N-Gram against the Machine: On the Feasibility of the N-Gram Network Analysis for Binary Protocols

Dina Hadžiosmanović[1], Lorenzo Simionato[2,⋆], Damiano Bolzoni[1], Emmanuele Zambon[1], and Sandro Etalle[1,3]

[1] University of Twente, The Netherlands
[2] Ca' Foscari University of Venice, Italy
[3] Technical University of Eindhoven, The Netherlands

Abstract. In recent years we have witnessed several complex and high-impact attacks specifically targeting "binary" protocols (RPC, Samba and, more recently, RDP). These attacks could not be detected by current – signature-based – detection solutions, while – at least in theory – they could be detected by state-of-the-art *anomaly-based* systems. This raises once again the still unanswered question of how *effective* anomaly-based systems are in practice. To contribute to answering this question, in this paper we investigate the effectiveness of a widely studied category of network intrusion detection systems: anomaly-based algorithms using *n-gram analysis* for payload inspection. Specifically, we present a thorough analysis and evaluation of several detection algorithms using variants of n-gram analysis on real-life environments. Our tests show that the analyzed systems, in presence of data with high variability, cannot deliver high detection and low false positive rates at the same time.

1 Introduction

While most of the current commercial network intrusion detection systems (NIDS) are *signature-based*, i.e., they recognize an attack when it *matches* a previously defined signature, there is a large body of literature on *anomaly-based* detection. An anomaly-based NIDS raises an alert when the observed input *does not match* the behavior that was previously observed; the underlying assumption being that attack payloads "look different" than normal network traffic. In principle, anomaly-based NIDS have *one great advantage* over signature-based ones: they can detect threats for which there exists no signature yet, including zero-day and targeted attacks. Targeted attacks are so complex and evasive that *by definition* cannot be detected by signature-based systems (false negative problem). One famous example of targeted attack is Stuxnet [14], a malware designed to hit specific embedded systems used in Iranian installations for uranium enrichment, discovered in the late 2010. The subsequent analysis revealed that the malware

⋆ The author carried out part of this work while he was a visiting student at the University of Twente. Current affiliation: Google Inc.

exploits two previously unknown vulnerabilities in network services. Thus, while the final target was a component typical of industrial control systems, (some of) the vulnerabilities aimed at infecting local computers. Hence, those vulnerabilities could have been used to attack "regular" business and home systems. Other attacks specifically designed to target Industrial Control Systems (ICS - which includes of nuclear power plants, oil and gas extraction and distribution facilities) have been disclosed recently [3].

Given that signature-based systems are ineffective against targeted and zero-day attacks and that most likely there exists no signature yet for the great majority of the attacks that one can buy on the black market, an effective anomaly-based NIDS would be the silver bullet thousands of enterprises and governments are looking for.

Problem & Contribution. Although the field of anomaly detection is well established in research, to date there are only few actual deployments of anomaly-based NIDS worldwide. A common reason used for explaining this is that such systems show poor performance with respect to false positive rate in real-life environments. More generally, Sommer and Paxson [29] argue that many machine learning approaches (which are typically used in anomaly-based IDS) are not effective enough for real-life deployments.

With this paper we want to shed a new light on the detection capabilities of anomaly-based NIDS for payload inspection. We do so by focusing on systems employing a form of n-gram-analysis as anomaly detection engine. To perform the analysis, we apply selected algorithms to environments that widely utilize binary protocols. Specifically,

– We perform thorough benchmarks using real-life data from binary-based protocols, which have been lately targeted by high-impact cyber attacks,
– We include in the analysis a protocol that is specific for Industrial Control Systems (also known as "SCADA"),
– We analyze and discuss the reasons why certain attack instances are (not) detected by the chosen approaches,
– We discuss the feasibility of deploying such approaches in real-life environments, in particular w.r.t. the false positive rate, an issue that is seldom discussed by authors in their work.

Our experiments show that n-gram analysis cannot be indiscriminately applied to the whole network stream data, and that data with high variability are difficult to model and analyze, confirming the conclusion of some earlier work on the matter (see Section 6 for a more detailed discussion).

2 Preliminaries

In this section we introduce the concepts and the terminology that will be used in the remaining of the paper.

2.1 Anomaly-Based Network Intrusion Detection Systems

A detection system can use different sources to extract data features, namely network traffic or system/application activities. In this embodiment we focus on network-based approaches. These approaches monitor the network traffic in a transparent way, without affecting the host performance and are thus often preferred over host-based approaches. There are two types of anomaly-based NIDS: (1) systems that analyse network flows and (2) systems that analyse the actual payload. A flow-based approach takes into consideration features such as the number of sent/received bytes, the duration of the connection and the layer-4 protocol used. A payload-based approach considers features of individual packet or communication payloads. Despite being complementary, the payload-based approaches offer higher chances of detecting a broader set of threats and, unlikely flow-based, these approaches can capture "semantic attacks". Semantic attacks exploit "a specific feature or implementation bug of some protocol or application installed at the victim" [25]. In addition, most of the top security risks (such as in the "OWASP Top 10" [33]) require the injection of some data to exploit the vulnerability. Thus, we focus on payload-based approaches.

2.2 Binary Protocols

In contrast to text-based network protocols (such as HTTP, POP and SMTP), binary protocols are designed to be processed by a computer rather than a human. Such protocols are largely used in network services, such as distributed file systems, databases, etc. In practical terms, the network payload of a binary protocol is more compact if compared to text protocols, often unreadable by a human and may resemble to attack payloads (since malware packets often consists of binary fragments too). Due to these reasons the challenge of detecting attacks in binary-based data is typically greater than in text-based data.

2.3 N-Gram Analysis

N-gram analysis is a common technique for capturing features of data content. This technique is used in various areas, such as monitoring system calls [16], text analysis [10], packet payload analysis [35]. In the context of network payload analysis, the current approaches use the concept of n-grams in different ways. In particular, we distinguish two aspects:

1. The way an n-gram builds feature space - The extracted n-grams can be used for building different feature spaces [12]: (a) count embedding (count the number of different n-grams to describe the payload), (b) frequency embedding (use relative frequency of byte values of an n-gram to describe the payload, e.g. [1,6,36]) and (c) binary embedding (use the presence/absence of specific n-grams to describe the payload, e.g., [35]).
2. The accuracy of payload representation - N-grams can represent the payload in the following ways: (a) as an exact payload description (n-grams represent

continuous sequences of bytes, e.g. in [6,35,36]) and (b) as an approximated payload description (n-grams represent a compression or a reduction of the exact payload, e.g., [17,28]).

