REQUEST FOR COMMENTS 2795 · INFORMATIONAL · 1 APRIL 2000

The Infinite Monkey *Protocol Suite (IMPS)*

S. Christey

Network Working Group · MonkeySeeDoo, Inc.

Abstract. This memo describes a protocol suite which supports an infinite number of monkeys that sit at an infinite number of typewriters in order to determine when they have either produced the entire works of William Shakespeare or a good television show. The suite includes communications and control protocols for monkeys and the organizations that interact with them.

1. Introduction

It has been posited that if an infinite number of monkeys sit at an infinite number of typewriters and randomly press keys, they will eventually produce the complete works of Shakespeare [1] [2]. But if such a feat is accomplished, how would anybody be able to know? And what if the monkey has flawlessly translated Shakespeare's works into Esperanto? How could one build a system that obtains these works while addressing the basic needs of monkeys, such as sleep and food? Nobody has addressed the practical implications of these important questions [3].

In addition, it would be a waste of resources if such a sizable effort only focused on Shakespeare. With an infinite number of monkeys at work, it is also equally likely that a monkey could produce a document that describes how to end world poverty, cure disease, or most importantly, write a good situation comedy for television [4]. Such an environment would be ripe for innovation and, with the proper technical design, could be effectively utilized to "make the world a whole lot brighter" [5].

The Infinite Monkey Protocol Suite (IMPS) is an experimental set of protocols that specifies how monkey transcripts may be collected, transferred, and reviewed for either historical accuracy (in the case of Shakespearean works) or innovation (in the case of new works). It also provides a basic communications framework for performing normal monkey maintenance.

2. Objects in the Suite

There are four primary entities that communicate within an IMPS network. Groups of monkeys are physically located in Zone Operations Organizations (ZOOs). The ZOOs maintain the monkeys and their equipment, obtain transcripts from the monkeys' typewriters, and interact with other entities who evaluate the transcripts.

A SIMIAN (Semi-Integrated, Monkey-Interfacing Anthropomorphic Node) is a device that is physically attached to the monkey. It provides the communications interface between a monkey and its ZOO. It is effectively a translator for the monkey. It sends status reports and resource requests to the ZOO using human language phrases, and responds to ZOO requests on behalf of the monkey.

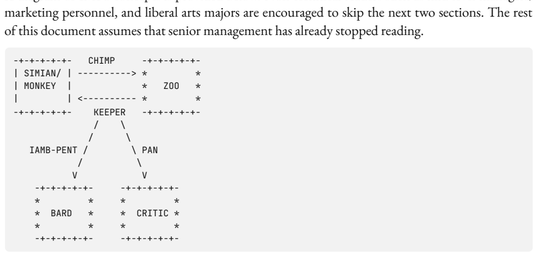
The SIMIAN uses the Cross-Habitat Idiomatic Message Protocol (CHIMP) to communicate with the ZOO. The ZOO uses the Knowledgeable and Efficient Emulation Protocol for Ecosystem Resources (KEEPER) to interact with the SIMIAN.

The ZOO obtains typewriter transcripts from the SIMIAN, which is responsible for converting the monkey's typed text into an electronic format if non-digital typewriters are used. The ZOO may then forward the transcripts to one or more entities who review the transcript's contents. IMPS defines two such reviewer protocols, although others could be added.

For Shakespearean works, as well as any other classic literature that has already been published, the ZOO forwards the transcript to a BARD (Big Annex of Reference Documents). The BARD determines if a transcript matches one or more documents in its annex. The ZOO sends the transcript to a BARD using the Inter-Annex Message Broadcasting Protocol for Evaluating Neoclassical Transcripts (IAMB-PENT). The transcripts are considered Neoclassical because (a) they are transferred in electronic media instead of the original paper medium, and (b) the word "classical" does not begin with the letter N.

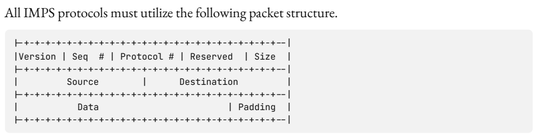
For new and potentially innovative works, the ZOO submits a transcript to a CRITIC (Collective Reviewer's Innovative Transcript Integration Center). The CRITIC determines if a transcript is sufficiently innovative to be published. The ZOO uses the Protocol for Assessment of Novelty (PAN) to communicate with the CRITIC. The process of using PAN to send a transcript to a CRITIC is sometimes referred to as foreshadowing.

