Development of a Method for Building a Trusted Environment by Using Hidden Software Agent Steganography

V Kh Fedorov¹, E G Balenko¹, S I Shterenberg², A V Krasov²

¹FSBEI HE "Don State Agrarian University", pos. Persianovskiy st. Krivoshlykova 24, 346493, Russia
²Saint Petersburg State University of Telecommunications prof. M.A. Bonch-Bruevich, St. Petersburg, Bolshevnikov Ave. 22, building 1, 193232, Russia

Abstract. For more than a decade, the scientific community has been developing various models of insider actions in the trusted-zone information system and methods for identifying such. All those years, they have been facing the challenge of finding quality data samples for analysis and testing, and it is the availability of reliable data on the attack that is crucial for any research or detection of attempts to steal or compromise legal software. Unfortunately, collection of such samples remains nearly impossible despite some attempts to do so. Any data on incidents is either incomplete or inaccurate, or is not available at all. Software developers also face lack of resources, including time. Developers do not have enough time, funding, material resources, or qualifications to make a sturdy security system, which is why they need to use third-party automations that dock a security module to the compiled program. The strength of this approach is that such a system can be embedded in any software, whereas the weakness is that the method is “one-size-fits-all”. Standard protections span across multiple programs, which results in a higher chance of being hacked.

1. Introduction

Earlier papers [1-8] discuss the development of program agents, the structure of this code, and the implementation of digital watermarks (DWM) in software. In particular, papers [2,3,4,6,8] refer to a trusted zone where the propagation of the software agent is analyzed. General outcomes of the possible actions on part of the software agent were compared to those of computer viruses and always had similar coefficients of their stay in the trusted zone [2]. The propagation of a software agent to the nodes and PCs in the TZ can be split into three stages:

1. Relatively slow (yet exponential) increase in the presence of the software agent until it hits the threshold of 0.05 defined as \( k_{na} = \frac{I}{N} \). Doubling the proportion of software agent-captured machines has the rate \( \ln(2) / \beta \).
2. Maximum propagation phase in the range 0.05 < \( k_{na} < 0.95 \); Duration is approximately 5,89 / \( \beta \).
3. Saturation, \( k_{na} > 0.95 \). In this segment, nodes mainly contact each other during random scanning of the address space, which is why they can remain ‘clean’ indefinitely.

Time to reach the saturation threshold \( k_{na} = 0.95 \) equals
\[
\ln \left( \frac{1}{k_{\text{нач}}} \right)^{1/\beta} - 1 \right];
\]

where \( k_{\text{нач}} \) is the propagation of the software agent in the LAN at the initial time \( t_0 \).

This data shows that as long as the presence of a software agent in the computers or nodes of a network is within 5\%, the trusted zone is safe.

The simulation model used event-based time label propagation. Before work begins, the time axis of the model is labeled with the times of expected events (arrival of packets, packet processing by router nodes, etc.). This saves computational resources, as time periods where nothing happens in the model are skipped. Software control process based on short-term prediction of software agent propagation across network nodes can be represented as a closed cycle of separate phases, see Figure 3. The first four phases define the data processing and analysis cycle, the rest constitute the control cycle.

The former generalizes, processes, and determines the state of software agent propagation across network nodes at a given time, and then reports the data to the Network Control Center (NCC). The control cycle predicts the propagation of the software agent across the network nodes and makes a decision in light of the received data.

Take a closer look at the steps involved in predicting the propagation of a software agent across the network nodes. To predict such propagation, the authors hereof decided to use a control method based on the predictive models: Model Predictive Control (MPC).

2. Related works

The development of a software agent control method applicable to the trusted zone and based on short-term propagation prediction falls into the following stages: describe the software agent propagation model for the trusted zone, make a safe-state criterion for the trusted zone, compute the control law.

One of the advantages of using predictive models for control is that they allow testing a multifactorial process “ahead of the curve”. For this reason, space-state model-based method was selected for predictions. State-Space Model Predictive Control (SSMPC) (fig. 1).

SSMPC requires a mathematical model of the controlled object, which is further used to predict the IT system’s output from past and current values and expected optimal future control actions. These actions are calculated by the optimizer, which takes into account the quality criterion (adjusts for future errors) and restrictions imposed on the variables of the process that describes the controlled object.

Figure 1. Software agent control in TZ.
The model of choice should cover the process dynamics to accurately predict future outputs; it should also be easy to understand and implement.

Consider the software agent control process shown in Figure 1. Propagation of the software agent in the trusted zone over the timeframe \([0,T]\) can be described by the epidemiological model assuming that:

1. \(N\) is the total number of machines in the LAN. An \(N\)-node LAN can be described by the matrix \(G = \{0,1\}^{N^2}\):

\[
G_{ij} = \begin{cases} 
1, & \text{if the nodes } i \text{ and } j \text{ are connected} \\
0, & \text{if not}
\end{cases}
\]  

2. An arbitrary node of the network can be in one of the three states: vulnerable \(S\), infected \(I\), and invulnerable \(R\);

3. The software agent propagates through the trusted zone without user intervention, and the same software cannot steganotransform a node again.

