3. *Mehdi K.L. Sencar H.T. Memon N.* Blind source camera identification. International Conference on Image Processing, 2004. – Vol. 1. – P. 709-712.
4. *Фёдоров В.М., Макаревич О.Б. Рублев Д.П.Чумаченко А.Б.* Идентификация звуковых плат по создаваемым аудиоданным с помощью нейросетевых методов // Материалы III Международной конференции «Моделирование устойчивого регионального развития». – 2009. – № 2. – С. 90-94.
5. *Дьяконов В.П.* Вейвлеты – от теории к практике. – М.: СОЛОН-ПРЕСС, 2004. – 440 с.

Федоров Владимир Михайлович
Технологический институт федерального государственного автономного образовательного учреждения высшего профессионального образования «Южный федеральный университет» в г. Таганроге.
E-mail: vladmih@rambler.ru.
Россия, 347928, г. Таганрог, ул. Чехова, 2.
Тел.: 88634371905.

Рублёв Дмитрий Павлович
E-mail: rublev-d@yandex.ru.

Макаревич Олег Борисович
E-mail: mak@tsure.ru**.**

Панченко Евгений Михайлович
НИИ физики Южного федерального университета в г. Ростов-на Дону.
E-mail: kordon@kordon-rnd.ru.
344090, г. Ростов-на Дону, пр. Стачки, 194.
Тел.: 88632905121.

Fedorov Vladimir Mikhailovich
Taganrog Institute of Technology – Federal State-Owned Autonomy Educational Establishment of Higher Vocational Education “Southern Federal University”.
E-mail: vladmih@rambler.ru.
2, Chekhova street, Taganrog, 347928, Russia.
Phone: +78634371905.

Rublev Dmitry Pavlovich
E-mail: rublev-d@yandex.ru.

Makarevich Oleg Borisovich
E-mail: mak@tsure.ru.

Panchenko, Eugene Mikhailovich
Institute of Physics, Southern Federal University in Rostov-on-Don.
E-mail: kordon@kordon-rnd.ru.
194, pr. Strikes, Rostov-on-Don, 344090, Russia.
Phone: +78632905121.

УДК 681.324

И.Ю. Половко, Е.С. Абрамов

МАТЕМАТИЧЕСКАЯ МОДЕЛЬ АРХИТЕКТУРЫ СИСТЕМЫ ЗАЩИТЫ ИНФОРМАЦИИ, УСТОЙЧИВОЙ К АТАКАМ

Развертывание в сети комплекса из брандмауэров и сетевых системах обнаружения вторжений (СОА) и уверенность, что все эти компоненты безопасности правильно настроены, является сложной задачей. Хотя модели разрабатываются и для анализа эффективности брандмауэров, и для СОА, не существует общей математической модели для анализа их взаимодействия. В работе представлен комплексный подход к моделированию и

исследованию этих конфигураций, он рассматривает зависимость между этими двумя типами компонентов и может автоматически анализировать их совместное поведение и проводить верификацию сетей, которые включают несколько брандмауэров и СОА.

Формальная спецификация и анализ данных; обнаружение сетевых атак; межсетевые экраны; конфигурирование средств сетевой безопасности.

I.Yu. Polovko, E.S. Abramov

MATHEMATICAL MODEL OF INTRUSION-TOLERANT SECURITY SYSTEM ARCHITECTURE

Given a network that deploys multiple firewalls and network intrusion detection systems (COAs), ensuring that these security components are correctly configured is a challenging problem. Although models have been developed to reason independently about the effectiveness of firewalls and COAs, there is no common framework to analyze their interaction. This paper presents an integrated, constraint-based approach for modeling and reasoning about these configurations. Our approach considers the dependencies among the two types of components, and can reason automatically about their combined behavior.

Formal specification and analysis; network intrusion detection; firewalls; network configuration and security

Графовая модель. Для моделирования топологии ЛВС, учитывающей расположение межсетевых экранов и СОА, используется графовая модель [1].