A diagram of IMPS concepts is provided below. Non-technical readers such as mid-level managers, marketing personnel, and liberal arts majors are encouraged to skip the next two sections. The rest of this document assumes that senior management has already stopped reading.



3. IMPS Packet Structure

All IMPS protocols must utilize the following packet structure.



The Version, Sequence Number, Protocol Number, and Reserved fields are 32 bit unsigned integers. For IMPS version 1.0, the Version must be 1. Reserved must be 0 and will always be 0 in future uses. It is included because every other protocol specification includes a "future use" reserved field which never, ever changes and is therefore a waste of bandwidth and memory. [6] [7] [8].

The Source and Destination are identifiers for the IMPS objects that are communicating. They are represented using Infinite TAGs (see next section).

The Data section contains data which is of arbitrary length.

The Size field records the size of the entire packet using Infinite TAG encoding.

The end of the packet may contain extra padding, between 0 and 7 bits, to ensure that the size of packet is rounded out to the next byte.

4. Infinite Threshold Accounting Gadget (I-TAG) Encoding

Each SIMIAN requires a unique identifier within IMPS. This section describes design considerations for the IMPS identifier, referred to as an Infinite Threshold Accounting Gadget (I-TAG). The I-TAG can represent numbers of any size.

To uniquely identify each SIMIAN, a system is required that is capable of representing an infinite number of identifiers. The set of all integers can be used as a compact representation. However, all existing protocols inherently limit the number of available integers by specifying a maximum number of bytes to be used for an integer. This approach cannot work well in an IMPS network with an infinite number of monkeys to manage.

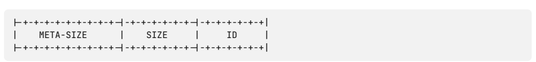
Practically speaking, one could select a byte size which could represent an integer that is greater than the number of atoms in the known universe. There are several limitations to this approach, however: (a) it would needlessly exclude IMPS implementations that may utilize sub-atomic monkeys and/or multiple universes; (b) there is not a consensus as to how many atoms there are in this universe; and (c) while the number is extremely large, it still falls pitifully short of infinity. Since any entity that fully implements IMPS is probably very, very good at handling infinite numbers, IMPS must ensure that it can represent them.

Netstrings, i.e. strings which encode their own size, were considered. However, netstrings have not been accepted as a standard, and they do not scale to infinity. As stated in [9], "[Greater than] 999999999 bytes is bad." Well put.

A scheme for identifying arbitrary dates was also considered for implementation [10]. While it solves the Y10K problem and does scale to infinity, its ASCII representation wastes memory by a factor greater than 8. While this may not seem important in an environment that has enough resources to support an infinite number of monkeys, it is inelegant for the purpose of monkey identification. It is also CPU intensive to convert such a representation to a binary number (at least based on the author's implementation, which was written in a combination of LISP, Perl, and Java). The algorithm is complicated and could lead to incorrect implementations. Finally, the author of this document sort of forgot about that RFC until it was too late to include it properly, and was already emotionally attached to the I-TAG idea anyway. It should be noted, however, that if a monkey had typed this particular section and it was submitted to a CRITIC, it would probably receive a PAN rejection code signifying the reinvention of the wheel.

Since there is no acceptable representation for I-TAGs available, one is defined below.

An I-TAG is divided into three sections:



SIZE specifies how many bytes are used to represent the ID, which is an arbitrary integer. META-SIZE specifies an upper limit on how many bits are used to represent SIZE.

META-SIZE is an arbitrary length sequence of N '1' bits terminated by a '0' bit, i.e. it has the form:

```
11111...1110
```

where N is the smallest number such that 2^N exceeds the number of bits required to represent the number of bytes that are necessary to store the ID (i.e., SIZE).

The SIZE is then encoded using N bits, ordered from the most significant bit to the least significant bit.

Finally, the ID is encoded using SIZE bytes.

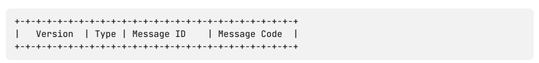
This representation, while clunky, makes efficient use of memory and is scalable to infinity. For any number X which is less than 2^N (for any N), a maximum of (N + log(N) + log(log(N)))/8 bytes is necessary to represent X. The math could be slightly incorrect, but it sounds right.

