ИМИТАЦИОННЫЕ ИССЛЕДОВАНИЯ ТЕХНИЧЕСКИХ СИСТЕМ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ (ИБ)

Требования к моделям исследования технических систем обеспечения информационной безопасности:

1. Имитационные модели (Имитаторы) исследования ИБ воздушного судна представляют собой программные средства, моделирующие потоки угроз совместно с функционированием СЗИ, противодействующим таким угрозам.
2. Имитация осуществляется в соответствии с математическим аппаратом теории массового обслуживания (*Queue Theory*).
3. Имитационные модели позволяют выполнять количественную оценку защищённости информации при использовании СЗИ.
4. Имитационные модели реализуются в интерпретирующем платформонезависимом средстве GPSS (General Ppurpose Ssimulation Systems).
5. Интерфейс Имитатора определяется диалоговой средой GPSS World.
6. Имитационные модели представляются в виде функциональных блоков, объединенных в группы, соответствующие основным объектам моделируемой системы: «Нарушитель», «СЗИ» и «Защищаемые ресурсы»
7. Имитатор может быть установлен как на отдельном компьютере, так и в локальной сети компьютеров.
8. В случае локальной сети Имитатор может быть установлен на специально выделенном компьютере либо на одной из рабочих станций автоматизированной системы управления.

9. Имитатор может быть использован для имитации функционирования системы управления воздушным судном в целом.
10. Имитатор может функционировать как на базе проприетарной ОС (Microsoft Windows, Macintosh), так и на базе свободно распространяемой ОС (семейства Linux) при использовании специальных эмуляторов (Wine).

РАЗРАБОТКА ЭЛЕМЕНТОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИМИТАТОРА

Язык GPSS (General Purpose Simulation System), ориентированный на процессы, разработан еще в 1969 г., но продолжает широко использоваться. Язык реализован в ряде программ имитационного моделирования под различные версии операционных систем.

Модель (программа) на языке GPSS представляет собой последовательность операторов (их называют блоками), отображающих события, происходящие в моделируемой системе массового обслуживания (СМО) при перемещениях транзактов. Поскольку в интерпретаторах GPSS реализуется событийный метод и в СМО может быть одновременно много транзактов, то интерпретатор будет попеременно исполнять разные фрагменты программы, имитируя продвижения транзактов в текущий момент времени до их задержки в некоторых устройствах или очередях.

Операторы (блоки) GPSS имеют следующий формат:

<метка> <имя_оператора> <поле_операндов> [<комментарий>]

Метка может занимать позиции, начиная со второй, имя оператора — с восьмой, поле операндов — с девятнадцатой, комментарий обязательно отделяется от поля операндов пробелом.

Поле операндов может быть пусто, иметь один или более операндов, обозначаемых ниже при описании блоков символами **A**, **B**, **C** и т.д.

Операндами могут быть идентификаторы устройств, накопителей, служебные слова и стандартные числовые атрибуты (СЧА). К СЧА относятся величины, часто встречающиеся в разных задачах. Это, например, такие операнды, как **S** — объем занятой памяти в накопителе, **F** — состояние устройства, **Q** — текущая длина очереди, **P** — параметр транзакта (каждый транзакт может иметь не более **L** параметров, где **L** зависит от интерпретатора), **V** — целочисленная переменная (вещественная и булева переменные обозначаются **FV** и **BV** соответственно), **X** — хранимая переменная (переменная, для которой автоматически подсчитывается статистика), **K** — константа, **AC1** — текущее время, **FN** — функция, **RN** — случайная величина, **RN1** — случайная величина, равномерно распределенная в диапазоне [0, 1] и др. При этом ссылки на СЧА записываются в виде **<СЧА>$<идентификатор>**. Например, **Q$ORD** означает очередь ORD или **FN$COS** — ссылка на функцию COS.

Рассмотрим наиболее часто встречающиеся операторы, сопровождая знакомство с ними простыми примерами моделей.

Источники заявок описываются блоком

```
GENERATE   A,B,C,D,E
```

Здесь **A** и **B** служат для задания интервалов между появлениями заявок, при этом можно использовать один из следующих вариантов:

- интервал — равномерно распределенная в диапазоне [A_B, A+B] случайная величина;
- интервал — значение функции, указанной в B, умноженной на A;

C — задержка в выработке первого транзакта; **D** — число вырабатываемых источником заявок; **E** — приоритет заявок. Если **D** пусто, то число вырабатываемых транзактов неограниченно. Например,

```
GENERATE   6,FN$EXP,,15
```

Этот оператор описывает источник, который вырабатывает 15 транзактов с интервалами, равными произведению числа 6 и значения функции EXP;

```
GENERATE   36,12
```

Здесь число транзактов неограниченно, интервалы между транзактами — случайные числа в диапазоне [24, 48].

Функции, на которые имеются ссылки в операторах, должны быть описаны с помощью блока следующего типа:

```
M    FUNCTION   A,B
```

За ним следует строка, начинающаяся с первой позиции:

```
X1,Y1/X2,Y2/X3,/.../Xn,Yn
```

Здесь метка **M** — идентификатор функции, **A** — аргумент функции, **B** — тип функции, **Xi** и **Yi** — координаты узловых точек функции, заданной таблично. Например:

```
EXP   FUNCTION   RN1,C12
0,0/.2,.22/.4,.51/.5,.6/.6,.92/.7,1.2/.8,1.61/.9,2.3/.95,3/.99,4.6/.999,6.9/1,1000
```

Это описание непрерывной (**C**) функции EXP, заданной таблично 12-ю узловыми точками, аргументом является случайная величина (RN1), равномерно распределенная в диапазоне [0, 1]; или:

```
BBB   FUNCTION   *4,D6
1,2/2,5/3,11/4,20/5,18/6,12/7,9
```

Дискретная (**D**) функция BBB задана 6-ю узловыми точками, аргумент — четвертый параметр транзакта, возбудившего обращение к функции BBB.

Здесь аргумент задан с использованием косвенной адресации, признаком которой является символ *, т.е. запись *4 означает, что аргументом является величина, указанная в 4-м параметре транзакта, вызвавшего функцию (в данном примере можно было бы использовать равноценную запись *p4). В общем случае косвенная адресация выполняется путем записи операнда в виде СЧА*СЧА, например, S*X2, что означает емкость памяти, занятая в

накопителе, именем которого является значение переменной X2, или Q*p5 — длина очереди с именем, записанным в параметре 5 транзакта.

Транзакты могут порождаться и оператором размножения:

```
SPLIT     A,B,C
```

Новые транзакты порождаются, когда в данный блок входит некоторый транзакт. При этом создается семейство транзактов, включающее основной (вошедший в блок) транзакт и A его копий. Основной транзакт переходит в следующий по порядку блок, а его копии переходят в блок с меткой B. Для различения транзактов параметр C основного транзакта увеличивается на 1, а транзактов-копий — на 2, 3, 4,... и т.д.

Обратное действие — сборка транзактов выполняется операторами:

```
ASSEMBLE   A
GATHER     A
```

Согласно оператору ASSEMBLE, первый из вошедших в блок транзактов выйдет из него только после того, как в этот блок придут еще A-1 транзактов того же семейства. Второй оператор отличается от предыдущего тем, что из блока выходят все A транзактов.

Операторы занятия транзактом и освобождения от обслуживания устройства A:

```
SEIZE     A
RELEASE   A
```

Задержка в движении транзакта по СМО описывается оператором:

```
ADVANCE   A,B
```

A и B имеют тот же смысл, что и в операторе **GENERATE**.

Пример 1

Обслуживание транзакта в устройстве WST продолжительностью a единиц времени, где a — равномерно распределенная в диапазоне [7,11] случайная величина, описывается следующим фрагментом программы:

```
SEIZE     WST
ADVANCE   9,2
```

```
RELEASE    WST
```

Аналогично описывается занятие транзактом памяти в накопителе:

```
ENTER      A,B
```

Здесь помимо имени накопителя (A) указывается объем занимаемой памяти (B).