Мы моделируем сеть как ориентированный граф, в котором пакеты перемещаются от узла к узлу. Граф имеет двунаправленные рёбра, где узлами являются или маршрутизаторы (роутеры), или области-подсети (areas), а каждое ребро соединяет маршрутизатор и подсеть. Межсетевые экраны смоделированы на узлах-маршрутизаторах, которые могут фильтровать трафик, проходящий через них. На рис. 1 показан пример графа корпоративной сети [2, 3].

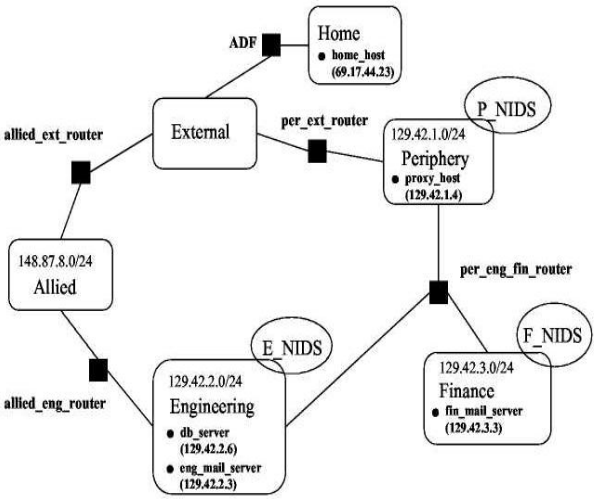

Рис. 1. Пример графа корпоративной сети

Подсети-области содержат непересекающиеся множества хостов, которые могут быть адресом источника или назначения отдельных пакетов. В этой модели отдельные хосты сами не являются узлами в графе, но их адреса используются для ограничения адресации пакетов, которые могут передаваться из одной подсети в другую.

Роутер имеет отдельный сетевой интерфейс для каждой подсети-области.

Определены *подмножества хостов*. Каждое подмножество может быть описано при помощи диапазона (CIDR) или как явное указание на входящие в него хосты.

Например, хосты в подсети engineering могут быть представлены как [129.42.2.*], или, если выбрано более абстрактное представление, как абстрактный набор из трех компьютеров, [eng_mail_server, db_server, eng_untrusted_host].

Определим операции на этих конечных множествах. Для этого графа определим:

Defhostset – операция определения (объявления) *подмножества* хостов,

Union – операция объединения (для создания *подмножества*),

Defhostset (internal,union(engineering,finance)),

Defhostset (corporate,union(periphery,internal)). def hostset (non_corporate, complement (corporate)). defhostset(mail_hosts,[eng_mail_server,fin_mail_server]). defhostset (trusted_hosts,

union(periphery,union(mail_hosts,[db_server]))). Defhostset (eng_untrusted, difference (engineering,trusted_hosts)).

Если у **finance** адреса из **[129.42.3.*],** а **periphery [129.42.1.*],** то полученная область **corporate** отождествляется с адресом **[129.42. [1,2,3].*].**

Представление пакетов. Пакеты – структуры с фиксированным набором полей:

- source port – порт источника;
- destination port – порт приёмника;
- IP addresses – IP-адреса;
- type –тип сервиса (ftp, http);
- orientation – от сервера или к серверу;
- payload – указание на тип атаки в терминах семейств сигнатур атак, принятых в Snort (например, smtp_attack_payload) [4].

Если используются цифровая форма адреса IPv4, каждая область адреса делится на четыре части, каждая ограничена в диапазоне от 0 . . . 255.

Язык пакетов (packet language) L представляет собой набор пакетов. Язык описывает адреса источников, которые, предположительно имеют отправленные пакеты, независимо от того, были ли они подделаны (spoofed). Язык представляется как ограниченная пакетная структура:

R = packet with [from_ip:F,to_ip:T,service:S,payload:P,...],

где каждое поле ассоциировано с соответствующим ему полем структуры пакета. Таким образом, R представляет собой набор:

$$\{p \mid from\_ip(p) \in F, to\_ip(p) \in T, service(p) \in S, payload(p) \in P, \ldots\}$$