Also, various systems employ different architectures and combinations of approaches to analyze n-grams (e.g., Markov models in [1], Self-Organizing Maps in [6], hashing in [17]).

For performing our benchmarks we choose algorithms that are conceptually different in the way the n-gram analysis is performed. Unfortunately, our choice is also limited to the availability of implementations and the level of details in algorithm descriptions.

2.4 Description of Analysed Systems

In the remaining of the section we introduce four algorithms that we select for testing: PAYL, POSEIDON, Anagram and McPAD. These algorithms are 1) general-purpose enough to be used with multiple application-level protocols, 2) proposed by often cited papers in the IDS community or 3) claiming to improve over the previous ones. Each algorithm requires as an input only the incoming network traffic, and does not perform any correlation between different packets.

PAYL. Wang and Stolfo in [36] present their 1-gram-based payload anomaly detector (PAYL). The system detects anomalies by combining 1-gram analysis algorithm with a classification method based on clustering of packet payload data length. The system employs a set of *models*: a model stores incrementally the resulting values of the 1-gram analysis for packet payloads of length l, thus each payload length has a different model. Each model stores two data series: mean byte frequency (i.e., relative byte frequencies span across several payloads of length l) and byte frequency standard deviation for each byte value (i.e., how relative byte frequencies change across payloads). During the detection phase, the same values are computed for incoming packets and then compared to model values: a significant difference from the model parameters produces an alert.

When PAYL fails to detect an attack. Fogla et al. [15] show that PAYL's detection can be evaded by mimicry attacks. PAYL is vulnerable to mimicry attacks since it models only 1-gram byte distributions. By carefully crafting an attack payload, an attacker is able to deceive the algorithm with additional bytes, which are useless to carry on the attack, but match the statistics of normal models.

POSEIDON. Bolzoni et al. present POSEIDON [6], a system built upon a modified PAYL architecture. PAYL uses data length field for choosing the right model. By contrast, POSEIDON employs a neural network to classify packets (and thus choose the most similar model) during the preprocessing phase. The authors use Self-Organizing Maps (SOMs) [19] to implement the unsupervised clustering. First, the full packet payload is analyzed by the SOM, which returns the value of the most similar neuron. That neuron model is then used for the calculation of byte frequency and standard deviation values, as in PAYL.

When POSEIDON fails to detect an attack. Differently from PAYL, POSEIDON is more resilient to mimicry attacks due to the combination of SOM and PAYL. The SOM analyzes the input by taking into consideration byte value at i-th position within the whole payload. Thus, extra bytes inserted by the attacker would be taken into consideration as well, resulting in a different classification than normal traffic. However, the granularity of the classification done by the SOM is coarse. Thus, if the attack portion of the sample payload is small enough, then the sample could be assigned to one of the clusters containing models of regular traffic, and may go unnoticed because of a similar byte frequency distribution.

Anagram. Wang et al. [35] present Anagram. The basic idea behind Anagram is that the usage of higher-order n-grams (i.e., n-grams where $n > 1$) helps to perform a more precise analysis. However, the memory needed to store average and standard deviation values for each n-gram grows exponentially (256^n, where n is the n-gram order). For instance, 640GB of memory would be needed to store 5-grams statistics. To solve this issue, the authors propose to use a binary-based n-gram analysis and store the occurred n-grams efficiently in a Bloom filters [5]. The binary-based approach implies a simple recording of the presence of distinct n-grams during training. Since less information is stored in the memory, it becomes possible to effectively use higher-order n-grams for the analysis. Authors show that this approach is more precise than the frequency-based analysis (e.g., used in PAYL) in the context of network data analysis. This is because higher-order n-grams are more sparse than low-order n-grams, and gathering accurate byte-frequency statistics becomes more difficult as the n-gram order increases.

When in detection mode, the current input is ranked using the number of previously unseen n-grams.

When Anagram fails to detect an attack. There are two main reasons why Anagram may fail to detect attack attempt. Firstly, the Bloom filter could saturate during training. This is because the user may underestimate the number of unique n-grams and allocates a small Bloom filter, during testing any n-gram would be considered as normal. Secondly, Anagram will likely miss the detection if the attack leverages a sequence of n-grams that have been observed during testing.

McPAD. Perdisci et al. present "Multiple-classifier Payload-based Anomaly Detector" (McPAD) [28] with a specific goal of an accurate detection of shellcode attacks. The authors use a modified version of the 2-gram analysis, combined with a group of one-class Support Vector Machine (SVM) classifiers [34]. The 2-gram analysis is performed by calculating the frequency of bytes that are

ν positions apart from each other. By contrast, a typical 2-gram analysis measures the frequency of 2 consecutive bytes. By varying the parameter ν, McPAD constructs several representation of the payload in different feature spaces. For example, for ν=*0..m*, McPAD builds m different representations of the packet payload. When in testing mode, a packet is flagged as anomalous if a combination (e.g., majority) of SVM outputs acknowledge the payload as anomalous.

When McPAD fails to detect an attack. By design, McPAD tries to give a wide representation of the payload (i.e. add more context by constructing byte pairs that are several positions apart). This may represent a difficulty in two cases. First, this is an approximate representation and that may imply a poorly described payload in case of slight differences between the training sample and an attack [1]. This may lead to a high false positive rate and a low detection rate. Secondly, McPAD uses different classifiers that have to come into an "agreement" to decide if a particular packet is anomalous or not. A problem may arise when, due to an approximate payload representation, several classifiers are misled by the byte pair representation and result in outvoting "correct" classifiers. In such case, the system might miss the detection.

3 Approach

We believe that one of the main reasons for poor performance of anomaly-based NIDS lies in the intrinsic limitation of commonly applied algorithm for content analysis: *n-gram analysis.* Since performing a comprehensive test to verify the ability of an IDS of identifying (all) attacks and to spot its weaknesses is unfeasible [22], we proceed to experimentally address our claim. We present a comparative analysis and evaluate the effectiveness of anomaly-based algorithms that analyse network payloads by using some form of n-gram analysis.