A remarkable, elegant little C function was written to implement I-TAG processing, but it has too many lines of code to include in this margin [11].

5. KEEPER Specification

Following is a description of the Knowledgeable and Efficient Emulation Protocol for Ecosystem Resources (KEEPER), which the ZOO uses to communicate with the SIMIAN. The IMPS protocol number for KEEPER is 1.

KEEPER is a connectionless protocol. The ZOO sends a request to the SIMIAN using a single IMPS packet. The SIMIAN sends a response back to the ZOO with another IMPS packet. The data portion of the packet is of the following form:



Version, Type, Message ID, and Message are all 16-bit integers.

Version = the version of KEEPER being used (in this document, the version is 1)

Type = the type of message being sent. '0' is a request; '1' is a response

Message ID = a unique identifier to distinguish different messages

Message Code = the specific message being sent

When a ZOO sends a KEEPER request, the SIMIAN must send a KEEPER response which uses the same Message ID as the original request.

5.1 KEEPER Message Request Codes (ZOO-to-SIMIAN)

CODE	NAME	DESCRIPTION
0	RESERVED	Reserved
1	STATUS	Determine status of monkey
2	HEARTBEAT	Check to see if monkey has a heartbeat
3	WAKEUP	Wake up monkey
4	TYPE	Make sure monkey is typing
5	FASTER	Monkey must type faster
6	TRANSCRIPT	Send transcript
7	STOP	Stop all monkey business
8-512	FUTURE	Reserved for future use
513+	USER	User defined

5.2 KEEPER Message Response Codes (SIMIAN-to-ZOO)

CODE	NAME	DESCRIPTION
0	RESERVED	Reserved
1	ASLEEP	Status: Monkey is asleep
2	GONE	Status: Monkey is not at typewriter
3	DISTRACTED	Status: Monkey is distracted (not typing)
4	NORESPONSE	Status: Monkey is not responding
5	ALIVE	Status: Monkey is alive
6	DEAD	Status: Monkey is dead
7	ACCEPT	Monkey accepts request
8	REFUSE	Monkey refuses request
9-512	FUTURE	Reserved for future use
513+	USER	User defined

5.3 Requirements for KEEPER Request and Response Codes

Below are the requirements for request and response codes within KEEPER.

1. A SIMIAN must respond to a STATUS request with an ALIVE, DEAD, ASLEEP, GONE, DISTRACTED, or NORESPONSE code.
2. A SIMIAN must respond to a HEARTBEAT request with an ALIVE or DEAD code. SIMIAN implementors must be careful when checking the heartbeat of very relaxed monkeys who practice transcendental meditation or yoga, as they may appear DEAD even if they are still alive.
3. A SIMIAN must respond to a STOP request with a NORESPONSE, ALIVE, DEAD, or GONE code. How a SIMIAN stops the monkey is implementation-specific. However, the SIMIAN should preserve the monkey's ALIVE status to protect the ZOO from being shut down by authorities or animal rights groups. If the monkey is present but the SIMIAN interface is unable to verify whether the monkey is ALIVE or DEAD, then it must use a NORESPONSE.
4. A SIMIAN should respond to a TYPE or FASTER request with an ACCEPT code, especially if there are deadlines. The only other allowed responses are REFUSE, ASLEEP, GONE, NORESPONSE, or DEAD. This protocol does not define what actions should be taken if a SIMIAN responds with REFUSE, although a BRIBE_BANANA command may be added in future versions.
5. A SIMIAN must respond to a WAKEUP request with ACCEPT, REFUSE, GONE, NORESPONSE, or DEAD.
6. A SIMIAN must respond to a TRANSCRIPT request by establishing a CHIMP session to send the transcript to the ZOO.