Множество пакетных языков, могут быть выражены таким образом:

$$R_1 \cap R_2 = \{p \mid \ from\_ip(p) \in F_1 \cap F_2, to\_ip(p) \in T_1 \cap T_2, service(p) \in S_1 \cap S_2, \ldots\}$$

Фильтры (межсетевые экраны) и их состояния. Фильтр определяется для каждого направления ребра. Для ребра *e* фильтром *filter(e)* будет являться множество пакетов, которым разрешено проходить по этому ребру. Каждый роутер может иметь разные входящие и исходящие фильтры для каждой подсети-области, к которой он присоединён, в зависимости от своих сетевых интерфейсов.

Состояние фильтра (*filtering posture*) – установленные на данный момент (итерация модели) входящие и исходящие правила фильтрации для каждого сетевого интерфейса роутера. Например, следующий фильтр:

declare_filter(allied_eng_router,inbound,engineering, [policy(allied, any, []),
policy(union(engineering,finance),allied,all_tcp_services)])

определяет, что роутер между подсетями engineering и allied блокирует (отсекает) любой входящий со стороны engineering трафик, адрес источника которого при-

надлежит подсети allied, но при этом разрешает (пропускает) все TCP-пакеты, адрес источника которых принадлежит engineering или finance, а адрес назначения (т.е. направление) указан как allied. При этом все остальные пакеты блокируются по умолчанию.

Пути, траектории, события. Для *path P:(ei,... ,e_n)*, множество пакетов, которые могут пройти по этому пути, определяется как *packets(P)* = $\cap_{e \in P}$ *filter(e)*, т.е. функция от $e \in P$, которые удовлетворяют правилам фильтра, что как раз и определяется функцией.

Траектория определяется для непустого пакетного языка *L* и пути *P* и задаётся парой *(L,P).* Траектория *осуществима (возможна),* если *L* ⊆*packets(P)* (*L* входит или равно *P*). Для наборов подсетей (area sets) *S1* и *S2* можно определить пути *paths(S1,S2)* – наборы «не зацикленных» пакетов, которые исходят из *S1* и идут в *S2*.

Для подсетей a_1, a_2 определяем *paths*(a_1, a_2) для *paths*({a_1}, {a_2}) (т.е. для двух множеств, каждое из которых состоит из одного элемента).

Для наборов хостов *H1* и *H2* пути определяются как

$$paths(H_1, H_2) \stackrel{\text{def}}{=} \bigcup \{paths(a_1, a_2) \mid \ (hosts(a_1) \cap H_1) \neq \emptyset \text{ and } (hosts(a_2) \cap H_2) \neq \emptyset\}$$

т.е. пути (*H1, H2*) – все пути подсетей a_1, a_2, для которых имеется непустое множество пересечений *hosts*(a_1) и H_1, *hosts* (a_2) и H_2.

Для пути *P* и вершины графа *n*, полагаем, что $n \in P$, если *n* появляется в *P*.

Событие *event* (H_1,H_2,*L*) определяется на наборах хостов H_1,H_2 и пакетном языке *L*. Событие описывает пакет, который направлен из хоста, входящего в H_1 на хост, входящий в H_2. Для события *E*: *(H1,* H_2*, L)* набор *возможных (осуществимых)* траекторий определяется как

$$trajectories(E) \stackrel{\text{def}}{=} \{\langle L', P\rangle \mid \ P \in paths(H_1, H_2) \text{ and } L' = (packets(P) \cap L) \neq \emptyset\} \ .$$

т.е. определяются траектории, которые теоретически могут привести к возникновению этого события [5].

Проверка возникновения события. В модели используется сигнатурная IDS (СОА). Конфигурация СОА задаётся её положением в сети и набором пакетов, которые вызывают генерацию предупреждения (т.е. возникновение *события*) [6].

С учётом графа сети легко вычислить пакетный язык *packets(P)* для любого пути *path P*:

$$packets(\langle\rangle) := \text{any}$$

это пустой путь, с которого начинается вычисление. В этом случае разрешены все пакеты.

