

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CROWDSTRIKE, INC.

Petitioner

v.

TAASERA

Patent Owner

Case No. IPR2024-00027

Patent No. 7,673,137

DECLARATION OF JUNE ANN MUNFORD

1. My name is June Ann Munford. I am over the age of 18, have personal knowledge of the facts set forth herein, and am competent to testify to the same.
2. I earned a Master of Library and Information Science (MLIS) from the University of Wisconsin-Milwaukee in 2009. I have over ten years of experience in the library/information science field. Beginning in 2004, I have served in various positions in the public library sector including Assistant Librarian, Youth Services Librarian and Library Director. I have attached my Curriculum Vitae as Appendix CV.
3. During my career in the library profession, I have been responsible for materials acquisition for multiple libraries. In that position, I have cataloged, purchased and processed incoming library works. That includes purchasing materials directly from vendors, recording publishing data from the material in question, creating detailed material records for library catalogs and physically preparing that material for circulation. In addition to my experience in acquisitions, I was also responsible for analyzing large collections of library materials, tailoring library records for optimal catalog

search performance and creating lending agreements between libraries during my time as a Library Director.

4. I am fully familiar with the catalog record creation process in the library sector. In preparing a material for public availability, a library catalog record describing that material would be created. These records are typically written in Machine Readable Catalog (herein referred to as “MARC”) code and contain information such as a physical description of the material, metadata from the material’s publisher, and date of library acquisition. In particular, the 008 field of the MARC record is reserved for denoting the date of creation of the library record itself. As this typically occurs during the process of preparing materials for public access, it is my experience that an item’s MARC record indicates the date of an item’s public availability.
5. Typically, in creating a MARC record, a librarian would gather various bits of metadata such as book title, publisher and subject headings among others and assign each value to a relevant numerical field. For example, a book’s physical description is tracked in field 300 while title/attribution is tracked in field 245. The 008 field of the MARC record is reserved for denoting the creation of the library record itself. As this is the only date reflecting the inclusion of said materials within the library’s collection, it is my experience

that an item's 008 field accurately indicates the date of an item's public availability.

6. I have reviewed Exhibit 1005, an IBM Research Report entitled *ChakraVyuha (CV): A Sandbox Operating System Environment for Controlled Execution of Alien Code* by Asit Dan, Ajay Mohindra, Rajiv Ramaswami and Dinkar Sitaram.
7. Attached hereto as Appendix DAN01 is a true and correct copy of the MARC record for *ChakraVyuha (CV): A Sandbox Operating System Environment for Controlled Execution of Alien Code* as held by the Carnegie Mellon University Library. I secured this record myself from the library's public catalog. The MARC record contained within Appendix DAN01 accurately describes *ChakraVyuha (CV): A Sandbox Operating System Environment for Controlled Execution of Alien Code*.
8. My determination that the MARC record contained within Appendix DAN01 accurately describes *ChakraVyuha (CV): A Sandbox Operating System Environment for Controlled Execution of Alien Code* is based upon the accuracy of several fields within this MARC record. The 088 field, describing the work's report number, matches the report number of

ChakraVyuha (CV): A Sandbox Operating System Environment for Controlled Execution of Alien Code as represented in Exhibit 1005. The 245 field, describing the title and authorship of the work, matches the title and authors as represented in Exhibit 1005. The 260 field describing the publisher of the work is properly attributed to the IBM Corporation, as found in Exhibit 1005. The 300 field, describing the physical attributes of the work, matches the page number as represented in Exhibit 1005.

9. Attached hereto as Appendix DAN02 is a true and correct copy of *ChakraVyuha (CV): A Sandbox Operating System Environment for Controlled Execution of Alien Code* as held by the Carnegie Mellon University Library. I secured these scans myself from the library's collection. In comparing Exhibit 1005 to Appendix DAN02, it is my determination that Exhibit 1005 is a true and correct copy of *ChakraVyuha (CV): A Sandbox Operating System Environment for Controlled Execution of Alien Code* by Asit Dan, Ajay Mohindra, Rajiv Ramaswami and Dinkar Sitaram.

10. While Appendix DAN01 does accurately describe the report number, title, author, publisher and page number of *ChakraVyuha (CV): A Sandbox Operating System Environment for Controlled Execution of Alien Code*

found within Exhibit 1005, this record also accurately describes the report number, title, author, publisher and page number of *ChakraVyuha (CV): A Sandbox Operating System Environment for Controlled Execution of Alien Code* as presented in Appendix DAN02. The MARC record presented in Appendix DAN01 also describes several proprietary elements exclusive to the Carnegie Mellon University copy of this report presented in Appendix DAN02. For instance, this paper is organized on Carnegie Mellon's shelves by call number – a combination of letters and numbers that ensure library materials are shelved near other materials with similar topics. This call number – IBMC 20742 – can be found on page 1 of the MARC record in field 099, page 2 of the MARC record in the 'location items' field, and in two places on page 1 of Appendix DAN02 – the top left of the page and a shelving sticker in the bottom left of the page. On page 2 of Appendix DAN02, a post-it note can be found reading as follows: '1. client/server computer, 2. computer security, 3. coding theory'. These phrases are subject headings, typically used to describe the topic of a work when creating a MARC record. These subject headings identically correspond to the subject heading fields found in field 650 of Appendix DAN01.

11. The 008 field of the MARC record in Appendix DAN01 indicates the date of record creation. The 008 field of Appendix DAN01 indicates Carnegie Mellon University library first acquired this book on July 3, 1997.

Considering this information, it is my determination that *ChakraVyuha (CV): A Sandbox Operating System Environment for Controlled Execution of Alien Code* was made available to the public shortly after its initial acquisition as of July 3, 1997.

12. In addition to the 008 field of Appendix DAN01 indicating the first date of library acquisition, there is additional information regarding availability to be found within Carnegie Mellon's copy of *ChakraVyuha (CV): A Sandbox Operating System Environment for Controlled Execution of Alien Code* as presented in Appendix DAN02. On page 25 of Appendix DAN02, there is a date stamped reading 'JUL 22 1997'. In my experience preparing library material for public accessibility, this marking is known as a date stamp and is added by library staff after the material has been cataloged. In my experience cataloging and preparing materials for the shelf, this 19-day gap between the two dates is typical of the cataloging process.

13. On page 1 of Appendix DAN02, there is a stamp on the bottom of the page reading 'Room Use Only Until AUG 15 1997'. In my experience preparing

library materials for public accessibility, this stamp represents a usage restriction typical of some library materials - the need for the library user to interact with the material on-site at the library in some type of private reading room. The date visible on this stamp has been struck, implying that the usage restriction has expired. I note that this usage restriction would have no impact on the public accessibility of the material.

14. The MARC record for *ChakraVyuha (CV): A Sandbox Operating System Environment for Controlled Execution of Alien Code* found within Appendix DAN01 in combination with features present in the Appendix DAN02 provide a clear timeline of this material's accessibility via the Carnegie Library University library. The 008 field of *ChakraVyuha (CV): A Sandbox Operating System Environment for Controlled Execution of Alien Code's* MARC record indicates this paper was first cataloged by Carnegie Mellon Librarians as of July 3, 1997. The date stamp on page 25 of Appendix DAN02 indicates this book was prepared for public accessibility as of July 22, 1997. The usage restriction stamp on page 1 of Appendix DAN01 indicates this material was made available to the public at first with some usage restrictions, which expired as of August 15, 1997.

15. I am familiar with the Internet Archive, a digital library formally certified by the State of California as a public library. Among other services that the Internet Archive makes available to the general public is the Wayback Machine, an online archive. The Internet Archive's Wayback Machine service archives webpages as of a certain capture date to track changes in the web over time. The Internet Archive has been in operation as a nonprofit library since 1996 and has hosted the Wayback Machine service since its inception in 2001. During my time as a librarian, I frequently used the Internet Archive's Wayback Machine for research and instruction purposes. This includes teaching instructional classes on using the Wayback Machine to library patrons and using the Wayback Machine to research reference inquiries that require hard-to-find online resources. I consider the Internet Archive's recordskeeping to be as rigorous and detailed as other formal library recordskeeping practices such as MARC records, OCLC records and Dublin Core.

16. Attached hereto as Appendix CMU01 is a screen capture of the Internet Archive Wayback Machine entry for <http://hathorne.library.cmu.edu/uhtbin/cgiirsi/0/1/0>. I secured these screen captures myself from

https://web.archive.org/web/19970301000000*/http://hathorne.library.cmu.edu/uhtbin/cgiirsi/0/1/0.

17. Appendix CMU01 is a webpage entitled *Unicorn WebCat – Unicorn Public Access Choices* as published on the Carnegie Mellon University library website, captured by the Internet Archive on April 24, 1997. This page describes user options when interacting with the Carnegie Mellon University library catalog as such: ‘Library materials are described in indexed records in the online catalog. Under the library catalog you can lookup library materials using a particular word or phrase, or review lists of the library’s authors, titles, subjects, series, keywords, or call numbers.’

18. Attached hereto as Appendix CMU02 is a screen capture of the Internet Archive Wayback Machine entry for <http://www.library.cmu.edu/Research/Online/index.html>. I secured these screen captures myself from <https://web.archive.org/web/19971009135520/http://www.library.cmu.edu/Research/Online/index.html>.

19. Appendix CMU02 is a webpage entitled *Catalogs and Databases* as published on the Carnegie Mellon University library website, captured by

the Internet Archive on October 9, 1997. This page describes the software used to manage the library's online public-facing catalog, a commonly used software program by the Sirsi Corporation known as WebCat. Carnegie Mellon's use of this software indicates that the catalog includes shelving locations for library materials held at different library facilities. This page also confirms that the catalog is updated daily, that users can interface with this catalog via a web browser and that access is unrestricted aside from a few exceptions detailed within.

20. To summarize, Appendix CMU01 indicates that Carnegie Mellon library materials are described in indexed library catalog records and presented as an online catalog that allows users to search by words/phrases and review lists of the library's authors, titles, subjects, series, keywords, or call numbers. Appendix CMU02 indicates that Carnegie Mellon University's library records are designed to correspond with the library's holdings in individual libraries, such as the Hunt Library where *ChakraVyuha (CV): A Sandbox Operating System Environment for Controlled Execution of Alien Code* can be found. Appendix CMU02 also indicates that Carnegie Mellon University are using Sirsi's Unicorn and WebCat software to present their online catalog.

21. Attached hereto as Appendix SIRSI01 is a screen capture of the Internet Archive Wayback Machine entry for <http://www.sirsi.com/Products/ulms.html>. I secured these screen captures myself from <https://web.archive.org/web/19970627002415/http://www.sirsi.com/Products/ulms.html>.

22. Appendix SIRSI01 is a webpage entitled *Unicorn Library Management System* as published on the Sirsi Corporation website, captured by the Internet Archive on June 27, 1997. This page describes the Unicorn Library Management software used by Carnegie Mellon University as of 1997, visible in Appendix CMU01. The page describes the public accessibility features of the Unicorn software as follows: 'UNICORN's highly evolved public access catalog is designed to address the needs of the beginning searcher, as well as the researcher. Separate choices are available for browsing and keyword searching. A full-text index is available as a standard feature of UNICORN, providing the best possible approach for accessing all library material. Precise search results are easily retrieved using UNICORN's sophisticated truncation, boolean, relational, adjacency, and proximity operators.'

23. Attached hereto as Appendix SIRSI02 is a screen capture of the Internet Archive Wayback Machine entry for <http://www.sirsi.com/Products/webcat.html>. I secured these screen captures myself from <https://web.archive.org/web/19970627002554/http://www.sirsi.com/Products/webcat.html>.

24. Appendix SIRSI02 is a webpage entitled *WebCat with Z39.50* as published on the Sirsi Corporation website, captured by the Internet Archive on June 27, 1997. This page describes the WebCat Library Catalog software used by Carnegie Mellon University as of 1997, visible in Appendices CMU01 and CMU02. Some of the WebCat features are detailed as follows: ‘The easy-to-use, familiar look and feel of today’s web browsers host WebCat’s powerful features: quickly create and execute simple or complex search statements, browse and select terms from heading lists, search related information by clicking on hypertext terms, view cross-reference information, sort search results, export selected records to networked printers, files or email, link to library holdings information on one or more servers [and] execute previous searches from search history.’

25.To further summarize, Appendix SIRSI01 describes the Unicorn software implemented by Carnegie Mellon University libraries as of 1997. Based on Sirsi's descriptions of the Unicorn software, library catalog users are presented a full-text index of the library's holdings and can review those holdings via browsing or search features. Specifically, the Unicorn program includes advanced search features such as keyword searching, boolean operators and proximity operators. Appendix SIRSI02 describes the WebCat software implemented by Carnegie Mellon University libraries as of 1997. Based on Sirsi's descriptions of WebCat, users are presented with an interactive library catalog when interacting with a WebCat page. Specifically, the WebCat online catalog supports simple and complex search statements, cross-references between similar materials, the ability to sort search results based on user preferences and the ability to browse one's search history.

26.Appendices CMU01 and CMU02 indicate the Carnegie Mellon University Library implemented Sirsi's Unicorn and WebCat software to create a detailed inventory of the library's holdings and present users with an interactive library catalog to search this inventory. Appendices SIRSI01 and SIRSI02 indicate that Sirsi's library software enables users to search through library catalog records using various complex search strategies. As such, it is

my determination that *ChakraVyuha (CV): A Sandbox Operating System Environment for Controlled Execution of Alien Code* was sufficiently indexed at the time of its acquisition by the Carnegie Mellon University Library in July 1997. This indexing via Carnegie Mellon University's Library catalog records allows any library catalog user to locate *ChakraVyuha (CV): A Sandbox Operating System Environment for Controlled Execution of Alien Code* via common search engines practices and access this paper within the Carnegie Mellon University Library's collection. Thus, any members of the interested public exercising reasonable diligence would have been able to locate *ChakraVyuha (CV): A Sandbox Operating System Environment for Controlled Execution of Alien Code* in the Carnegie Mellon University Library as of July 22, 1997.

27. I have been retained on behalf of the Petitioner to provide assistance in the above-illustrated matter in establishing the authenticity and public availability of the documents discussed in this declaration. I am being compensated for my services in this matter at the rate of \$200.00 per hour plus reasonable expenses. My statements are objective, and my compensation does not depend on the outcome of this matter.

28.I declare under penalty of perjury that the foregoing is true and correct. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: 10/20/2023

A handwritten signature in black ink, appearing to read "June Ann Munford", written in a cursive style.