Освобождение B ячеек памяти в накопителе A выполняется оператором:

```
LEAVE      A,B
```

Для накопителей в модели нужно задавать общий объем памяти, что делается в следующем описании накопителя:

```
M    STORAGE    A
```

Здесь: M — имя накопителя, A — объем его памяти.

Если транзакт приходит на вход занятого устройства или на вход накопителя с недостаточным объемом свободной памяти, то транзакт задерживается в очереди к этому устройству или накопителю. Слежение за состоянием устройств и очередей выполняет интерпретатор. Но если в модели требуется ссылаться на длину очереди или собирать статистику по ее длине, то требуется явное указание этой очереди в модели. Делается это с помощью операторов входа в очередь и выхода из очереди:

```
QUEUE      A,B
DEPART     A,B
```

Согласно этим операторам очередь A увеличивается и уменьшается на B единиц соответственно, если B=1, то поле B можно оставить пустым.

Движение транзактов выполняется по маршруту, заданному последовательностью операторов в модели. Если требуется изменение естественного порядка, то используется оператор перехода. Оператор условного перехода имеет вид:

```
TEST XX    A,B,C
```

В соответствии с ним переход к оператору, помеченному меткой C, происходит, если не выполняется условие A XX B, где XX ∈ {**E**, **NE**, **L**, **LE**, **G**, **GE**}; **E** — равно; **NE** — неравно; **L** — меньше; **LE** —

меньше или равно; **G** — больше; **GE**— больше или равно (XX всегда размещается в позициях 13 и 14).

Пример 2

Приходящие пользователи ожидают обслуживания, если длина очереди не более 4, иначе от обслуживания отказываются. Соответствующий фрагмент программы

```
TEST LE    Q$STR,4,LBL
QUEUE      STR
SEIZE      POINT
DEPART     STR
ADVANCE    50,16
RELEASE    POINT
...
LBL   TERMINATE  1
```

В примере 2 использован оператор выхода транзактов из СМО:

```
TERMINATE  A
```

Согласно этому оператору из итогового счетчика вычитается число A. С помощью итогового счетчика задается длительность моделирования. В начале исполнения программы в счетчик заносится число, указанное в операнде A оператора

```
START      A,B,C
```

Моделирование прекращается, когда содержимое счетчика будет равно или меньше нуля. Операнд C — шаг вывода статистики на печать. Если B=0 и C=0, то выполняется только стандартная печать по окончании моделирования. В стандартную печать входят собранные за время моделирования статистические данные по основным параметрам модели: средние и максимальные значения длин очередей, объемов занимаемой памяти в накопителях, времени занятого состояния устройств и др. От печати можно отказаться, указав B=NP.

Пример 3

Общая структура программы на GPSS имеет вид

```
SIMULATE
<описания, в том числе функций, накопителей, массивов и т.п.>
<операторы, моделирующие движение транзактов>
START     A,B,C
END.
```

Оператор безусловного перехода записывается следующим образом:

```
TRANSFER   ,B
```

Здесь B — метка оператора, к которому следует переход.

Используется ряд других разновидностей оператора **TRANSFER**. Например:

```
TRANSFER   P,B,C
```

Переход происходит к оператору с меткой, равной сумме значения параметра B транзакта и числа C.

```
TRANSFER   FN,B,C
```

То же, но вместо параметра транзакта слагаемым является значение функции B.

```
TRANSFER   PICK,B,C
```

Это оператор равновероятного перехода к операторам, метки которых находятся в интервале [B,C].

Важное место в СМО занимает переход по вероятности:

```
TRANSFER   A,B,C
```

Здесь A — вероятность перехода к оператору с меткой C, переход к оператору с меткой B будет происходить с вероятностью 1 – A.

Оператор перехода в циклических процедурах выглядит так:

```
LOOP      A,B
```

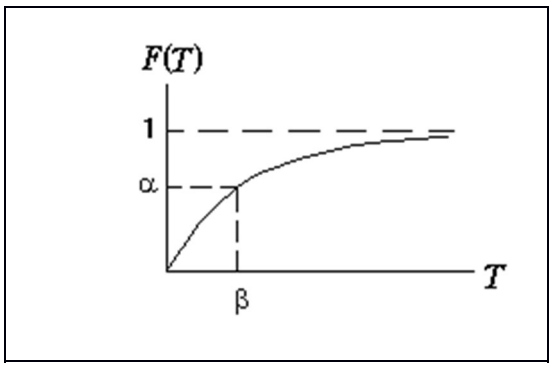
Здесь A — номер параметра транзакта, в котором содержится число повторений (витков) цикла, B — метка оператора, с которого начинается повторяющаяся часть.

Пример 4

Заказы, поступающие в СМО в случайные моменты времени в диапазоне [20,40], выполняет сначала бригада WGR1, затем параллельно работают бригады WGR2 и WGR3, каждая над своей частью заказа. Заданы экспоненциальные законы для времен выполнения работ бригадами WGR1, WGR2 и WGR3 с интенсивностями 0,05, 0,1 и 0,125 соответственно. Моделирование нужно выполнить на временном отрезке, соответствующем выполнению 1000 заказов.

Программа:

```
SIMULATE
EXP   FUNCTION   RN1,C12
0,0/.2,.22/.4,.51/.5,.6/.6,.92/.7,1.2/.8,1.61/.9,2.3/.95,3/.99,4.6/.999,6.9/1,10
00
GENERATE   30,10
SEIZE      WGR1
ADVANCE    20,FN$EXP
RELEASE    WGR1
SPLIT      1,MET1
SEIZE      WGR2
ADVANCE    10,FN$EXP
RELEASE    WGR2
TRANSFER   ,MET2
MET1  SEIZE      WGR3
ADVANCE    8,FN$EXP
RELEASE    WGR3
MET2  ASSEMBLE   2
TERMINATE  1
START      1000
END.
```



Функция экспоненциального закона распределения

В этом примере использован экспоненциальный закон распределения с плотностью

$$P(t) = \lambda \exp(-\lambda t),$$

где λ — интенсивность. Функция распределения экспоненциального закона

$$F(T) = \int_0^T p(t)\,dt = 1 - \exp(-\lambda T).$$

Задавая значение α, как равномерно распределенной в диапазоне $[0, 1]$ случайной величины, по формуле

$$\beta = \frac{1}{\lambda} \ln \frac{1}{1-\alpha} \qquad (1)$$

находим искомое значение. Именно в соответствии с (1) в операторах **ADVANCE** множителями были значения $\frac{1}{\lambda}$.

Приведем еще несколько операторов языка GPSS.

Оператор изменения параметров транзактов:

ASSIGN A,B

Здесь A — номер параметра транзакта, B — присваиваемое ему значение.

В следующих операторах параметр A увеличивается (уменьшается) на значение B:

ASSIGN A+,B

ASSIGN A-,B

Оператор фиксации времени события

```
MARK      A
```

его выполнение заключается в запоминании значения текущего модельного времени в параметре вошедшего в оператор MARK транзакта, в поле A указывается номер параметра.