To verify the effectiveness of different algorithms we execute a number of steps: 1) collect network and attack data, 2) obtain a working implementation of each algorithm, 3) run the algorithms and analyse the results.

Obtaining the data. We acknowledge that optimal conditions for evaluating the performance of an IDS consist of running tests on unprocessed data from real networks [2]. Thus we first collect real-life data from different network environments, which are currently being operated. The past research is typically focused on benchmarking the algorithms with the HTTP protocol, although the authors do not explicitly restrict the scope of their algorithms to this protocol. We focus on the analysis of binary protocols. In particular, we analyse an example of a binary protocol found in a typical LAN (such as a Windows-based network service) and an example of a binary protocol typically found in an ICS.

Windows is the most used OS in the world, and every instance runs by default certain network services that are often used within LANs. For instance, the SMB/CIFS protocols [20] are used to exchange files between two computers, while other services (e.g. RPC) run on the top of it to provide additional functionalities. Although Windows systems are usually secured against abuses of such service from the Internet, corporate users take advantage of this feature quite often. An attacker that would develop an exploit for a zero-day vulnerability leveraging this protocol could potentially affect a large number of systems. In the last decade several malware [8,14] exploited SMB/CIFS to operate botnets and carry on other malicious activities.

As described in the introduction, ICS have lately become a valuable target for cyber attackers. Considering the sensitive character of such environments, the detection of cyber attacks plays a crucial role, sometimes even in homeland security. Lately, we have witnessed an increasing number of vulnerabilities discovered in software used in critical facilities, mainly due to the poor software development cycles that several vendors adopted, and the "security by obscurity" paradigm used to "protect" legacy devices.

We collect attacks in two different datasets. Our focus is on data injection attacks that have a high impact (see [27]).

Obtaining the implementations. Secondly, to carry out the benchmarks, we need working prototypes of all the algorithms we want to test. We could obtain an implementation of POSEIDON and McPAD from the authors. For the other two algorithms, we write our own implementation[1] based on the description found in the papers. To be sure that our implementation is correct, we need to verify that our results resemble the ones shown in the benchmarks of the respective original papers.

Analysing the results. The last step of our evaluation is the analysis of results with a focus on the identification of reasons for (un)successful detection.

3.1 Evaluation Criteria

The effectiveness of an IDS is mainly determined by the detection and false positive rates. The detection rate indicates the number of attack instances correctly identified by the IDS (true positives), w.r.t. the total number of attack instances. The false positive rate indicates the amount of samples that the IDS flags as attacks when they are actually not. False positives are a major limiting factor in this domain because, differently from other classification problems, their cost is high [4].

Detection rate. To provide a detailed overview of the detection capabilities of each algorithm, we consider both the number of correctly detected packets in the attack set and the number of detected attack instances. In fact, not all attacks show malicious payload within one single packet. Although an algorithm that exhibits a high per-packet detection rate has a higher chance of detecting attack instance, we do not argue that a low per-packet detection rate implies an equivalently low per-instance detection. In summary, we consider an alarm

[1] We intend to disclose the implementations in near future.

as a true positive if the algorithm is able to trigger at least one alert packet per attack instance.

False positive rate. The usual approach to document the performance of an IDS is to relate the false positive rate with the detection rate. This is done by drawing so called Receiver Operating Characteristic (ROC) curves. The benchmarks from the original papers of the proposed algorithms express the false positive rate as a percentage. However, such number has little meaning to the final users. A better way to express the false positive rate is in terms of the number of false positives per time unit. We establish two different thresholds: 10 false positives per day and 1 false positive per minute. The former value is proposed in [21] as a reasonable number for a user to maintain trust in the system. The latter is, in our opinion, the highest rate at which a human can verify alerts generated by an IDS. It is worth noting that anomaly-based IDSes, unlikely signature-based ones, do not provide information regarding the attack classification. Thus, the user might require additional time to investigate whether the alert is a true or a false positive. For each data set we compute these two thresholds based on the number of actual packets included in the verification sub data set after having split the original data set.

Since we do not make the data sets attack-free before hand, and thus some "noise" could have been collected as well, we need a way to verify that the alerts generated while processing the verification sub data set are actual false positives. To do that, we use a signature-based IDS (the popular open-source Snort), which is automatically fed with the network stream for which an alert was triggered during verification.

Commonly, in IDS evaluation papers the authors stop their analysis by reporting on the true and false positive rates. We believe that inspecting which attacks are detected, which are not detected and why, would provide useful information to fully understand when an algorithm could perform better than others (and for which threats). This kind of analysis can provide insights for future improvements. Finally we aim at evaluating the effectiveness of combining diverse algorithms to boost the detection rate.

4 Description of Network Data

In this section we describe in detail the background data set and attack data sets that will be used for benchmarking the detection algorithms. The chosen data sets comprise network traffic taken from two environments. We use the publicly available vulnerabilities and high impact exploits to run the tests.

4.1 Web Data Set

The following data sets are a collection of network traces of web traffic (in particular, the HTTP protocol).

DS_{DARPA}. The DARPA 1999 data set [21] is a standard data set used as reference by a number of researchers. Despite being anachronistic (and criticized in several works [23]), three of the algorithms we test have used this data set to compare their performance to previous works. Thus, we use the DARPA data set to verify that our own implementations of PAYL and Anagram offer comparable detection and false alert rates with the tests reported in the research papers of the detection algorithms.

AS_{HTTP}. This attack set is presented by Ingham and Inoue in [18], and has been used also by the authors of McPAD for their benchmarks. It comprises 66 diverse attacks, including 11 shellcodes, which were collected from public attack archives. The attacks are instances of buffer overows, input validation errors (other than buffer overflows), signed interpretation of unsigned values and URL decoding errors.

4.2 LAN Data Set

DS_{SMB} This data set includes network traces from a large University network. Samples have been collected through a week of observations. The average data rate of incoming and outgoing packets is ~40Mbps.

In particular, we focus on the SMB/CIFS protocol, and even more on SMB/CIFS messages which encapsulate RPC messages (see Section 5.2). The average packet rate for this traffic is ~22/sec. Based on this we calculate the false positive rate threshold for obtaining 10 alerts per day as 0.0005% and the one for obtaining 1 alert per minute as 0.073%.