June Ann Munford

Appendix CV

June A. Munford
Curriculum Vitae

Education

University of Wisconsin-Milwaukee - MS, Library & Information Science, 2009
Milwaukee, WI

- Coursework included cataloging, metadata, data analysis, library systems, management strategies and collection development.
- Specialized in library advocacy, cataloging and public administration.

Grand Valley State University - BA, English Language & Literature, 2008
Allendale, MI

- Coursework included linguistics, documentation and literary analysis.
- Minor in political science with a focus in local-level economics and government.

Professional Experience

Researcher / Expert Witness, October 2017 – present
Freelance ● Pittsburgh, Pennsylvania & Grand Rapids, Michigan

- Material authentication and public accessibility determination. Declarations of authenticity and/or public accessibility provided upon research completion. Experienced with appeals and deposition process.
- Research provided on topics of public library operations, material publication history, digital database services and legacy web resources.
- Past clients include Alston & Bird, Arnold & Porter, Baker Botts, Fish & Richardson, Erise IP, Irell & Manella, O'Melveny & Myers, Perkins-Coie, Pillsbury Winthrop Shaw Pittman and Slayden Grubert Beard.

Library Director, February 2013 - March 2015
Dowagiac District Library ● Dowagiac, Michigan

- Executive administrator of the Dowagiac District Library. Located in

Southwest Michigan, this library has a service area of 13,000, an annual operating budget of over \$400,000 and total assets of approximately \$1,300,000.

- Developed careful budgeting guidelines to produce a 15% surplus during the 2013-2014 & 2014-2015 fiscal years while being audited.
- Using this budget surplus, oversaw significant library investments including the purchase of property for a future building site, demolition of existing buildings and building renovation projects on the current facility.
- Led the organization and digitization of the library's archival records.
- Served as the public representative for the library, developing business relationships with local school, museum and tribal government entities.
- Developed an objective-based analysis system for measuring library services - including a full collection analysis of the library's 50,000+ circulating items and their records.

November 2010 - January 2013

Librarian & Branch Manager, Anchorage Public Library ● Anchorage, Alaska

- Headed the 2013 Anchorage Reads community reading campaign including event planning, staging public performances and creating marketing materials for mass distribution.
- Co-led the social media department of the library's marketing team, drafting social media guidelines, creating original content and instituting long-term planning via content calendars.
- Developed business relationships with The Boys & Girls Club, Anchorage School District and the US Army to establish summer reading programs for children.

June 2004 - September 2005, September 2006 - October 2013

Library Assistant, Hart Area Public Library
Hart, MI

- Responsible for verifying imported MARC records and original MARC

cataloging for the local-level collection as well as the Michigan Electronic Library.

- Handled OCLC Worldcat interlibrary loan requests & fulfillment via ongoing communication with lending libraries.

Professional Involvement

Alaska Library Association - Anchorage Chapter

- Treasurer, 2012

Library Of Michigan

- Level VII Certification, 2008
- Level II Certification, 2013

Michigan Library Association Annual Conference 2014

- New Directors Conference Panel Member

Southwest Michigan Library Cooperative

- Represented the Dowagiac District Library, 2013-2015

Professional Development

Library Of Michigan Beginning Workshop, May 2008

Petoskey, MI

- Received training in cataloging, local history, collection management, children's literacy and reference service.

Public Library Association Intensive Library Management Training, October 2011

Nashville, TN

- Attended a five-day workshop focused on strategic planning, staff management, statistical analysis, collections and cataloging theory.

Alaska Library Association Annual Conference 2012 - Fairbanks, February 2012

Fairbanks, AK

- Attended seminars on EBSCO advanced search methods, budgeting, cataloging, database usage and marketing.

Depositions

2019 ● Fish & Richardson

Apple v. Qualcomm (IPR2018-001281, 39521-00421IP, IPR2018-01282 and 39521-00421IP2)

2019 ● Erise IP

Implicit, LLC v. Netscout Systems, Inc (Civil Action No. 2:18-cv-53-JRG)

2019 ● Perkins-Coie

Adobe Inc. v. RAH Color Technologies LLC (Cases IPR2019-00627, IPR2019-00628, IPR2019-00629 and IPR2019-00646)

2020 ● O'Melveny & Myers

Maxell, Ltd. v. Apple Inc. (Case No. 5:19-cv-00036-RWS)

2021 ● Pillsbury Winthrop Shaw Pittman LLP

Intel v. SRC (IPR2020-1449)

2022 ● Perkins-Coie

Realtek v. Future Link (IPR2021-01182)

2023 ● Fish & Richardson

Neuroderm Ltd. v. Abbvie, Inc (Case No. PGR2022-00040)

2023 ● Fish & Richardson

Nearmap US Inc. v. Pictometry International Corp. (IPR2022-00735)

2023 ● Fish & Richardson

Samsung Electronics v. MemoryWeb LLC (Case No. 39843-0136PS1)

Limited Case History & Potential Conflicts

Alston & Bird

- Ericsson
 - v. Collision Communications (IPR2022-01233)
- Nokia
 - v. Neptune Subsea, Xtera (Case No. 1:17-cv-01876)
- Universal Electronics Inc
 - v. Roku Inc (IPR2022-00818)

Arnold & Porter

- Ivantis
 - v. Glaukos (Case No. 8:18-cv-00620)
- Samsung
 - v. Jawbone (Case No. 2:21-cv-00186)

Benesch Friedlander Coplan & Aronoff

- Voyis
 - v. Cathx (Case No. 5:21-cv-00077-RWS)

Erise I.P.

- Apple
 - v. Ericsson Inc. (IPR2022-00715)

- v. Future Link Systems (IPRs 6317804, 6622108, 6807505, and 7917680)
- v. INVT (Case No. 20-1881)
- v. Navblazer LLC (IPR2020-01253)
- v. Qualcomm (IPR2018-001281, 39521-00421IP, IPR2018-01282, 39521-00421IP2)
- v. Quest Nettech Corp (Case No. 2:19-cv-00118-JRG)
- v. Telefonaktiebolaget LM Ericsson (IPR2022-00275)

- Fanduel

- v. CGT (Case No. 19-1393)

- Garmin

- v. Phillips North America LLC (Case No. 2:19-cv-6301-AB-KS)

- Netscout

- v. Longhorn HD LLC (Case No. 2:20-cv-00349)
 - v. Implicit, LLC (Civil Action No. 2:18-cv-53-JRG)

- Sony Interactive Entertainment LLC

- v. Bot M8 LLC (IPR2020-01288)
 - v. Infernal Technology LLC (Case No. 2:19-CV-00248-JRG)

- Unified Patents

- v. GE Video Compression (Civil Action No. 2:19-cv-248)

Fish & Richardson

- Apple

v. AliveCor (3:21-cv-03958)
v. LBS Innovations (Case No. 2:19-cv-00119-JRG-RSP)
v. Koss Corporation (IPR2021-00305)
v. Masimo (IPR 50095-0012IP1, 50095-0012IP2, 50095-0013IP1,
50095-0013IP2, 50095-0006IP1, 50095-0135IP1)
v. Neonode (Case No. 21-cv-08872-EMC)
v. Qualcomm (IPR2018-001281, 39521-00421IP, IPR2018-01282,
39521-00421IP2)

- Dell

v. Neo Wireless (IPR2022-00616)

- Dish Network

v. Realtime Adaptive Streaming (Case No. 1:17-CV-02097-RBJ)

v. TQ Delta LLC (Case No. 18-1798)

- Evapco Dry Cooling

v. SPG Dry Cooling (IPR2021-00688)

- Genetec

v. Sensormatic Electronics (Case No. 1:20-CV-00760)

- Huawei

v. Bell Northern Research LLC (IPR2019-01174)

- Kianxis

v. Blue Yonder (Case No. 3:20-cv-03636)

- LG Electronics
 - v. Bell Northern Research LLC (Case No. 3:18-cv-2864-CAB-BLM)
- Metaswitch
 - v. Sonus Networks (IPR2018-01719)
- Microsoft
 - v. Throughputer Inc (IPR2022-00757)
- MLC Intellectual Property
 - v. MicronTech (Case No. 3:14-cv-03657-SI)
- Nearmap Inc
 - v. EagleView Technologies (IPR2022-01009)
- Neuroderm Ltd.
 - v. Abbvie, Inc (Case No. PGR2022-00040)
- Realtek Semiconductor
 - v. Future Link (IPR2021-01182)
- Quectel
 - v. Koninklijke Philips (Case No. 1:20-cv-01710)
- Samsung
 - v. Aire Technology (IPR2022-00877)
 - v. Bell Northern Research (Case No. 2:19-cv-00286-JRG)
 - v. Communication Technologies Inc (IPR2022-01221)

- v. Jawbone Innovations (IPR2022-00865)
- v. MemoryWeb LLC (IPR2022-00885)
- v. Telefonaktiebolaget LM Ericsson (IPR2021-00615)

- Texas Instruments

- v. Vantage Micro LLC (IPR2020-01261)

- Xilinx

- v. Sentient Sensors LCC (Case No. 1:22-cv-00173)

Irell & Manella

- Curium

O'Melveny & Myers

- Apple

- v. Maxell (Case No. 5:19-cv-00036-RWS)

- Samsung

- v. Daedalus Prime LLC (1335 Investigation)

Perkins-Coie

- Heru Industries

- v. The UAB Research Foundation (IPR2022-01148)

- Intel Corporation

- v. BitMICRO (Case No. 5:23-cv-00625)

- Realtek Semiconductor

v. Future Link (IPR2021-01182)

- Twitter Inc

v. VOIP-Pal.com (Case No. 3:20-cv-02397-JD)

- TCL Industries

v. Koninklijke Philips NV (IPR2021-00495, IPR2021-00496
and IPR2021-00497)

Pillsbury Winthrop Shaw Pittman

- Intel

v. FG SRC LLC (Case No. 6:20-cv-00315)

- Gravel Rating Systems

v. Costco (Case No. 4:21-cv-149)

v. Lowe's Home Centers (Case No. 4:21-cv-150)

v. T-Mobile USA (Case No. 4:21-cv-152)

v. Kohl's Inc. (Case No. 4:21-cv-258)

v. Under Armor (Case No. 4:21-cv-356)

Appendix DAN01

leader 02477nam a2200409Ia 4500
001 991003215149704436
005 20231005194100.0
008 970703s1997 nyua tb 000 0 eng d
035 ##\$z490706-o1cmu_inst
035 ##\$a(OCOLC)37237058
040 ##\$aPMC \$cPMC
049 ##\$aPMCC
088 ##\$aRC 20742 (91875)
099 ##\$aIBMC 20742
245 00\$aChakra Vyuha (CV) : \$ba sandbox operating system environment for controlled execution of alien code / \$cAsit Dan ... [et al.].
246 13\$aChakraVyuha (CV) : \$ba sandbox operating system environment for controlled execution of alien code
260 ##\$aYorktown Heights, N.Y. : \$bIBM T.J. Watson Research Center, \$c[1997].
300 ##\$a22 p. : \$bill. ; \$c28 cm.
490 1##\$aResearch report RC. International Business Machines Corporation. Research Division ; \$v20742
500 ##\$a"2/20/97."
500 ##\$aCover title.
504 ##\$aIncludes bibliographical references.
520 ##\$aAbstract: "Sharing of unknown programs creates an atmosphere of untrust and hence exacerbates the need to secure information and physical
596 ##\$a9
650 #0\$aClient/server computing.
650 #0\$aCoding theory.
650 #0\$aComputer security.
700 1##\$aDan, Asit.
810 2##\$aInternational Business Machines Corporation. \$bResearch Division. \$tResearch report ; \$vRC 20742.
901 ##\$aocm37237058
910 ##\$ajh/z 6/27/97
916 ##\$a19970703
917 ##\$a19970703
918 ##\$aCATALOGER
919 ##\$a0
945 ##\$he&s-tech rept \$i38482009794286
999 ##\$aIBMC 20742 \$wALPHANUM \$c1 \$i38482009794286 \$d7/3/1997 \$lBY-REQUEST \$mOFFSITE \$rY \$sY \$tTECH-RPT \$u7/3/1997 FORM=MARC

Carnegie Mellon University
Libraries

SEARCH TIPS

BROWSE SEARCH

DATABASES A-Z

JOURNAL SEARCH

COLLECTION DISCOVERY

INTERLIBRARY LOAN

...

SIGN IN

Menu

Search anything

Everything

ADVANCED SEARCH



BOOK

Chakra Vyuha (CV) : a sandbox operating system environment for controlled execution of alien code

Dan, Asit.

1997

Not Available

TOP

Find in Library

FIND IN LIBRARY

LINKS

Please sign in to check if there are any request options.

Sign in

DETAILS

BACK TO LOCATIONS

SEND TO

LOCATION ITEMS

VIRTUAL BROWSE

Offsite Repository

Out of library , BY-REQUEST ; IBMC 20742

(1 copy, 0 available, 1 request)

On hold shelf until 10/23/2023

Sign in for loan information

TOP

FIND IN LIBRARY

LINKS

Report a Problem

Signed in and still can't find what you're looking for? Let us know

SEND TO

Display source record

VIRTUAL BROWSE

Details

Title

Chakra Vyuha (CV) : a sandbox operating system environment for controlled execution of alien code

Creator

Dan, Asit.

Subject

Client/server computing

Coding theory

Computer security

Description

Abstract: "Sharing of unknown programs creates an atmosphere of untrust and hence exacerbates the need to secure information and physical resources from any alien code. A sandbox approach to execution of such foreign code is proposed in this paper. An alien code is trusted with a set of privileges for accessing logical (e.g., callable functions and services) and physical (e.g., rate and maximum consumption amount of CPU, memory, disk space, etc.) resources via a third party authentication. During installation of any code in the sandbox environment, the associated privileges (per user basis) are stored in a secure area and enforced by the Operating System during execution. The alien code may also access resources defined by other subsystems (e.g., database), and hence, in an integrated environment the subsystem specific privileges are transferred securely to the subsystem for monitoring. A prototype version based on Linux OS code has been developed."

Other title

ChakraVyuha (CV) : a sandbox operating system environment for controlled execution of alien code

Series

International Business Machines Corporation. Research Division. Research report ; RC 20742.

Research report RC. International Business Machines Corporation. Research Division ; 20742

TOP

Publisher

Yorktown Heights, N.Y. : IBM T.J. Watson Research Center

Creation Date

1997

Format

22 p. : ill. ; 28 cm.

