Complete Characterization of Security Protocols by Pattern Refinement*
(Work in Progress)

Cas Cremers

ETH Zurich, Dept. of Computer Science
Zurich, Switzerland
cremersc@inf.ethz.ch

Abstract. Recently, the notion of *complete characterizations* of security protocols was introduced by Guttman and Thayer. We provide an alternative definition of this concept, and extend an existing protocol verification tool (Scyther) to compute our notion of complete characterization. We present both notions of complete characterization, discuss their relative merits, and provide preliminary empirical results using an extended version of the Scyther tool.

Keywords. Security Protocols, Analysis, Complete Characterization, Tools

1 Introduction

Many tools exist to verify whether or not a given security protocol satisfies a specific property, e.g. [1–3]. In general, recent developments in security protocol analysis tools have focussed on improving performance, in order to improve the coverage of the analysis [4], or to allow for verification of larger protocols [3].

A protocol designer using such tools, upon finding an attack, needs to locate the fault in the protocol that caused the attack, and repair it. The repaired protocol is then again verified by the tool, until no more attacks are found. Hence it is crucial to be able to understand how the protocol works (or why it doesn't) from the output of such a tool. However, in general these tools share some major drawbacks with other counterexample generating analysis methods: the counterexample is often not minimal, and it is usually not clear how to find the fault that causes it. Furthermore, if the particular counterexample is in some sense not harmful, or will not occur in practical applications, the tools do not provide a means to show all possible counterexamples/attacks (because this set is usually in the order of the size of the state space, e.g. exponential or infinite).

An alternative approach for analysing and verifying protocols was started in [5–7]. In this approach, the idea is not to verify a specific property, but to give a concise finite representation of all possible behaviours of a security protocol,

* This work was supported by the Hasler Foundation, under ManCom project 2071.

in such a way that certain classes of security properties can be trivially verified. Such a finite representation is called the *complete characterization* of a security protocol. A tool called CPSA[1] (Cryptographic Protocol Shape Analyzer) was developed to compute the complete characterizations. The approach is described in e.g. [5] and uses and extended version of the Strand Space method [8], in which the finite representations are expressed in terms of *shapes* using the so-called *skeletons-and-homomorphisms* approach. We describe these concepts in Section 2, and provide a brief introductory example below.

Example 1 (Needham-Schroeder (Skeletons, CPSA)). Consider the two-party version of the Needham-Schroeder protocol. From the point of view of a responder Bob, upon the completion of the protocol, the following must have happened: there must be an honest agent that has performed the initiator role of the protocol, and received Bob's nonce. However, this agent may or may not have been communicating with Bob (representing either normal behaviour or the man-in-the-middle attack). This message flow is represented below in Figure 1. Hence, if we focus on honest agent behaviour only, the complete characterization of the responder role is captured by a single message flow.

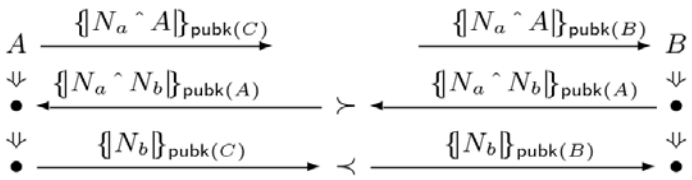

Fig. 1. Needham-Schroeder Shape for B ($privk(A)$ uncompromised, N_b fresh)

This single message flow captures the following execution traces: (1) assume the private key of the initiator's intended partner is known to the intruder (in case $privk(C)$ is known to the intruder), and (2) assume this private key is not known to the intruder, and infer that the intended partner must be Bob (in case $C = B$).

In this paper we show how to achieve an alternative form of complete characterization, using a slightly different choice of finite representation for the possible behaviours of the protocol, in terms of *realizable patterns*. We modify an existing protocol verification tool, called Scyther[2] [2], to compute these alternative complete characterizations.