AS_{SMB}. This attack data set is made of seven attack instances which exploit four different vulnerabilities in the Microsoft SMB/CIFS protocol: *ms04-011*, *ms06-040*, *ms08-067* and *ms10-061* [7].

ms04-011 is a vulnerability of certain Active Directory service functions in LSASRV.DLL of the Local Security Authority Subsystem Service (LSASS) of several Microsoft Windows versions. We select this vulnerability because it is used by the worm Sasser [26]. We collect two different attack instances for this vulnerability. One trace is downloaded from a public repository of network traces [13] where the attack payload is split in three fragments and contains a shellcode of 3320 bytes. The shellcode is made of a number of NOP instructions (byte value `0x90`), followed by valid x86 instructions and a sequence of the ASCII character '1'. The second instance is generated through the Metasploit framework [24]. The attack payload is split into three fragments and contains a shellcode of 8204 bytes to remotely launch a command shell in the victim host.

m06-040 is a vulnerability of the Microsoft Server RPC service. In particular, the vulnerability allows a stack overflow during the canonicalization of a network resource path. The specified path can be crafted to execute arbitrary code after the exploitation. We collect the attack instance from a public repository of network traces [13].

ms08-067 is a vulnerability of the Microsoft Server RPC service which exploits a similar weakness as the one described in *m06-040*, with the same effects. We select this vulnerability because it is used by Conficker [8] and Stuxnet, two high-impact pieces of malware. We collect two different attack instances for this vulnerability. One instance was downloaded from a public repository [13], while the other one was generated by us using the metasploit framework. In the first instance the payload contains a shellcode of 684 bytes, while in the second instance the shellcode is 305 bytes long only.

ms10-061 is a vulnerability of the PrintSpooler RPC service. When printer sharing is enabled, the PrintSpooler service does not properly validate spooler access permissions. Remote attackers can create files in a system directory, and consequently execute arbitrary code, by sending a crafted print request over RPC. We select this vulnerability because it was used by Stuxnet to successfully propagate in both regular backoffice LANs as well as in industrial control system environments. We collect two different attack instances for this vulnerability, both of them are generated through the metasploit framework. In one instance the attack payload is a binary file (the meterpreter executable), which accounts for 69832 bytes spanned over 18 fragments, while in the other instance the payload consists of a DLL file, which accounts for 1735 bytes spanned over three fragments.

4.3 ICS Data Set

DS_{Modbus} To test the anomaly detection algorithms on ICS networks we collect a data set of traces from the industrial control network of a real-world plant over 30 days of observation. The average throughput on this network is ∼800Kbps.

This data set includes network traces of one of the most common protocols used in such environments, Modbus/TCP [32]. Modbus was developed more than 30 years ago initially as a protocol used in serial channels, while the TCP/IP variant was introduced approximately 15 years ago to allow the serial protocol to be used in TCP/IP networks. Modbus/TCP features basic instructions and functions. Its structure is relatively simple, and operators of critical infrastructures usually repeat a limited set of operations, thus reducing the variability of the transmitted data. In fact, the maximum size of a Modbus/TCP message is 256 bytes. Thus, a Modbus/TCP message is always contained in one single TCP segment. Observations on DS_{Modbus} reveal that the average size of Modbus/TCP messages is 12.02 bytes (in DS_{SMB} it is 535.5 bytes). In the observed DS_{Modbus}, the number of duplicated TCP segment payloads is high (96.08%), in contrast to the other data set (e.g., DS_{SMB} has 31.37%). In other words, more than 9 TCP segments out of 10 carry a Modbus/TCP message that is a perfect duplicate of some other message already observed. The average packet rate for Modbus incoming traffic is ∼96/sec. Based on this observation, we calculate the false positive rate threshold for generating 10 alerts per day as 0.00012% and the threshold for generating 1 alert per minute as 0.017%.

AS_{Modbus}. The attack data set is made of 163 attack instances, which exploit diverse vulnerabilities of the Modbus/TCP implementations. There are fewer publicly known attacks against Modbus/TCP devices than SMB/CIFS attacks. Network traces for a good deal of these attacks can be downloaded from the website of an ICS security firm [11]. The exploited vulnerabilities can be categorised in two large families: *unauthorised use* and *protocol errors*.

Unauthorized use consist of two attack types: (a) "weird" clients talking to the Modbus server and (b) messages used only for diagnostics and special maintenance, which are thus seldom seen in the network traffic. By issuing these special messages, the attacker is often able to achieve a complete take over of the device.

Protocol errors are mainly fuzzing attempts against a device. For instance, these attacks are carried out by sending data not compliant with the protocol specifications (e.g. a too short protocol data unit). The outcome of such attacks can range from the unavailability of the device up to the control of the execution flow (see Cui and Stolfo [9] for a more detailed discussion).

5 Benchmarks

In this section we show the results of our benchmarks and compare the performances of the algorithms for each data set.

Setting up and tuning the algorithms. For each dataset we split the background traffic into two sets, one for training and one for verification. The splitting is performed randomly, by sampling the network streams. Each split sub data set accounts for nearly 50% of the original data. The training sub data set is used to build the detection profiles for each algorithm, while the verification one is used to evaluate the number of false positives raised by the algorithm. Finally, we run the algorithms on the attack data set.

For performing the benchmarks we need to set up several starting parameters for each algorithm.

PAYL. As introduced in the original paper, the size of the PAYL model can be reduced by merging profiles when their number becomes too large. For each data set we perform several runs using different values of the merging parameter. Finally, for the DS_{DARPA} data set we set the parameter value to 0.12. For the remaining data sets we do not merge profiles, as the total number of profiles remains low (up to 150, compared to 480 in DS_{DARPA}). As the "smoothing factor", we use the same value (0.001) suggested by Ingham and Inoue in [18]. We apply the algorithm to individual TCP segment payloads.

Anagram. For each data set we run tests with different n-gram sizes (n size of 3, 5, 7, 9 and 12). Since we obtain the best results with the 3-grams, we set this as the standard n-gram size. As in the original paper, we set the size of the Bloom filter to 2MB. We do not use the "bad content model" proposed by the authors because it would be ineffective as they build it with virus samples, and our attack data sets do not include viruses.

POSEIDON. For all tests we use a SOM with fixed number of neurones (96). Also, we set the number of instances for training the SOM at 10000.

