УДК 681.3

В.А. Артамонов, Е.В. Артамонова

# ФОРМАЛЬНАЯ МОДЕЛЬ ПОЛИТИКИ БЕЗОПАСНОСТИ МЕХАНИЗМА ЗАЩИТЫ ОТ УГРОЗЫ ОТКАЗА В ОБСЛУЖИВАНИИ

Информационная безопасность определяется выполнением требований известной «триады»: ***конфиденциальности, целостности и доступности.***

Большинство исследований безопасности информационных технологий (ИТ) традиционно связаны с угрозами раскрытия и целостности. Вместе с тем, развитие систем реального времени (в особенности военные и ядерные приложения) и информационно-справочных систем диктуют пристальное внимание к проблеме обеспечения доступности к ресурсам систем ИТ.

Отказ в обслуживании (Denial of Servise – ***DoS***), без всякого сомнения, является наиболее известной формой атаки на доступность к ресурсам систем ИТ. Когда атака данного типа проводится одновременно через множество узлов распределённой системы имеет место распределённая атака ***DoS*** (***DDoS*** – distributed ***DoS***). Атаки ***DoS*** отличаются от атак других типов. Они не нацелены на получение доступа к системе ИТ или на получение из этой системы какой либо информации. ***DoS*** делает систему ИТ недоступной для обычного (штатного) использования за счёт превышения допустимых пределов параметров функционирования системы или её приложений. В случае использования некоторых серверных приложений (таких как ***Web-***сервер или ***FTP-***сервер) атаки ***DoS*** могут заключаться в том, чтобы занять все соединения, доступные для этих приложений и держать их в занятом состоянии, не допуская обслуживания обычных пользователей. Большинство атак ***DoS*** опирается не на программные ошибки или бреши в системе безопасности, а на общие слабости системной (сетевой) архитектуры ИТ. Некоторые атаки сводят к нулю производительность систем (сетей) ИТ, переполняя их нежелательными и ненужными пакетами или сообщая ложную информацию о текущем состоянии системных ресурсов. Этот тип атак трудно предотвратить, так как для этого требуется координация действий с провайдером сетевых (интернет) услуг. Практические правила в виде политик безопасности механизмов защиты от ***DoS*** можно найти на сайте группы

экстренного реагирования на компьютерные проблемы (***CERT*** – Computer Emergency Response Team) [1]. В основе разработки политик безопасности, как правило, лежат формальные модели безопасности ИТ. Соответственно, с этих позиций перейдём к рассмотрению формализмов реализации политики безопасности механизма защиты от атаки ***DoS.***

Для того чтобы понять угрозу отказа в обслуживании, необходимо иметь в виду, что безопасные ИТ-системы служат промежуточным звеном при запросе услуг пользователей посредством монитора обращений (пересылок). Такой промежуточный монитор обращений позволяет рассмотреть запросы услуг в терминах простой модели, в которой пользователи являются зарегистрированными либо незарегистрированными, и обеспечиваются запрашиваемой услугой либо получают отказ. В случаях, когда зарегистрированным пользователям не предоставляется запрашиваемая услуга, говорят, что имеет место отказ в обслуживании.

В понятии предоставления или отказа в обслуживании должно быть включено время. Добавление времени к простой модели запроса услуг основано на том, что каждая услуга должна быть связана с некоторым периодом времени, называем ***максимальным временем ожидания*** (***MWT***). Для некоторой услуги ***MWT*** определяется как длина промежутка времени после запроса услуги, в течение которого считается приемлемым предоставление этой услуги.

Можно трактовать приведенное выше определение по-другому. А именно, рассматривать ***MWT*** как период времени, в течение которого запрашиваемая услуга не устаревает. Иными словами, если после запроса услуга обеспечивается в течение слишком долгого времени, то можно считать, что она не предоставлена или ее нельзя больше использовать. Заметим, что хотя приведенное выше определение и не выражено в терминах пользователей, запрашивающих услуги, может оказаться, что для данной услуги ***MWT*** будет различным для различных пользователей. Дав определение ***MWT***, мы можем теперь ввести точное определение угрозы ***DoS***, а именно, будем говорить, что имеет место угроза всякий раз, когда услуга с соответствующим ***DoS*** максимальным временем ожидания (***MWT***) запрашивается зарегистрированным пользователем в момент времени **t** и не предоставляется этому пользователю к моменту времени (***t + MWT***). Это означает, что ответы на запросы зарегистрированных пользователей не должны опаздывать.