Пример 5

На вход производственной линии поступают и проходят обработку на станке TOOL1 детали типов X и Y. Далее детали типа X обрабатываются на станке TOOL2, а детали типа Y — на станке TOOL3. Интервал моделирования соответствует обработке 600 деталей (ниже у операторов **GENERATE и ADVANCE** значения операндов не конкретизированы):

```
SIMULATE
GENERATE   A,B
ASSIGN     1,LBL4

TRANSFER   ,LBL1
GENERATE   A,B
ASSIGN     1,LBL2
LBL1  SEIZE     TOOL1
ADVANCE    A,B
RELEASE    TOOL1
TRANSFER   P,1
 LBL4  SEIZE     TOOL2
ADVANCE    A,B
RELEASE    TOOL2
LBL3  TERMINATE  1
LBL2  SEIZE     TOOL3
ADVANCE    A,B
RELEASE    TOOL3
TRANSFER   ,LBL3
START      600
```

```
END.
```

Расширение возможностей управления движением транзактов достигается благодаря операторам, реализующим механизм семафора:

```
LOGIC X    A
GATE XX    A,B
```

Первый оператор при X = S устанавливает переключатель A в единичное состояние, при X = R сбрасывает его в нулевое состояние, при X = I инвертирует значение состояния. Второй оператор при XX = LR и значнии переключателя, указанного в A, равном 1, или при XX = LS и состоянии переключателя 0 передает транзакт оператору с меткой B (или задерживает его в блоке **GATE**, если поле B пусто), а при других сочетаниях XX и значений переключателя — направляет к следующему оператору.

Вычислительный оператор присваивает переменной с номером M значение арифметического выражения A:

```
M    VARIABLE   A
```

Например, в следующем операторе переменной номер 3 присваивается разность числа 216 и объема занятой памяти в накопителе MEM2:

```
XINIT VARIABLE   K216-S$MEM2
```

Знаки арифметических операций сложения, вычитания, умножения, деления +, -, #, / соответственно. В случае логических выражений имя оператора должно быть BVARIABLE, а знаками операций дизъюнкции и конъюнкции являются + и #. Если операции выполняются над числами типа real, то имя оператора FVARIABLE.

Следующий оператор присваивает хранимой переменной, указанной в A. значение, записанное в B:

```
SAVEVALUE  A,B
```

Прерывание обслуживания заявки в устройстве A происходит при входе некоторой другой заявки в блок:

```
PREEMT     A
```

а возобновление прерванного обслуживания заявки — при входе в блок:

```
RETURN    A
```

Оператор синхронизации, реализующий механизм рандеву, имеет, например, вид:

```
LBL   MATCH     NUMB
```

Приходящий в него транзакт задерживается до тех пор, пока в некоторой другой части модели в сопряженный оператор не войдет транзакт того же семейства. Сопряженный оператор выглядит так:

```
NUMB  MATCH     LBL
```

Часто сведения о некоторых величинах, характеризующих моделируемый процесс, удобно представлять в виде гистограмм. Задание гистограммы выполняют в разделе описаний с помощью оператора:

```
M    TABLE     A,B,C,D
```

Здесь M — имя гистограммы; A — табулируемая величина; B — верхняя граница левого интервала гистограммы; C — ширина интервалов; D — число интервалов. Формирование гистограммы происходит с помощью оператора:

```
TABULATE   A
```

Выполнение этого оператора увеличивает на единицу число попаданий в i-й интервал гистограммы, имя которой указано в A. При этом i-й интервал соответствует текущему значению переменной, являющейся аргументом для гистограммы.

Среда *GPSS* World предназначена для имитационного моделирования систем с дискретными и непрерывными процессами. Языком моделирования в ней является язык *GPSS*, улучшенный встроенным языком программирования низкого уровня PLUS.

Язык *GPSS* построен в предположении, что модель сложной системы можно представить совокупностью элементов и логических правил их взаимодействия в процессе функционирования моделируемой системы. Набор абстрактных элементов, называемых объектами, небольшой. Также набор логических правил ограничен и может быть описан стандартными операциями. Комплекс программ, описывающих функционирование объектов

и выполняющих логические *операции*, является основой для создания программной модели.

РАЗРАБОТКА ИМИТАЦИОННЫХ МОДЕЛЕЙ УГРОЗ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

В основе проектирования подсистем ИБ автоматизированных систем лежит разработка модели угроз. Рассмотрим возможный подход к ее построению.

В соответствии с блочно-иерархическим подходом в большинстве АС введено описание структуры модели большинства объектов проектирования и собственно структуры АС, включающее пять уровней: архитектурный, функционально-логический, системотехнический, схемотехнический, физический.

Для каждого из данных уровней характерна своя степень детализации объекта проектирования и, соответственно, методов и средств проектирования. Функционал архитектурного уровня соответствует самой низкой степени детализации объекта проектирования. На данном уровне иерархической структуры рассматриваются модели топологий, правила и условия их построения.

Модели функционально-логического уровня строятся для решения задач согласования подсистем, входящих в состав объекта проектирования.

Системотехнический уровень соответствует степени детализации в приближении моделей «черный ящик» или «серый ящик». На схемотехническом уровне в модельном представлении объекта проектирования не учитывается физическая природа компонент объекта проектирования.

На схемотехническом уровне проводятся проектные работы, направленные на разработку документации, частично учитывающей физическую природу подсистем и компонент объекта проектирования.

На физическом уровне выполняются работы, определяемые ГОСТ, как рабочее проектирование (по существу – конструкторские работы).

- Анализ состояния проблемы автоматизированного проектирования каналов передачи защищенных сообщений показал, что необходимо применять блочно-иерархический подход к рассмотрению существующих в настоящее время методик проектирования. Решение поставленных в рамках данной работы задач позволит получить методику оценку защищенности сообщений в проектируемых АС от помех и атак.
- Анализ методик проектирования в соответствии с блочно-иерархическим подходом к описанию структуры каналов на архитектурном уровне позволил выделить топологии соединений, которые применимы при использовании рассматриваемых стандартов локальной беспроводной связи. Методика проектирования, являющаяся решением поставленных задач, позволяет синтезировать топологии децентрализованного типа.
- В результате сравнительного анализа результатов проектирования при помощи рассматриваемых методик на функционально-логическом уровне структуры каналов выявлена необходимость проведения имитационного моделирования.

ВЫБОР СРЕДСТВ МОДЕЛИРОВАНИЯ

Анализ российских и мировых исследований показывает, что, несмотря на огромное разнообразие различных инструментов для проведения

имитационного моделирования (например, типичная среда AnyLogic), получили преимущественное распространение имитационные модели и их реализация на GPSS. Моделирование на GPSS содержит большие потенциальные возможности для формализованного описания и имитационного моделирования *бортовой цифровой вычислительной системы* (БЦВС).

В структурной схеме бортовой цифровой вычислительной системы (БЦВС), представленной на рис. 1: МКИО - мультиплексный канал информационного обмена, БЛС – бортовая локальная сеть, МВ – модуль электропитания, преобразующий $U_{\text{бс}}$ – напряжение бортовой сети во $U_{\text{втор}}$ – вторичные напряжения, ПК - последовательный канал, РК – разовые команды, ФМ – функциональный модуль, МГ – графический модуль.

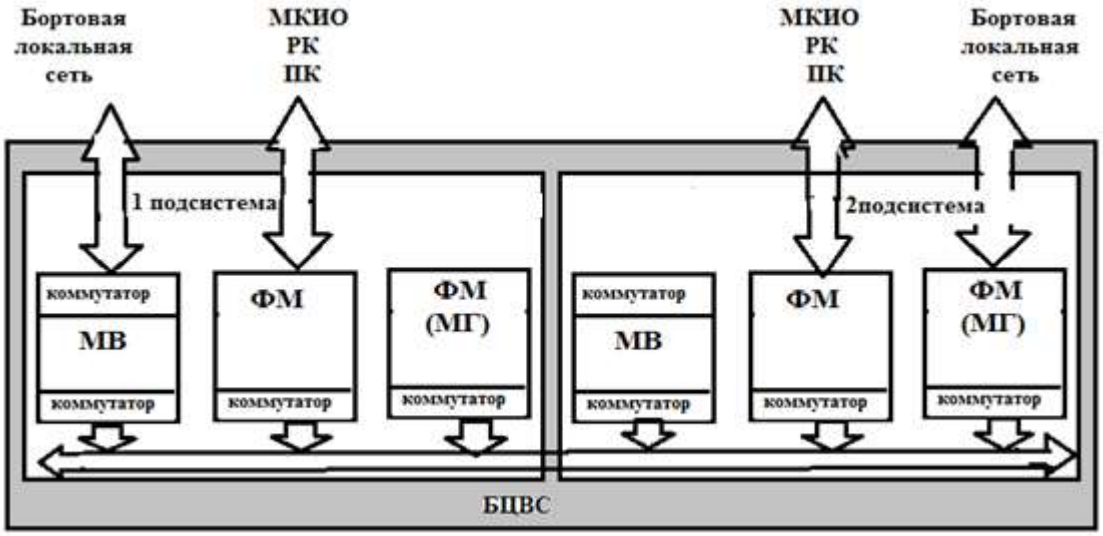


Рис.1. Структурная схема БЦВС

На основе структурной схемы должна быть разработана GPSS модель, которая позволит получить необходимые числовые характеристики.