McPAD. For all tests we use the best performing parameters as proposed in [28]. Those are: number of clusters $k = 160$, desired false positive rate for each SVM classifier set to 1% and *maximum probability* as combination rule for the output of the SVM classifiers.

For each algorithm we vary the value of the threshold to observe how the false positive and detection rates change.

5.1 Implementation Verification

To verify the correctness of our implementations we run initial tests using DS_{DARPA} data set (since that was the only common data set used in 3 original algorithm benchmarks). Instead of using DARPA attack dataset, we use the AS_{HTTP} for testing. There are two main reasons for this: (1) the original attack instances of the DARPA data set do not reflect at all modern attacks, and (2) not all the algorithms have been benchmarked against the attack set of DARPA (Anagram is the exception). Thus, it would be impossible to faithfully reproduce the previous experiments. In Table 1 we summarize the results of this first round of benchmarks: for each algorithm we report the highest detection rate we achieve, and the corresponding false positive rate.

Table 1. Test results on DS_{DARPA} and testing with AS_{HTTP}

	FPR (packet-based)	DR (packet-based)
PAYL	0.00%	90.73%
POSEIDON	0.004%	92.00%
Anagram	**0.00%**	**100.00%**
McPAD	0.33%	87.80%

All the algorithms show high detection and low false positive rates. When compared to the original papers, these results match our expectations, thus we can be reasonably sure that our re-implementations are not (too) dissimilar from the original ones in terms of completeness and accuracy.

5.2 Tests with LAN Data Set

We first perform the test on DS_{SMB} by using all the SMB/CIFS packets directed to the TCP ports 139 or 445. However, none of the algorithm can perform well enough under these conditions. For example, the Bloom filter used by Anagram saturates with 3-grams during the training phase. Consequently, no attack instance is further detected, even with the lowest threshold. Increasing either the size of n-grams or the size of the Bloom filter cannot solve the issue of undetected attacks. In the former case, the Bloom filter saturates even with fewer training packets. In the latter case, even with a filter size of 20MB (10 times bigger than the size suggested by the authors) which does not cause full saturation, no attack is detected with false positives rates lower than 4%. Other algorithms exhibit similar detection problems, with at least one attack instance detected only with false positive rates higher than 1%.

This result alone would imply a complete failure for this protocol. We believe the reason why the algorithms poorly perform on SMB/CIFS is because of the high variability of the analyzed payload. In fact, SMB/CIFS is mainly used to transfer files between Windows computers. Such files are contained in the request messages processed by the algorithms and can be of any type, from simple text files to compressed binary archives or even encrypted data.

We acknowledge that all attack instances publicly available exploit vulnerabilities of the Windows RPC service by leveraging SMB/CIFS, which can encapsulate RPC messages. Thus, we re-run the test on a filtered data set. In particular, we extract only SMB/CIFS messages that carry RPC data (~1% of the original SMB/CIFS traffic). By doing this, we are implicitly providing a semantical hint to the algorithms. We expect this to improve both the detection and false positive rates.

The results of this round of tests are summarized on Table 2. Anagram achieves a 0.00% false positive rate while still being able to detect 3 attack instances. Anagram also achieves the lowest false positive rate among the tested algorithms when detecting all of the attack instances, a rate lower than the adjusted false positive of 1 alert per minute. McPAD generates the highest false positive rate, and it is impossible to lower that no matter which combination of parameters we choose. We believe that the main reason for this lies in the fact that McPAD implements the approximate payload representation, that, in such variable conditions, provides poor payload description.

There is no need to evaluate how a combined approach, i.e., using all the algorithms simultaneously, would perform since Anagram is already outperforming the other algorithms.

Finally, we process all false positives with SNORT to verify that none of them is actually a true positive.

Analysis of detected and undetected attacks. By considering which attack instance is detected the most, we observe that all the algorithms can detect an attack instance exploiting the *ms04-011* vulnerability. In particular, of the three segments composing the attack payload, only one is always identified as anomalous. Here we report a fragment of it:

```
0230   59 35 1c 59 ec 60 c8 cb cf ca 66 4b c3 c0 32 7b  Y5.Y.‘....fK..2{
0240   77 aa 59 5a 71 76 67 66 66 de fc ed c9 eb f6 fa  w.YZqvgff.......
0250   d8 fd fd eb fc ea ea 99 da eb fc f8 ed fc c9 eb  ................
0260   f6 fa fc ea ea d8 99 dc e1 f0 ed cd f1 eb fc f8  ................
0270   fd 99 d5 f6 f8 fd d5 f0 fb eb f8 eb e0 d8 99 ee  ................
0280   ea ab c6 aa ab 99 ce ca d8 ca f6 fa f2 fc ed d8  ................
0290   99 fb f0 f7 fd 99 f5 f0 ea ed fc f7 99 f8 fa fa  ................
02a0   fc e9 ed 99 fa f5 f6 ea fc ea f6 fa f2 fc ed 99  ................
02b0   90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
02c0   90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
02d0   90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
02e0   90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
02f0   90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0300   90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
...
05a0   90 90 90 90 90 90 90 90                          ........
```

Table 2. Test results on DS_{SMB} and testing with AS_{SMB}

	FPR (packet-based)	DR (packet-based)	DR (instance-based)
PAYL	0.004%	1.43%	2/7
	0.007%	3.35%	3/7
	0.01%	6.65%	4/7
	0.05%	23.92%	5/7
	4.41%	66.51%	6/7
	11.05%	85.02%	7/7
POSEIDON	0.007%	6.22%	2/7
	0.007%	10.04%	3/7
	1.27%	37.32%	4/7
	2.28%	50.23%	6/7
	3.32%	58.37%	6/7
	5.39%	66.98%	7/7
Anagram	0.00%	0.95%	2/7
	0.00%	22.48%	3/7
	0.02%	37.32%	7/7
	2.34%	55.50%	7/7
	8.39%	63.64%	7/7
McPAD	19.02%	60.38%	7/7
	20.62%	60.86%	7/7
	22.31%	61.35%	7/7
	27.61%	65.21%	7/7
	97.39%	90.00%	7/7

We verify that this particular segment is flagged as anomalous by all the algorithms because of the long sequence of the byte value `0x90`, which corresponds to the NOP instruction of the shellcode. Although there are individual bytes with value `0x90` in the training set, we verify that there is never a sequence of three bytes with this value. This explains why Anagram can identify as anomalous a vast majority of the 3-grams in the aforementioned payload. Also PAYL and POSEIDON will identify an anomalous byte frequency distribution, in which the byte value `0x90` peaks above all the others. Similarly, the fact that the payload consists of continuous `0x90` bytes implies that McPAD classifiers will be able to recognize the peak in the frequency of these 2-grams.