Example 2 (Needham-Schroeder (Patterns, Scyther)). Consider again the two-party version of the Needham-Schroeder protocol. From the point of view of a responder Bob, upon the completion of the protocol, the following must have

[1] At the moment of writing this, CPSA is not available for download.
[2] Scyther can be downloaded at http://people.inf.ethz.ch/cremersc/scyther/index.html

happened: there must be an honest agent that has performed the initiator role of the protocol, and received Bob's nonce. If we take intruder actions into account, there are two distinct cases: either the intruder knows the secret key of the partner of this agent, or this agent is talking to Bob. In this view, the complete characterization of the responder role is captured by two distinct message flows: First, the message flow in which the intruder knows the secret key of the initiator's supposed partner, and the intruder performs decryption and encryption of messages. Second, the message flow in which the protocol runs as expected.

Hence, in this more fine-grained approach we distinguish between the two sub-cases of the first example, and there is no need to infer agent equalities to unfold the execution traces.

In this paper we discuss the relation between the two representations. In short, the shapes approach provides a slightly more coarse, and therefore sometimes more concise representation than the pattern approach, but has a more implicit relation with the actual behaviours of the protocol.

Furthermore, we provide extensive empirical results. We compare these results to the currently reported results of the CPSA tool, revealing three advantages of the Scyther approach. First, the pattern refinement algorithm allows for a wider range of protocols to be characterized. Second, if complete characterization is not achieved for a protocol, the pattern output still provides insight into the protocol behaviour. Third, Scyther is more efficient at computing the characterizations.

Purpose of the paper In this paper, we aim at providing a high-level idea of the concept of complete characterization, and our novel way of computing it. We report on our current work-in-progress and provide preliminary experimental results. We will omit technical details and explain by example where possible. The full technical details will be explained in the full paper.

Outline of the paper We start off by giving a high-level sketch of the complete characterization concept, and the shapes and skeletons theory, in Section 2. We describe the alternative characterization in terms of patterns in Section 3. We provide a preliminary empirical analysis in Section 4 and conclude in Section 5.

2 Complete characterizations as shapes

The concept of complete characterization was introduced in [6]. The idea is to consider a role of a particular role of a protocol, and give a concise finite representation of all essentially different possible behaviours of the protocol that include this particular role. Intuitively, the complete characterization provides a concise answer to the question "which other events must also have occurred, if we assume that this role was executed?"

Strand spaces In this paper we omit technical details. We therefore assume the reader is at least to some extent familiar with the basic of the Strand Space method as described in e.g. [8]. There is one property we recall here, which concerns capturing the generation of fresh terms (nonces). The Strand Space method does not distinguish between different terms (except for having an explicit key set), but Nonce generation is captured by the concept of *unique origination*: a term t can be said to uniquely originate from node n. This effectively connects nonces to the first send event in the role instance in which they occur as a subterm. A set of such bindings of terms to a specific event is called the *unique origination set*, and is written as unique.

Skeletons and homomorphisms Until recently, the Strand Space model was used only to model possible execution traces of security protocols in general: as there are no notions of e.g. protocol or variable instantiation in the original Strand Space model. Hence, in order to be able to specify a protocol, and its relation to protocol execution, modifications to the strand space model are introduced.

In the skeletons and homomorphisms approach, the strand space approach is extended with a number of new notions.

First, in the original Strand Space method there is no way to indicate which agents are honest, which in this method boils down to indicating which private keys are not known to the intruder. In the skeletons and homomorphisms approach, there is an explicit set of private keys not known to the intruder, written as non.

Second, the concept of variable instantiation is replaced by the notion of *information preserving homomorphisms*. In e.g. process calculus approaches, role descriptions contain variables, which are instantiated in each of its possible executions. In the Strand Space method no such variables occur. Instead, the protocol contains terms, which can be substituted by other (atomic) terms, as long as the substitution is said to be information-preserving with respect to the unique origination set unique, as well as the private key set non.

In the strand spaces terminology, a set of strands with a partial ordering is referred to as a bundle. However, in the shapes setting, one only considers regular strands (and thus only the behaviour of honest agents, and not that of the intruder). As a result, a *skeleton* can correspond to the regular part of a bundle, extended with the information in the associated nonand uniquesets: it consists of a four tuple $(nodes, \leq, \mathsf{non}, \mathsf{unique})$. In particular, a skeleton is said to be *realizable* if it corresponds to an execution trace of the protocol when all agents whose keys are not in unique are considered to be compromised. Finally, *shapes* are minimal realizable skeletons.

Example 3. In Figure 1 we have that $\mathsf{unique} = \{N_b\}$ and $\mathsf{non} = \{privk(A)\}$. The figure is a realizable skeleton (execution trace if we assume C to be compromised), and also a shape.