ОПИСАНИЕ МОДЕЛИ

В модели учтено следующее:

- затраты по времени на направление запроса, тестирования и ожидания ответа от БЦВС;
- возможность распознания отрицательных результатов тестирования;
- иерархия, так как некоторые переходы не являются мгновенными и имеют вложенные цепи.

Описываемая модель является примером, подтверждающим работоспособность выбранной методики, поэтому время прохождения переходов устанавливается условно, сообразно соответствующим им задачам, а не снимаются с реальной БЦВС.

КОМПЛЕКСИРОВАНИЕ ЭЛЕМЕНТОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ - СОЗДАНИЕ СТЕНДА ИССЛЕДОВАНИЯ ИБ

Программа GPSS устанавливается сама после запуска инсталляционного файла, который в оригинале называется student.exe. Программа размещает себя в каталоге на диске C:\Program Files\Minuteman Software в каталоге, носящем название фирмы-разработчика программы Minuteman Software. В этом каталоге размещаются каталоги более низкого уровня: GPSS World Student Version → Sample Models.

В каталоге GPSS World Student Version находится сама программа GPSSW.exe, а также вспомогательные файлы.

Также программа может работать в автономном режиме, без установки на локальную машину. Для этого необходимо наличие flash-накопителя с имеющимися на нём файлами GPSS, организованными в такую же иерархическую структуру, как и при установке на локальный диск.

В рамках исследования информационной безопасности бортовой цифровой вычислительной системы летательного аппарата (БЦВС ЛА) потребуется моделирование различных ситуаций, возникающих в системе во время полёта, в частности, моделирование отказа СЗИ.

Также будет рассмотрена модель работы универсального СЗИ, анализирующего угрозы разных типов для демонстрации механизма его работы.

МОДЕЛИРОВАНИЕ ОТКАЗА БЛОКА СЗИ И ЕГО ЗАМЕНЫ

Здесь представлена модель процессов возникновения и устранения неисправностей в некоторой технической системе, состоящей из множества однотипных блоков; в запасе имеется один исправный блок; известны статистические данные об интенсивностях возникновения отказов и длительностях таких операций, как поиск неисправностей, замена и ремонт отказавшего блока. Поиск и замену отказавшего блока производит бригада TEAM1, а ремонт замененного блока — бригада TEAM2.

```
SIMULATE
GENERATE   A,B   моделируется возникновение отказов
SEIZE      TEAM1
ADVANCE    A,B   поиск неисправности
ENTER      MEM,1 получение запасного блока из резерва
ADVANCE    A,B   замена блока
RELEASE    TEAM1
SEIZE      TEAM2
ADVANCE    A,B   ремонт
LEAVE      MEM,1 восстановление резерва
RELEASE    TEAM2
TERMINATE  1
START      1000
END.
```

МОДЕЛИРОВАНИЕ РАБОТЫ УНИВЕРСАЛЬНОГО БЛОКА СЗИ

Требуется разработать модель блока СЗИ, обрабатывающего, к примеру, 30 угроз А1 и 16 угроз типа А2, поступающих от независимых экспоненциальных источников с интенсивностями λ, равными 0,1 и 0,04 сек соответственно. Длительностью работы по анализу и устранению находится в пределах [12,18] сек. Промоделируем обработку 600 подобных блоков угроз.

```
SIMULATE
MEM1  STORAGE     30
MEM2  STORAGE     16
TAB   TABLE       MEM2,32,16,6
EXP   FUNCTION    RN1,C12
0,0/.2,.22/.4,.51/.5,.6/.6,.92/.7,1.2/.8,1.61/.9,2.3/.95,3/.99,4.6/.999,6.9/1,10
00
GENERATE    10, FN$EXP
ENTER       MEM1,1
TRANSFER    ,MMM
GENERATE    25, FN$EXP
ENTER       MEM2,1
MMM   TEST GE     S$MEM1,30,LLL
TEST GE     S$MEM2,16,LLL
TABULATE    TAB
SEIZE       MONT
ADVANCE     15,3
RELEASE     MONT
TERMINATE   1
LLL   TERMINATE
START       600
END.
```

На этих примерах можно увидеть, что выбранное ПО подходит для моделирования множества ситуаций, характеризующих работу бортовой системы информационной безопасности.

GPSS позволяет получать множество отчётов по моделированию, что позволить оценить работу всех подсистем и блоков, исправляя недостатки и проверяя разработанную систему на соответствие функциональным требованиям.

ВЫПОЛНЕНИЕ МОДЕЛИРОВАНИЯ

Для описания явлений распространения деструктивных воздействий на автоматизированные системы бортовых компьютерных сетей предлагается использовать модель распространения разрушающего программного кода в автоматизированных системах. Разрушающий программный код (РПК) представляет собой машинную реализацию разрушающего программного воздействия. РПК может поступить в среду бортовых ВС по каналам радиосвязи и навигации.

Разрушающему программному коду присущи числовые параметры, часть из которых зависит от свойств РПК, а часть может быть задана атакующим. Наиболее важным параметром является средняя вероятность успеха атаки на компьютерное оборудование. Другой немаловажный параметр – это вероятность использования захваченных процессоров для дальнейших атак, т.е. для формирования т.н. «лавинной атаки». Если атака с помощью РПК оказалась успешной, необходимо знать какое количество атак может быть спровоцировано на следующем заданном отрезке времени. Дополнительный параметр – время активации деструктивных функций РПК. Понятно, что если оно будет слишком мало, то ограничит вероятность использования захваченных ресурсов для дальнейших атак. С другой стороны, если этот период будет очень большим, РПК подвергнется большему риску обнаружения и обезвреживания и с меньшей вероятностью сможет осуществить свою функциональность.

Согласно международным стандартом, эффективность защиты информации определяется классом защищённости автоматизированной системы (АС). Класс защищённости, в свою очередь, определяет набор механизмов защиты (МЗ), которые должны быть реализованы в АС. Такой подход к оценке эффективности защиты информации не позволяет учитывать качество самих МЗ, констатируя лишь факт их наличия или отсутствия, также вне критериев оценки остаётся такое понятие, как изменение условий функционирования СЗИ. Примерами таких изменений могут служить модификация аппаратной и программной среды, изменение условий информационного взаимодействия объектов и субъектов защиты, числа пользователей системы, возникновение информационных конфликтов в АС.

Существуют методы, позволяющие выполнять количественную оценку защищённости информации при использовании СЗИ. Количественно защищённость информации оценивается, как правило, рядом вероятностных показателей, основной из которых — некий интегральный показатель.

В общем случае СЗИ представляется в виде сетевой модели или сети *массового обслуживания* (СМО), рис. 2, состоящей из некоторого набора средств защиты S_i. На вход средств защиты поступают потоки запросов НСД, определяемые моделью нарушителя на множестве потенциальных угроз $\{U_i\}$. Каждое из средств защиты отвечает за защиту от угрозы определённого типа и использует соответствующий защитный механизм. Его задача состоит в том, чтобы распознать угрозу и заблокировать несанкционированный запрос.