При более тщательном рассмотрении угрозы ***DoS*** и предложенного понятия ***MWT*** возникают некоторые вопросы. Например, угрозы ***DoS*** можно полностью избежать, если определить ***MWT*** для всех услуг равным бесконечности. Однако этот подход не

годится для тех случаев, когда величина ***MWT*** определяется некоторой значимой операционной характеристикой среды. Кроме того, значение ***MWT*** можно определить только для конкретного набора услуг, для которых оно необходимо, то есть можно определить некоторое подмножество активов системы как особенно критическое, тогда значения ***MWT*** будут соответствовать услугам, связанными с этими критическими активами.

Для того чтобы проиллюстрировать приведенные выше понятия ***DoS***, рассмотрим простое требование ***DoS*** для типичной вычислительной системы и выразим его, используя расширение логики предикатов первого порядка, называемое временной логикой [2].

Требование ***DoS*** будет выражено по отношению к системе, которая состоит из набора субъектов, активно инициирующих запросы услуг в различных состояниях системы. Значение ***MWT*** для каждого запроса услуги ***r*** может быть получено вычислением функции ***mwt(r),*** областью определения которой являются все запросы услуг, а областью значений – множество положительных целых чисел. Если максимальное время ожидания услуги равняется некоторому целому числу ***n***, то удовлетворение запросов этой услуги должно происходить раньше, чем через ***n+1*** единицу времени.

Требование заключается в том, что если ***mwt(r) = n,*** то удовлетворение запроса услуги ***r*** в момент времени ***t*** должно произойти к моменту времени ***t+n***. Это требование легко формализуется в терминах временной логики использованием эвентуального оператора (обозначаемого ***a***). Этот оператор определяется следующим образом: если ***P*** – предикат в логике первого порядка, то ***aP*** – утверждение во временной логике. Если утверждение ***aP*** истинно для некоторого состояния ***s***, это означает, что ***P*** должно быть истинно для состояния ***s*** или для некоторого состояния, имеющего место после ***s.*** Можно установить простые временные границы, определяя оператор a следующим образом: если ***a(t)P*** истинно для некоторого состояния, это означает, что ***P*** должно быть истинно в некотором последующем состоянии, которое достигается раньше, чем через ***t*** единиц времени.

Для заданных операторов можно определить требование в приведенном выше примере. Предположим, что всякий раз, когда субъект производит запрос услуги ***r*** в некотором состоянии, предикат ***REQ(r)*** является истинным. Можно предположить, что этот предикат не будет истинным в последующих состояниях, в которых ответ на запрос может все еще ожидаться. Более того, предположим, что всякий раз, когда запрос услуги r удовлетворяется в некотором состоянии, истинным является предикат ***GRANT(r).*** Для упрощения примера предположим далее, что каждый запрос услуги единственен (это позволит облегчить измерение времени от

запроса до завершения). Наш пример можно описать следующим образом:

$$REQ(r) \rightarrow \Diamond (mwt(r))\ GRANT(r).$$

Далее в терминах субъектов и объектов рассмотрим мандатную модель механизма защиты от угрозы отказа в обслуживании [3].

Субъектам системы соответствуют приоритеты, которые могут быть одинаковы, ниже или выше по сравнению с приоритетом любого другого субъекта. Объектам соответствуют степени критичности, имеющие аналогичную иерархическую структуру. Субъект может требовать услугу у вычислительной системы, запрашивая доступ к объектам системы. Говорят, что субъект получает отказ в обслуживании, если его запрос зарегистрирован, но не удовлетворен в течение соответствующего ***MWT***.

При описании модели необходимо рассмотреть условия, при которых один субъект может отказать в обслуживании другому субъекту. Такой отказ может быть вполне приемлем для одних случаев и совершенно не допустим для других. Правила, составляющие модель, нацелены на то, чтобы определить условия, при которых отказ в обслуживании был бы недопустим.

Рассмотрим правила, описывающие мандатную модель механизма защиты от ***DoS***. Первое правило – "никаких отказов вверх" (***NDU***) – основано на том наблюдении, что никаким объектам с более низким приоритетом не позволено отказывать в обслуживании субъектам с более высокими приоритетами. Однако некоторым субъектам с более высоким приоритетом (например администраторам системы) должна предоставляться возможность отказывать в обслуживании объектам с более низким приоритетом, если первые того желают.

Второе правило, являющееся просто обобщением первого, представляет собой альтернативу, которую можно использовать в тех приложениях, для которых защищенным от угроз отказа в обслуживании должно быть лишь некоторое подмножество объектов. Таким образом, это правило учитывает то, что проблема отказа в обслуживании может стоять только для объектов из некоторого конкретно определенного множества, то есть это правило позволяет обеспечить защиту от отказа в обслуживании только для особого множества объектов.

Это более общее правило требует, чтобы субъекты с более низкими приоритетами не препятствовали запросам услуг субъектов с более высокими приоритетами, производимыми через объекты из некоторого конкретно определенного множества. Это множество, которое мы обозначим через ***C***, обычно содержит те объекты, которые являются наиболее критическими и для которых

предоставляемые услуги никогда не должны устаревать.