We also observe that both PAYL and POSEIDON fail to detect one attack instance exploiting the *ms06-040* vulnerability, when the false positive rate is below 2%. A fragment of the payload of such attack instance is reported below:

```
0170   52 df 47 2c 0c 86 de fe fe b9 f6 68 ae 23 4f a5  R.G,.......h.#O.
0180   81 53 79 43 fc fc 31 58 af ad 6e 30 29 f7 50 8a  .SyC..1X..n0).P.
0190   4a e1 78 43 30 6a 55 75 58 72 6e 4f 42 48 4c 42  J.xC0jUuXrnOBHLB
01a0   36 34 33 4a 51 38 69 42 52 37 39 46 59 49 79 71  643JQ8iBR79FYIyq
01b0   7a 62 38 48 4e 47 68 48 7a 52 59 6e 38 76 55 78  zb8HNGhHzRYn8vUx
01c0   41 4d 57 61 57 66 68 30 48 4c 30 61 76 73 61 6b  AMWaWfh0HL0avsak
01d0   7a 56 65 4d 32 42 76 64 64 43 64 41 45 75 75 53  zVeM2BvddCdAEuuS
01e0   4f 7a 41 4f 70 6b 30 37 4c 45 70 66 73 44 73 49  OzAOpk07LEpfsDsI
01f0   66 57 39 65 31 59 45 6e 43 38 52 62 76 57 65 50  fW9e1YEnC8RbvWeP
0200   59 63 54 77 7a 63 32 4f 50 4f 52 6b 71 4c 33 4b  YcTwzc2OPORkqL3K
0210   65 7a 69 62 72 57 4e 6d 58 33 4b 56 70 50 6c 45  ezibrWNmX3KVpPlE
```

This fragment contains a large majority of printable characters, thus one would expect that, since RPC over SMB/CIFS messages are mostly binary, a detection algorithm based on byte frequency distribution would be able to detect such payload. However, RPC over SMB/CIFS is also used to feed remote print spoolers with files to print. For example, in the filtered data set used for training we can identify the following fragment which is part of a PDF file sent to the spooler:

```
0200   09 2f 40 6f 70 53 74 61 63 6b 4c 65 76 65 6c 20  ./@opStackLevel
0210   40 6f 70 53 74 61 63 6b 4c 65 76 65 6c 20 31 20  @opStackLevel 1
0220   61 64 64 20 64 65 66 0d 0a 09 09 63 6f 75 6e 74  add def....count
0230   64 69 63 74 73 74 61 63 6b 20 31 20 73 75 62 0d  dictstack 1 sub.
0240   0a 09 09 40 64 69 63 74 53 74 61 63 6b 43 6f 75  ...@dictStackCou
0250   6e 74 42 79 4c 65 76 65 6c 20 65 78 63 68 20 40  ntByLevel exch @
0260   64 69 63 74 53 74 61 63 6b 4c 65 76 65 6c 20 65  dictStackLevel e
0270   78 63 68 20 70 75 74 0d 0a 09 09 2f 40 64 69 63  xch put..../@dic
0280   74 53 74 61 63 6b 4c 65 76 65 6c 20 40 64 69 63  tStackLevel @dic
0290   74 53 74 61 63 6b 4c 65 76 65 6c 20 31 20 61 64  tStackLevel 1 ad
```

Similar to the previous fragment, this fragment also contains a vast majority of printable characters. Due to the high variability of data, the threshold for detecting such fragment had to be set in such a way that many normal samples were classified as anomalous.

5.3 Tests with ICS Data Set

The results of this round of tests are summarized on Table 3. Anagram shows outstanding results in this test, and this does not come unexpected. The messages in this data set are rather short and the number of duplicates is also high (more than 95%). This is a perfect combination for Anagram and its binary-based approach. Anagram detects most of attack instances with a false positive rate that is lower than the adjusted false positive rate of 10 alerts per day. When detecting all of the attack instances, the false positive rate is still one order of magnitude lower than the adjusted false positive rate of 1 alert per minute.

McPAD also performs well with respect to the false positive. This is expected because the analysed Modbus traffic is expressed in messages of fixed length structure and with a limited range of values in used bytes. This results in McPAD accurately modeling relationships in the message structure.

PAYL seems to have a better packet-rate detection rate than POSEIDON. However, POSEIDON always performs better with respect to the instance-based detection rate as well as lower false positive. When the two algorithms are tuned to detect all of the attack instances, they both generate an overwhelming number of false positives, triggering on almost every packet.

With respect to the false positives generated during the verification phase, no raised alert turned out to be a true positive when processed with Snort. This is largely expected due to 1) the small number of available signatures for the Modbus protocol, and 2) the fact that the industrial control network from which we collected traffic from is highly isolated from other networks, with only a fixed number of hosts communicating. Thus, the chance of "noise" is substantially low.

Similarly to the previous test, there is no reason to test how a combination of algorithms would perform, because Anagram outperforms all the other algorithms in terms of detection and false positive rates.