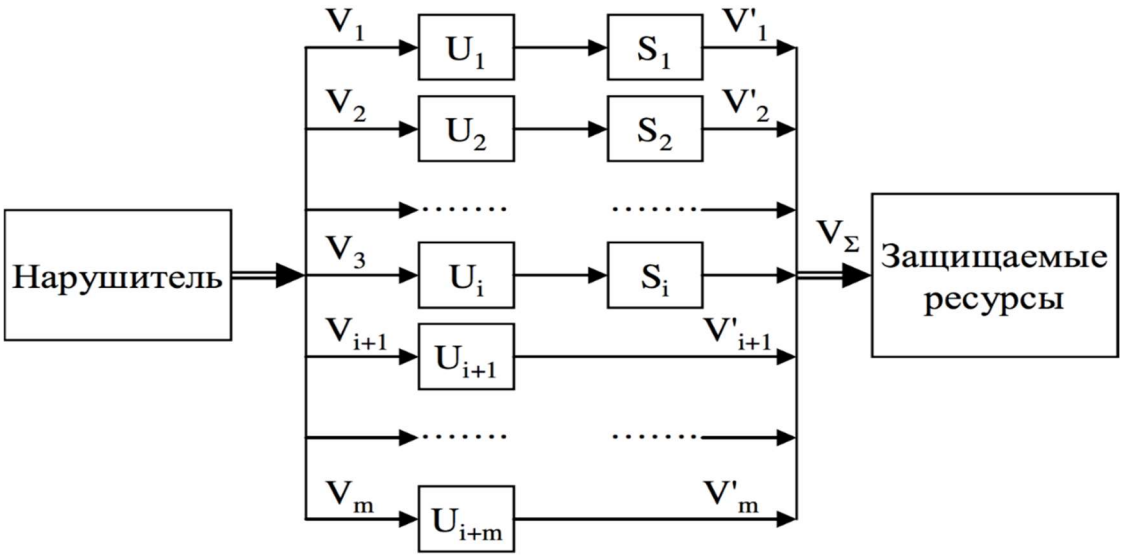


Рис. 2. Концептуальная модель СЗИ от НСД

В результате функционирования системы защиты исходный поток НСД разрежается, образуя выходной поток. Входные потоки несанкционированных запросов обозначены как $V_i(t)$, i = {1, ..., n}, а потоки нераспознанных (пропущенных) системой защиты НСД — V_i'. Факт неполного закрытия системой защиты всех возможных каналов проявления угроз учитывается отсутствием для m входных потоков средств защиты, что означает $V_i'(t) = V_i(t)$. Потоки запросов на НСД, поступающие по i-м каналам, разрежаются с вероятностями $p_i(y)$, которые зависят от используемого способа обнаружения и блокирования НСД.

На выходе СЗИ образуется выходной поток — объединение выходных потоков i-средств защиты и потока НСД-запросов, приходящих по m неконтролируемым каналам.

Каждое средство (механизм) защиты характеризуется вероятностью пропуска НСД — q и, соответственно, вероятностью обеспечения защиты (отражения НСД) p = 1-q.

Нарушитель характеризуется вектором интенсивностей $\lambda = \{\lambda_1, \lambda_2, ... \lambda_{i+m}\}$ попыток реализации соответствующих угроз $U_1 \ldots U_{i+m}$.

Для реализации системного подхода к решению проблемы обеспечения информационной безопасности необходимо комплексное использование методов моделирования систем и процессов защиты информации. Цели такого моделирования: поиск оптимальных решений управления МЗ, оценки эффективности использования средств и методов защиты и т.п.

Модель представляет логическое или математическое описание компонентов и функций, отображающих существенные свойства моделируемого объекта или процесса.

Моделирование системы заключается в построении некоторого её образца, адекватного с точностью до целей моделирования исследуемой системы, и получения с помощью построенной модели необходимых характеристик реальной системы.

Для реализации комплексного подхода к моделированию СЗИ необходимо построить имитационную модель атак на устройство и противодействия этим атакам.

Для описания моделей СМО разработаны специальные языки и системы имитационного моделирования для ЭВМ. Существуют общецелевые языки, ориентированные на описание широкого класса СМО в различных предметных областях, и специализированные языки, предназначенные для анализа систем определенного типа. Примером общецелевых языков служит широко распространённый язык GPSS (General Purpose Simulation System).

Для дальнейшей работы остановим выбор на языке GPSS и среде GPSS World, основываясь на их распространенности и положительном опыте работы с ними.

Представим модель СЗИ, показанную на рисунке 2 в виде функциональных блоков, объединенных в три группы, соответствующие трем основным объектам моделируемой системы: «Нарушитель», «СЗИ» и «Защищаемые ресурсы». Модель показана на рисунке 3.

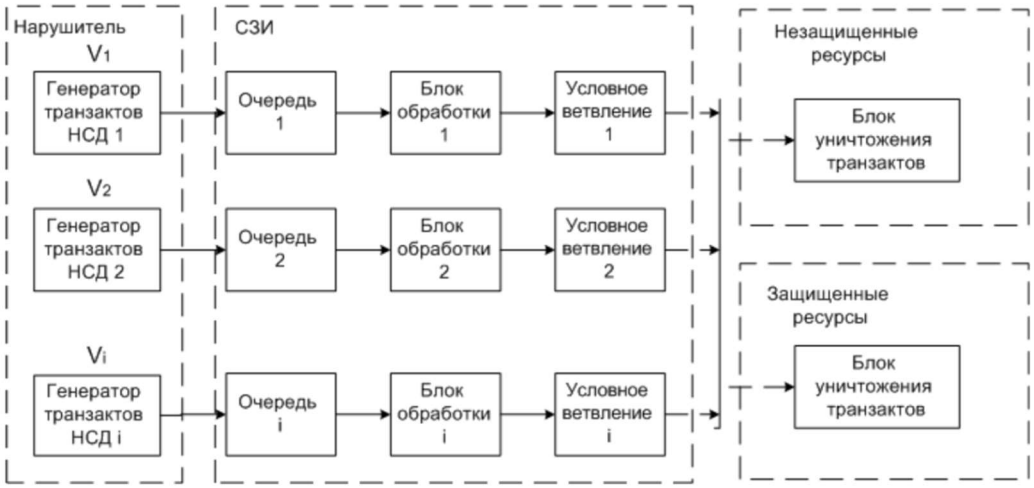


Рис. 3. Имитационная модель СЗИ от НСД

«Нарушитель» — это первый блок модели, в общем случае он не подвергается входному воздействию. Задача функционирования этого блока — генерация потока (потоков) запросов НСД (транзактов) с заданной интенсивностью λ. Согласно модели нарушителя, разработанной ранее, злоумышленник пытается реализовывать разные угрозы защищённости информации с соответствующими интенсивностями.

Блок «СЗИ» имитирует функционирование СЗИ от НСД (МЗ). Элементы этого блока могут имитировать очереди запросов НСД на входах МЗ, задержки на обслуживание, выход МЗ из строя (аппаратной части) и т.д. Однако главная задача функционирования этого блока — отсеивание запросов НСД с определённой (заданной) вероятностью. Разреженный поток запросов НСД на выходе блока «СЗИ» имеет интенсивность λ'.

Последний блок модели — «Защищаемые ресурсы» — не выполняет самостоятельных функций и может быть использован в имитационной модели для уничтожения запросов НСД (транзактов). Представим функции блоков модели СЗИ (табл. 1).