Bibliography Note

Includes bibliographical references.

General Note

"2/20/97."

Cover title.

Source

Library Catalog

Send to

FIND IN LIBRARY

EXPORT TO EXCEL

LINKS

DETAILS

PRINT

EMAIL

PERMALINK

QR

CITATION

EXPORT RIS

EXPORT BIBTEX

CASTIDID

SEND TO

VIRTUAL BROWSE

Virtual Browse

Deriving texture feature set for content-based retrieval of ...
1997

ForestaPC (Scalable-VLIW) user instruction set architecture ...
1997

Efficient retrieval of composite multimedia objects in the ...
1997

Dynamic load balancing in geographically distributed heterogeneous ...
1997

Chakra Vyuha (CV) : a sandbox operating system ...
1997

Hybrid pyramidal/vector-quantized volume compression ...
1997

A comparison of the applicability of ...
1997

Appendix DAN02

IBMC
20742

RC 20742 (91875) 2/20/97
Computer Science/Mathematics 22 pages

COMPUTER SCIENCE DEPT.
TECHNICAL REPORT FILE

Research Report

*Chakra Vyuha*¹ (CV): A Sandbox Operating System Environment for
Controlled Execution of Alien Code

Asit Dan, Ajay Mohindra, Rajiv Ramaswami, Dinkar Sitaram
IBM Research Division
T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

UNCLASSIFIED
DATE 01-15-2001 BY SP10
PITTSBURGH, PENNSYLVANIA 15211

LIMITED DISTRIBUTION NOTICE

This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g., payment of royalties).

IBM Research Division
Almaden • T.J. Watson • Tokyo • Zurich • Austin

IBMC
20742

ROOM USE ONLY
UNTIL 100151997

1. Client/server computing.
2. Computer security.
3. Coding theory.

*ChakraVyuha*¹ (CV): A Sandbox Operating System Environment for Controlled Execution of Alien Code

Asit Dan, Ajay Mohindra, Rajiv Ramaswami and Dinkar Sitaram

IBM T. J. Watson Research Center

Yorktown Heights, NY 10598

{asit, ajay, rajiv, sitaram}@watson.ibm.com

Abstract

Sharing of unknown programs creates an atmosphere of untrust and hence exacerbates the need to secure information and physical resources from any *alien* code. A sandbox approach to execution of such foreign code is proposed in this paper. An alien code is trusted with a set of privileges for accessing *logical* (e.g., callable functions and services) and *physical* (e.g., rate and maximum consumption amount of CPU, memory, disk space, etc.) resources via a third party authentication. During installation of any code in the sandbox environment, the associated privileges (per user basis) are stored in a secure area and enforced by the Operating System during execution. The alien code may also access resources defined by other subsystems (e.g., database), and hence, in an integrated environment the subsystem specific privileges are transferred securely to the subsystem for monitoring. A prototype version based on Linux OS code has been developed.

1 Introduction

The rapid growth in connectivity and sharing of information via the World-Wide Web has dramatically increased the vulnerability of client machines to *alien* codes. Once the alien code is installed on the client machine, it can run with all the privileges of the invoking user. The invoking user may not be aware of the privileges being used. The alien code can silently obtain unauthorized access to all the system resources (i.e., *logical resources* such as files, network ports) that can be accessed by the invoking user. It may also propagate itself, attempt illegal break-ins or maliciously alter files in the client system [10]. Even if the executing alien code is restricted to accessing authorized logical resources only, it may still engage in a *denial of service* attack by monopolizing *physical resources* such as disk, physical memory, CPU, network bandwidth. For example, the alien code may execute a wasteful tight loop which results in preventing any other program from obtaining fair access to the CPU. It may also create many junk files to fill up the available disk space. If this attack is carried out with sophistication (e.g., the tight loop is executed at times when the alien code is supposed to be inactive) the user may be unable to determine the source of the attack. This problem is critical in multimedia systems, where the system attempts to guarantee Quality of Service (QoS) for all streams by reservation of shared resources. QoS guarantee can be provided to all streams only if all streams adhere to their allocated quota. Deliberate or accidental overuse of a shared resource by a stream can result in a failure to meet QoS guarantees for other streams and hence, interrupted stream delivery. The development and popularity of the Java programming language [5] has made it relatively

¹In Hindu mythology, Mahabharata, *ChakraVyuha* (pronounced as chakra-view-ah) refers to the dynamic formation of multi layer wheel made of shields that is very difficult to penetrate in a battlefield.

easy to transport alien code over the World-Wide Web. However, even in computer systems not connected to the Web, programs can be downloaded from remote sites or installed from diskettes or CDROMs. Hence, the problem of providing a secure environment for the execution of programs exists even in traditional environments.

One solution to the above problem is to restrict via a *reference monitor* [8] the privileges that can be acquired by any alien code and to ensure that it cannot compromise the security of the system during execution. Java *applets*, for example, cannot write files on the host machine, or send messages to nodes other than the machine they were downloaded from [5]. This approach cannot be used in general, since it is difficult to devise a set of restrictions without limiting the usefulness of an alien code. Removing the restrictions (as is done, for example with Java *programs*) re-instates the security problem. A similar security problem exists with importing data for word processors and spreadsheets, where the imported data could be infected with *active content macro viruses*, and these viruses in turn, could try to access system resources without the knowledge of the user. To alleviate the problem, the macros associated with the data need to be monitored. Operating system support is also required to associate varying privileges with different programs of a user, rather than simply with individual users. For example, a multimedia application may require access to video files, and hence, guaranteed access to disks at a specified rate, while a non-multimedia application may only require access to non-video files. Therefore, an important requirement for any solution is that an alien code be granted the privileges necessary for its execution, but not be allowed any unrestricted access to the system.

In this paper, we propose a solution to the above security problem. The solution combines certification of alien code together with explicit granting of privileges for execution. In Section 2, we provide an overview of our approach embodied in the *ChakraVyuha* environment. A detailed discussion of the approach is provided in Section 3. Section 4 describes prototype user environments that can be built on top of *ChakraVyuha*. Section 5 contains our concluding remarks.

1.1 Motivating Examples

We now describe three example application environment scenarios that rely on varying degrees of trusts or use different enforcement processes for verification. These are the special cases of the general approach described in this paper. All three examples are viable applications in the near future.

Intranet environments: A simple security model that relies primarily on trust rather than complex runtime enforcement mechanism [10] may be employed in an intranet environment with inexpensive Network Computers. The network computer environment could be customized for a specific organization allowing execution of only the trusted applications (authorized code with proper certificates) without further monitoring of their execution.

Multimedia support: This requires runtime monitoring of execution for the purpose of controlling resource consumption by any code (alien or local). Typically, the code

would be required to declare its resource needs, and the runtime environment would enforce resource usage accordingly. For example, video conferencing applications require multimedia support and access to physical resources such as the network.

Sandboxed browser environment: A more general environment than the simple trust model is required for the execution of alien code. Different helper applications and/or downloaded components may require different access capabilities to logical and physical resources, and a runtime environment would enable execution of the code in a controlled environment. Untrusted code would be run with limited access privileges. A Java or ActiveX enabled browser is an example of this environment.

1.2 Related Work

Enforcing security for networked computers has been an area of active research for quite some time. Most of the work, however, has focussed on preventing unauthorized accesses to the computer systems from the network using challenge/response passwords and authenticated token schemes such as Kerberos [11]. Other work has focussed on addressing trust issues for applications that can be downloaded from the network. Based on the level of trust placed by the runtime system, the work can be classified into three categories. Systems in the first category do not trust any program, i.e., the runtime system views all programs to be potentially malicious, and takes adequate measures to ensure that the programs do not cause any damage. The KeyKOS [7] system and the Domain and Type Enforcement (DTE) system [12] fall under this category. These systems are pure capability based systems unlike most commercial operating systems (e.g., Unix, Windows NT, etc.). In KeyKOS system, all applications are executed in protected *domains* that preserve system integrity. Similarly, the DTE system uses a runtime database incorporated into the operating system structure to control access of processes to system objects. These systems use statically defined capabilities for enforcing strict security, and thereby making them inflexible in their usage. For example, in DTE any change in application capabilities to be effective requires a complete reboot of the system.

Systems in the second category trust certified authors to write programs that do not exhibit malicious intent. Issues such as trust-worthiness of applications have become important due to the growing popularity of the World-Wide Web and widespread availability of downloadable freeware such as ghostview and mpeg player. The Betsi [10] system and the work on application specific-interpreters [6] fall under this category. The Betsi system provides an infrastructure wherein the software developer can attach a digital certificate to the application code, which the user can then use to verify the authenticity and integrity of the software. The system thus provides a mechanism for users to verify whether any unauthorized modifications have been made to the software by hackers. In the work on application-specific interpreters [6] the authors describe an architecture that allows the software developers to specify the resource and access control requirements of the software. This information is then used to generate an application-specific interpreter which monitors

the accesses of the downloaded applications. Their system architecture also defines system services for access control of the downloaded application.

In general, a higher level interpreter cannot monitor/control fine granularity accesses to various system resources as well as logical and physical resources owned by another subsystem (e.g., database, physical resources of the server node). Without a detailed execution plan corresponding to a subsystem access that enumerates the usage of various sub-system resources (e.g., mapping of an SQL query execution to database records and fields), it is impossible to monitor all accesses. Hence, sub-systems are the best place to monitor/control accesses to resources owned or defined by them.

In the absence of mechanisms for resource access control per application per user, the systems in the third category trust the operating system to provide adequate mechanisms and policies (via a higher level reference monitor) to protect users from suspect programs. The Janus system [4] falls under this category. Janus is a user-level implementation of a "sandbox" architecture that enables a user to restrict an application's access to the operating system resources, and thereby, reducing the security exposure to a malicious application. In a configuration file, the user specifies the access control list for all untrusted applications. Such untrusted applications are then executed in the sandboxed environment that restricts the application's access to operating system resources based on the access control lists specified in the configuration file. The sandbox environment acts as a reference monitor for controlling accesses to system calls. The difference between their scheme and our work lies in the overall design and the scope of resources that are protected. The Janus system is primarily concerned with protecting application's access to the file system, and communication resources. The system does not handle *denial of service* attacks by untrusted applications. Our work provides a generic framework that allows the operating system to authenticate and verify installed applications, dynamically control access to both logical and physical resources, and allow privileges associated with an application to be transferred to any other application and/or subsystem.

2 Overview of the Approach

Our approach incorporates three major items:

- specifying in a flexible manner in a *resource capability list* (RCL) the set of permissions and resource access privileges required by the code,

- having the code and its RCL authenticated by a trusted third party, before a client receives them, and finally,

- enabling the client system to agree and allocate all or a subset of the permissions and resources specified in the RCL and enforcing these capabilities, i.e., ensuring that the permissions are not violated or resource consumptions have not exceeded specified limits.

The overall system that implements this approach is shown in Fig. 1. The system consists of one or more code production systems, certification agencies, servers and clients. A code production system communicates with a certification agency, which is a trusted third party. The certification agency issues a certificate for the code identifying its source and a certificate for the RCL of that code. The separate certificates for the code and the RCL allows the two entities to be dealt separately, i.e., they could be downloaded separately. Once the certificate is issued, it is not possible for any party to modify the code or RCL without invalidating the certificate. The code and its RCL are stored on a server along with their certificates. Note that while the above figure describes the most general framework, all of the components need not be present in all implementations. For example, the RCL may not be defined in advance for all alien codes. In this case, the RCL may be the default RCL associated with the sandbox environment in the client system, or it may be defined on-the-fly during installation at the client system. Further details on RCL definition are provided in Section 4.

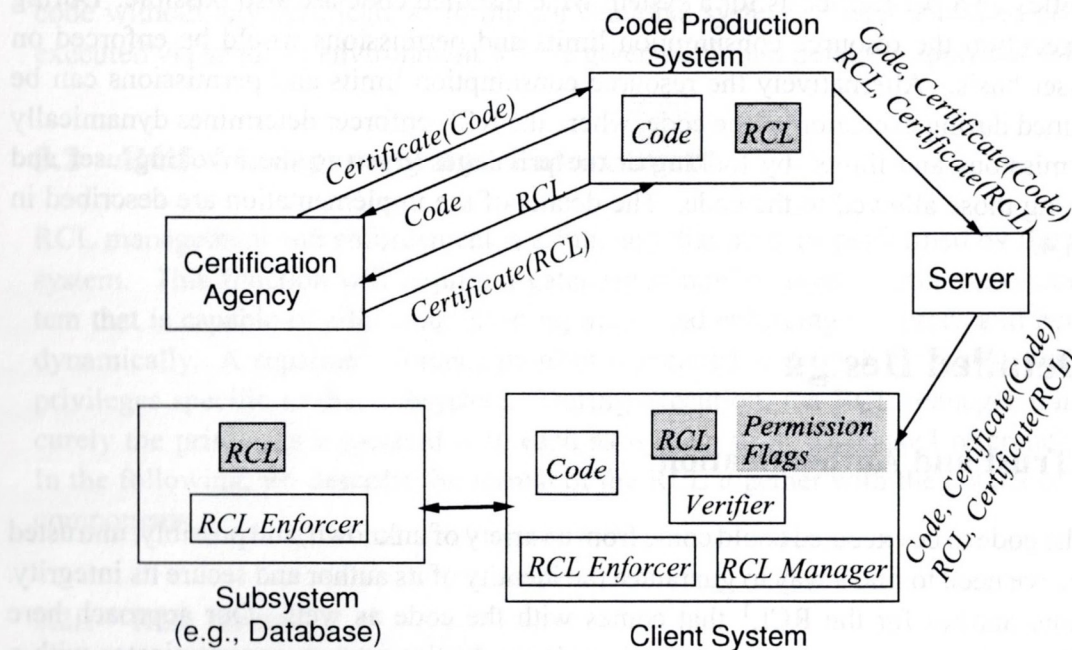


Figure 1: A. Block diagram of the proposed system.

The main focus of this paper is on the client system. Within the client system, the functions described below can be implemented either within the operating system kernel or as another layer on top of the operating system (see, Figure 1). The shaded areas represent entities such as code, RCL or environment specific permission flags. The system monitor layer (i.e., the CV layer) consists of a verifier of code certification, RCL manager that maintains RCLs and a RCL enforcer that performs a “gatekeeping” function, i.e., controls and monitors access to system resources. When a client receives the code and its RCL, it invokes the verifier to verify that the code and RCL are valid (not tampered with). Once the code is verified, the RCL manager is responsible for determining which subset of parameters

within the RCL are to be allowed. It then stores the code and its modified RCL in a secure location. The RCL manager can also be invoked subsequently by a privileged user to alter these capabilities. The RCL enforcer is invoked when the code is executed. It continues to monitor accesses by the code to ensure that the permissions and resource consumption limits are not violated. The subsystems may have their own set of resources to protect. Hence, the RCL manager is also responsible for transferring securely to other subsystems relevant capabilities (e.g., data access capabilities relevant to the database subsystem).