Для путей длиной n вычисляются ограничения для всех путей, расширяющих (продляющих) их на следующее ребро, «пересекая» их пакетные языки с фильтрами для нового ребра:

$$packets(\langle e_1, \ldots, e_n, e_{n+1}\rangle) := \\ \quad packets(\langle e_1, \ldots, e_n\rangle) \cap filter(e_{n+1}) \qquad \text{(extend)}$$

Набор пакетов *packets*(a_1,a_2), которые могут входить в информационный поток от подсети a_1 к подсети a_2, – это объединение *packets(P)* для всех путей *paths P* ∈ *paths(a1,a2).* Фильтры могут также осуществлять маршрутизацию – отдельные пакеты могут проходить только по указанным рёбрам.

В этом случае можно вычислить все возможные пути, по которым пакетный язык *L* может проходить через сеть, используя *L* в качестве начального значения

packets(<>) вместо *any*. Таким образом, можно вычислить *траектории*, для которых IDS *I* будет генерировать тревогу в узле *N* для данных, содержащихся в пакетном языке *L*:

1. Выбираем требуемый *alerts(I, N)*.
2. Вычисляем все возможные пути для в пакетного языка *L*.
3. Выбираем все пути, содержащие *N*.

С точки зрения отдельного узла *n* вычисляется набор пакетов, которые могут его достичь:

$$reach(n) = \bigcup_{\langle i,n\rangle \in Edges} (reach(i) \cap filter(\langle i,n\rangle)) \ .$$

Требования политики безопасности будут соблюдаться, если определённые сети или группы хостов будут порождать только определённые типы пакетов или будет обнаруживаться и исключаться спуфинг [1].

Набор пакетов *O(a)*, которые могут попасть в сеть *a*, определяется как *any*, если *a* – недоверенная сеть, и как *{P | source(P) ∈ a}* в ином случае. Таким образом, пакеты, которые могут достичь *a*, могут быть найдены как

$$reach(a) = \bigcup_{a' \neq a} (\mathcal{O}(a') \cap packets(a', a)) \ .$$

т.е. пересечение множества пакетов, могущих появиться в *a* пакетов, действительно предназначенных *a*.

Множество содержащихся в *a* пакетов *contents(a)* определяется как *contents(a) = O(a) ∪ reach(a).*

Проверка обнаружения событий. Проверить возможность генерации тревоги *E: (H_1,H_2, L)* можно следующим образом:

1. Вычисляя *trajectories(E),* представляющую все пути, по которым пакеты из пакетного языка *L* могут попасть в H_1, H_2.
2. Проверяя *L' ⊆guaranteedalert(P)* для каждого *< L',P >∈ trajectories(E).*

Траектории рассчитываются как *L ∩ packets(P)* для всех *P∈paths(H_1,H_2).*

Анализ сетевого потока. Рассмотрим возможную атаку **http** по направлению от **external** на веб-сервер, расположенный в **finance**.

Анализируя конфигурацию межсетевого экрана:

declare_filter(per_eng_fin_router,inbound,engineering, periphery, [policy(external, any, []),

можно полагать, что такая атака невозможна.

Атака теоретически возможна по двум путям – через **periphery** и **engineering**. Фильтр между **external** и **finance** блокирует **http**-трафик.

Проверка правильности конфигурации COA. Для иллюстрации основных принципов представления COA в модели рассмотрим политику для обнаружения атак, использующих **SMTP**-уязвимости, с хостов вне **corporate** на **engineering**. Как показывает рис. 1, полагаем, что три области, а именно **engineering**, **finance**, и **periphery**, соответствуют трем подсетям, так что каждая может контролироваться COA (область можно разложить на составляющие).

Во-первых, определяем пакетный язык для обобщённого сценария нападения:

define_packets(smtp_attack, [source:any, dest:any, service:smtp, orientation:any, payload:smtp_attack_payload]).

Это определение, с учётом ограничений на источники пакетов, описывает все пакеты, содержащие smtp_attack. Это значит, что любой пакет, исходящий не из **corporate**, должен гарантированно генерировать тревогу.

define_detection_policy(detect_smtp, [leaves:non_corporate, arrives:engineering, packets: smtp_attack]).