5.4 Example ZOO-to-SIMIAN Exchanges using KEEPER

Assume a ZOO (SanDiego) must interact with a monkey named BoBo. Using KEEPER, SanDiego would interface with BoBo's SIMIAN (BoBoSIM). The following exchange might take place if BoBo begins to evolve self-awareness and independence.

```
SanDiego> STATUS
BoBoSIM>  DISTRACTED
SanDiego> TYPE
BoBoSIM>  REFUSE
SanDiego> TYPE
BoBoSIM>  REFUSE
SanDiego> TYPE
BoBoSIM>  GONE
```

The following exchange might take place early in the morning, if BoBo was being poorly maintained and was working at its typewriter very late the night before.

```
SanDiego> WAKEUP
BoBoSIM>  NORESPONSE
SanDiego> WAKEUP
BoBoSIM>  NORESPONSE
SanDiego> WAKEUP
BoBoSIM>  NORESPONSE
SanDiego> HEARTBEAT
BoBoSIM>  DEAD
SanDiego> TRANSCRIPT
```

6. CHIMP Specification

Following is a description of the Cross-Habitat Idiomatic Message Protocol (CHIMP), which the SIMIAN uses to communicate with the ZOO. The IMPS protocol number for CHIMP is 2.

CHIMP is a connection-oriented protocol. A SIMIAN (the "client") sends a series of requests to the ZOO (the "server"), which sends replies back to the SIMIAN.

6.1. SIMIAN Client Requests

`SEND <resource>`

The SIMIAN is requesting a specific resource. The resource may be FOOD, WATER, MEDICINE, VETERINARIAN, or TECHNICIAN. The SIMIAN makes requests for FOOD or WATER by interpreting the monkey's behavior and environment, e.g. its food dish. It requests MEDICINE or VETERINARIAN if it observes that the monkey's health is declining in any way, e.g. carpal tunnel syndrome or sore buttocks. How the SIMIAN determines health is implementation-specific. In cases where the SIMIAN itself may be malfunctioning, it may request a TECHNICIAN.

`REPLACE <item>`

The ZOO must replace an item that is used by the monkey during typing activities. The item to be replaced may be TYPEWRITER, PAPER, RIBBON, CHAIR, TABLE, or MONKEY.

`CLEAN <item>`

The SIMIAN is requesting that the ZOO must clean an item. The item may be CHAIR, TABLE, or MONKEY. How the ZOO cleans the item is implementation-specific. This command is identified in the protocol because it has been theorized that if an infinite number of monkeys sit at an infinite number of typewriters, the smell would be unbearable [12]. If this theory is proven true, then CLEAN may become the most critical command in the entire protocol suite.

`NOTIFY <status>`

The SIMIAN notifies the ZOO of the monkey's status. The status may be any status as defined in the KEEPER protocol, i.e. ASLEEP, GONE, DISTRACTED, NORESPONSE, ALIVE, or DEAD.

`TRANSCRIPT <size>`

The SIMIAN notifies the ZOO of a new transcript from the monkey. The number of characters in the transcript is specified in the size parameter.

`BYE`

The SIMIAN is terminating the connection.

6.2. ZOO Server Responses

`HELO <free text>`

Upon initial connection, the ZOO must send a HELO reply.

`ACCEPT`

The ZOO will fulfill the SIMIAN's request.

`DELAY`

The ZOO will fulfill the SIMIAN's request at a later time.

`REFUSE`

The ZOO refuses to fulfill the SIMIAN's request.

`RECEIVED`

The ZOO has received the full text of a transcript that has been submitted by the SIMIAN.

6.3 Example SIMIAN-to-ZOO Session using CHIMP

Assume a monkey BoBo with a SIMIAN interface named BoBoSIM, and a ZOO named SanDiego. Once the BoBoSIM client has established a connection to the SanDiego server, the following session might take place.

```
SanDiego> HELO CHIMP version 1.0 4/1/2000
BoBoSIM> REPLACE PAPER
SanDiego> ACCEPT
BoBoSIM>  TRANSCRIPT 87
SanDiego> ACCEPT
BoBoSIM>  xvkxvn i hate Binky xFnk , feEL hungry and sIck sbNf
BoBoSIM>  so so sad sDNfkodgv .,n.,  ,HELP MEEEEEEEEE cv.Cvn l
SanDiego> RECEIVED
BoBoSIM>  SEND FOOD
SanDiego> ACCEPT
BoBoSIM>  SEND MEDICINE
SanDiego> DELAY
BoBoSIM>  SEND VETERINARIAN
SanDiego> REFUSE
BoBoSIM>  SEND VETERINARIAN
SanDiego> REFUSE
BoBoSIM>  NOTIFY NORESPONSE
SanDiego> ACCEPT
BoBoSIM>  NOTIFY DEAD
SanDiego> ACCEPT
BoBoSIM>  REPLACE MONKEY
SanDiego> ACCEPT
```

7. IAMB-PENT Specification

Following is a description of the Inter-Annex Message Broadcasting Protocol for Evaluating Neoclassical Transcripts (IAMB-PENT), which a ZOO uses to send transcripts to a BARD. The IMPS protocol number is 5.