It is possible for different users to be allocated different types and/or amounts of resources and permissions in the recipient system for the same code. This can be done at the time the code is installed by the RCL manager. (The issues on ease-of-use are addressed in Section 4.) The RCL manager looks at the privileges given to different users and combines (i.e., intersects) those capabilities with the resource access privileges and permissions allowed to the code. In this case the set of resource access privileges and permissions for each user would have to be stored separately. Additional enhancements or reductions of capabilities on a per-user basis for a system wide installed code are also possible. During code execution the resource consumption limits and permissions would be enforced on a per-user basis. Alternatively the resource consumption limits and permissions can be determined during execution of the code, where the RCL enforcer determines dynamically the permissions and limits, by looking at the privileges given to the invoking user and combining those allowed to the code. The details of the implementation are described in Section 4.

3 Detailed Design

3.1 Trust and Authentication

Since the code to be executed could come from a variety of unknown, and possibly, untrusted sources, we need to find a way to guarantee the identity of its author and secure its integrity. The same applies for the RCL¹ that comes with the code as well. Our approach here is similar to the one proposed in [10]. A code production system communicates with a Certification Agency (CA), which is a trusted third party. The CA issues a certificate for the code and a certificate for the resource list of that code. Once the certificate has been issued, it is not possible for any party to modify the code or resource list without invalidating the certificate. The code and its RCL, along with their certificates are stored on a server. A client downloading the code or access list can verify the integrity of the code/resource list.

The CA provides a public key K that is widely available and known to the client system. In addition, the CA has a private key P . To provide a certificate for the code, the CA creates

¹Here, the RCL is the declaration of permissions and physical resources requested by this code for its execution. Alternative interpretations are possible for a RCL associated with a downloaded code, where it may represent already granted capabilities.

a certificate containing the code name and the cryptographic hash of the code and signs it using its private key. The certificate cannot now be changed without invalidating the CA's signature. Similarly the CA creates and signs another certificate for the RCL associated with that code (if desired there could be a single certificate for both the code and its RCL).

Upon obtaining the code/RCL and its certificate, the verifier in the client system first checks to see if the CA's signature on the certificate is valid (using CA's known public key). It then computes the cryptographic hash of the code/RCL and verifies that it matches that in the certificate. If the signature is not valid, (i.e., the hash does not match,) the code and RCL are rejected.

Once the code and its RCL have been verified, the client system may elect to allow certain privileged users to modify the code or the RCL (See, Section 4 for further details). However other users should not be allowed to modify them. Thus the RCL and permission flags must be stored in a secure area; reading or updating this area is itself a privilege enforced by the RCL enforcer. Finally, CV allows downloading (or installing from CDROM) of any code without any certificate as in the conventional systems. Such untrusted codes when executed via sandbox environment will be given minimum default resource capabilities.

3.2 RCL Management and Enforcement

RCL management and enforcement is a function that must be performed by the operating system. This function will require a gatekeeper/monitor layer within the operating system that is capable of allocating, keeping track, and enforcing resources and permissions dynamically. A separate reference monitor associated with each subsystem will enforce privileges specific to that subsystem. During execution, the RCL manager transfers securely the privileges associated with each subsystem to its associated reference monitor. In the following, we describe the format of the RCL together with the details of the other components.

3.2.1 RCL format

The RCL is divided into several categories. Apart from the capabilities related to the system resources, there may be various other subsystem related capabilities. The capabilities for the system resources can be divided into two parts: (a) the capabilities for the physical resources specifying the required physical resources and (b) those for the logical resources (e.g., files, network ports) that need to be accessed. Each item in the physical resource list contains four fields. The first field is the name of the physical component (e.g. disk, physical memory, CPU). The attribute field specifies the actual resource in the case that there is more than one resource associated with a physical component (e.g. space or bandwidth in the case of disk). The other two fields specify the maximum allowable consumption and consumption rate of the resource, respectively. Note that while some resources refer to the same amount in all machines (e.g., disk space), specification of other resources (e.g.,

required CPU MIPS) may be machine dependent. Either the amount needs to be expressed for a reference machine, and translated for the client machine during code installation, or the amount should be expressed in a machine independent manner (e.g., Bandwidth or data movement per unit time).

Logical Resources

Syntax:

```

<Statement> ::= <Logical resource name> <Parameter Definition> <Range List>
<Logical resource name> ::= <Package name> <Function name>
<Parameter definition> ::= <Parameter type>,...
<Parameter type> ::= INT | STR
<Range list> ::= | <ParmSet range>, ... |
<ParmSet range> ::= {<Parm range>, ...}
<Parm range> ::= <Integer range> | <String range> |
<Integer range> ::= <Integer> | <Integer> - <Integer> |
                    <Integer>, <Integer range> |
                    <Integer>-<Integer>, <Integer range>
<Integer> ::= <Simple integer> | <Environment variable>
<String range> ::= <Environment variable>, <String range> |
                    <String>, <String range>
                    <epsilon>

```

Figure 2: BNF for the language used to specify the logical resource list.

Access to logical resources, or client system facilities, is controlled by specifying the list of functions external to the program that may be invoked, together with restrictions on the allowable parameter combinations. For example, restrictions on the files that can be accessed in read-write mode can be enforced by specifying the allowable values for the filename parameter in the open call when the open mode is read-write. In many situations, it may be necessary to allow access to resources whose name is client-system dependent. For example, a database query program may need access to a database, whose name may vary in different client systems or between different users on the same system. This is handled by allowing the use of environment variables whose values can be determined during execution. Changing of such environment variables is itself a privileged operation.

Figure 2 shows, in BNF, the syntax of the language used to specify the logical resource list. Informally, each statement in the language specifies a valid combination of parameters for a particular function. The statement consists of the function name, followed by the types of the parameters and the range of allowable values for each parameter. The function name is a combination of package name and the actual function name to handle the case where the same actual function name (e.g. open) is used by multiple subsystems. The parameter types may be integer or string, denoted by INT or STR, respectively. For integer parameters, the allowable values are specified as a range of integers. For string parameters, regular expressions (in the Unix sense) are used to specify the allowable values. Environment variables are specified by prefixing the name of the variable with \$; for example \$MAXINT

may specify the maximum integer value on the system and \$USER_DB may specify the name of a user-specific database.

Note that the logical resource list should not include specifications of the type "access all files except <file-list1>". Specifications of this form implicitly assume that the only files to which access needs to be restricted is <file-list1>. This approach is adequate if the only files that need to be protected are a well-known set of system files that are the same on every system. In general, however, the files that need to be protected on client systems may vary. Hence, it is safer for programs to explicitly specify the resources needed. However, set difference may provide ease of expression as shown in Section 4 (See, Figure 11).

The representation of subsystem capability will be specific for that subsystem, and is beyond the scope of the current paper.

3.2.2 RCL Enforcement

The RCL enforcer module at the client ensures that the permissions and resource consumption limits specified in the RCL are not violated. The module for enforcing system resource capabilities monitors two types of resources: physical and logical. In this section, we describe the algorithms and data structures used by the RCL enforcer for these two types of resources.

The RCL enforcer protects physical resources such as memory and CPU usage, disk storage, and disk bandwidth utilization. Access control information about physical and logical resources of the process executing an alien application code is maintained in data structures called the *Runtime Physical Resources Table (RPRT)* and the *Runtime Logical Resources Table (RLRT)*. For easy access to this information at runtime, pointers to these tables exist in the *u-area* of the process (see Figure 3). When an application is initially loaded for execution, the RPRT and RLRT data structures are initialized from the information specified in the RCL configuration file associated with this application. In the next two subsections, we describe in detail the RPRT and RLRT data structures and their use in monitoring of resource accesses.

3.3 Physical Resources

The RPRT data structure contains a row for each resource access capability (see Figure 4). The *current consumption rate* and *current usage* fields are used to keep track of the current consumption rate and the total resource consumption so far, respectively, of the code during run time. The RCL enforcer also keeps track of the code start time, *CodeStartTime*.

During process execution, the RCL enforcer references and updates the information stored in the RPRT data structure using the algorithm outlined in Figure 5 to allow or deny the request for accessing physical resources. Two parameters, the amount of requested resource, *REQAMT*, and the estimated time of consumption, *COMPT*, are passed for each

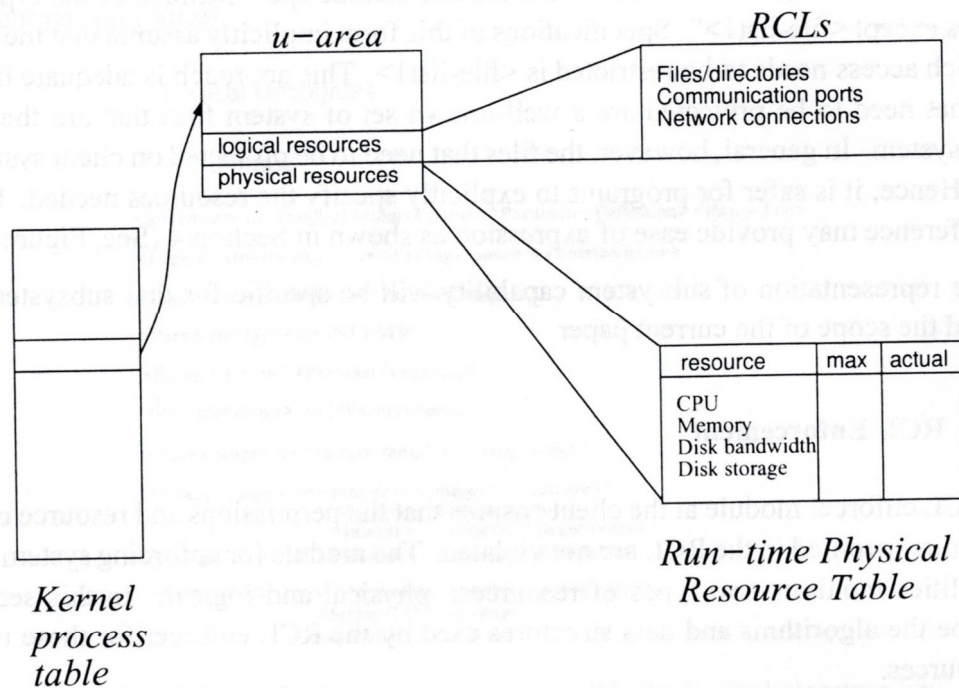


Figure 3: Data structures used by the RCL enforcer.

Physical Resource name	Physical Resource attribute	Maximum consump. rate	Maximum usage	Current consump. rate	Current usage
Disk	Space		100 MB		y1
Disk	I/O	100/s	10 MB	x2	y2
Physical Memory			8MB		y3
CPU	50 Sec	50 MIPS	1 sec	x4	y4

Figure 4: Runtime Physical Resource Table

access request for a physical resource. For disk I/O, the *REQAMT* is the amount of disk I/O and the *COMPT* is the estimated time for the I/O to complete. The RCL enforcer then locates the row in the RPRT for this resource and checks whether the maximum consumption amount, *MaximumUsage*, is specified, and if the $(CurrentUsage + REQAMT)$ exceeds the *MaximumUsage*. If so, it returns a failure indicating that the consumption is not allowed. Alternatively, if it does not exceed the *MaximumUsage*, the RCL enforcer computes the projected consumption rate, *NewConsumptionRate*, of this resource as $(CurrentUsage + REQAMT) / (CurrentTime + COMPT - CodeStartTime)$. The RCL enforcer then checks to see if the *MaximumConsumptionRate* is specified and if the *NewConsumptionRate* exceeds *MaximumConsumptionRate*. If the *MaximumConsumptionRate* is not specified or the projected consumption rate is not greater, the RCL enforcer returns with an indication that the consumption is allowed. If the projected consumption rate exceeds its limit, the RCL enforcer computes the required delay before this operation is allowed as $(CurrentUsage + REQAMT) / MaximumConsumptionRate - (CurrentTime + COMPT - CodeStartTime)$ and returns the required delay with an indication that the consumption has to be delayed for the computed delay.

CheckConsumptionRequest(){

 If ((*REQAMT* + *CurrentUsage*) > *MaximumUsage*))
 return(FAILURE);

 Compute *NewConsumptionRate* for this request;

 If (*NewConsumptionRate* > *MaximumConsumptionRate*) {

 Compute *MinimumDelay* before consumption;
 return(*MinimumDelay*);

 }

 return(SUCCESS);

}

UpdateCurrentUsage(){

CurrentUsage = *CurrentUsage* + *REQAMT* ;

}

Figure 5: Algorithm used by the RCL enforcer to manage physical resources.

Once the consumption request is allowed, it is put on the queue for that resource. If the resource is currently available, the request is serviced. Otherwise, the resource scheduling policy (as in traditional systems) prioritizes these allowable requests. For example, the multimedia requests may be given a higher priority to guarantee QoS [9].

After the consumption of the resource, the RCL enforcer is invoked again with a parameter specifying the amount of resource consumed, *CONSAMT*. The RCL enforcer then updates the actual resource consumption, *CurrentUsage*.

3.3.1 Enforcement of Limits for Child Processes

It is critical that any secure system enforces fairly the limits for all processes including child processes. There are various alternatives that can be used. First, each child process may be assigned the RCLs and resource consumption limits of the parent process. While the policy is simple to implement, and enforces logical accesses correctly, it creates a caveat for bypassing resource limits. A process can fork many child processes for circumventing these resource limits. The second alternative is to use the limits of the parent process as the maximum capability of the process and all its children, i.e., the sum total of capabilities of the parent process and all its children cannot exceed the original limit of the parent process. For ease of tracking, and for efficiency, each child process may be assigned a fraction of the resources available to the parent process. The allocation policy decides how to divide the available resources amongst the forked processes. In our implementation, we chose the second alternative. The RCL enforcer equally divides the limits of the parent process among all its children, and checks against this limit at runtime.