Правило ***NDU(C)*** утверждает, что ни один субъект не может отказать запросам, сделанным субъектом с более высоким приоритетом через объекты из множества ***C***. Второе правило особенно полезно для систем, в которых необходимо обеспечивать защиту от ***DoS*** только для выбранного множества критических услуг. Например, система, выполняющая некоторую определенную роль, может содержать лишь небольшое множество услуг, напрямую связанных с этой ролью. В результате защиту от ***DoS*** с использованием правила ***NDU(C)*** можно обеспечить только для этих критических услуг. Это значительно сократит стоимость и трудность реализации стратегии защиты от ***DoS***.

Главным преимуществом двух представленных правил является то, что они дают средство для предотвращения угрозы отказа в обслуживании на основе понятия (приоритета), предположительно уже существующего для данной системы. Например, для большинства операционных систем существует понятие приоритета процессов. Данные правила также являются гибкими в том смысле, что их легко приспособить к данной системе. Ограничение на объекты в правиле ***NDU(C)*** можно выразить в терминах какого-либо определения критичности объекта.

Недостаток этих правил заключается в том, что они имеют смысл только для систем, в которых можно определить несколько приоритетов. Если это не так, тогда должны быть определены подходящие аналогичные правила внутри уровня с одним приоритетом. Например, формулировка этих правил для ПЭВМ с одним единственным пользователем не будет иметь большого смысла.

Приведенные выше правила касаются условий, которых необходимо избегать, если необходимо гарантировать, что действия нарушителя не создадут условий для отказа в обслуживании. Однако среди ситуаций, которые могут вызвать угрозу отказа в обслуживании, встречаются такие которые затрагивают вопросы времени и пространства (векторного), не выявляемые простыми правилами типа приведенных выше.

Представим модель распределения ресурсов в виде политики безопасности, позволяющую определить правила и стратегии отказа в обслуживании в терминах детального распределения ресурсов, включающего предоставление услуг пользователям систем ИТ. Данная модель отличается от традиционных уровневых правил, поскольку в ее основе лежит идея о том, что для выполнения нужного задания субъектам необходимы определенные пространственные и временные требования к ресурсам. Отказ в обслуживании происходит в том случае, если распределение пространства и времени для некоторого процесса не отвечает соответствующим требованиям. Такие понятия, как

конечное время ожидания (***FWT***) и максимальное время ожидания (***MWT***) являются основополагающими.

Данная модель включает в себя вполне соответствующий стратегии отказа в обслуживании набор правил в виде политик безопасности, характеризующих семейство систем ИТ.

И так, пусть ***P*** – множество активных процессов, ***R*** - множество пассивных ресурсов, ***c*** – некоторая фиксированная граница, используемая для обозначения общего максимального числа единиц всех типов ресурсов, доступных в исследуемой системе. Через вектор распределения $A_p$ обозначим для каждого ресурса число единиц ресурса, выделенных для процесса ***p*** в некотором состоянии. Таким образом, вектор распределения можно рассматривать как своего рода отображение выделения ресурсов для процесса. Для формирования информации о том, является ли процесс текущим или застывшим, используется особый тип ресурсов – ресурс ***CPU***. В частности, всякий раз, когда $A_p(CPU) = 1$, будем говорить, что истинным является ***running(p),*** а всякий раз, когда $A_p(CPU) = 0$, будем говорить, что истинным является ***asleep(p).***

Вектор пространственных требований $^{S}Q_p$ означает число единиц каждого ресурса, требуемое процессом p для выполнения необходимого задания в некотором состоянии. Предполагается, что процессы могут определять множество ресурсов, необходимых им для завершения работы, до того, как они ее начнут. Функция ***T(p)*** показывает, когда в последний раз изменились часы для процесса с целью отражения реального времени. Вектор временных требований $^{T}Q_p$ обозначает объем времени, необходимого каждому ресурсу процесса ***p*** для выполнения работы. Как и в случае пространственных требований, предполагается, что процесс может определять временные требования для конкретного задания.