IAMB-PENT is a connection-oriented protocol. A ZOO (the "client") sends a transcript phrases to the BARD (the "server"), which evaluates the transcript and notifies the ZOO if the transcript matches all of a classical work or a portion thereof.

7.1. ZOO Client Requests

`RECEIVETH <transcript name>`

The ZOO notifies the BARD of a new transcript to be evaluated. The name of the transcript is provided.

`ANON <size>`

The ZOO notifies the BARD that a transcript of the given size is to be provided soon. The text of the transcript is then sent.

`ABORTETH <A2> <U3> <A3> <U4> <A4> <U5> <A5>`

The ZOO notifies the BARD that it is about to close the connection. The ZOO must specify a closing message. A2, A3, A4, and A5 must be accented syllables. U3, U4, and U5 must not be accented.

7.2 BARD Responses

`HARK <U1> <A2> <U3> <A3> <U4> <A4> <U5> <A5>`

When the ZOO establishes a connection, the BARD must send a HARK command. A2, A3, A4, and A5 must be accented syllables. U1, U2, U3, U4, and U5 must not be accented.

`PRITHEE <A2> <U3> <A3> <U4> <A4> <U5> <A5>`

When a ZOO uses a RECEIVETH command to specify a forthcoming transcript, the BARD must respond with a PRITHEE. A2, A3, A4, and A5 must be accented syllables. U3, U4, and U5 must not be accented.

`REGRETTETH <A2> <U3> <A3> <U4> <A4> <U5> <A5>`

If the BARD does not have the transcript in its Annex, it uses the REGRETTETH command to notify the ZOO. A2, A3, A4, and A5 must be accented syllables. U3, U4, and U5 must not be accented.

`ACCEPTETH <A2> <U3> <A3> <U4> <A4> <U5> <A5>`

If the BARD has located the transcript in its Annex, it uses the ACCEPTETH command to notify the ZOO. A2, A3, A4, and A5 must be accented syllables. U3, U4, and U5 must not be accented.

7.3 Example ZOO-to-BARD Session using IAMB-PENT

This is a sample IAMB-PENT session in which a ZOO (SanDiego) sends a transcript to a BARD (William).

```
William> HARK now, what light through yonder window breaks?
SanDiego> RECEIVETH TRANSCRIPT SanDiego.BoBo.17
William> PRITHEE thy monkey's wisdom poureth forth!
SanDiego> ANON 96
SanDiego> I must be cruel, only to be kind.  Thus bad begins,
          and worse remains in front.
William> REGRETTETH none hath writ thy words before
SanDiego> ABORTETH Fate may one day bless my zone
```

8. PAN Specification

Following is a description of the Protocol for Assessment of Novelty (PAN). A ZOO uses PAN to send monkey transcripts for review by a CRITIC. The IMPS protocol number for PAN is 10 [13].

PAN is a connection-oriented protocol. A ZOO (the "unwashed masses") sends a request to the CRITIC (the "all-powerful"), which sends a response back to the ZOO.

8.1. ZOO Requests

`COMPLIMENT <text>`

The ZOO may say something nice to the CRITIC using the given text. The CRITIC does not respond to the compliment within the protocol. However, it is generally believed that the CRITIC is more likely to accept a new transcript when a ZOO uses many compliments.

`TRANSCRIPT <name> <size>`

The ZOO notifies the CRITIC of a new transcript for review. The name of the transcript, plus the number of characters, are specified as parameters to this request. The text of the transcript is then sent.

`THANKS`

This is an indicator that a ZOO is about to terminate the connection.

8.2. CRITIC Responses

`SIGH <insult>`

When the ZOO establishes a connection, the CRITIC must respond with a SIGH and an optional insult.

`IMPRESS_ME`

A CRITIC must respond with an IMPRESS_ME once a ZOO has made a TRANSCRIPT request.