Table 3. Test results on DS_{Modbus} and testing with AS_{Modbus}

	FPR (packet-based)	DR (packet-based)	DR (instance-based)
PAYL	0.00%	62.57%	101/163
	0.00%	92.63%	150/163
	9.25%	95.70%	155/163
	95.00%	100.00%	163/163
POSEIDON	0.04%	3.07%	4/163
	0.07%	77.30%	125/163
	0.37%	95.09%	154/163
	7.65%	97.54%	158/163
	97.81%	100.00%	163/163
Anagram	0.00%	3.68%	5/163
	0.00005%	10.42%	16/163
	0.00007%	18.40%	29/163
	0.00007%	96.93%	157/163
	0.002%	100.00%	163/163
McPAD	0.041	6.31%	10/163
	0.044	96.93%	157/163
	0.045	96.93%	157/163
	0.046	96.93%	157/163

Analysis of detected and undetected attacks. To understand why Anagram works so well with Modbus traffic consider the following two Modbus messages. The first one is a valid read request (identified by the 8th byte with value `0x03` which corresponds to the request "function code"):

```
35 ae 00 00 00 06 00 03 0c 7f 00 64
```

The following fragment is an attack instance attempting to corrupt the PLC memory by invoking a vulnerable diagnostic function (byte value `0x08`) with invalid data (bytes `0x00 0x04 0x00 0x00`):

```
00 00 00 00 00 06 0a 08 00 04 00 00
```

We first observe that the anomalous value in this payload are the byte value of the function code, and the subsequent four bytes (never observed in the training set on that same positions). There are 6 3-grams over 10 (60%) which are not present in the valid request. The number of distinct 3-grams observed during training is not much bigger than the one observable in the aforementioned request, due to the large number of duplicated payloads. Thus, with such a small packet size, even a few bytes with unusual value can make a big difference.

On the other hand, from the results in Table 3 we see that for all the algorithms there is always a significant increase in the amount of false positives raised when the threshold is adjusted to detect all attack instances. We observe that the attack instances that do not get detected before the threshold is adjusted are similar to the one in the following example:

```
00 00 00 00 00 02 0a 11
```

This 8 bytes long message is the smallest possible Modbus message allowed by the protocol specification. We acknowledge that this request is not unusual only because of its size, but also because the function code value `0x02` (corresponding to the request "report slave ID") was never observed during training. We have verified that the only 3-gram in this payload not observed during the training is the last one `0x02 0x0a 0x11`. Thus, in spite of the small size of this payload, the threshold has to be lowered in order to detect it. Detecting such anomalous packet (with only one anomalous n-gram) in a bigger message would be much more difficult.

6 Related Work

To the best of our knowledge, Ingham and Inoue describe the most recent framework for testing the performance of IDS algorithms [18]. The authors focus on HTTP traffic. The framework is based on the general principle that testing different IDS algorithms on the same network environment and with the same network and attack data allows a better comparison of the algorithms' performance, which would be impossible by re-using the results of the unrelated, individual tests run by the algorithm developers. They collect background web traffic from four different websites and create a publicly available set of network traces. The traces contain instances of web attacks generated by running exploitation tools downloaded from popular vulnerability repositories (e.g., BugTraq, SecurityFocus and the Open Source Vulnerability Database). There are other IDS evaluation frameworks, as reported in [18], but are all quite dated.

In [31] Song et al. show that polymorphic behaviour in shellcodes is too greatly spread to be modeled effectively. The authors note that it is difficult to model data with high variability, especially in case the adversary is able to inject some "normal" looking n-grams into the attack payload to make it look legitimate. Our experiments with SMB traffic confirm that, not only this observation is true for polymorphic shellcode traffic, but even for regular attacks that do not leverage any evasion techniques.

7 Conclusion

In this paper we present a thorough analysis of several n-gram-based algorithms for network-based anomaly detection. We investigate the performance of state-of-the-art detection algorithms when analyzing network traffic from two binary protocols. We believe our analysis allows us to draw interesting observations and conclusions.

First, despite the fact that the attack instances on the SMB/CIFS protocols are correctly detected, all studied algorithms incur in a high penalty in terms of false positives they raise. Concretely, it would be expensive to deploy them independently in a real environment. On the other hand, if we restrict the field to the Modbus protocol alone, Anagram detects almost every attack instance with a rate of false positives lower than the 10 alerts per day threshold.

We believe that with such performance the algorithm could be deployed in a real environment.

Second, all studied algorithms trigger on the exploitation payload. We can observe this by selecting two different attack instances that exploit the same vulnerability, but using two different attack payloads. In several cases, while one instance is detected even with a low threshold, to detect both attacks one needs to increase the threshold significantly. The previously missed attack instance usually contains a small-size attack payload, and thus "blends" more easily with the normal payload data, thereby avoiding detection.

Third, there is no absolute best algorithm among the ones we studied. Anagram performs slightly better than the rest when analyzing the filtered SMB/CIFS and the Modbus protocols, but it is also the one performing worst when the filter is not applied. Technically, this is due to the fact that the unfiltered SMB/CIFS traffic contains several n-grams that are also present in the attack payloads. This supports the intuition that variability of the network traffic has a great impact on the performance of these systems. Indeed, *every* studied algorithm is affected by this, allowing us to conclude that, rather than the single implementation, it is the underlying principle of capturing regularity in the unstructured packet payload that does not hold true. Our results show that n-gram analysis quickly becomes incapable of capturing relevant content features when analysing moderately variable traffic. This problem could be partly alleviated by deploying the detection system in combination with some other sensor that will verify the correctness of alerts [35]. We believe that a more promising approach is the one focusing on identifying chunks of payload (that represent some kind of semantic units) and applying the n-gram analysis on those. For example, several authors propose to exploit the syntactical knowledge of the HTTP protocol to improve the overall performance of anomaly-based systems, e.g. in [30]. We foresee that a similar approach could be applied to binary protocols as well. Another issue that remains still open is how to "measure" traffic variability without having to run several empirical experiments.

Acknowledgements. We thank Davide Ariu for providing the McPAD source code and supporting during tests. The research leading to these results has been supported by the Ministry of Security and Justice of the Kingdom of the Netherlands through projects Hermes, Castor and Midas and by the European Commission through project FP7-SEC-285477-CRISALIS funded by the 7th Framework Program.

