ID	Vuln Type	State	Potential Attack
A3	ID	S2	A01 SELECT /../../<OTHER-U>/inbox
A4	BO	any	<A×2596>

a) Potentially detected vulnerabilities.

ID	Vuln Type	State	First Successful Attack
A1	BO	S2	A01 AUTHENTICATE <A×1296>
	BO	S2	A01 SELECT <A×1296>
A2	FS	any	<%s×10>
A5	FS	S2	A01 LIST <A×10> <%s×10>
A6	BO	S2	A01 CREATE <A×244>
	BO	any*	<A×1260>
A7	FS	S3	A01 SEARCH TOPIC <%s×10>
A8	BO	S2	A01 SELECT "{localhost/user=\"}"
A9	BO	S2	A01 EXAMINE <A×300>
A10	ID	S2	A01 SELECT /../../<OTHER-U>/inbox
A11	BO	S2	A01 SELECT <A×1296>
	ID	S2	A01 CREATE /<A×10>
A12	DoS	S2	A01 RENAME <A×10> <A×10>

b) Detected previously known vulnerabilities.

Application	Vuln Type	State	First Successful Attack
TrueNorth eMail-Server Corporate Edition 5.3.4	BO	S3	A01 SEARCH <A×560>

c) New vulnerability discovered with AJECT.

Table .1. Attacks generated by AJCET on IMAP Serves

For the two applications that we were unable to get, it was necessary to employ a different approach in the tests. The Injector was used to generate and carry out the attacks against a dummy IMAP server. Basically, this server only stored the contents of the received packets and returned simple responses. The packets were then later analyzed to determine if one of the attacks could activate the reported vulnerability. In Table 1 a) are presented the results of these experiments, and in both cases an attack was generated that could supposedly explore the vulnerabilities.

The vulnerabilities actually detected with AJECT are presented in Table 1 b). From the table it is possible to conclude that AJECT is capable of detecting several kinds of bugs, ranging from buffer overflows to information disclosure. Since we had a limited time for testing, and since we wanted to evaluate a large number of applications, we had to interrupt the tests as soon as a vulnerability was discovered so only the first successful attack is presented. In the few cases where experiments were run for a longer period, we noticed that several distinct attacks were able to uncover the same problem. For example, after 24500 injections against the GNU Mail utils, there were already more than 200 attacks that similarly crashed the application. This section gives a brief overview of the IMAP communication protocol that is utilized by the servers under test. It also describes the classes of attacks that were tried by the injector, and provides some information about the test cases.

Some times it was difficult to determine if distinct attacks were equivalent in terms of discovering the same flaw, specially in the cases where they used different IMAP commands. For example, if a bug is in the implementation of a validation routine that is called by the various commands, then the attacks would be equivalent. On the other hand, if no code was shared then there should be different bugs. Therefore, in order to find out exactly if attacks are equivalent, one would need to have access to the source code of the applications (something impossible to obtain for a majority of the products). Consequently, we decided to take a conservative approach, where all attacks were deemed equivalent except in the situations where they correspond without any doubt to different vulnerabilities.

During the course of our experiments, we were able to discover a previously unknown vulnerability (see Table 1 c)). The attack sends a large string in a SEARCH command that causes a crash in the server. This indicates that the bug is a boundary condition verification error, which probably corresponds to a buffer overflow. Several versions of the E-Mail Server application were tested, including the most recent one, and all of them were vulnerable to this attack.

V. CONCLUSIONS

The paper presents a tool for the discovery of vulnerabilities in server applications. AJECT simulates the behavior of a malicious adversary by injecting different kinds of attacks against the target server. In parallel, it observes the application while it runs in order to collect various information. This information is later analyzed to determine if the server executed in correctly, which is a strong indication that a vulnerability exists.

To evaluate the usefulness of the tool, several experiments were conducted with many IMAP products. These experiments indicate that AJECT could be utilized to locate a significant number of distinct types of vulnerabilities (e.g., buffer over flows, format strings, and information disclosure bugs).In addition, AJECT was able to discover a new buffer overflow vulnerability.

VI. REFERENCES

[1] A. Adelsbach, D. Alessandri, C. Cachin, S. Creese,Y. Deswarte, K. Kursawe, J. C. Laprie, D. Powell, B. Randell, J. Riordan, P. Ryan, W. Simmonds,R. Stroud, P. Ver´issimo, M. Waidner, and A. Wespi. Conceptual Model and Architecture of MAFTIA. Project MAFTIA deliverable D21. Jan. 2002. http://www.research.ec.org/maftia/deliverables/D21.pdf.

[2] A.Albinet, J.Arlat, and J.-C.Fabre. Characterization of the impact of faulty drivers on the robustness of the Linux kernel. In Proc. of the Int. Conference on Dependable Systems and Networks, pages 867–876, June2004.

[3] J. Arlat, A. Costes, Y. Crouzet, J.-C. Laprie, and D. Powell. Fault injection and dependability evaluation of fault-tolerant systems. IEEE Trans. on Computers, 42(8):913–923, Aug.1993.

[4] J. Arlat, Y. Crouzet, and J. Laprie. Fault injection for dependability validation of fault-tolerant computer systems. In Proc. of the Int. Symp. on Fault-Tolerant Computing, pages 348–355, June1989.

[5] M. Bishop and M. Dilger. Checking for race conditions in file accesses. Computing Systems, 9(2):131–152, Spring 1996.

[6] A. Brown, L. C. Chung, and D. A. Patterson. Including the human factor in dependability benchmarks. In Workshop on Dependability Benchmarking, in Supplemental Volume of DSN2002,pages F–9–14, June2002.

[7] J. Carreira, H. Madeira, and J. G. Silva. Xception: A technique for the experimental evaluation of dependability in modern computers. IEEE Trans. on Software Engineering,24(2):125–136, Feb.1998.

[8] J. Christ mansson and R. Chillarege. Generation of an error set that emulates software faults. In Proc. of the Int. Symp. On Fault-Tolerant Computing, pages304–313, June1996.

[9] C.Cowan, S.Beattie, J.Johansen, and P.Wagle. Point guard: Protecting pointers from buffer overflow vulnerabilities. In Proc.ofthe12th USENIX Security Symposium, Aug.2003.

[10] C. Cowan, C. Pu, D. Maier, J. Walpole, P. Bakke, S. Beattie, A. Grier, P. Wagle, Q. Zhang, and H. Hinton. Stack Guard: Automatic adaptive detection and prevention of buffer-overflow attacks. In Proc. of the 7th USENIX Security Conference, pages63–78, Jan.1998.

[11] M. Crispin. Internet Message Access Protocol – Version 4rev1. Internet Engineering Task Force, RFC 3501, Mar.2003.

[12] J.Duraes˜ and H.Madeira. Definition of software fault emulation operators: A field data study.In Proc. Of the Int. Conference on Dependable Systems and Networks, pages 105–114,June2003.

[13] J. Duraes˜ and H. Madeira. A methodology for the automate identification of buffer overflow vulnerabilities in executable software without source-code. In Proc. of the Second Latin American Symposium on Dependable Computing,Oct.2005.

[14] D. Farmer and E. H. Spafford. The COPS security checker system. In Proc. of the Summer USENIX Conference, pages 165–170, June1990.

[15] Found Stone Inc. Found Stone Enterprise, 2005. http://www.foundstone.com.

[16] K.Goswami, R.Iyer, and L.Young. Depend: A simulation based environment for system level dependability analysis. IEEE Trans. on Computers,46(1):60–74, Jan.1997.