3.4 Logical Resources

Similar to the RPRT, access control information for the logical resources is maintained in a data structure called the Runtime Logical Resources Table (RLRT). The logical resources constitute file, directories, communication ports and various other system service calls. The RLRT data structure is also initialized when an application is loaded into memory. The RLRT contains a row for each system service access capabilities. Each row has a flag that indicates whether this service call is allowed *always*, *never* or needs to be *verified* (*ALWAYS*, *NEVER* and *VERIFY*, respectively). If the system call needs to be verified, then a list is provided that contains both the allowable and non-allowable ranges of parameter combinations (See Figure 6).

3.4.1 Enforcement of Access Control to Logical Resources

The RCL enforcer monitors all systems calls, i.e., access requests to logical resources in the system. This monitoring requires that the RCL enforcer be invoked for each system call made by an application. The RCL enforcer then validates the parameters of the system call against those specified in the RLRT data structures and either allows or disallows the call from continuing.

The RCL enforcer references the information stored in the RLRT data structure using the algorithm outlined in Figure 7 to allow or deny the request for accessing logical resources. During each system call made by the application code, the RCL enforcer locates the number of parameters, their values, and the name of the system call being invoked. The exact method for doing this is implementation dependent (discussed later); for example, in Java, these are located on the operand stack. The RCL enforcer then locates the row for

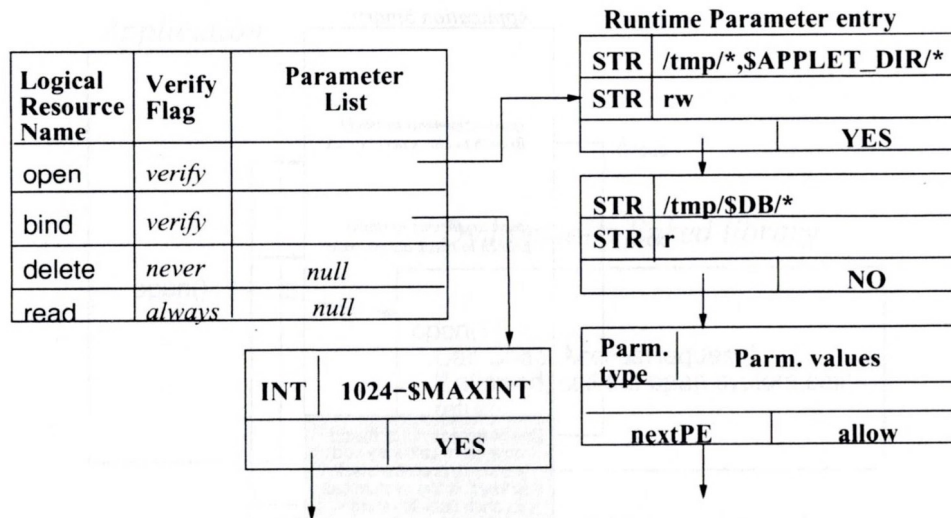


Figure 6: Runtime Logical Resource Table

this function in the RLRT and executes the algorithm in Figure 7. If the given parameter combination for this call does not lie in any of the allowable ranges then the call is not allowed. Otherwise, the *AllowFlag* is first set to *YES* and subsequently, further checked to see if the parameter combination lies in a non-allowable subrange (within the allowable range). The call is allowed only if no such subrange is found.

CheckAccessRequest(){

if (VerifyFlag == ALWAYS) return (YES);

if (VerifyFlag == NEVER) return (NO);

Set AllowFlag to NO ;

If found(allowable parameter range entry that includes the current parameter combination) then

Set AllowFlag to YES ;

If found(non-allowable parameter range entry that includes the current parameter combination) then

Set AllowFlag to NO ;

Return(AllowFlag);

}

Figure 7: Algorithm used by the RCL enforcer to manage logical resources.

3.4.2 Implementation Choices for Monitoring Accesses to System Resources

We now describe three different approaches to monitor systems calls:

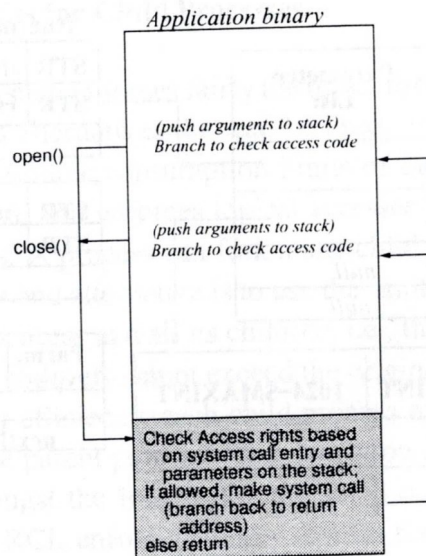


Figure 8: Preprocessor for modifying an application's binary code.

- Binary code modification:** — In this approach, the downloaded application code is preprocessed and modified to insert hooks in the binary code for calling RCL enforcer (i.e., `CheckAccessRequest()` function. See, Figure 7 and 8). Note that all the required parameters for system calls as well as the return address are already placed on the stack. The `CheckAccessRequest()` code verifies these parameters and then branches to the system call routine. At the end of the system call it returns to the original location in the main code. This approach does not require any modifications to the operating system in order to implement any security checks. On the downside, the user has to preprocess all untrusted applications. This may incur a lot of overhead. Also, this is not a full proof solution, since a program may use encryption to hide system calls in its code.
- Modification of service libraries:** — The idea here is to provide on each machine a duplicate set of libraries for various services that may make system calls (e.g., file system, language support, etc.). The new set of libraries calls the `CheckAccessRequest()` routine to verify the parameters before making any system call. Figure 9 shows the structure of the modified system call library approach. The advantage of such a scheme is that no changes to the operating system are required. On the downside, the system cannot enforce security for applications that have system call libraries statically linked into the application.
- System call modification:** — In this approach, the operating system's call interface is modified to call the `CheckAccessRequest()` upon entry to verify the parameters of the system call. Thus, regardless of how the applications are structured, the system call entry point in the operating systems will ensure consistent security checks for all applications that are executed on the machine. The disadvantage of this scheme

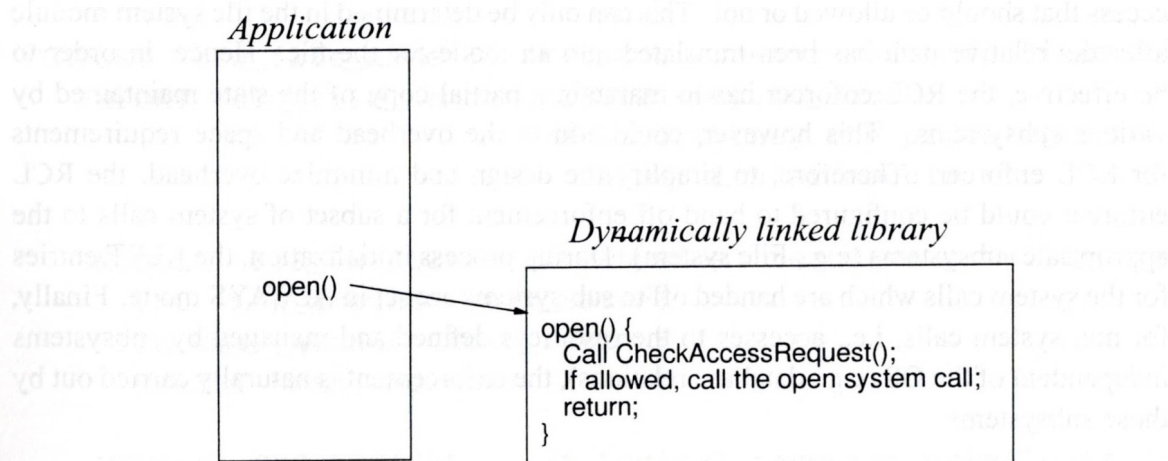


Figure 9: The modified system call library approach.

is that the operating system needs to be modified. We selected this approach to manage logical resources in the system primarily because of the consistency it offers for enforcing security. Figure 10 shows the structure of this approach.

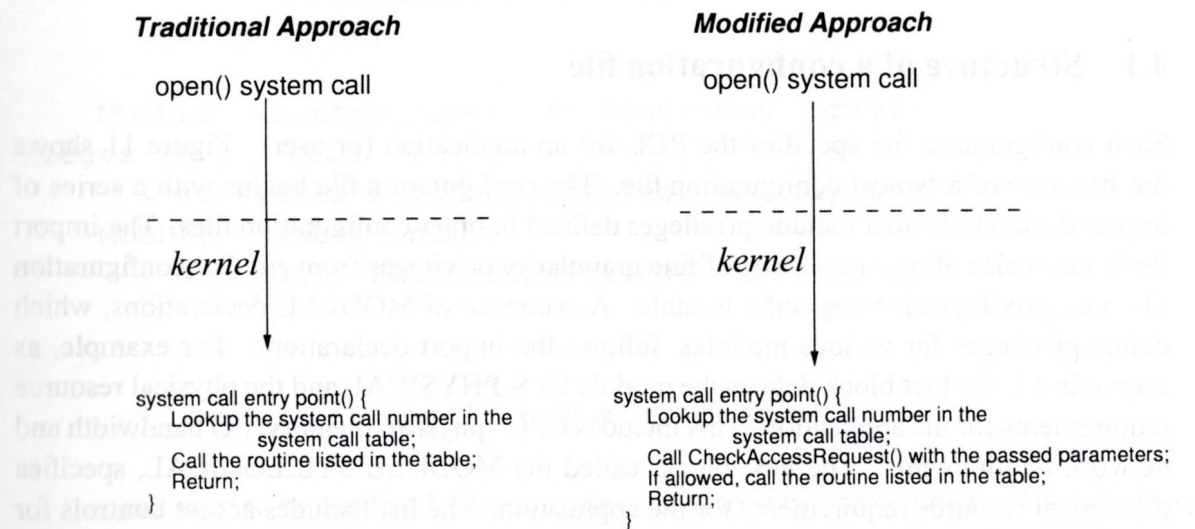


Figure 10: The modified operating system call interface approach.

3.4.3 RCL Enforcement by Subsystems

The common entry point to the Kernel for all system calls provides an effective mechanism for enforcing access control for logical resources. However, enough information may not be available to adequately enforce security at the time a system call is being made. For example, in case of filesystems, it is very difficult to resolve at the system call layer whether a request for a file specified using relative paths (e.g., `../foo`) translates to an

access that should be allowed or not. This can only be determined in the file system module after the relative path has been translated into an inode for the file. Hence, in order to be effective, the RCL enforcer has to maintain a partial copy of the state maintained by various subsystems. This however, could add to the overhead and space requirements for RCL enforcer. Therefore, to simplify the design and minimize overhead, the RCL enforcer could be configured to hand off enforcement for a subset of system calls to the appropriate subsystems (e.g., File system). During process initialization, the RLRT entries for the system calls which are handed off to sub systems are set in ALWAYS mode. Finally, for non-system calls, i.e., accesses to the resources defined and managed by subsystems independent of the OS (e.g., database relations), the enforcement is naturally carried out by those subsystems.

4 Installation and Use of Applications under ChakraVyuha

We first describe the structure of a configuration file, and various types of default (e.g., system wide, per user) and application specific configuration files used by the CV. We will then detail the installation process of untrusted applications.

4.1 Structure of a configuration file

Each configuration file specifies the RCL for an application (or user). Figure 11 shows the structure of a typical configuration file. The configuration file begins with a series of import declarations that include privileges defined in other configuration files. The import declaration also allows importing of fine granularity privileges from another configuration file, i.e., privileges for a specific module. A sequence of MODULE declarations, which define privileges for various modules, follows the import declarations. For example, as shown in 11, the first block defines the module SYS_PHYSICAL and the physical resource requirements for the application. This includes CPU, physical memory, I/O bandwidth and network requirements. The next block, called the MODULE SYS_LOGICAL, specifies the logical resource requirements for the application. The list includes access controls for various system calls allowed to this application. The keywords reserved for defining this module are as follows:

- **ALWAYS** – Always allow the application to make the specific system call without any need for verification.
- **NEVER** – Never allow the application to make the specific system call.
- **VERIFY** – Verify the arguments before making the specific system call. The keywords following the verify reserved word describes how the privileges for the system call are specified. For example, the figure shows that arguments for the open system


```

IMPORT <cfg_file_name>; /* import one or more cfg files */
IMPORT <cfg_file_name>::MODULE <module_name>;
                                /* import only the privileges of
                                a particular module */

MODULE SYS_PHYSICAL /* Physical resource privileges */
    CPU <max_amount> <max_rate>;
    IO <max_amount> <max_rate> ;
    MEMORY <max_amount>;

MODULE SYS_LOGICAL /* Logical resource privileges */
    exit ALWAYS; /* Always allowed */
    setuid NEVER; /* Never allowed */
    open VERIFY STR, INT, INT /* verify calls */
        [{" /tmp/*", *, *} - {" /tmp/admin/*", *, *}],
        [{" /foo*", {0, 1, 2}, 2..3} -
{" /fooblitz", {1, 2}, 2..3} -
        {" /fookazam", *, *}];
    unlink NEVER;

MODULE <subsys_name> /* Sybsystem privi-
leges (e.g., DB2) */
    < subsystem related privilege description>
MODULE <subsys_name>
    < subsystem related privilege description>

```

Figure 11: A sample configuration file.

call need to be verified before allowing the system call to complete. The privileges associated with the open system call are described in the form of a *<string, int, int>* corresponding to the types of the parameters that the open system call takes, i.e., *pathname, flags, mode*. Following this syntax specification, a list of comma-separated entries are described which specify the application privileges for an open call to files in various directories. The example shows that the open call should be allowed in any mode for all files in the /tmp directory and not be allowed for any files in the /tmp/admin subdirectory. The privileges associated with a system call is a union of all the privileges in the comma-separated list.

The application writer can also specify privileges for various subsystems that the application would use. For example, if an application communicates with a database subsystem

such as DB2 then the application writer can specify the requirements for that subsystem using the MODULE DB2 declaration.

4.2 Types of configuration files

The different types of configuration files used in ChakraVyuha are described below. The first two configuration files (*cvmin.cfg* and *cvmax.cfg*) relate to a specific user environment, and describe the bounds on access privileges allowed to any code execution instantiated by this user. The configuration files (*sysdefault.cfg* and *userapp.cfg*) refer to the specific privileges granted to a specific application. Finally, the user *cvrc* file allows an user to specify to the CV use of a different configuration file for that instantiation.