Конфигурация СОА определяет местоположение и пакеты, для которых она поднимает тревогу:

define_ids_config(candidate_ids, [location: periphery, packets: smtp_attack]).

В этом случае СОА развернута на **periphery** и настроена на обнаружение **SMTP-**атак, направленных на хосты в engineering. Однако эта конфигурация не покрывает путь для атак из **allied** в **engineering**. Проверяя конфигурацию при помощи функции candidate_ids на эффективность политики detect_smtp, получаем следующий результат:

Path: allied->allied_eng_router->engineering.

Undetected: packet with [service:smtp, payload:smtp_attack_payload,

from_ip:allied_host,

to_ip:[eng_mail_server, fin_mail_server]].

Если перенести расположение СОА с **periphery** на **engineering**, функция candidate_ids сообщит, что никакого нарушения политики detect_smtp нет.

Обнаружение многоэтапных атак. Сложные атаки могут состоять из нескольких этапов (шагов). Рассмотрим следующий сценарий. Злоумышленник из **external** планирует получить исходный код из **engineering** для переноса в **external**.

Чтобы обойти контроль доступа к сети на межсетевых экранах и обнаружение на NIDS, злоумышленник использует три этапа атаки:

1. Использование уязвимости на email server в **corporate**.
2. Использование скомпрометированного email server для получения исходного кода их хранилища на **engineering**.
3. Шифрование исходного кода и передача по email на хост **external**.

Шаг 3 не может быть обнаружен NIDS. Однако задачу можно решить настройкой нескольких NIDS, которые бы обнаруживали другие шаги атаки.

Чтобы обнаружить многоступенчатую атаку, наши политики обнаружения могут указывать неупорядоченные множества событий, за исключением одного события, описывающие сценарий атаки. Следующая политика определяет, что будет сгенерирована тревога, если будет обнаружена email-атака против **corporate mail host** и перенос кода на mail host.

define_packets(code_xfer_to_mailhost, [services:any,

source:corporate, dest:mail-hosts, payload:source_code]).

define_detection_policy(detect_sourcecode_exfiltration,

[event: sequence([leaves:non-corporate, arrives:mail-hosts, packets:smtp_attack], [leaves:corporate, arrives:mail-hosts, packets:code_xfer_to_mailhost])]).

Также можно сконфигурировать NIDS на **finance** и **engineering** для обнаружения email-атак против **corporate mail host**-ов и переноса кода на них.

define_ids_config(e_nids, [location: engineering, packets:

[source: non_corporate, destination: mail-hosts,

payload: smtp_attack_payload]])

define_ids_config(f_nids, [location: finance, packets:

[source: non_corporate, destination: mail-hosts,

payload: smtp_attack_payload]])

define_ids_config(e_nids, [location: engineering, packets: code_xfer_to_mailhost])

define_ids_config(f_nids, [location: finance, packets: code_xfer_to_mailhost]).

Расширение модели для учёта трансляции сетевых адресов (NAT). NAT (от англ. Network Address Translation – преобразование сетевых адресов) – это механизм в сетях TCP/IP, позволяющий преобразовывать IP-адреса транзитных пакетов. Также имеет названия IP Masquerading, Network Masquerading и Native Address Translation.

Преобразование адресов методом NAT может производиться почти любым маршрутизирующим устройством – маршрутизатором, сервером доступа, межсетевым экраном. Наиболее популярным является SNAT, суть механизма которого состоит в замене адреса источника (англ. *source*) при прохождении пакета в одну сторону и обратной замене адреса назначения (англ. *destination*) в ответном пакете. Наряду с адресами источник/назначение могут также заменяться номера портов источника и назначения.

Помимо source NAT (предоставления пользователям локальной сети с внутренними адресами доступа к сети Интернет) часто применяется также destination NAT, когда обращения извне транслируются межсетевым экраном на сервер в локальной сети, имеющий внутренний адрес и потому недоступный извне сети непосредственно (без NAT).