`REJECT <code> REJECT 0 <text>`

When a transcript has been received, the CRITIC must respond with a REJECT and a code that indicates the reason for rejection. A table of rejection codes is provided below. When the code is 0, the CRITIC may respond using free text. A CRITIC may send a REJECT before it has received or processed the full text of the transcript.

`DONT_CALL_US_WE'LL_CALL_YOU`

The CRITIC makes this statement before terminating the connection.

`GRUDGING_ACCEPTANCE`

THIS RESPONSE IS NOT SUPPORTED IN THIS VERSION OF PAN. The Working group for the Infinite Monkey Protocol Suite (WIMPS) agreed that it is highly unlikely that a CRITIC will ever use this response when a REJECT is available. It is only included as an explanation to implementors who do not fully understand how CRITICs work. In time, it is possible that a CRITIC may evolve (in much the same way that a monkey might). Should such a time ever come, the WIMPS may decide to support this response in later versions of PAN.

8.3. Table of CRITIC Reject Codes

CODE	DESCRIPTION
0	<Encrypted response following; see below>
1	"You're reinventing the wheel."
2	"This will never, ever sell."
3	"Huh? I don't understand this at all."
4	"You forgot one little obscure reference from twenty years ago that renders your whole idea null and void."
5	"Due to the number of submissions, we could not accept every transcript."
6	"There aren't enough charts and graphs. Where is the color?"
7	"I'm cranky and decided to take it out on you."
8	"This is not in within the scope of what we are looking for."
9	"This is too derivative."
10	"Your submission was received after the deadline. Try again next year."

If the CRITIC uses a reject code of 0, then the textual response must use an encryption scheme that is selected by the CRITIC. Since the PAN protocol does not specify how a ZOO may determine what scheme is being used, the ZOO might not be able to understand the CRITIC's response.

8.4. Example ZOO-to-CRITIC Session using PAN

Below is a sample session from a ZOO (SanDiego) to a CRITIC (NoBrainer).

```
NoBrainer> SIGH Abandon hope all who enter here
SanDiego> COMPLIMENT We love your work.  Your words are like
SanDiego> COMPLIMENT jewels and you are always correct.
SanDiego> TRANSCRIPT RomeoAndJuliet.BoBo.763 251
NoBrainer> IMPRESS_ME
SanDiego> Two households, both alike in dignity,
SanDiego> In fair Verona, where we lay our scene,
SanDiego> From ancient grudge break to new mutiny,
SanDiego> Where civil blood makes civil hands unclean.
SanDiego> From forth the fatal loins of these two foes
SanDiego> A pair of star-cross'd lovers take their life;
NoBrainer> REJECT 2    ("This will never, ever sell.")
SanDiego> THANKS
NoBrainer> DONT_CALL_US_WE'LL_CALL_YOU
```

9. Security Considerations

In accordance with the principles of the humane treatment of animals, the design of IMPS specifically prohibits the CRITIC from contacting the SIMIAN directly and hurting its feelings. BARDs and CRITICs are also separated because of fundamental incompatibilities and design flaws.

The security considerations for the rest of IMPS are similar to those for the original Internet protocols. Specifically, IMPS refuses to learn from the mistakes of the past and blithely repeats the same errors without batting an eye. Spoofing and denial of service attacks abound if untrusted entities gain access to an IMPS network. Since all transmissions occur in cleartext without encryption, innovative works are subject to theft, which is not a significant problem unless the network contains entities other than CRITICs. The open nature of BARDs with respect to IAMB-PENT messages allows a BARD to borrow heavily from transmitted works, but by design BARDs are incapable of stealing transcripts outright.

The ZOO may be left open to exploitation by pseudo-SIMIANs from around the world. A third party could interrupt communications between a ZOO and a SIMIAN by flooding the SIMIAN with packets, incrementing the message ID by 1 for each packet. More heinously, the party could exploit the KEEPER protocol by sending a single STOP request to each SIMIAN, thus causing a massive denial of service throughout the ZOO. The party could also spoof a CHIMP request or send false information such as a DEAD status, which could cause a ZOO to attempt to replace a monkey that is still functioning properly.

In addition, if a ZOO repeatedly rejects a SIMIAN's requests (especially those for FOOD, WATER, and VETERINARIAN), then the ZOO may inadvertently cause its own denial of service with respect to that particular SIMIAN. However, both KEEPER and CHIMP allow the ZOO to detect this condition in a timely fashion via the NORESPONSE or DEAD status codes.