Табл. 1. Функции логических блоков имитационной модели

№ блока	Название блока	Функции блока
1	Нарушитель	Генерация запросов НСД с заданными интенсивностями, которые образуют входной поток блока «СЗИ».
2	СЗИ	1. Имитация буфера (очереди) запросов НСД. 2. Имитация обслуживания запросов НСД МЗ. 3. Разреживание входных и образование выходных потоков пропущенных и отсеянных запросов НСД.
3	Защищаемые ресурсы	Уничтожение запросов НСД (как отсеянных, так и пропущенных МЗ СЗИ).

Таким образом, для построения имитационной модели СЗИ от НСД представляется целесообразным использование следующих функциональных блоков:

- генератора транзактов — для имитации поступления запросов НСД;
- блока задержки — для имитации обработки МЗ поступающих запросов НСД;
- очереди — для имитации буфера запросов каждого из МЗ;
- блоков уничтожения транзактов — для уничтожения запросов НСД (как пропущенных, так и отсеянных МЗ).

Ниже представлена простая имитационная модель, иллюстрирующая процесс работы одного СЗИ, созданная при помощи языка GPSS:

```
N_  EQU             2

    generate        20
Mark1               test l    Q$Queue1,N_,Fail1
    queue           Queue1
    seize           Channel1
    depart          Queue1
    advance         20,1
    release         Channel1
    transfer ,Fail1

Fail1               terminate 1

    start 1000
```

В данной модели поток атак отражается полностью, при этом загрузка СЗИ составляет 100% от его возможностей. Это отражают следующие статистические показатели моделирования:

```
FACILITY          ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 CHANNEL1           1001    0.999      19.973  1     1001    0    0     0      0


QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 QUEUE1              1    1   1001     23     0.242      4.849      4.963   0
```

Представляя себе возможности групп атакующего, можно подобрать СЗИ таким образом, чтобы оно не расходовало лишние ресурсы. С другой стороны, всегда необходим запас по мощности СЗИ, чтобы оно могло справиться с внезапной атакой. Следующая модель представляет собой случай, когда СЗИ тратит на предотвращение угроз неполную мощность:

```
N_  EQU             2

    generate        30
Mark1               test l    Q$Queue1,N_,Fail1
    queue           Queue1
    seize           Channel1
    depart          Queue1
    advance         20,1
    release         Channel1
    transfer ,Fail1

Fail1               terminate 1

    start 1000
```

Уменьшившийся поток угроз приводит не только к высвобождению ресурсов, но и к уменьшению задержек в обслуживании:

```
FACILITY         ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 CHANNEL1          1000    0.666      19.993  1        0    0    0     0      0


QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 QUEUE1              1    0   1000   1000     0.000      0.000      0.000   0
```

В отличие от прошлой модели, на этот раз инциденты не формируют очередь. В ОС РВ, где задержки в ряде систем критичны для работы, следует подбирать СЗИ таким образом, чтобы в случае возникновения внештатной ситуации задержки не влияли на работу системы. Для этого можно либо закладывать дополнительные мощности в СЗИ, либо использовать дублирование.

```
N_  EQU             2

    generate        20
Mark1               test l    Q$Queue1,N_,Fail1
    queue           Queue1
    seize           Channel1
    depart          Queue1
    advance         30,1
    release         Channel1
    transfer ,Fail2

Fail1               test l    Q$Queue2,N_,Fail2
    queue           Queue2
    seize           Channel2
    depart          Queue2
    advance         20,1
    release         Channel2

Fail2               terminate 1

  start 1000
```

В данной модели используются два СЗИ – основной и резервный. Если основной СЗИ уже занят обработкой инцидента – то включается резервный СЗИ. Загрузка в приведённом примере выглядит следующим образом:

```
FACILITY          ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 CHANNEL1           669    0.999      29.964  1     1001    0    0     0      1
 CHANNEL2           333    0.331      19.918  1     1003    0    0     0      0


QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 QUEUE1              2    2    670      1     1.678     50.249     50.324   0
 QUEUE2              1    0    333    333     0.000      0.000      0.000   0
```

Как можно заметить, резервный СЗИ две трети времени простаивал, но в результате весь поток угроз был обработан СЗИ. В случае отсутствия резервного СЗИ часть угроз, находящаяся в очереди на обработку, могла бы быть потеряна, что привело бы к их реализации. Или же место, занимаемое входящими инцидентами, тратило ресурсы ОС РВ, приводя к проблемам в других частях системы.

Следует заметить, что, несмотря на приведённое выше общее описание системы, в данных моделях не рассматриваются угрозы, не покрываемые СЗИ: в реалиях бортовой ОС РВ оставлять какие-либо каналы незащищёнными нельзя. Необходимость такого подхода обоснована приведёнными в другом разделе требованиями безопасности. Также, согласно им, в данных моделях считается, что разные СЗИ полностью независимы друг от друга, поскольку в ином случае компрометация одного СЗИ означала бы уязвимость во всём классе СЗИ.

Можно использовать одно и то же СЗИ для обработки нескольких типов угроз, если это позволяют его мощности. Для одного типа угроз он будет основным, для другого – резервным. Модель в таком случае будет выглядеть следующим образом:

```
        generate        20
Mark1                   test le   Q$Queue1,1,
        queue           Queue1
        seize           Channel1
        depart          Queue1
        advance         30,1
        release         Channel1
        transfer ,Fail2

        generate        40
Mark2                   test le   Q$Queue2,1,Fail2 ;
        queue           Queue2
```

```
        seize           Channel2
        depart          Queue2
        advance         20,1
        release         Channel2
        transfer ,Fail2

Fail1                   test le   Q$Queue2,1,Fail2
        queue           Queue2
        seize           Channel2
        depart          Queue2
        advance         20,1
        release         Channel2

Fail2                   terminate 1

        generate 10000000
        terminate 1
    start 1000
```

Для понимания принципа работы программы на этот раз помимо статистики работы СЗИ стоит ещё вывести число обращений к блокам модели:

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	669	0	0
MARK1	2	TEST	669	0	0
	3	QUEUE	448	2	0
	4	SEIZE	446	0	0
	5	DEPART	446	0	0
	6	ADVANCE	446	1	0
	7	RELEASE	445	0	0
	8	TRANSFER	445	0	0
	9	GENERATE	334	0	0
MARK2	10	TEST	334	0	0
	11	QUEUE	334	0	0
	12	SEIZE	334	0	0
	13	DEPART	334	0	0
	14	ADVANCE	334	0	0

	15	RELEASE	334	0	0
	16	TRANSFER	334	0	0
FAIL1	17	TEST	221	0	0
	18	QUEUE	221	0	0
	19	SEIZE	221	0	0
	20	DEPART	221	0	0
	21	ADVANCE	221	0	0
	22	RELEASE	221	0	0
FAIL2	23	TERMINATE	1000	0	0
	24	GENERATE	0	0	0
	25	TERMINATE	0	0	0

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
CHANNEL1	446	0.999	29.997	1	1000	0	0	0	2
CHANNEL2	555	0.828	19.993	1	0	0	0	0	0

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(-0)	RETRY
QUEUE1	2	2	448	1	1.644	49.154	49.264	0
QUEUE2	1	0	555	250	0.180	4.349	7.913	0

Как можно заметить, на втором СЗИ (Mark2) обрабатывалась угроза, возникающая гораздо реже, чем та, которой занималось первое СЗИ (Mark1). За счёт этого поток необработанных угроз с первого СЗИ не помешал работе второго СЗИ, просто повысив его загрузку.