Complete characterization In terms of the notions described above, the *complete characterization* of a protocol role is the set of all shapes of this protocol. In other

words, if the role is executed in a particular trace, then one of the shapes will also occur in that trace. The complete characterization of a protocol is the complete characterization of each of its roles.

In general, most protocols only have only very few shapes. We have seen the responder role of the Needham-Shroeder protocol has only one shape. In general, protocol roles of correct protocols (i.e. that satisfy strong authentication properties) have usually only one shape. However, in theory protocols may also have an infinite amount of shapes.

If finite, the complete characterization allows for easy verification of a large class of authentication properties, e.g. if every shape of the responder role contains an initiator role, then existence of an initiator is guaranteed for all executions of the responder role. If one shape does not have an initiator role, it represents a class of counterexamples.

3 Complete characterization by pattern refinement

In this section we describe an alternative approach to achieve a complete characterization of a security protocols. This new approach uses patterns instead of skeletons.

The skeletons, as described in the previous section, fix the behaviour of the honest agents, and do not describe the intruder behaviour explicitly. Rather, a realizable skeleton implies that there exists intruder behaviour such that the events can be executed as described.

Patterns A pattern is a partially ordered set of protocol events. These protocol events may contain type variables, e.g.

$$send(A, R\sharp 0, V\sharp 0)\sharp 0$$

where $R\sharp 0$ is of type Agent and $V\sharp 0$ of type Message. This event represents any send event by the agent A in thread 0. In general, the notation $X\sharp Y$ denotes that the event or term X is local to the thready Y.

Realizable patterns A pattern is said to be *realizable* if for any substitution that meets the variable typing constraints, and any linearization of the partial order, the sequence of resulting events forms a valid execution trace of the protocol.

Given a pattern pt of a protocol P, the pattern refinement algorithm in Scyther [2] computes a set of realizable patterns such that the union of the traces of the protocol with the realizable patterns, is equal to the traces of the original pattern. For most patterns the set of realizable patterns that correspond to the same set of traces, is finite.

Pattern subsumption The set of realizable patterns computed by pattern refinement is not guaranteed to be minimal. In particular, it may be the case that the set includes two patterns $pt1, pt2$, such that $traces(P, pt1) \subset traces(P, pt2)$. In such cases, $pt1$ is redundant.

We have implemented *pattern subsumption* tests in Scyther. These tests try to determine for some patterns pt, pt' of a protocol P whether or not $traces(P, pt') \subset traces(P, pt)$. Using these tests, the result set of the refinement algorithm is minimized. This has only effect in a very limited number of cases, but has been implemented to guarantee a form of minimality over the characterization.

Complete pattern characterization Using the pattern refinement algorithm in combination with the subsumption tests, it is straightforward to construct a complete characterization algorithm.

1. Given a protocol and a role, a pattern is constructed, with an arbitrary thread identifier, e.g. 0, which is a direct translation of the description of the role into trace events: roles are replaced by local role variables, e.g. R is replaced by the variable $R\sharp 0$, and local constants and variables are similarly bound to the thread, i.e. a constant ni is replaced by the constant $ni\sharp 0$, and a variable X is replaced by the variable $X\sharp 0$. (Note that in terms of Scyther internals, this corresponds to simply placing a "reachable" claim at the end of the role.)
2. The resulting pattern is refined into the full set of realizable patterns, if possible. (In terms of Scyther internals, this is captured by requesting all attacks.)
3. Finally, the realizable pattern set is minimized by using the subsumption tests.

Example 4. In Figure 2 we provide an example characterization of the responder role of the Needham-Schroeder protocol. The result is two patterns, which correspond to either the expected behaviour of the protocol on the left, or the man-in-the-middle attack on the right. As a result, any trace of the protocol that includes the events of the responder role, will necessarily also include the events from one of the patterns in the complete characterization.

Comparing both characterization concepts At first sight, upon considering the Needham-Schroeder example, it seems that both concepts differ significantly, in the sense that the shape characterization is more coarse and therefore more concise, than the pattern characterization.

The shape characterization is more coarse (and thus may be more concise) because the agents whose keys are not in the non set, may be instantiated by honest or dishonest agents. In the cases where both these options can lead to traces of the protocol, the pattern characterization will always distinguish between them. As a result, we find that the two Needham-Schroeder patterns for the responder role are subsumed by one shape in the skeletons-and-homomorphisms approach.