Разработанная математическая модель может быть расширена для учёта трансляции сетевых адресов. Сам механизм трансляции описывается следующим образом.

Фактически, в модели необходимо рассчитать постсостояние пакетного языка L после прохождения им узла n. Для этого введём функцию f_n, описывающую любую трансформацию пакетов в узле (NAT-ирование, порт-маппинг, шифрование или туннелирование), которая трансформирует пакет p в $f_n(p)$:

$$post(n, L) \stackrel{\text{def}}{=} \{p' \mid p' = f_n(p) \text{ for some } p \in L\} \quad .$$

Далее для учёта трансформации пакета в узле (или нескольких последовательных узлах), введём историю пакетов в каждую траекторию. Вместо пары *T:(L,P)*, траектория *T* примет вид

$$T : \langle \langle L_1, n_1 \rangle, \langle L_2, n_2 \rangle, \ldots, \langle L_k, n_k \rangle \rangle ,$$

где каждое значение T_i описывает пакеты в узле n_i.

Теперь можно отличить пакеты, которые ещё в источнике, от пакетов, уже прибывших по адресу.

В случае атаки или нарушения конфиденциальности, мы можем отличить начальные и конечные пакеты друг от друга. Например, можно написать такую политику, которая нарушается, если конфиденциальный пакет P, вышедший из защищённой подсети, приходит в незащищённую.

Примеры использования. Определим сетевые события, которые должны быть выявлены и зарегистрированы в подсистеме NIDS. Политика обнаружения нарушается, если описываемые её сетевые события могут произойти без генерации тревоги NIDS. Ниже перечислены некоторые события, которые необходимо обнаруживать:

AC: Нарушение политики контроля доступа (host из **engineering** обращается к Web-серверу в **finance)**;

EC-FHT-C: Атака из **external** на **corporate**, эксплуатирующая известные уязвимости **ftp, http** или **telnet** клиентов;

EC-S-SC: Атака из **external** на **corporate,** эксплуатирующая известные уязвимости **smtp** серверов и клиентов;

AC-S-SC: Атака из **allied** на **corporate,** эксплуатирующая известные уязвимости **smtp** серверов и клиентов;

SC: Передача исходного кода из **engineering** в **external**.

Решение:

AC:

Чтобы проверить политику контроля доступа, мы используем основанную на правилах сетевую СОА, чтобы обнаружить следующие типы пакетов:

define_packets(AC1,[source:external,destination:[proxy_host],

orientation: to_server, service: [ftp,http,telnet])
define_packets(AC2,[source:external,destination:[proxy_host],
orientation: any, service:complement([ftp,http,telnet]))
define_packets(AC3,[source:[proxy_host],destination:internal,
orientation: to_server, service: [ftp,http,telnet])
define_packets(AC4,[source:[proxy_host],destination:internal,
orientation: any, service:complement([ftp,http,telnet])).

Мы используем P_NIDS, настроенную на обнаружение этих пакетов для контроля **periphery**:

define_ids_config(periphery_ids, [location: periphery, packets: disjoin([AC1,AC2,AC3,AC4])]).

EC-FHT-C:

Создаём пакетный язык для реализации политики **EC-FHT-C**:

define_packets(EC_FHT_C1, [source:external, destination:corporate, orientation: from_server, service:[ftp,http,telnet], payload:known_attack])

SC:

Политика обнаружения может быть описана следующим образом:

define_detection_policy(detect_direct_sourcecode_leak, [leaves:engineering, arrives:external, packets: [payload: source-code]]).

Мы используем E_NIDS, настроенную на обнаружение этих пакетов для контроля **engineering,** чтобы обнаруживать передачу исходного кода через **ftp, http, smtp** или **telnet**.