Однако же в данной ситуации второе СЗИ является уязвимым местом системы: при его выходе из строя система окажется уязвимой не только перед угрозами, которые обрабатывались на нём, но и перед угрозами, за которые отвечает первое СЗИ. Для предотвращения таких ситуаций можно использовать полное дублирование, как показано в следующей модели:

```
        generate        10
Mark1                   test le   Q$Queue1,1,Fail1 ;
        queue           Queue1
        seize           Channel1
        depart          Queue1
        advance         16,1
        release         Channel1
        transfer ,Fail2
```

```
Fail1           test le   Q$Queue2,1,Fail2
    queue       Queue2
    seize       Channel2
    depart      Queue2
    advance     15,1
    release     Channel2

Fail2           terminate 1

    generate    10
Mark2           test le   Q$Queue3,1,Fail3 ;
    queue       Queue3
    seize       Channel3
    depart      Queue3
    advance     8,1
    release     Channel3
    transfer ,Fail3

Fail3           terminate 1

    generate    10
Mark3           test le   Q$Queue4,1,Fail4 ;
    queue       Queue4
    seize       Channel4
    depart      Queue4
    advance     20,1
    release     Channel4
    transfer ,Fail5

Fail4           test le   Q$Queue5,1,Fail5
    queue       Queue5
    seize       Channel5
    depart      Queue5
```

```
        advance         12,1
        release         Channel5

Fail5                   terminate 1

        generate        10
Mark4                   test le   Q$Queue6,1,Fail6 ;
        queue           Queue6
        seize           Channel6
        depart          Queue6
        advance         12,1
        release         Channel6
        transfer ,Fail7

Fail6                   test le   Q$Queue7,1,Fail7
        queue           Queue7
        seize           Channel7
        depart          Queue7
        advance         16,1
        release         Channel7

Fail7                   terminate 1

        start 4000
```

Здесь используются целых семь СЗИ для защиты от четырёх типов угроз: три пары СЗИ работают в связке основной-резервный, и одно СЗИ целиком закрывает свой тип угроз. Полное распределение нагрузки выглядит следующим образом:

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	1002	0	0
MARK1	2	TEST	1002	0	0
	3	QUEUE	628	2	0
	4	SEIZE	626	0	0

	5	DEPART	626	0	0
	6	ADVANCE	626	1	0
	7	RELEASE	625	0	0
	8	TRANSFER	625	0	0
FAIL1	9	TEST	374	0	0
	10	QUEUE	374	0	0
	11	SEIZE	374	0	0
	12	DEPART	374	0	0
	13	ADVANCE	374	0	0
	14	RELEASE	374	0	0
FAIL2	15	TERMINATE	999	0	0
	16	GENERATE	1002	0	0
MARK2	17	TEST	1002	0	0
	18	QUEUE	1002	0	0
	19	SEIZE	1002	0	0
	20	DEPART	1002	0	0
	21	ADVANCE	1002	0	0
	22	RELEASE	1002	0	0
	23	TRANSFER	1002	0	0
FAIL3	24	TERMINATE	1002	0	0
	25	GENERATE	1002	0	0
MARK3	26	TEST	1002	0	0
	27	QUEUE	503	1	0
	28	SEIZE	502	0	0
	29	DEPART	502	0	0
	30	ADVANCE	502	1	0
	31	RELEASE	501	0	0
	32	TRANSFER	501	0	0
FAIL4	33	TEST	499	0	0
	34	QUEUE	499	0	0
	35	SEIZE	499	0	0
	36	DEPART	499	0	0
	37	ADVANCE	499	1	0
	38	RELEASE	498	0	0
FAIL5	39	TERMINATE	999	0	0
	40	GENERATE	1002	0	0
MARK4	41	TEST	1002	0	0
	42	QUEUE	837	1	0
	43	SEIZE	836	0	0
	44	DEPART	836	0	0
	45	ADVANCE	836	1	0
	46	RELEASE	835	0	0
	47	TRANSFER	835	0	0
FAIL6	48	TEST	165	0	0
	49	QUEUE	165	0	0
	50	SEIZE	165	0	0

	51	DEPART	165	0	0
	52	ADVANCE	165	0	0
	53	RELEASE	165	0	0
FAIL7	54	TERMINATE	1000	0	0

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
CHANNEL4	502	0.999	19.958	1	3994	0	0	0	1
CHANNEL6	836	0.999	11.984	1	4003	0	0	0	1
CHANNEL1	626	0.999	16.004	1	3996	0	0	0	2
CHANNEL3	1002	0.800	8.011	1	0	0	0	0	0
CHANNEL5	499	0.596	11.980	1	4008	0	0	0	0
CHANNEL2	374	0.558	14.953	1	0	0	0	0	0
CHANNEL7	165	0.262	15.932	1	0	0	0	0	0

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(-0)	RETRY
QUEUE4	2	1	503	1	1.816	36.202	36.274	0
QUEUE6	2	1	837	1	1.572	18.834	18.857	0
QUEUE1	2	2	628	1	1.686	26.931	26.974	0
QUEUE3	1	0	1002	1002	0.000	0.000	0.000	0
QUEUE5	1	0	499	495	0.001	0.014	1.706	0
QUEUE2	1	0	374	374	0.000	0.000	0.000	0
QUEUE7	1	0	165	165	0.000	0.000	0.000	0

Для наглядности изобразим полученные результаты в виде графика процента отражённых атак и загрузки:

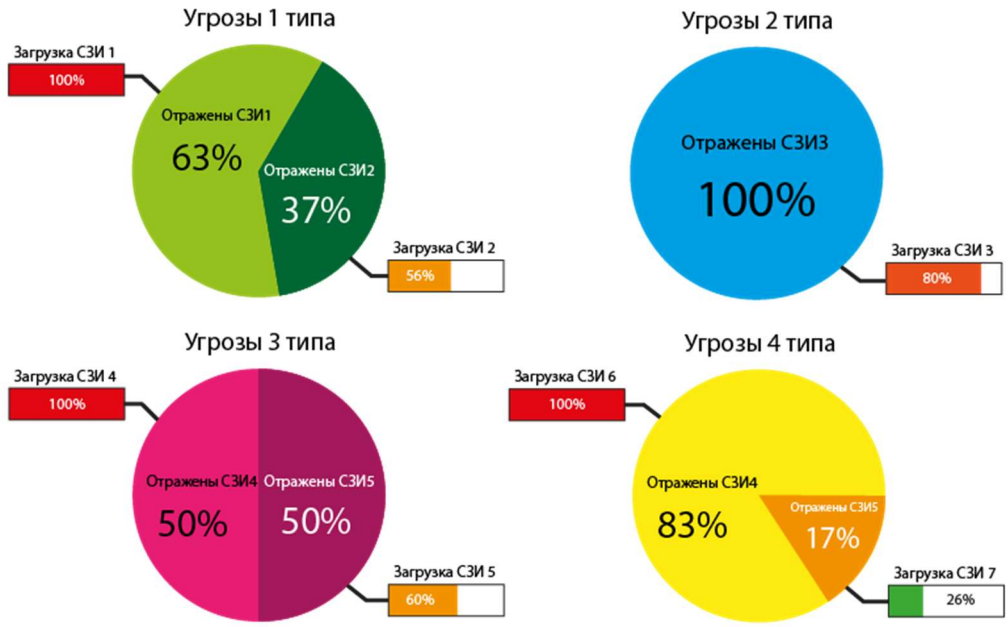


Рис. 4. Иллюстрация работы модели обеспечения информационной безопасности

Как можно заметить, часть СЗИ простаивает значительное время – как основных, так и резервных. Таким образом, реализация данной модели, несмотря на большую надёжность, потребует использования гораздо большего числа ресурсов.

ЗАКЛЮЧЕНИЕ

В целом, результаты моделирования позволяют подтвердить правомерность требований безопасности высокого уровня, а также позволяют оценить разные варианты построения системы ИБ бортового оборудования, позволяя комбинировать разные варианты использования СЗИ, исходя из известных данных об источниках угроз, имеющихся в распоряжении мощностей и топологии сети.

ЗАДАНИЕ:

1. Создать схему, имитирующую работу трёх СЗИ для защиты от двух независимых типов атак. Для 1го типа атак СЗИ1 – основной, СЗИ2 –

резервный. Для 2го типа атак СЗИ3 – основной, СЗИ2 – резервный. Для 1го типа атак generate=20, для второго типа атак generate=15. СЗИ1 имеет advance=25, СЗИ2 имеет advance=20, СЗИ3 имеет advance=20. Показать загрузку очередей, процент отражённых и прошедших атак каждого типа и по каждому СЗИ. Количество атак – 10000.