We note that the more coarse definition in terms of shapes has a disadvantage: the relation between the shape and its possible executions is slightly more complicated. Consider e.g. the example in Figure 1. In the case that $privk(C)$ is not known to the intruder, and $C \neq B$, the shape does not represent a valid execution. Hence, in some sense, the shape is an over-approximation of all possible

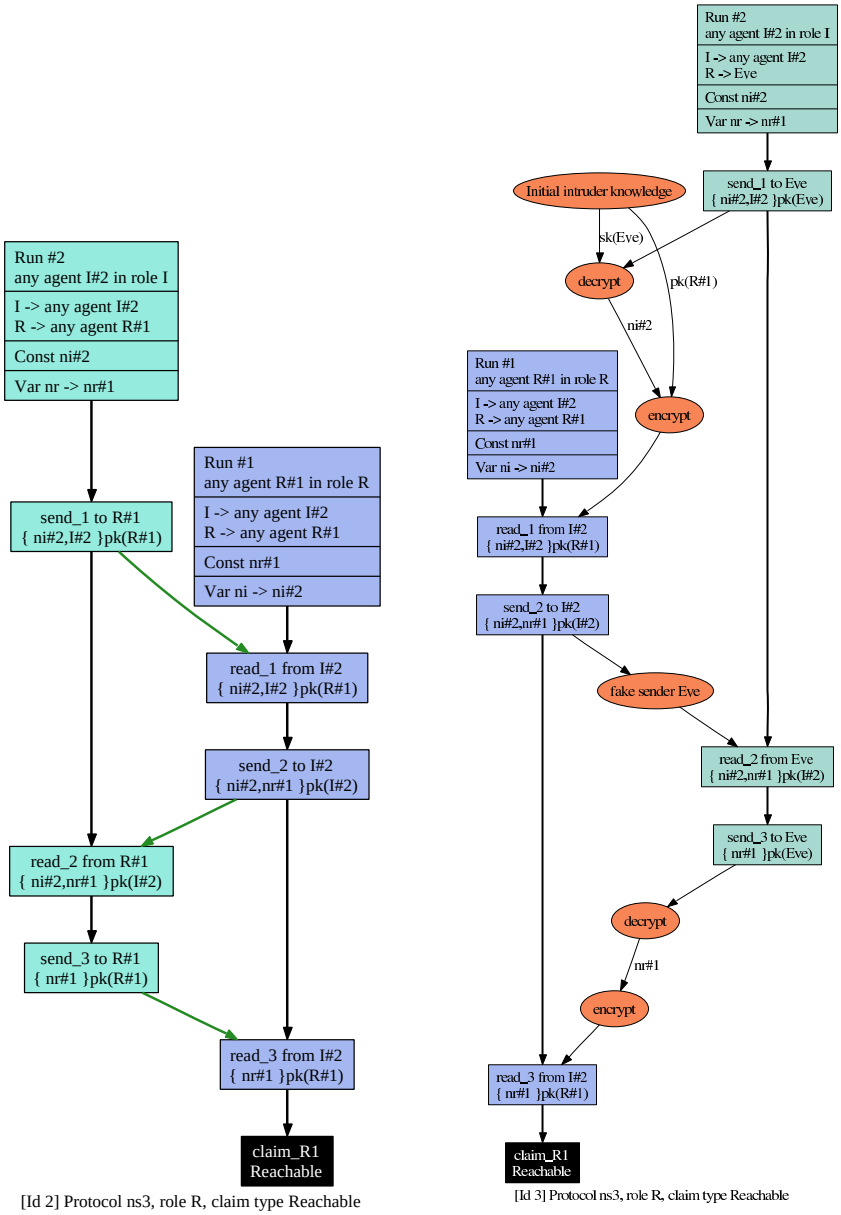

Fig. 2. Needham-Schroeder, role R complete characterization: exactly two patterns

valid executions. In the case of patterns, any well-typed substitution of variables is by definition a valid execution trace.