- User Min Configuration File: The User Min Configuration File (hereafter abbreviated as *cvmin.cfg*) specifies the maximum resource consumption of a process running in a ChakraVyuha sandbox for which a configuration file could not be located. Ideally, this file will never be used since proper configuration files would always be in place. The user may customize this file. The location of *cvmin.cfg* file is illustrated in Figure 12 (see, for instance, */cv/config/u/susan/cvmin.cfg*).

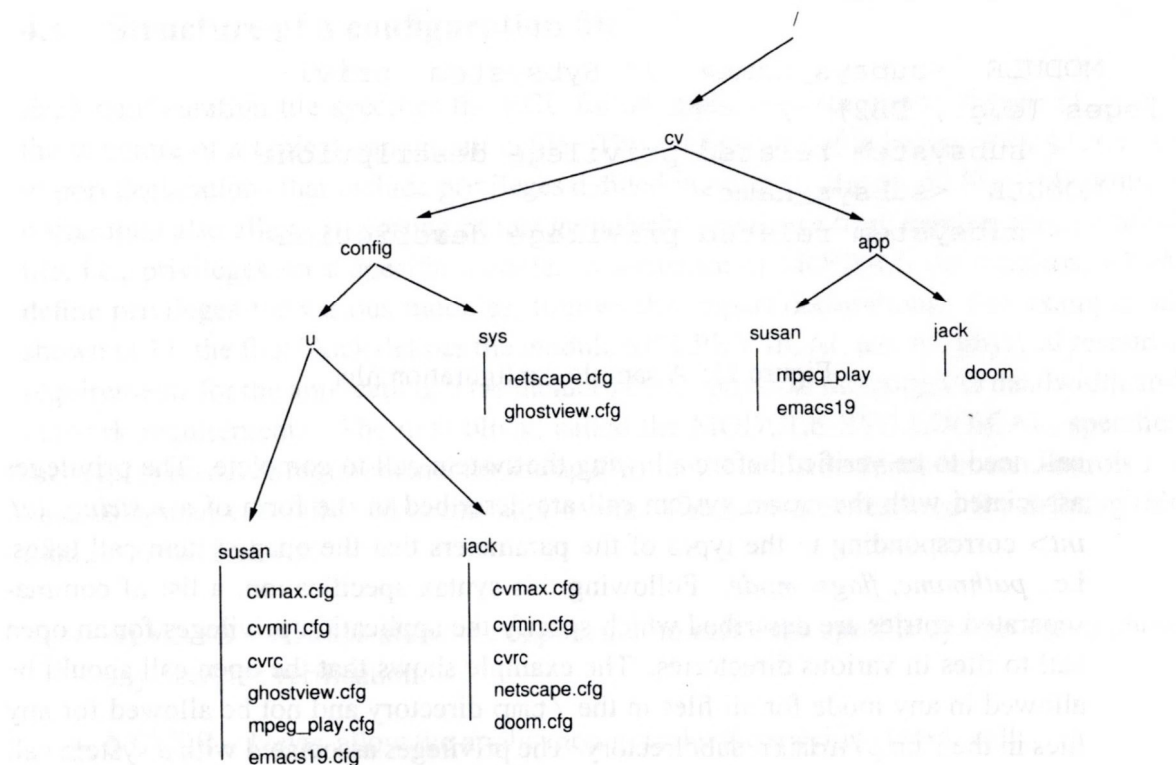


Figure 12: Configuration file hierarchy.

- **User Max Configuration File:** The User Max Configuration File (hereafter abbreviated as *cvmax.cfg*) specifies the maximum resource consumption (e.g., CPU rate) of any process instantiated by the user. The parameters in *cvmax.cfg* are set by the system administrator and cannot be modified by the user. In Figure 12, */cv/config/u/susan/cvmax.cfg* is a *cvmax.cfg* file.
- **System Default Application Configuration File:** The system default application configuration file (hereafter, abbreviated as *sysdefault.cfg*) specifies system defaults for an application. It is generated during installation from the application RCL. The *sysdefault.cfg* file is optional and is provided to facilitate the use of the CV. For example, in Figure 12, */cv/config/sys/netcape.cfg* is a *sysdefault.cfg* file for the Netscape browser.
- **User Application Configuration File:** The user application configuration file (hereafter abbreviated as *userapp.cfg*) allows the user to customize the sandbox. A *userapp.cfg* will *replace* a *sysdefault.cfg*. That is, if a *userapp.cfg* is specified when the sandbox is invoked, the *sysdefault.cfg* will be disregarded. However, if the user wishes to build upon the restrictions specified in the *sysdefault.cfg*, the user can import the *sysdefault.cfg* and specify enhancements to and overrides of its specifications on the granularity of each resource.

An example snippet of a *userapp.cfg* file follows. In the directory structure illustrated in Figure 12, this file would be */cv/config/u/susan/ghostview.cfg*.

```
# Snippets of Susan's ghostview.cfg file
#
# The following line imports the system default
# ghostview.cfg file.
import /cv/config/sys/ghostview.cfg

MODULE SYS_LOGICAL
# The following line enhances the restrictions on
# the open system call.
open ENHANCE VERIFY STR, INT, INT
    [ - {''~/Private/''', *, *} ]

# The following line overrides the restrictions on
# the unlink system call.
unlink OVERRIDE VERIFY STR [ {''/tmp/'''} ]
```

- **User cvrc File:** The User cvrc File allows an user to specify CV parameters without retyping them at every CV invocation. It is similar in nature to a *.cshrc* file or a *.Xdefaults* file. An example snippet of a *cvrc* file follows. In the directory structure illustrated in Figure 12, this file would be */cv/config/u/susan/cvrc*.

```
# Snippets of Susan's cvrc file
#
# The following option requires that a user app
# config file be processed before the CV sandbox
# may be constructed.

# The following line specifies the pathname for the
# user app config file to be used for emacs19.
emacs19*UserAppCfg: /cv/config/u/susan/emacs19.cfg

# The following line specifies that Susan wants to use
# Jack's config file whenever she plays Doom.
doom*UserAppCfg: /cv/config/u/jack/doom.cfg
```

4.3 Installation of Untrusted Applications

For effective enforcement of security, all untrusted applications are installed in the system via an uniform installation process. An utility called the InstallCop is used for this purpose. The steps involved in the installation process are as follows:

1. The application and RCL are downloaded to the host (via network, diskette, CD-ROM, satellite, etc.).
2. The InstallCop is invoked upon receiving the application and RCL.
3. The InstallCop authenticates the executable and RCL.
4. The InstallCop then generates a system default configuration file from the authenticated RCL² if the installation is made system wide. Otherwise, a configuration file specific to the installing user will be generated for this application.
5. The InstallCop names the configuration file <application name>.cfg.
6. In case of system wide installation (by the superuser), the InstallCop places the configuration file in the /cv/config/sys directory. Otherwise the file is placed in the /cv/config/sys/<username> directory. If there is a name conflict, the installer is prompted by InstallCop for a new application name. For instance, if a new version of emacs is installed, it can be called emacs2 or emacsnew. Alternatively, the obsolete version of emacs can be reinstalled as, for example, emacs.old. Note that we do not use the complete path name for identifying an application. By taking this approach, we allow the flexibility for an application

²Optionally, the InstallCop can allow the system administrator to edit the system default configuration file before the application is made available to users.

to move from one directory to another without having to completely reinstall the application.

7. For system wide installation, the InstallCop prompts the installer for an installation directory. The installation directory must be writable only by the superuser (e.g. /usr/bin or /usr/local/bin, etc). For installation by an user, either the file may already exist in the user specified directory, or it may be copied to the user specified directory at this time.
8. The application is placed in the installation directory. If there is a name conflict, the installer is prompted by the InstallCop for a new application name. The config file just created must also be renamed. The configuration file specifies the physical and logical resource requirements as well as the RCLs related to other subsystems. The configuration file is stored in a hidden directory on the system, whose location is known only to the InstallCop and the "sandbox" program. As described earlier, this file is used by the sandbox program to read in the physical and logical resource requirements and limits for this application.

5 Summary and Conclusion

Sharing of unknown programs creates an atmosphere of untrust and hence, exacerbates the need to secure information and physical resources from *alien* codes. Once an alien code is installed on a client machine, it can run with all the privileges of the invoking user, without any notification to the user of the privileges being used. Hence, the alien code can silently obtain unauthorized accesses to all the system resources (e.g. files, network ports) that can be accessed by the invoking user. The alien code may propagate itself, attempt illegal break-ins or maliciously alter files. It may also engage in a *denial of service* attack by monopolizing various *physical resources* (e.g. disk, physical memory, CPU).

A sandbox approach to execution of such foreign codes is proposed in this paper. Any alien code is trusted with a set of privileges for accessing various resources via a third party authentication. The list of resources may include *logical* system resources (e.g., callable functions and services), *physical* resources (e.g., rate and maximum consumption amount of CPU, memory, disk space, etc.) as well as various resources defined or owned by other subsystems (e.g., database relations). During installation of an alien code in the sandbox environment, the associated privileges (per user per code basis) are stored in a secure area and enforced by the OS and various subsystem monitors during execution.

A prototype version based on Linux OS code has been developed.

Acknowledgments: The authors would like to thank David Dion of the University of Washington, Seattle for participating in the discussion of user environment issues, and in the development of prototype based on Linux. We would also like to thank David Safford for his comments on the earlier revisions of the paper.

References

- [1] Anderson, D. P., "Metascheduling for Continuous Media," *ACM Transactions on Computer Systems*, Vol. 11, No. 3, August 1993, pp. 226–252.
- [2] Benantar, M., R. Guski, K. M. Troidle, "Access Control Systems: From Host-centric to Network-centric Computing", *IBM Systems Journal*, Vol. 35, No. 1, 1996, pp. 94–112.
- [3] Dion, D., A. Dan, A. Mohindra, D. Sitaram, "ChakraVyuha: Design and Implementation ", *IBM Research Report*, (in preparation).
- [4] Goldberg, I., Wagner, D., Thomas, R., and Brewer, A. B., "A Secure Environment for untrusted helper applications," *To appear in Proceedings of the 6th USENIX UNIX Security Symposium*.
- [5] van Hoff A., S. Shaio and O. Starbuck (eds), "Hooked on Java", Addison-Wesley, 1996.
- [6] Jaeger, T., Rubin, A. D., and Prakash, A., "Building Systems That Flexibly Control Downloadable Executable Content," *To appear in Proceedings of the 6th USENIX UNIX Security Symposium*.
- [7] Key Logic., "KeyKOS Architecture,"
<http://www.agorics.com/agorics/KeyKos/architecture/Welcome.html>.
- [8] Lampson, B. W., "Protection", *Proc. of the fifth Princeton Symposium on Information Sciences and Systems*, 1971.
- [9] Ramakrishnan, K. K., L. Vaitzblit, C. Gray, U. Vahalia, D. Ting, P. Tzelnic, S. Glaser, W. Duso "Operating System Support for a Video-On-Demand File Service." *Multimedia Systems*, Vol. 3 No. 2, May 1995, pp.53–65.
- [10] Rubin, Aviel. D., "Trusted Distributed of Software Over the Internet," *Symposium on Network and Distributed System Security*, February 1995, pp. 4 7–53.
- [11] Steiner, J. G., Neuman, C., and Schiller, J. I., "Kerberos: An authentication service for open network systems," *USENIX Winter Conference Proceedings*, February 1988.
- [12] Walker, K. M., Sterne, D. F., Badger, M. L., Petkac, M. J., Sherman, D. L., Oostendorp, K. A., "Confining Root Programs with Domain and Type Enforcement," *6th USENIX Security Symposium*, July 1996.


JUL 22 1997










Copies may be requested from:


IBM Thomas J. Watson Research Center
Publications Office, 16-220
Post Office Box 218
Yorktown Heights, NY 10598

Appendix CMU01





SIGN UP | LOG IN

UPLOAD

ABOUTBLOGPROJECTSHelpDONATECONTACTJOBSVOLUNTEERPEOPLE



INTERNET ARCHIVE

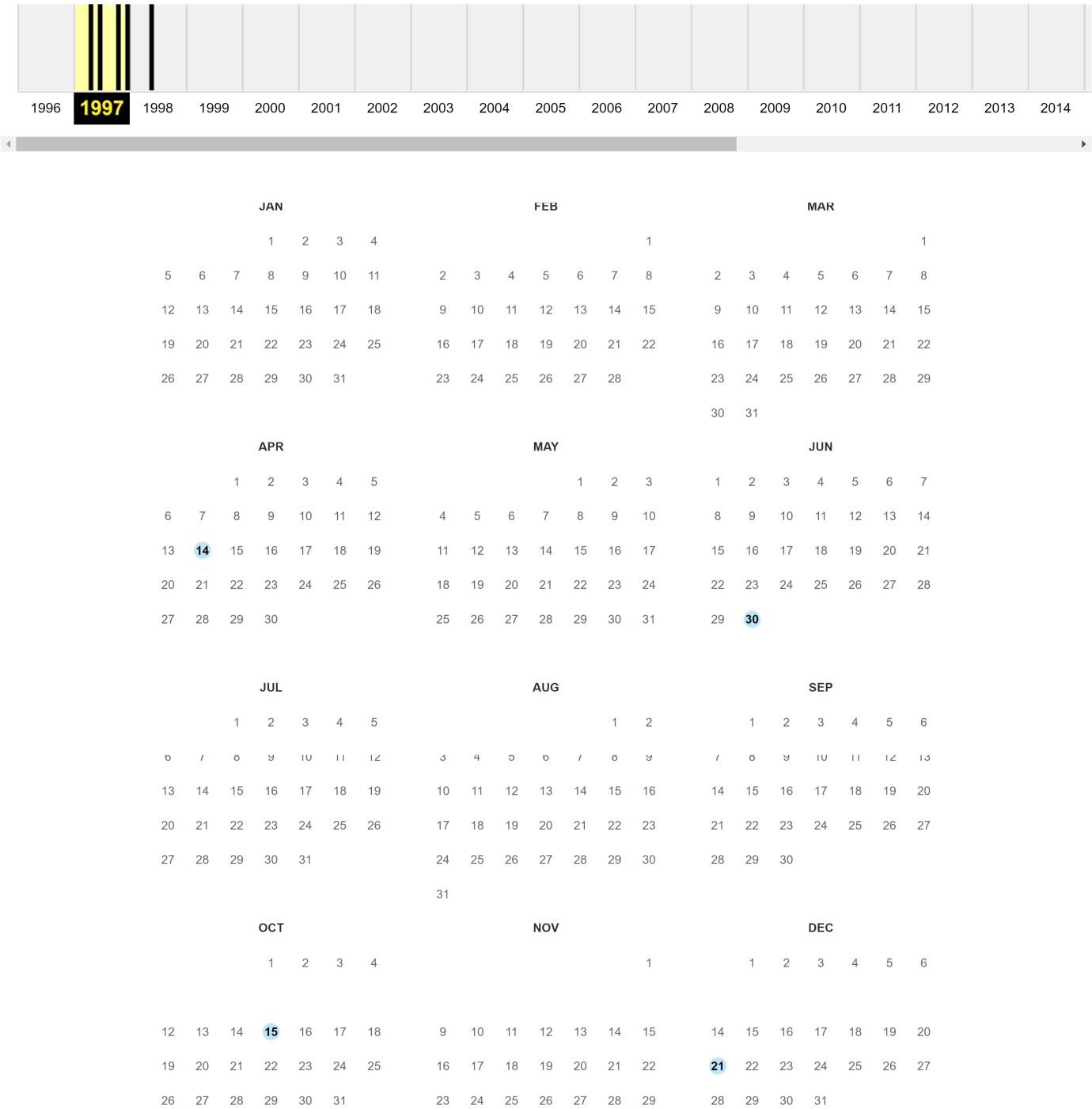
DONATE

WayBackMachine

Explore more than 849 billion [web pages](#) saved over time

- Calendar
- Collections
- Changes
- Summary
- Site Map
- URLs

Saved 5 times between April 14, 1997 and May 6, 1998.