2. Создать схему, имитирующую работу трёх СЗИ для защиты от трёх независимых типов атак. Для 1го типа атак СЗИ1 – основной, СЗИ2 – резервный. Для 2го типа атак СЗИ2 – основной, СЗИ3 – резервный. Для 3го типа атак СЗИ3 – основной, СЗИ1 – резервный. Для 1го типа атак generate=30, для второго типа атак generate=25, для третьего типа атак generate=20. СЗИ1 имеет advance=25, СЗИ2 имеет advance=20, СЗИ3 имеет advance=25. Показать загрузку очередей, процент отражённых и прошедших атак каждого типа и по каждому СЗИ. Количество атак – 10000.
3. Создать схему, имитирующую работу трёх СЗИ для защиты от двух независимых типов атак. Для 1го типа атак СЗИ1 – основной, СЗИ2 – резервный. Для 2го типа атак СЗИ2 – основной, СЗИ3 – резервный. Для 1го типа атак generate=20, для второго типа атак generate=25. СЗИ1 имеет advance=30, СЗИ2 имеет advance=15, СЗИ3 имеет advance=25. Показать загрузку очередей, процент отражённых и прошедших атак каждого типа и по каждому СЗИ. Количество атак – 10000.
4. Создать схему, имитирующую работу трёх СЗИ для защиты от трёх независимых типов атак. Для 1го типа атак СЗИ1 – основной, СЗИ2 – резервный. Для 2го типа атак СЗИ2 – основной, СЗИ3 – резервный. Для 3го типа атак СЗИ3 – основной. Для 1го типа атак generate=30, для второго типа атак generate=40, для третьего типа атак generate=20. СЗИ1 имеет advance=35, СЗИ2 имеет advance=30, СЗИ3 имеет advance=20. Показать загрузку очередей, процент отражённых и

прошедших атак каждого типа и по каждому СЗИ. Количество атак – 10000.

5. Создать схему, имитирующую работу четырёх СЗИ для защиты от трёх независимых типов атак. Для 1го типа атак СЗИ1 – основной, СЗИ4 – резервный. Для 2го типа атак СЗИ2 – основной, СЗИ4 – резервный. Для 3го типа атак СЗИ3 – основной, СЗИ4 – резервный. Для 1го типа атак generate=30, для второго типа атак generate=25, для третьего типа атак generate=20. СЗИ1 имеет advance=40, СЗИ2 имеет advance=30, СЗИ3 имеет advance=20, СЗИ4 имеет advance=35. Показать загрузку очередей, процент отражённых и прошедших атак каждого типа и по каждому СЗИ. Количество атак – 10000.
6. Создать схему, имитирующую работу четырёх СЗИ для защиты от трёх независимых типов атак. Для 1го типа атак СЗИ1 – основной, СЗИ2 – резервный. Для 2го типа атак СЗИ2 – основной, СЗИ4 – резервный. Для 3го типа атак СЗИ3 – основной, СЗИ4 – резервный. Для 1го типа атак generate=30, для второго типа атак generate=25, для третьего типа атак generate=20. СЗИ1 имеет advance=35, СЗИ2 имеет advance=20, СЗИ3 имеет advance=25, СЗИ4 имеет advance=25. Показать загрузку очередей, процент отражённых и прошедших атак каждого типа и по каждому СЗИ. Количество атак – 10000.
7. Создать схему, имитирующую работу четырёх СЗИ для защиты от трёх независимых типов атак. Для 1го типа атак СЗИ1 – основной, СЗИ3 – резервный. Для 2го типа атак СЗИ2 – основной, СЗИ3 – резервный. Для 3го типа атак СЗИ3 – основной, СЗИ4 – резервный. Для 1го типа атак generate=25, для второго типа атак generate=30, для третьего типа атак generate=30. СЗИ1 имеет advance=35, СЗИ2 имеет advance=35, СЗИ3 имеет advance=25, СЗИ4 имеет advance=20. Показать загрузку очередей, процент отражённых и прошедших атак каждого типа и по каждому СЗИ. Количество атак – 10000.

8. Создать схему, имитирующую работу четырёх СЗИ для защиты от трёх независимых типов атак. Для 1го типа атак СЗИ1 – основной, СЗИ3 – резервный. Для 2го типа атак СЗИ2 – основной, СЗИ4 – резервный. Для 3го типа атак СЗИ3 – основной. Для СЗИ4 – СЗИ3 является резервным. Для 1го типа атак generate=20, для второго типа атак generate=25, для третьего типа атак generate=30. СЗИ1 имеет advance=25, СЗИ2 имеет advance=30, СЗИ3 имеет advance=20, СЗИ4 имеет advance=30. Показать загрузку очередей, процент отражённых и прошедших атак каждого типа и по каждому СЗИ. Количество атак – 10000.
9. Создать схему, имитирующую работу четырёх СЗИ для защиты от трёх независимых типов атак. Для 1го типа атак СЗИ1 – основной, СЗИ2 – резервный. Для 2го типа атак СЗИ2 – основной, СЗИ3 – резервный. Для 3го типа атак СЗИ3 – основной, СЗИ4 – резервный. Для 1го типа атак generate=25, для второго типа атак generate=25, для третьего типа атак generate=30. СЗИ1 имеет advance=30, СЗИ2 имеет advance=35, СЗИ3 имеет advance=25, СЗИ4 имеет advance=30. Показать загрузку очередей, процент отражённых и прошедших атак каждого типа и по каждому СЗИ. Количество атак – 10000.
10. Создать схему, имитирующую работу четырёх СЗИ для защиты от трёх независимых типов атак. Для 1го типа атак СЗИ1 – основной, СЗИ4 – резервный. Для 2го типа атак СЗИ2 – основной, СЗИ4 – резервный. Для 3го типа атак СЗИ3 – основной. Для СЗИ4 – СЗИ3 является резервным. Для 1го типа атак generate=20, для второго типа атак generate=20, для третьего типа атак generate=30. СЗИ1 имеет advance=30, СЗИ2 имеет advance=25, СЗИ3 имеет advance=25, СЗИ4 имеет advance=40. Показать загрузку очередей, процент отражённых и прошедших атак каждого типа и по каждому СЗИ. Количество атак – 10000.

Результаты моделирования отправить на электронную почту

Почта Медведева Н.В.

nikmed4591@gmail.com

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Имитационное моделирование на GPSS : учеб.-метод. пособие для студентов технических специальностей / Д. Н. Шевченко, И. Н. Кравченя ; М-во образования Респ. Беларусь, Белорус. гос. ун-т трансп. – Гомель : БелГУТ, 2007. – 97 с.
2. Бронов, С. А. Имитационное моделирование : учеб. пособие / С. А. Бронов; ФГОУ ВПО "Сибирский федеральный университет", кафедра "Системы автоматизированного проектирования". — Красноярск: СФУ, 2007. — 82 с.
3. Боев, В. Д. Моделирование систем. Инструментальные средства GPSS World : учеб. пособие / В. Д. Боев. — СПб.: БХВ-Петербург, 2007. — 368 с.
4. Кудрявцев, Е. М. GPSS World. Основы имитационного моделирования различных систем / Е. М. Кудрявцев. — М.: ДМК Пресс, 2008. — 320 с. — (Серия "Проектирование").
5. Шрайбер, Т. Дж. Моделирование на GPSS / Т. Дж. Шрайбер. — М.: Машиностроение, 1979. — 592 с.
6. Советов, Б. Я. Моделирование систем. Практикум / Б. Я. Советов, С. А. Яковлев. — М.: Высшая школа, 1999. — 224 с.