However, as the more abstract shapes can be more concise, an obvious next question is whether we can abstract from patterns in the same way, to obtain a coarser characterization, which could correspond more closely to the shapes characterization. Hence we introduce the notion of *shape approximation*. In this approximation, an agent communicating with a compromised agent (usually represented by Eve in Scyther) may also choose to communicate with any honest agent. Internally, this involves two changes:

- All occurrences of Eve are turned into a variable *Anybody* of type Agent
- All patterns are projected onto their regular behaviour only, i.e. intruder events are filtered out

After this, pattern subsumption is tested again, and certain patterns will be subsumed by others: e.g. the normal behaviour of the Needham-Schroeder protocol is now a subclass of the regular behaviour of the man-in-the-middle attack, in which *Anybody* is mapped to Bob.

We discuss empirical results obtained from this approximation in Section 4.

4 Preliminary empirical results

We have applied the modified Scyther algorithm to the protocol models found in the Scyther distribution. These include a large subset of the SPORE repository [9]. We have characterized each role of these protocols, by default named as **I** (Initiator), **R** (Responder) and **S** (Server). The names of the protocols correspond directly to the names of the protocols as chosen in the Scyther distribution.

The Scyther tool was run using its default settings (maxruns=5, typed variables).

In this default setup, not all roles could be completely characterized. Of the 115 roles characterized in this set, Scyther produced complete characterizations for 75 roles. We summarize the results in Table 1 and 2.

In Table 1 we list the protocols for which Scyther was able to characterize all roles. For each role, the number of patterns in the characterization is reported. E.g. the two party Needham Schroeder protocol (ns3) has 2 patterns for its responder role, representing the correct behaviour and the man-in-the-middle attack. The Needham-Schroeder-Lowe protocol (nsl3) has only 1 pattern for the responder, reflecting that only the correct behaviour is possible.

In Table 2 we list the protocols for which the tool did not yield full characterizations of each role. For roles for which the characterization was not complete, we write $\geq X$ to denote that Scyther finds X patterns, but cannot determine whether these represent a *complete* characterization, or that more patterns are needed.

Approximating Shape Characterization using patterns We have experimented with an approximation of the Complete Shape Characterization of protocols. In

Table 1. Complete Pattern Characterizations (Scyther)

Protocol ID	I	R	S
andrew	2	1	-
andrew-Ban	1	1	-
andrew-LoweBan	1	1	-
ccitt509-1	1	1	-
ccitt509-1c	1	1	-
ccitt509-ban3	1	1	-
ns3	1	2	-
nsl3	1	1	-
ksl-Lowe	1	1	1
neustub-GuttmanHwang	1	2	1
neustub-GuttmanHwang-Repeat	1	1	1
otwayrees	3	5	5
woolamPi-1	1	5	2
woolamPi-2	1	6	2
woolamPi-3	1	4	1
woolamPi-f	1	2	1
yahalom	1	1	1
yahalom-BAN	2	3	1
yahalom-Lowe	1	1	1
yahalom-Paulson	1	2	1

this approximation, an agent communicating with a compromised agent (represented by Eve) may also choose to communicate with any honest agent.

In our test set, this shape approximation did not cause many changes. We summarize the changes in Table 3. Note that as we did not have access to the exact protocol descriptions used in CPSA, the numbers in the last table may not be comparable to the numbers produced by Scyther.

5 Preliminary conclusions and future work

We have sketched an alternative formulation of the complete characterization of a protocol role in terms of *realizable patterns*. This approach can result in a larger characterization set than the shapes-based approach by Doghmi, Guttman and Thayer, but has the advantage of having a more direct relation to the execution traces.

The protocol verification tool Scyther was extended to generate the complete characterizations in patterns. Furthermore, we have implemented an alternative type of patterns, intended to approximate the coarseness of the shapes approach. Extensive empirical testing has already been performed with a large set of protocols. Our approach can deal with all reported CPSA examples, and also with protocols that have been reported to be problematic for CPSA, such as Otway-Rees, for which CPSA currently does not terminate.