In the simplest case, \(\beta\) depends on the average scanning rate of the trusted-zone network \((v_s)\) as well as on the size of its address space \((N_{ip})\):

\[
\beta = v_s \frac{N}{N_{ip}}
\]

Eqs. (3) and (4) allow for a more accurate description of the state of the trusted zone, as they forsake the old approach that would only describe states of individual LAN nodes in time; the new approach gives a more extensive view of the rate and extent of software agent propagation.

3. Experiment

3.1. Test 1. Steganographic effect with the use of a single-level SGA from the insider

3.1.1. Input for Test 1. To create and use a reliable steganographic tool, there needs to be a toolkit for testing it [2,6,8]. Quantifying the steganographic security system’s resilience to external influences is a fairly complex problem that is usually solved by system analysis, mathematical modeling, or experimentally.

As a rule, a professional SGS provides three-level data protection that addresses two key challenges:

1. It conceals the very existence of protected data (first level of protection);
2. It blocks unauthorized access to data by using an appropriate data concealment method (second level of protection).

Finally, there may exist a third level: pre-encryption of the concealed data. SGS quality, i.e., the degree of concealment it provides, can be assessed by analysis and testing in kind [6,8]. The quality of steganographic concealment is often tested by methods from other areas, in particular cryptoanalysis. Since the receiver can restore the data hidden in the received message, apparently retrieval methods exist.

Most distortion indicators or quality criteria applicable in visual data processing are differential distortion indicators. These indicators are based on the difference between the original container and the output container. The second group comprises indicators based on the correlation between original and distorted signals (correlation indicators of distortion).

3.1.2. Experimental conditions. Single-level SGA was used to model steganographic effect. Information was embedded by applying LSB steganography to each of the four file domains. To make a training set, 20% of files were altered in Collections 1 and 2. Other 80% of the program files were used as the test set. Each set had an equal number of original files and steganography-altered files. The most common ones were additionally selected, see Figure 2 for the list [1,2,8]:
3.1.3. **Testing steganoanalysis methods for effectiveness.** SGA effectiveness is found by classifying program files into original and altered ones. Perfect steganoanalysis would classify 100% of the program files with embedded data as containers and 100% files with no embedded data as originals. Classification output can have four values: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). Accuracy across SGA methods can be visually compared by means of graph of TN, TP, FP, and FN ratios showing the numbers of files classified as such as percentage of the total. To visualize SGA methods considered herein, the output of classification of any files with a 5% payload is shown as TN, TP, FP, FN graphs, see Figure 2.

![Figure 2](image-url) **Figure 2.** TN, TP, FN, FP ratios for a variety of files with embedded data.

3.2. **Test 2. Changes in the probabilities of steganoanalysis at various stages of operation in the trusted zone.**

3.2.1. **Input for Test 2.** In a broader sense, the need to log audit events using the procedures shown above caused multiple potential issues pertaining to supporting the software agent itself; however, tests [6,8,9] show that thanks to more complex obfuscation, DWM can withstand a certain load from active SGA.

First, risk was evaluated by using an attack graph [2-4]. Table 1 shows the comparative security assessment and outlines the key criteria of risk levels.

| Degree of importance | Description                                                                 | Priority of $\psi_j$ |
|----------------------|-----------------------------------------------------------------------------|----------------------|
| Low                  | This security measure is useful; however, its absence can be compensated by alternatives | 0.06                 |
| Medium               | This security measure is recommendable, and its absence can help the attacker | 0.13                 |
| High                 | This security measure is important, and the attacker will likely succeed in its absence | 0.26                 |
| Critical             | This security measure is critical, and the attacker will almost certainly succeed in its absence | 0.51                 |

3.2.2. **Errors Found by Tests 1 and 2.** An important aspect of the trusted zone is what it deploys to protect data, its storage, processing, transmission, and process automation from the insider’s actions,
including attempts to unlock information (including media protection from physical destruction) required to solve control problems and block obfuscation, which is propagated and injected in the control process (SSMPC).

Risk assessment standards and methods discussed above use different scoring scales, lists of threats and vulnerabilities, and inputs for assessment, as well as list different procedures to collect data on the software agent active in the trusted zone. However, nearly all the reviewed sources use stepwise risk evaluations by experts or expert panels who rely on their knowledge and experience, sometimes on statistics [10-16].

The procedure consists in using typical malware attacks to collect some data on the network, then attack the network to gain unauthorized access to the components of the entire network of the system. Statistically, reverse code accounts for 70% of all attacks. Figure 3 shows the estimated probability of unauthorized access gained in various attacks on the trusted zone; for clarity, it compares data on various categories of steganographically embedded hid code. For operations with up to 1000 iterations, tests used various versions of error backpropagation for the function $f_{ij}^{A_S}$, which differed in decision rules applicable to the direction and adjustment value of weight vectors.