Note

This calendar view maps the number of times

<http://hathorne.library.cmu.edu/uhtbin/cgiirsi/o/1/o> was crawled by the Wayback Machine, *not* how many times the site was actually updated. More info in the [FAQ](#).

[FAQ](#) | [Contact Us](#) | [Terms of Service \(Dec 31, 2014\)](#)



The Wayback Machine is an initiative of the [Internet Archive](#), a 501(c)(3) non-profit, building a digital library of Internet sites and other cultural artifacts in digital form. Other [projects](#) include [Open Library](#) & [archive-it.org](#).

Your use of the Wayback Machine is subject to the Internet Archive's [Terms of Use](#).

http://hathorne.library.cmu.edu/uhtbin/cgisirsi/0/1/0 JUN OCT DEC
5 captures 14 Apr 1997 - 6 May 1998 1996 15 1997 1998 About this capture

Unicorn WebCat

UNICORN PUBLIC ACCESS CHOICES



Choose one of the following:

- [LIBRARY CATALOG](#)
- [COURSE RESERVES](#)
- [HOURS AND WHAT'S NEW](#)
- [USER SELF SERVICE INFORMATION](#)

Find library materials using a particular word or phrase through the [LIBRARY CATALOG](#). View the holdings of the [COURSE RESERVES](#) to see materials

Unicorn WebCat

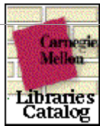
UNICORN PUBLIC ACCESS CHOICES



Choose one of the following:

- [LIBRARY CATALOG](#)
- [COURSE RESERVES](#)
- [HOURS AND WHAT'S NEW](#)
- [USER SELF SERVICE INFORMATION](#)

Find library materials using a particular word or phrase through the [LIBRARY CATALOG](#). View the holdings of the [COURSE RESERVES](#) to see materials set aside by an instructor for a course. Hours and What's New Use any of the library's interactive patron functions, including checking your circulation status with [USER SELF SERVICE INFORMATION](#).



Library materials are described in indexed records in the online catalog. Under the [LIBRARY CATALOG](#) you can lookup library materials using a particular word or phrase, or review lists of the library's authors, titles, subjects, series, keywords, or call numbers.

[PUBLIC ACCESS CHOICES](#) Reserve materials are items put aside at the [COURSE RESERVES](#) at the request of an instructor for students taking a course. See a list of reserve items a particular course by typing the course name. See a list of items reserved by an instructor by typing the instructor's name.



Our [HOURS AND WHAT'S NEW](#) is like a bulletin board where the library puts information you may find interesting or useful. The Information Desk has library memos such as Meetings, Library Services, or Community Resources, or lists can be of library material in the catalog, such as New Books, Great Videos, or Recommended Reading.




The library offers a variety of interactive patron functions, including displaying current circulation status and library messages, and letting you change your PIN code. To use any of the [USER SELF SERVICE INFORMATION](#), you must identify yourself by your library card number and your PIN.







This is the NEWROOT footer





WebCat(tm) © 1995 - 1997 Sirsi Corporation

Appendix CMU02






SIGN UP | LOG IN UPLOAD

ABOUTBLOGPROJECTSHelpDONATECONTACTJOBSVOLUNTEERPEOPLE



INTERNET ARCHIVE

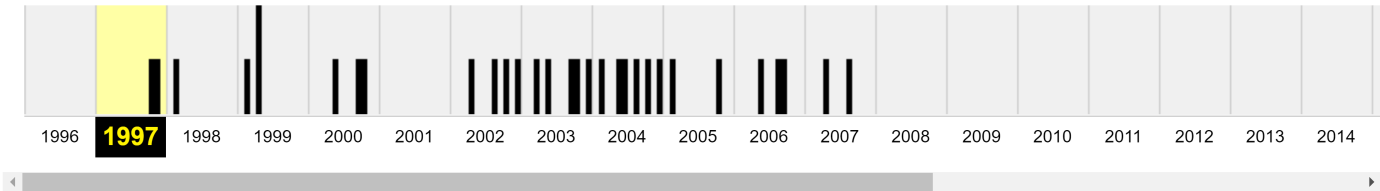


Explore more than 849 billion [web pages](#) saved over time

DONATE

- Calendar
- Collections
- Changes
- Summary
- Site Map
- URLs

Saved 31 times between [October 9, 1997](#) and [August 3, 2007](#).



JAN							FEB							MAR										
			1	2	3	4							1										1	
5	6	7	8	9	10	11			2	3	4	5	6	7	8			2	3	4	5	6	7	8
12	13	14	15	16	17	18			9	10	11	12	13	14	15			9	10	11	12	13	14	15
19	20	21	22	23	24	25			16	17	18	19	20	21	22			16	17	18	19	20	21	22
26	27	28	29	30	31				23	24	25	26	27	28				23	24	25	26	27	28	29
																		30	31					
APR							MAY							JUN										
			1	2	3	4	5						1	2	3			1	2	3	4	5	6	7
6	7	8	9	10	11	12			4	5	6	7	8	9	10			8	9	10	11	12	13	14
13	14	15	16	17	18	19			11	12	13	14	15	16	17			15	16	17	18	19	20	21
20	21	22	23	24	25	26			18	19	20	21	22	23	24			22	23	24	25	26	27	28
27	28	29	30						25	26	27	28	29	30	31			29	30					
JUL							AUG							SEP										
			1	2	3	4	5						1	2				1	2	3	4	5	6	
6	7	8	9	10	11	12			3	4	5	6	7	8	9			7	8	9	10	11	12	13
13	14	15	16	17	18	19			10	11	12	13	14	15	16			14	15	16	17	18	19	20
20	21	22	23	24	25	26			17	18	19	20	21	22	23			21	22	23	24	25	26	27
27	28	29	30	31					24	25	26	27	28	29	30			28	29	30				
									31															
OCT							NOV							DEC										
			1	2	3	4							1					1	2	3	4	5	6	
12	13	14	15	16	17	18			9	10	11	12	13	14	15			14	15	16	17	18	19	20
19	20	21	22	23	24	25			16	17	18	19	20	21	22			21	22	23	24	25	26	27
26	27	28	29	30	31				23	24	25	26	27	28	29			28	29	30	31			

Note

This calendar view maps the number of times

<http://www.library.cmu.edu/Research/Online/index.html> was crawled by the Wayback Machine, *not* how many times the site was actually updated. More info in the [FAQ](#).

[FAQ](#) | [Contact Us](#) | [Terms of Service \(Dec 31, 2014\)](#)



The Wayback Machine is an initiative of the [Internet Archive](#), a 501(c)(3) non-profit, building a digital library of Internet sites and other cultural artifacts in digital form. Other [projects](#) include [Open Library](#) & [archive-it.org](#).

Your use of the Wayback Machine is subject to the Internet Archive's [Terms of Use](#).

http://www.library.cmu.edu/Research/Online/index.html
Go
SEP
OCT
NOV
09
1996
1997
1999
About this capture

31 captures
9 Oct 1997 - 3 Aug 2007

University Libraries: Online Research

Catalogs and Databases

Home

Research

Info

Services

Libs/Archs

Search

Comment

Database List

Library Catalogs

Carnegie Mellon University Libraries

The Carnegie Library of Pittsburgh

Center for Research Libraries

Library of Congress

University of Pittsburgh Library System

WebCat includes bibliographic records for more than 900,000 books, journals, films, sound recordings, and other items owned by libraries at Carnegie Mellon University.

Catalog records indicate the library and shelving location for items in:


- Hunt Library (HUNT)
- Engineering and Science Library (ENGR&SCI;)
- Mellon Institute Library (MELLON)
- Hunt Institute for Botanical Documentation (HIBD)
- Software Engineering Institute Library (SEI)







Interdisciplinary



August 1997 -- http://www.library.cmu.edu/Research/Online/webmaster@www.library.cmu.edu

Carnegie Mellon University Libraries

Appendix SIRSI01





SIGN UP | LOG IN UPLOAD

ABOUTBLOGPROJECTSHelpDONATECONTACTJOBSVOLUNTEERPEOPLE



INTERNET ARCHIVE

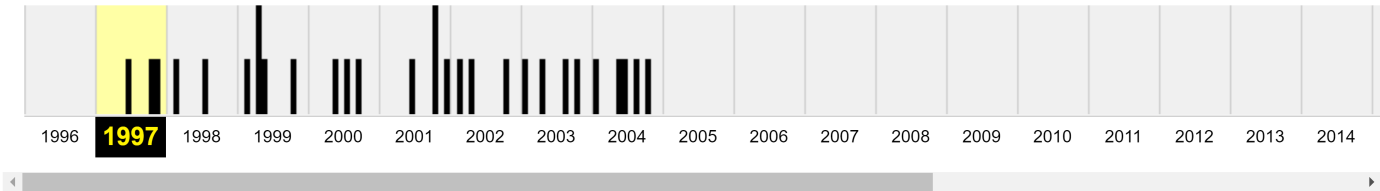
DONATE

WayBackMachine

Explore more than 849 billion [web pages](#) saved over time

Calendar · Collections · Changes · Summary · Site Map · URLs

Saved 29 times between June 27, 1997 and October 31, 2004.



JAN							FEB							MAR								
			1	2	3	4						1						1				
5	6	7	8	9	10	11		2	3	4	5	6	7	8		2	3	4	5	6	7	8
12	13	14	15	16	17	18		9	10	11	12	13	14	15		9	10	11	12	13	14	15
19	20	21	22	23	24	25		16	17	18	19	20	21	22		16	17	18	19	20	21	22
26	27	28	29	30	31			23	24	25	26	27	28			23	24	25	26	27	28	29
																30	31					
APR							MAY							JUN								
			1	2	3	4	5					1	2	3		1	2	3	4	5	6	7
6	7	8	9	10	11	12		4	5	6	7	8	9	10		8	9	10	11	12	13	14
13	14	15	16	17	18	19		11	12	13	14	15	16	17		15	16	17	18	19	20	21
20	21	22	23	24	25	26		18	19	20	21	22	23	24		22	23	24	25	26	27	28
27	28	29	30					25	26	27	28	29	30	31		29	30					
JUL							AUG							SEP								
			1	2	3	4	5					1	2			1	2	3	4	5	6	
6	7	8	9	10	11	12		3	4	5	6	7	8	9		7	8	9	10	11	12	13
13	14	15	16	17	18	19		10	11	12	13	14	15	16		14	15	16	17	18	19	20
20	21	22	23	24	25	26		17	18	19	20	21	22	23		21	22	23	24	25	26	27
27	28	29	30	31				24	25	26	27	28	29	30		28	29	30				
								31														
OCT							NOV							DEC								
			1	2	3	4						1				1	2	3	4	5	6	
12	13	14	15	16	17	18		9	10	11	12	13	14	15		14	15	16	17	18	19	20
19	20	21	22	23	24	25		16	17	18	19	20	21	22		21	22	23	24	25	26	27
26	27	28	29	30	31			23	24	25	26	27	28	29		28	29	30	31			

Note

This calendar view maps the number of times

<http://www.sirsi.com/Products/ulms.html> was crawled by the Wayback Machine, *not* how many times the site was actually updated. More info in the [FAQ](#).

[FAQ](#) | [Contact Us](#) | [Terms of Service \(Dec 31, 2014\)](#)



The Wayback Machine is an initiative of the [Internet Archive](#), a 501(c)(3) non-profit, building a digital library of Internet sites and other cultural artifacts in digital form. Other [projects](#) include [Open Library](#) & [archive-it.org](#).

Your use of the Wayback Machine is subject to the Internet Archive's [Terms of Use](#).

MAY **JUN 27 1997** OCT



Library Management System

- [Product Overview](#)
- [Modules Available](#)
- [Product Demonstration](#)
- [Companion Products](#)
- [Clients](#)
- [Let us configure a system for you!](#)

Product Overview

The way the world finds and shares information is changing fast. It is a world without barriers. To provide your clients and patrons with the absolute finest in library services and support -- and extend those services around the world -- you need UNICORN from SIRSI



Library Management System

- [Product Overview](#)
- [Modules Available](#)
- [Product Demonstration](#)
- [Companion Products](#)
- [Clients](#)
- [Let us configure a system for you!](#)

Product Overview

The way the world finds and shares information is changing fast. It is a world without barriers. To provide your clients and patrons with the absolute finest in library services and support -- and extend those services around the world -- you need UNICORN from SIRSI.

From the beginning, the right decisions have been made to ensure that UNICORN users would always be able to meet change head-on. A perfect blend of power and versatility, UNICORN is today's continuous source for technological breakthroughs.

- Balanced client/server architecture for growth
- UNIX server for freedom of choice
- Core design that will never become obsolete

Client Software - Something for Everyone

To make the most of your library system investment, you'll need flexible options for speeding resources to your staff and patrons. With UNICORN's many client options, no one is left out. Whether you are using a terminal, a personal computer with Windows, or a Web browser, the UNICORN family of InfoVIEW, WebCat, and Z39.50 client software is designed to work for you.