Table 2. Incomplete Pattern Characterizations (Scyther)

Protocol ID	I	R	S
andrew-Concrete	≥ 2	≥ 2	-
ccitt509-3	1	≥ 20	-
smartright	1	≥ 1	-
denningSacco	≥ 1	≥ 1	1
denningSacco-Lowe	≥ 2	≥ 2	1
kaochow	1	≥ 2	1
kaochow-2	≥ 1	≥ 2	1
kaochow-3	≥ 1	≥ 2	1
ksl	≥ 2	≥ 2	1
needhamschroederpk	≥ 20	≥ 20	1
needhamschroederpk-Lowe	≥ 20	≥ 20	1
needhamschroedersk	≥ 1	≥ 1	1
needhamschroedersk-amend	≥ 1	≥ 1	≥ 1
neustub-Hwang	2	≥ 2	1
spliceAS	≥ 18	≥ 20	1
spliceAS-CJ	≥ 18	≥ 19	1
spliceAS-HC	≥ 18	≥ 20	1
tmn	≥ 15	1	12
wmf	1	≥ 4	≥ 4
wmf-Lowe	≥ 4	≥ 4	≥ 4
woolam	3	≥ 3	3
woolamPi	1	≥ 7	≥ 3

Table 3. Scyther: from patterns to shape approximation

Protocol ID	Role	Patterns	Shape approx.
ccitt509-3	R	≥ 20	≥ 5
ns3	R	2	1
needhamschroederpk	I	≥ 20	≥ 10
needhamschroederpk	R	≥ 20	≥ 10
needhamschroederpk-Lowe	I	≥ 20	≥ 10
needhamschroederpk-Lowe	R	≥ 20	≥ 10
spliceAS	I	≥ 18	≥ 12
spliceAS	R	≥ 20	≥ 16
spliceAS-CJ	I	≥ 18	≥ 12
spliceAS-CJ	R	≥ 19	≥ 9
spliceAS-HC	I	≥ 18	≥ 12
spliceAS-HC	R	≥ 20	≥ 16
tmn	I	≥ 15	≥ 9
tmn	S	12	1

As future work we will further analyze the empirical results, and optimize the pattern subsumption testing, which is currently the most time-consuming step in our method. Nevertheless, all characterizations performed here ran in less than a few seconds. In particular, in the cases which were also handled by CPSA, it seems Scyther is at least twice as fast.

Our preliminary results indicate that the pattern approach for complete characterization is both useful and feasible. It is useful in the sense that the relation to execution traces is straightforward, and feasible in the sense that many protocols have finite pattern characterizations, and that Scyther can efficiently compute the complete characterizations in terms of patterns.

References

1. Blanchet, B.: An efficient cryptographic protocol verifier based on Prolog rules. In: Proc. 14th IEEE Computer Security Foundations Workshop (CSFW), IEEE Computer Society (2001) 82–96
2. Cremers, C.: Scyther - Semantics and Verification of Security Protocols. Ph.D. dissertation, Eindhoven University of Technology (2006)
3. Armando, A., Basin, D., Boichut, Y., Chevalier, Y., Compagna, L., Cuellar, J., Drielsma, P.H., Heám, P.C., Kouchnarenko, O., Mantovani, J., Mödersheim, S., von Oheimb, D., R., M., Santiago, J., Turuani, M., Viganò, L., Vigneron, L.: The AVISPA tool for the automated validation of internet security protocols and applications. In: Proc. of CAV'2005. LNCS 3576. Springer-Verlag (2005) 281–285
4. Cremers, C., Lafourcade, P.: Comparing state spaces in automatic security protocol verification. In: Proc. 7th International Workshop on Automated Verification of Critical Systems (AVoCS). Electron. Notes Theor. Comput. Sci., Elsevier Science Publishers B. V. (2007)
5. Doghmi, S., Guttman, J., Thayer, F.: Searching for shapes in cryptographic protocols. In: Proc. 13th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Volume 4424 of Lecture Notes in Computer Science., Springer-Verlag (2007) 523–537
6. Doghmi, S., Guttman, J., Thayer, F.: Skeletons and the shapes of bundles. http://www.ccs.neu.edu/home/guttman/skeletons.pdf (2006)
7. Doghmi, S., Guttman, J., Thayer, F.: Completeness of the authentication tests. In: Proc. 12th European Symposium on Research in Computer Security (ESORICS). (2007)
8. Thayer, F., Herzog, J., Guttman, J.: Strand spaces: Proving security protocols correct. Journal of Computer Security **7** (1999) 191–230
9. Jacquemard, F.: Security Protocols Open Repository (SPORE) (2007) Available at http://www.lsv.ens-cachan.fr/spore/index.html.