References

1. Ariu, D., Tronci, R., Giacinto, G.: HMMPayl: An intrusion detection system based on Hidden Markov Models. Computers and Security 30(4), 221–241 (2011)
2. Athanasiades, N., Abler, R., Levine, J., Owen, H., Riley, G.: Intrusion Detection Testing and Benchmarking Methodologies. In: IWIA 2003: Proc. 1st IEEE International Workshop on Information Assurance, pp. 63–72. IEEE Computer Society Press (2003)

3. Auriemma, L.: Advisories (March 2011), http://aluigi.altervista.org/ (accessed March 2012)
4. Axelsson, S.: The base-rate fallacy and the difficulty of intrusion detection. ACM Transactions on Information and System Security 3(3), 186–205 (2000)
5. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM 13(7), 422–426 (1970)
6. Bolzoni, D., Zambon, E., Etalle, S., Hartel, P.H.: POSEIDON: a 2-tier Anomaly-based Network Intrusion Detection System. In: IWIA 2006: Proc. 4th IEEE International Workshop on Information Assurance, pp. 144–156. IEEE Computer Society Press (2006)
7. Microsoft Security Response Center. Microsoft Security Bulletin, http://technet.microsoft.com/en-us/security/bulletin/ (accessed March 2012)
8. Microsoft Security Response Center. Conficker Worm: Help Protect Windows from Conficker (April 2009), http://technet.microsoft.com/en-us/security/dd452420.aspx (accessed March 2012)
9. Cui, A., Stolfo, S.J.: Defending Embedded Systems with Software Symbiotes. In: Sommer, R., Balzarotti, D., Maier, G. (eds.) RAID 2011. LNCS, vol. 6961, pp. 358–377. Springer, Heidelberg (2011)
10. Damashek, M.: Gauging similarity with n-grams: Language-independent categorization of text. Science 267(5199), 843–848 (1995)
11. Digital Bond, Inc. QuickDraw SCADA IDS, http://www.digitalbond.com/tools/quickdraw/ (accessed March 2012)
12. Dussel, P., Gehl, C., Laskov, P., Busser, J., Störmann, C., Kästner, J.: Cyber-Critical Infrastructure Protection Using Real-Time Payload-Based Anomaly Detection. In: Rome, E., Bloomfield, R. (eds.) CRITIS 2009. LNCS, vol. 6027, pp. 85–97. Springer, Heidelberg (2010)
13. Mu Dynamics. pcapr, http://pcapr.net (accessed March 2012)
14. Falliere, N., Murchu, L.O., Chien, E.: W32.Stuxnet Dossier. Technical report, Symantec (September 2010)
15. Fogla, P., Sharif, M., Perdisci, R., Kolesnikov, O., Lee, W.: Polymorphic blending attacks. In: Proc. 15th USENIX Security Symposium, pp. 241–256. USENIX Association (2006)
16. Forrest, S., Hofmeyr, S.A.: A Sense of Self for Unix Processes. In: S&P 1996: Proc. 17th IEEE Symposium on Security and Privacy, pp. 120–128. IEEE Computer Society Press (2002)
17. Gu, G., Porras, P., Yegneswaran, V., Fong, M., Lee, W.: BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation. In: Proc. 16th USENIX Security Symposium (Security 2007). USENIX Association (2007)
18. Ingham, K.L., Inoue, H.: Comparing Anomaly Detection Techniques for HTTP. In: Kruegel, C., Lippmann, R., Clark, A. (eds.) RAID 2007. LNCS, vol. 4637, pp. 42–62. Springer, Heidelberg (2007)
19. Kohonen, T.: Self-Organizing Maps, Second Extended Edition. Springer Series in Information Sciences, vol. 30. Springer (1995)
20. MSDN Library. [MS-CIFS]: Common Internet File System (CIFS) Protocol Specification, http://msdn.microsoft.com/en-us/library/ee442092v=prot.13.aspx (accessed March 2012)
21. Lippmann, R.P., Haines, J.W., Fried, D.J., Korba, J., Das, K.: The 1999 DARPA off-line intrusion detection evaluation. Computer Networks: The International Journal of Computer and Telecommunications Networking 34(4), 579–595 (2000)

22. Loscocco, P.A., Smalley, S.D., Muckelbauer, P.A., Taylor, R.C., Turner, S.J., Farrell, J.F.: The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments. In: NISSC 1998: Proc. 21st National Information Systems Security Conference, pp. 303–314 (1998)
23. Mahoney, M.V., Chan, P.K.: An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection. In: Vigna, G., Kruegel, C., Jonsson, E. (eds.) RAID 2003. LNCS, vol. 2820, pp. 220–237. Springer, Heidelberg (2003)
24. Metasploit Penetration Testing Software, `http://metasploit.com/` (accessed March 2012)
25. Mirkovic, J., Reiher, P.: A taxonomy of DDoS attack and DDoS defense mechanisms. SIGCOMM Comput. Commun. Rev. 34, 39–53 (2004)
26. Nakayama, T.: W32.Sasser.Worm. Technical report, Symantec (April 2004)
27. NIST: National Institute of Standards and Technologies. National Vulnerability Database, `http://nvd.nist.gov` (accessed March 2012)
28. Perdisci, R., Ariu, D., Fogla, P., Giacinto, G., Lee, W.: McPAD: A multiple classifier system for accurate payload-based anomaly detection. Computer Networks 53(6), 864–881 (2009)
29. Sommer, R., Paxson, V.: Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In: S&P 2010: Proc. 31st IEEE Symposium on Security and Privacy, pp. 305–316. IEEE Computer Society (2010)
30. Song, Y., Stolfo, S.J., Keromytis, A.D.: Spectrogram: A Mixture-of-Markov-Chains Model for Anomaly Detection in Web Traffic. In: NDSS 2009: Proc. 16th ISOC Symposium on Network and Distributed Systems Security. The Internet Society (2009)
31. Song, Y., Locasto, M.E., Stavrou, A., Keromytis, A.D., Stolfo, S.J.: On the infeasibility of modeling polymorphic shellcode. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS 2007, pp. 541–551. ACM, New York (2007)
32. Swales, A.: Open MODBUS/TCP Specification (March 1999)
33. The OWASP Foundation. OWASP: The Open Source Web Application Security Project, `https://www.owasp.org` (accessed March 2012)
34. Vapnik, V.N., Lerner, A.: Pattern recognition using generalized portrait method. Automation and Remote Control 24 (1963)
35. Wang, K., Parekh, J.J., Stolfo, S.J.: Anagram: A Content Anomaly Detector Resistant to Mimicry Attack. In: Zamboni, D., Kruegel, C. (eds.) RAID 2006. LNCS, vol. 4219, pp. 226–248. Springer, Heidelberg (2006)
36. Wang, K., Stolfo, S.J.: Anomalous Payload-Based Network Intrusion Detection. In: Jonsson, E., Valdes, A., Almgren, M. (eds.) RAID 2004. LNCS, vol. 3224, pp. 203–222. Springer, Heidelberg (2004)