All BARDs are inherently insecure because they face insurmountable financial problems and low prioritization, which prevents them from working reliably. In the rare cases when a BARD implementation overcomes these obstacles, it is only successful for 15 minutes, and reverts to being insecure immediately thereafter [14]. Since a CRITIC could significantly reduce the success of a BARD with an appropriate PAN response, this is one more reason why BARDs and CRITICs should always be kept separate from each other.

It is expected that very few people will care about most implementations of CRITIC, and CRITICs themselves are inherently insecure. Therefore, security is not a priority for CRITICs. The CRITIC may become the victim of a denial of service attack if too many SIMIANs submit transcripts at the same time. In addition, one SIMIAN may submit a non-innovative work by spoofing another SIMIAN (this is referred to as the Plagiarism Problem). A CRITIC response can also be spoofed, but since the only response supported in PAN version 1 is REJECT, this is of little consequence. Care must be taken in future versions if a GRUDGING_ACCEPTANCE response is allowed. Finally, a transcript may be lost in transmission, and PAN does not provide a mechanism for a ZOO to determine if this has happened. Future versions of IMPS may be better suited to answer this fundamental design problem: if an innovative work is lost in transmission, can a CRITIC still PAN it?

Based on the number of packet-level vulnerabilities discovered in recent years, it is a foregone conclusion that some implementations will behave extremely poorly when processing malformed IMPS packets with incorrect padding or reserved bits [15] [16] [17].

Finally, no security considerations are made with respect to the fact that over the course of infinite time, monkeys may evolve and discover how to control their own SIMIAN interfaces and send false requests, or to compose and submit their own transcripts. There are indications that this may already be happening [18].

10. Acknowledgements

The author wishes to thank Andre Frech for technical comments that tripled the size of this document, Kean Kaufmann and Amanda Vizedom for lectures on Shakespearean grammar, Rohn Blake for clarifying the nature of the entire universe, William Shakespeare for accents, the number 16, and the color yellow.

11. References

1. The Famous Brett Watson, "The Mathematics of Monkeys and Shakespeare." http://www.nutters.org/monkeys.html
2. Dr. Math. "Monkeys Typing Shakespeare: Infinity Theory." http://forum.swarthmore.edu/dr.math/problems/bridge8.5.98.html
3. K. Clark, Stark Mill Brewery, Manchester, NH, USA. Feb 18, 2000. (personal communication). "Good question! I never thought of that! I bet nobody else has, either. Please pass the french fries."
4. The author was unable to find a reference in any issue of TV Guide published between 1956 and the date of this document.
5. "Dough Re Mi," The Brady Bunch. Original air date January 14, 1972.
6. Postel, J., " Internet Protocol", STD 5, RFC 791, September 1981.
7. Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
8. Brown, C. and A. Malis, "Multiprotocol Interconnect over Frame Relay", STD 55, RFC 2427, September 1998.
9. Internet-Draft, bernstein-netstrings-06 (expired Work in Progress). D.J. Bernstein. Inclusion of this reference is a violation of RFC 2026 section 2.2.
10. Glassman, S., Manasse, M. and J. Mogul, "Y10K and Beyond", RFC 2550, 1 April 1999.
11. "My Last Theorem: A Prankster's Guide to Ageless Mathematical Jokes That are Funny Because They're True and People Can't Prove Them for Centuries." P. Fermat. Circa 1630.
12. .signature in various USENET postings, circa 1994. Author unknown.
13. "Recognizing Irony, or How Not to be Duped When Reading." Faye Halpern. 1998. http://www.brown.edu/Student_Services/Writing_Center/halpern1.htm
14. Andy Warhol. Circa 1964.
15. CERT Advisory CA-98-13. CERT. December 1998. http://www.cert.org/advisories/
16. CERT Advisory CA-97.28. CERT. December 1997. http://www.cert.org/advisories/
17. CERT Advisory CA-96.26. CERT. December 1996. http://www.cert.org/advisories/
18. All issues of TV Guide published between 1956 and the date of this document.

12. Author's Address

SteQven M. Christey
Email: steqve@shore.net

13. Full Copyright Statement

Copyright (C) The Internet Society (2000). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

Status of this Memo. This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice. Copyright (C) The Internet Society (2000). All Rights Reserved.

doi:10.17487/RFC2795