Ниже перечислены восемь правил, составляющих политику безопасности [3]. Каждое правило ограничивает семейство систем, соответствующих этой модели. Прежде чем вводить эти правила, важно заметить, что "помеченные" переменные ***(***например ***running(p))*** показывают значение переменной после одного перехода.

$$(R1)\ \mathbf{S}\ A_p \le c.$$
$$\{p \in P$$

Правило ***R1*** утверждает, что сумма единиц выделенных ресурсов для всех процессов из ***P*** должна быть меньше системной границы ***c***. Стратегии, нарушающие это правило, являются неправдоподобными, поскольку невозможно выделить ресурсов больше, чем имеется в наличии.

$$\text{(R2) if running(p) then } {}^{S}Q_p = 0.$$

Правило ***R2*** утверждает, что текущие процессы должны иметь нулевые пространственные требования. Аргументируется это тем, что если процесс не обладает всеми ресурсами, которые необходимы ему в некотором состоянии, то не имеет смысла продолжать этот процесс, пока требования не будут выполнены. "Зависание" происходит, когда процесс имеет ненулевые пространственные требования, которые никогда не будут выполнены (или будут выполнены поздно).

$$\text{(R3) if running(p) and running(p)' then } A_p' = A_p.$$

Конструкция ***"running(p) and running(p)' "*** в правиле ***R3*** означает, что в некотором состоянии процесс ***p*** является текущим и остается текущим и в следующем состоянии. Это правило утверждает, что для текущих процессов распределение ресурсов не меняется. Это действительно мощное предположение, поскольку из него следует, что во время выполнения текущие процессы не выгружаются в случае, если произойдет какое-либо перераспределение ресурсов (в отличие от перераспределения ресурсов ***CPU***).

$$\text{(R4) if } A_p(CPU)' = A_p(CPU) \text{ then } T(p)' = T(p).$$

Правило ***R4*** утверждает, что часы процесса изменяются только с изменением распределения ***CPU***. Предполагается, что единицы времени - это положительные целые числа, всегда возрастающие с изменением времени (как будет определено ниже).

$$\text{(R5) if } A_p(CPU)' \neq A_p(CPU) \text{ then } T(p)' > T(p).$$

Правило ***R5*** утверждает, что часы изменяются только для того, чтобы отразить увеличение во времени. Заметим, что у каждого процесса имеются свои собственные часы, а синхронизации различных часов друг с другом или с какими-либо часами, показывающими реальное время, не предпринимается.

$$\text{(R6) if asleep(p) then } {}^{S}Q_p' = {}^{S}Q_p + A_p - A_p'.$$

Правило ***R6*** утверждает, что пространственные требования устанавливаются для застывших процессов. Другими словами, если процесс не является текущим, он должен определить те ресурсы, которые ему потребуются для выполнения задания. Как только все эти ресурсы получены, процесс возобновляется и становится текущим.

$$\text{(R7) if asleep(p) then } {}^{T}Q_p' = {}^{T}Q_p.$$

Правило ***R7*** утверждает, что для застывших процессов временные требования не устанавливаются, то есть тогда как

пространственные требования для застывших процессов устанавливаются, временные требования для таких процессов не устанавливаются.

(R8) if running(p) and asleep(p)' then $A_p' = A_p$ - CPU.

Правило ***R8*** утверждает, что переходы, в результате которых процесс останавливается, перераспределяют только ресурсы ЦПУ. Другие изменения в распределении должны происходить после возобновления процесса.

Можно использовать данную модель распределения ресурсов для описания некоторых стратегий. Например, применяя ***MPP***, можно выразить стратегию конечного времени ожидания (***FWT***). Для указания интервалов, которые могут возникнуть после многократных переходов, мы используем оператор временной логики ***leads_to*** (то есть ***A leads_to B*** означает, что во всех последующих состояниях из ***A*** в конце концов следует ***B***). Для указания интервалов, которые могут возникнуть после многократных переходов, можно выразить максимальное время ожидания (***MWT***) аналогичным образом.

*FWT*: " p, s : ∃ s' : s'(running(p)) and s *leads_to* s'.

В выражении, приведенном выше, ***s(x)*** означает, что ***x*** истинно для состояния ***s***, а ***s leads_to s'*** означает, что ***" s, s' : s((T(p) =n))*** и ***s'((T(p) =m))*** и ***m>n***. ***FWT*** означает, что пользователи в конце концов получат запрашиваемые ресурсы, то есть они в конечном итоге получат ***CPU*** для выполнения работы. Аналогичным образом можно выразить максимальное время ожидания (***MWT***).

*MWT*: ∃ b : ∀p, s : ∃ s' : s'(running(p)) and s *leads_to*(b) s'.

В этом выражении ***s leads_to(b) s'*** означает, что ***Vs, s' : s((T(p) =n)) и s'((T(p) =m)) и m-n ≤ b. MWT*** отличается от ***FWT*** тем, что здесь накладывается ограничение на время ожидания пользователями возможности продвижения в выполнении задания.

Литература

1.http://www.cert.org/tech_tips/denial_of_service.html

2. Справочная книга по математической логике: В 4-х частях/Под ред. Дж. Барвайса. – Ч.1. Теория моделей: Пер. с англ. – М.: Наука, Главная редакция физико-математической литературы, 1982 г. – 392 с.

3. J.Millen "Resource allocation model for denial of service", IEEE simposium on research in security and privacy, 1992.

МНОО «МАИТ», г. Минск