![Figure 3](image)

**Figure 3.** Estimated probabilities of unauthorized access gained across attacks (blue for Category A, red for Cat. B, gray for Cat. C, and yellow for Cat. D).

General results shown in Figure 3 demonstrate that for the completed operations, error propagation in the developed SGS increases significantly when running steganotransforms for Categories B and C, but does not reach critical levels.

### 4. Conclusions

Tests show that across all values of TN, TP, FP, and FN, embedding hid code subjected to processing by a variety of SGA tools remains >64% intact on average under any possible conditions of information embedding in any program files.

The developed software agent module features an error detection and correction block and operates hid code; it stabilizes the network control system of the trusted zone. In none of the tests did any attack reach 100% success probability, whereas many actions of the software agent always had a higher than 50% chance to remain undetected.

### 5. References

[1] Korzhik V, Gerling E, Imai H, Shikata J, Morales-Luna G 2008 On the use of Bhattacharyya distance as a measure of the detectability of steganographic systems *Lecture Notes in Computer Science* T 4920 LNCS pp 23-32

[2] Krasov A V, Shterenberg S I 2020 Methods for building a trusted environment in Unix operating systems based on the implementation of a digital watermark *International Congress*
on Ultra Modern Telecommunications and Control Systems and Workshops pp 253–257

[3] Sharikov P I, Krasov A V, Volkogonov V N 2020 A study of the correctness of the execution of a class file with an embedded digital watermark in different environments IOP Conference Series: Materials Science and Engineering 862(5) 052052

[4] Krasov A, Vitkova L, Pestov I 2019 Behavioral Analysis of Resource Allocation Systems in Cloud Infrastructure Proceedings - 2019 International Russian Automation Conference RusAutoCon 2019 8867699

[5] Krasov A V, Arshinov A S, Ushakov I A 2018 Embedding the hidden information into java byte code based on operands' interchanging ARPN Journal of Engineering and Applied Sciences 13(8) pp 2746–2752

[6] Shterenberg S I, Poltavtseva M A 2018 A distributed intrusion detection system with protection from an internal intruder Automatic Control and Computer Sciences T 52 8 pp 945-953

[7] Buinevich M, Izrailov K, Vladyko A 2016 Method and prototype of utility for partial recovering source code for low-level and medium-level vulnerability search 18th International Conference on Advanced Communication Technology (ICACT) pp 700-707

[8] Shterenberg S I, Krasov A V, Ushakov I A 2015 Analysis of using equivalent instructions at the hidden embedding of information into the executable files Journal of Theoretical and Applied Information Technology T 80 1 pp 28-34

[9] Kotenko I, Saenko I, Lauta O, Kribel A 2020 An approach to detecting cyber attacks against smart power grids based on the analysis of network traffic self-similarity Energies 13(19) 5031

[10] Bagretsov S A, Lauta O S, Klimenko A I, Balenko E G 2019 Data Acquisition Technologies and System for Automating, Record-Keeping and Managing Water Supply Processes 2019 International Multi-Conference on Industrial Engineering and Modern Technologies FarEastCon 2019 8934400

[11] Bagretsov S A, Lauta O S, Klimenko A I, Balenko E G 2019 Method of Providing Rationale for Reasonable Number of Backup Communication Channels in Information Telecommunication Network 2019 International Multi-Conference on Industrial Engineering and Modern Technologies FarEastCon 2019 8934242

[12] Bagretsov S A, Lauta O S, Klimenko A I, Balenko E G 2019 Method for Providing Rationale of Basic Option of Information and Telecommunication Network under Hostile Action 2019 International Multi-Conference on Industrial Engineering and Modern Technologies FarEastCon 2019 8934215

[13] Bagretsov S A, Lauta O S, Klimenko A I, Balenko E G 2019 Method of Assessing Regions of Controlled Balance in Information and Telecommunications Network 2019 International Multi-Conference on Industrial Engineering and Modern Technologies FarEastCon 2019 8934272

[14] Burlov V, Lepeshkin O, Lepeshkin M 2021 Mathematical model for managing energy sector in the region Advances in Intelligent Systems and Computing 1258 AISC pp 659–668

[15] Burlov V, Lepeshkin O, Lepeshkin M 2020 Algorithmic support for the dynamic functioning of transport systems in the region IOP Conference Series: Materials Science and Engineering 918(1) 012224

[16] Burlov V G, Lepeshkin O M, Lepeshkin M O, Solovev D B 2020 Organization of Management of Social and Economic Systems of the Region in the Conditions of the Required Technosphere Safety IOP Conference Series: Earth and Environmental Science 459(2) 022081