One Look, One Feel

SIRSI understands libraries. We know that productivity demands consistency. That's why we offer consistent interfaces across all UNICORN functions. So whether you're cataloging new material, or adding a new patron to your circulation files, UNICORN looks and feels the same.

Modules Available

- [Bibliographic and Inventory Control](#)
- [Authority Control](#)
- [SmartPORT](#)
- [Enhanced Public Access](#)
- [Circulation](#)
- [Academic Reserves](#)
- [Materials Booking](#)
- [Acquisitions](#)
- [Serials](#)
- [Information Gateway](#)

Bibliographic and Inventory Control

The heart of UNICORN is complete bibliographic control. For consistency, all catalog information is created, maintained and stored in one place -- your catalog. In addition to full MARC import and export capabilities, UNICORN includes sophisticated techniques for addressing special collections, uncataloged items, and bound-with titles. SIRSI's Z39.50 version 3 server is also included as a standard UNICORN catalog feature.

Authority Control

The UNICORN Authority Control module provides an online, interactive control system to establish a single, authoritative form for headings. Support for multiple authority files is standard, and any type of bibliographic heading may be controlled, with validation occurring immediately upon entry. To aid in catalog searching, a full ANSI thesaurus is also generated to provide SEE and SEE-ALSO cross references.

SmartPORT

Streamlining the process of copy cataloging is what SIRSI's [SmartPORT](#) is all about. Fully integrated with the UNICORN bibliographic catalog and authority file, SmartPORT uses the Z39.50 protocol to revolutionize the capture of MARC records for acquisitions and cataloging. Sources for records can be any MARC-based Z39.50 server including OCLC, RLIN, ISM and Library of Congress.

Enhanced Public Access

UNICORN's highly evolved public access catalog is designed to address the needs of the beginning searcher, as well as the researcher. Separate choices are available for browsing and keyword searching. A full-text index is available as a standard feature of UNICORN, providing the best possible approach for accessing all library material. Precise search results are easily retrieved using UNICORN's sophisticated truncation, boolean, relational, adjacency, and proximity operators. As an added benefit, the UNICORN Enhanced Public Access module also includes an Information Desk, a Request Desk, and a User Services Desk.

Circulation

SIRSI's powerful circulation control system is designed to meet the production demands of any circulation department. To speed patrons on their way, every function -- checkout, checkin, renewals, holds and payments -- is immediately available. Staff never spend time moving through multiple levels of computer screens. Extensive policy matrices handle single or multi-library requirements, and provide an unlimited number of user profiles, item types, locations, and loan periods.

Academic Reserves

SIRSI recognizes the importance of managing temporary reserve collections in academic libraries. The UNICORN Academic Reserves module supports multiple reserve collections across multiple libraries. Items can be placed on reserve for several reservers and courses simultaneously. In addition to catalog access, patrons can retrieve reserve items by instructor name, course name, and course ID. Hooks to electronic reserves is also included.

Materials Booking

The UNICORN Materials Booking module provides control over the allocation of time-scheduled resources. Separate booking calendars are generated for each bookable item. Custom cataloging forms are provided for describing equipment inventories and meeting rooms.

Acquisitions

UNICORN acquisitions and fund accounting tracks the purchase of materials through ordering, claiming, receiving, invoicing, and processing. Managing approval items, gifts, standing orders, and subscriptions, as well as any other library-defined order types is straight-forward and complete. On-order records may include full MARC or brief catalog descriptions that are fully integrated with the public access catalog. Time-saving order processing features include automatic vendor discount calculations and currency conversion, unlimited space for staff notes, fund splits by percentage and amount, multi-volume/copy ordering, and multi-library/location holdings distributions.

Serials

Serials checkin and control manages the prediction, receipt, and routing of all serial subscriptions. Managing orders and renewals is fully integrated with the UNICORN Acquisitions system. Receiving individual issues automatically increments the prediction calendar, and generates and maintains a separate MARC holdings record for each subscription. In addition, UNICORN also produces a consolidated holdings display for public access catalog users which conforms to the Z39.44 standard. Barcode checkin of issues is also supported for those publishers that meet SISAC's guidelines for defining an issue's chronology in SICI barcode format.

Information Gateway

Organizing and accessing sources of information found beyond the library's catalog is easy with SIRSI's Information Gateway. Library-controlled lists of Internet sites to visit are readily available for patrons at the click of a button. Defining site classification schemes based on subject, geographic, or other library-defined groupings is also supported. Destinations requiring telnet, World Wide Web, and Z39.50 protocols can be easily added using the SIRSI Gateway.

Companion Products

- [WebCat](#)
 - [InfoBASE](#)
 - [VIZION with Z39.50](#)
 - [Digital Media Archive](#)
-

WebCat

The Internet's World Wide Web has more potential for delivering interactive information than any other software. Available as an add-on for UNICORN, or any other Z39.50-compatible server, SIRSI's award-winning [WebCat](#) turns your library catalog and all public access services into a full-fledged World Wide Web catalog.

InfoBASE

Today information services often extend well beyond the library catalog. Sources of information consist of on-line services and locally generated material. [InfoBASE](#) is designed to support local data warehousing and provide total control over auxiliary databases. Available as an integrated component of any UNICORN library management system, or as a standalone server, InfoBASE is your Z39.50 server solution.

VIZION with Z39.50

SIRSI's [VIZION with Z39.50](#) for Windows provides a single user interface to a world of information resources. VIZION provides a sophisticated, customizable, icon-based site capture and retrieval management system for your favorite Internet resources. VIZION allows searching of research databases, locally developed databases, library catalogs, and the rapidly growing body of information resources available through Z39.50 servers.


Digital Media Archive







SIRSI's [digital media archive system](#) addresses the media capture, registration, search, retrieval, and administrative requirements of the new virtual library. Available as a standalone archive, or integrated with any Unicorn library management system, SIRSI's digital media archive is the next logical step in library services.





© 1996, Sirsi Corporation, 689 Discovery Drive, Huntsville, AL 35806
info@sirsi.com, webmaster@sirsi.com

Appendix SIRSI02






SIGN UP | LOG IN UPLOAD

ABOUTBLOGPROJECTSHelpDONATECONTACTJOBSVOLUNTEERPEOPLE



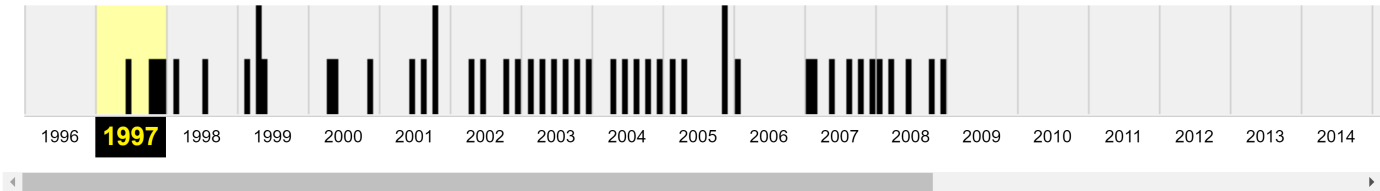
INTERNET ARCHIVE



Explore more than 849 billion [web pages](#) saved over time

- Calendar
- Collections
- Changes
- Summary
- Site Map
- URLs

Saved **50 times** between [June 27, 1997](#) and [July 10, 2023](#).



JAN							FEB							MAR									
			1	2	3	4							1								1		
5	6	7	8	9	10	11		2	3	4	5	6	7	8		2	3	4	5	6	7	8	
12	13	14	15	16	17	18		9	10	11	12	13	14	15		9	10	11	12	13	14	15	
19	20	21	22	23	24	25		16	17	18	19	20	21	22		16	17	18	19	20	21	22	
26	27	28	29	30	31			23	24	25	26	27	28			23	24	25	26	27	28	29	
																30	31						
APR							MAY							JUN									
			1	2	3	4	5						1	2	3		1	2	3	4	5	6	7
6	7	8	9	10	11	12		4	5	6	7	8	9	10		8	9	10	11	12	13	14	
13	14	15	16	17	18	19		11	12	13	14	15	16	17		15	16	17	18	19	20	21	
20	21	22	23	24	25	26		18	19	20	21	22	23	24		22	23	24	25	26	27	28	
27	28	29	30					25	26	27	28	29	30	31		29	30						
JUL							AUG							SEP									
			1	2	3	4	5						1	2			1	2	3	4	5	6	
6	7	8	9	10	11	12		3	4	5	6	7	8	9		7	8	9	10	11	12	13	
13	14	15	16	17	18	19		10	11	12	13	14	15	16		14	15	16	17	18	19	20	
20	21	22	23	24	25	26		17	18	19	20	21	22	23		21	22	23	24	25	26	27	
27	28	29	30	31				24	25	26	27	28	29	30		28	29	30					
								31															
OCT							NOV							DEC									
			1	2	3	4							1				1	2	3	4	5	6	
12	13	14	15	16	17	18		9	10	11	12	13	14	15		14	15	16	17	18	19	20	
19	20	21	22	23	24	25		16	17	18	19	20	21	22		21	22	23	24	25	26	27	
26	27	28	29	30	31			23	24	25	26	27	28	29		28	29	30	31				

Note

This calendar view maps the number of times

<http://www.sirsi.com/Products/webcat.html> was crawled by the Wayback Machine, *not* how many times the site was actually updated. More info in the [FAQ](#).

[FAQ](#) | [Contact Us](#) | [Terms of Service \(Dec 31, 2014\)](#)



The Wayback Machine is an initiative of the [Internet Archive](#), a 501(c)(3) non-profit, building a digital library of Internet sites and other cultural artifacts in digital form. Other [projects](#) include [Open Library](#) & [archive-it.org](#).

Your use of the Wayback Machine is subject to the Internet Archive's [Terms of Use](#).

<http://www.sirsi.com/Products/webcat.html>

MAY

JUN 27 1997

OCT

1996

1998

50 captures

27 Jun 1997 - 10 Jul 2023

About this capture

WebCat with Z39.50

WebCat Demonstration

SIRSI's award-winning WebCat harnesses the power of the World Wide Web to extend your library's services to the four corners of the earth. When you see WebCat, you will see that the library walls are coming down. You'll see that in their place stand five million users of the World Wide Web. With your Z39.50 server and SIRSI's WebCat, your library becomes the center of your patrons' information world.

Instant Availability

All your patrons need is a Web browser such as Mosaic or Netscape. WebCat does the rest, turning your library catalog and all public access services into a full-fledged Web catalog. There is no proprietary client software to distribute or support, and no training required for patrons to access your library's collections through your Web home page.



WebCat with Z39.50

WebCat Demonstration

SIRSI's award-winning WebCat harnesses the power of the World Wide Web to extend your library's services to the four corners of the earth. When you see WebCat, you will see that the library walls are coming down. You'll see that in their place stand five million users of the World Wide Web. With your Z39.50 server and SIRSI's WebCat, your library becomes the center of your patrons' information world.

Instant Availability

All your patrons need is a Web browser such as Mosaic or Netscape. WebCat does the rest, turning your library catalog and all public access services into a full-fledged Web catalog. There is no proprietary client software to distribute or support, and no training required for patrons to access your library's collections through your Web home page.

Multi-Media Access

WebCat brings bibliographic citation records to life by providing hyperlinks to multi-media files. SIRSI uses the power of WebCat and the World Wide Web to display photographs, images, full-text, and sound files. Access to these exciting information resources is just a mouse click away from the electronic address (URL) you supply in your bibliographic record.

One Look, One Feel

WebCat with Z39.50 gives users a single interface for accessing and organizing information. The easy-to-use, familiar look and feel of today's Web browsers host WebCat's powerful features.

- Quickly create and execute simple or complex search statements
- Browse and select terms from heading lists
- Search related information by clicking on hypertext terms
- View cross-reference information
- Sort search results
- Export selected records to networked printers, files or email
- Link to library holdings information on one or more servers
- Execute previous searches from search history

Patron Services

When used with any of SIRSI's Unicorn library management systems, WebCat provides a straight-forward method to allow your patrons to help themselves.

Checkouts

Users may see a list of all items they currently have checked out. In addition to author, title and call number, WebCat displays date charged, date due, and any accruing fines.

Holds

WebCat provides information for titles on hold, along with current availability.

Bills

Patrons can review all bills they have received from the library. This includes fees for overdues, as well as for services such as photocopy or fax charges.

Renewals

WebCat allows patrons to be responsible for their own renewals. Individual items or a batch of items may be renewed, with any conflicts immediately reported for each item.

Request Desk

Communication between library patrons and library staff is critical. WebCat's Request Desk helps you keep information flowing by providing custom forms for electronic communication. Whenever possible, WebCat extracts data from bibliographic records to complete each form automatically. Electronic forms include, but are not limited to:

- Order requests
- Suggestions

- Change of address

Information Desk

Libraries often need a way to present structured data to their patrons. The WebCat Information Desk gives you the ability to pre-define search strategies. The searches may be executed against your local catalog, or against any other Z39.50 server. The Information Desk also allows you to organize searches into collected bibliographies. For users of SIRSI's Unicorn library management products, the Information Desk provides an additional feature for broadcasting information referral data such as announcements, community information and new book lists.

Reserves Desk

Locating information associated with Academic Reserves collections is easy with WebCat. For users of SIRSI's Unicorn library management products, WebCat permits retrieval of reserve items by course name, course number and instructor name. Reserve items may be titles in the Unicorn catalog or material in electronic format.

Gateway Manager

Finding a balanced presentation of functionality that gives new users enough information to make intelligent choices, without getting in the way of experienced users is a goal SIRSI shares with libraries. Flexibility is the key to WebCat's success in this effort. The Gateway Manager allows the library to customize user interfaces. Through a simple question and answer interview session, WebCat's forms, icons, instructions and functionality can be arranged to suit local needs.

Access Control

You want to make your bibliographic resources available to users of the World Wide Web, but you need to protect certain files from unauthorized access. WebCat is the answer with its precise user authentication features. Administrative policies can be established for each class of user. Local patrons may be given unlimited access privileges, for example, while external library users may be restricted to catalog material only.

Meeting Your Needs

WebCat is available in several configurations. Depending on your local needs, WebCat, along with your Web server software, can be installed on the same host server as your Unicorn or other Z39.50 server, or on a standalone server with networked access to your local Z39.50 catalogs and services. Access to the Internet is also required.



© 1996, Sirsi Corporation, 689 Discovery Drive, Huntsville, AL 35806
info@sirsi.com, webmaster@sirsi.com