Пакетный язык SC1 описывает **smtp**-соединения между **engineering,** и **external,** SC2 описывает **ftp-, http-** и **telnet**-соединения, инициированные в **engineering** к серверу в **external**:

define_packets(SC1, [source:engineering, destination:external, orientation: any, service:[smtp], payload:source-code])

define_packets(SC2, [source:engineering, destination:proxy_host, orientation: to_server, service: [ftp,http,telnet], payload:source-code])

Конфигурация E_NIDS:

define_ids_config(eng_nids, [location: engineering, packets: disjoin([SC1,SC2]))

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Guttman J.D.* Filtering postures: Local enforcement for global policies.
2. *Guttman J.D., Herzog A.L.* Rigorous automated network security man.
3. Com. 3Com Embedded Firewall. Software for the 3CR990 Network Interface Card (NIC) Family, Dec. – 2001.
4. *Roesch M.* Snort: Lightweight intrusion detection for networks.
5. *Porras P., Neumann P.* EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In Proceedings of the 20th National Information Systems Security Conference, Baltimore, MD, Oct. 1997. – P. 353-365.
6. *Cheadle M., Harvey W., Sadler A.J., Schimpf J., Shen K., Wallace M.G.* ECLiPSe: An Introduction. Technical Report IC-Parc-03-1, IC-Parc, Imperial College London, 2003.

Половко Иван Юрьевич
Технологический институт федерального государственного автономного образовательного учреждения высшего профессионального образования «Южный федеральный университет» в г. Таганроге.
E-mail: ivan.polovko@gmail.com.
347928, г. Таганрог, ул. Чехова, 2.
Тел.: 88634371905.

Абрамов Евгений Сергеевич
E-mail: abramoves@gmail.com.

Polovko Ivan Yur'evich
Taganrog Institute of Technology – Federal State-Owned Autonomy Educational Establishment of Higher Vocational Education "Southern Federal University".
E-mail: ivan.polovko@gmail.com.
2, Chexova street, Taganrog, 347928, Russia.
Phone: +78634371905.

Abramov Evgeny
E-mail: abramoves@gmail.com.

УДК 004.732.057

Н.Е. Горло, О.Ю. Пескова

ПОДХОДЫ К ПОСТРОЕНИЮ ЗАЩИЩЕННОЙ СИСТЕМЫ УПРАВЛЕНИЯ КОНТЕНТОМ

Рассмотрены основные подходы к построению защищенной системы управления контентом с упором на разработку сайтов для поддержки электронных библиотек. Представлена разработанная система управления контентом электронных публикаций, которая обеспечивает функции регистрации, аутентификации, разграничения доступа, публикации и редактирования материалов, защищенное хранение пользовательских данных.

Электронные библиотеки; CMS; аутентификация и разграничение доступа.

N.E. Gorlo, O.Yu. Peskova

APPROACHES TO THE CONSTRUCTION OF PROTECTED CONTENT MANAGEMENT SYSTEM

The paper discusses the main approaches to the construction of secure content management systems, with emphasis on the development site to support digital libraries. The developed content management system of electronic publications, which provides the functions of registration, authentication, access control, publishing and editing of materials protected storage of user data.

Digital libraries; CMS; authentication and access control.

Введение. В наше время публикации все чаще появляются не на бумажных носителях, а в глобальной сети Интернет. Статьи располагаются на специализированных порталах, на сайтах учебных заведений, персональных страницах авторов. Достоинство электронной публикации прежде всего в том, что она дает возможность автору постоянно развивать, дополнять и править свой материал – в соответствии с собственным развитием и с критическими комментариями и рецензиями на выложенный в открытый доступ материал. Кроме того, электронная публикация дает возможность использовать элементы мультимедиа, перекрестные ссылки и другие элементы, позволяющие автору максимально полно и доступно представить свои идеи. Доступ к выложенным в сеть материалам открыт миллионам читателей благодаря поисковым сервисам. Поэтому не удивительно, что в сети постоянно появляются новые специализированные ресурсы, предназначенные для размещения и рецензирования научных публикаций (например, сайт научных публикаций сотрудников Российской академии наук www.ras.ru, журнал научных публикаций аспирантов и докторантов www.jurnal.org), и для каждого нового участника «научной паутины» возникает закономерный вопрос – строить ли сайт на основе уже существующей системы управления контентом (CMS) или разработать свой собственный движок.