), который используется в программном приложении.

Данное правило используется если на вход класса Goal_for_factor подается один из анализируемых критериев угрозы, была возможность увидеть цели дальнейших перспективных действий.

Благодаря проведенному анализу появляется возможность не только отследить пути, которые привели к принятому решению, но и увидеть предполагаемые цели (рис. 9). Например, для SystemModification (Изменения системы) предлагается цель Integrity (Обеспечения доступности). Также анализ показал, что сопутствующей целью выступает Correctness (Корректность). Кроме того, отслеживается динамика принятого решения для получения результата правила Danger, которое было рассмотрено ранее.

Применение SWRL-правил обеспечивает получение детальной картины принятого решения и предоставляет возможность оценить степень влияния на систему взаимодействующих факторов.

ЗАКЛЮЧЕНИЕ

Проблема интеграции разнородных данных по типам, источникам происхождения, надежности в единую модель описания предметной области на основе онтологий имеет большой потенциал для построения систем поддержки принятия решений для различных прикладных областей. Такой подход позволяет учитывать стандарты (реализованные онтологиями), следовательно, интегрироваться с существующими системами в данной предметной области. Возможно расширение модели знаний на основе объединения онтологий разного уровня представления или обучения, а также расширение аналитических возможностей таких систем через добавление правил логических выводов (заключений) как экспертным методом, так и через использование технологий DataMining и машинное обучение.

СПИСОК ЛИТЕРАТУРЫ

1. Документация стандарта по информационной безопасности ISO/IEC 27001: база данных. – Режим доступа: http://www.standardsdirect.org/iso17799.htm.

2. *Herzog, A.* An Ontology of Information Security / A. Herzog // International Journal of Information Security and Privacy, 1(4):1-23, 2007.

3. Security Ontology: база данных. – Режим доступа: https://www.ida.liu.se/divisions/adit/security/projects/secont/.

4. *Данковцева, А. Ю.* Система оценки информационной защищенности компании на основе интеграции разнородных данных и построения выводов по онтологии стандарта

ISO/IEC 27001 / А. Ю. Данковцева, В. В. Гаршина // Сборник студенческих научных работ факультета компьютерных наук ВГУ; под ред. Д. Н. Борисова, Воронежский государственный университет. – Вып. 13. – Воронеж : Издательский дом ВГУ, 2018. – С. 130–137.

5. Apache Jena. – Режим доступа: https://jena.apache.org/ (дата обращения 11.09.2018).

6. OWL API. – Режим доступа: http://owlapi.sourceforge.net/ (дата обращения 11.09.2018).

Гаршина Вероника Викторовна – канд. техн. наук, доцент, доцент кафедры технологий обработки и защиты информации Воронежского государственного университета.
Тел.: +7(903)854-05-87
E-mail: garshina@cs.vsu.ru

Степанцов Вячеслав Алексеевич – канд. техн. наук, доцент, доцент кафедры технологий обработки и защиты информации Воронежского государственного университета.
Тел.: +7(920)414-47-58
E-mail: mrstep@yandex.ru

Данковцева Анастасия Юрьевна – магистр 1 года обучения по программе «Безопасность информационных систем» факультета компьютерных наук Воронежского государственного университета.
Тел.:+7(920)455-65-53
E-mail: dankovtseva.nastya@yandex.ru

Garshina Veronika V. – Ph.D., Associate Professor, Department of Processing Technology and Information Security, Computer Sciences Faculty, Voronezh State University.
Tel.: +7(903)854-05-87
E-mail: garshina@cs.vsu.ru

Stepantsov Vyacheslav A. – Ph.D., Associate Professor, Department of Processing Technology and Information Security, Computer Sciences Faculty, Voronezh State University.
Tel.: +7(920)414-47-58
E-mail: mrstep@yandex.ru

Dankovtseva Anastasia Yu. – Master of 1 year training program «Information Systems Security», Computer Sciences Faculty, Voronezh State University.
Тел.:+7(920)455-65-53
E-mail: dankovtseva.nastya@yandex.ru